arXiv:2506.14054v1 [cs.LG] 16 Jun 2025

Scientifically-Interpretable Reasoning Network (ScIReN): Uncovering the Black-Box of Nature

Joshua Fan[1*] **Haodi Xu**[2*] **Feng Tao** [3*] **Md Nasim**[1]
Marc Grimson[1] **Yiqi Luo**[2] **Carla P. Gomes**[1]
[1]Cornell University, Department of Computer Science
[2]Cornell University, School of Integrative Plant Science, Soil and Crop Sciences Section
[3]Cornell University, Department of Ecology and Evolutionary Biology
{jyf6,hx293,feng.tao,mn637,mg2422,yl2735}@cornell.edu, gomes@cs.cornell.edu

Abstract

Neural networks are a powerful tool for learning patterns from data. However, they do not respect known scientific laws, nor can they reveal novel scientific insights due to their black-box nature. In contrast, scientific reasoning distills biological or physical principles from observations and controlled experiments, and quantitatively interprets them with process-based models made of mathematical equations. Yet, process-based models rely on numerous free parameters that must be set in an ad-hoc manner, and thus often fit observations poorly in cross-scale predictions. While prior work has embedded process-based models in conventional neural networks, discovering interpretable relationships between parameters in process-based models and input features is still a grand challenge for scientific discovery. We thus propose *Scientifically-Interpretable Reasoning Network (ScIReN)*, a fully-transparent framework that combines interpretable neural and process-based reasoning. An interpretable encoder predicts *scientifically-meaningful latent parameters*, which are then passed through a differentiable process-based decoder to predict labeled output variables. ScIReN leverages Kolmogorov-Arnold networks (KAN) to ensure the encoder is fully interpretable and reveals relationships between input features and latent parameters; it uses smoothness penalties to balance expressivity and simplicity. ScIReN also uses a novel hard-sigmoid constraint layer to restrict latent parameters to meaningful ranges defined by scientific prior knowledge, further enhancing its interpretability. While the embedded process-based model enforces established scientific knowledge, the KAN-based encoder reveals new scientific mechanisms and relationships hidden in conventional black-box models. We apply ScIReN on two tasks: simulating the flow of organic carbon through soils, and modeling ecosystem respiration from plants. In both tasks, ScIReN outperforms black-box networks in predictive accuracy while providing substantial scientific interpretability – it can infer latent scientific mechanisms and their relationships with input features.

1 Introduction

Deep learning has emerged as a powerful technique to learn complex patterns from data [29], leading to advancements in many areas of science, such as protein modeling [22], genetics [43], and climate science [48]. However, the black-box nature of neural networks makes it challenging to translate their predictive ability into scientific insights that can be understood by humans [40]. Neural networks can even produce predictions that violate basic scientific laws like mass conservation [24]. It is still a

[*]Equal contribution.

Preprint. Under review.

grand challenge to leverage existing scientific knowledge – insights that have been gained through centuries of experimentation and theorizing – to improve the neural network [23].

As a motivating example, consider the task of modeling the soil organic carbon (SOC) cycle. Soils play a crucial role in the global carbon cycle – they store more carbon than plants and the atmosphere combined [21], and have the potential to mitigate climate change by sequestering carbon dioxide from the atmosphere [27]. Yet it is difficult to understand how carbon flows through the soil, leading to high uncertainties in future climate projections [38]. Out of the box, neural networks can predict the total amount of organic carbon stored in the soil at each location [52]. However, neural networks do not respect prior scientific knowledge about the carbon cycle, nor do they give us new insights into the biogeochemical processes governing the soil carbon cycle – which control how much carbon can be stored in the soil and how long it will remain sequestered there.

To gain insight into these biogeochemical processes, ecologists have developed process-based models to simulate the soil carbon cycle [37]. These models have been meticulously developed through years of research – they may contain hundreds of pools representing different types of carbon, and matrices specifying flux rates between each pair of pools [39, 18, 34]. However, these models contain many unknown parameters, which traditionally must be tuned by human experts through an inefficient trial-and-error process [36]. These models often cannot fit observed data well, especially in cross-scale predictions, mainly due to a poor understanding of the relationships between environmental conditions and the parameter values. This is a key bottleneck in using process-based models for verifiable predictions on the global soil carbon cycle and its response to climate change.

A few prior works in hydrology [55, 15], phenology [56], ecosystem modeling [46], and soil biogeochemistry [59] have proposed implementing process-based models in a differentiable way so that poorly-understood parameters can be optimized by backpropagation or replaced with neural networks. While these approaches aim to combine process-based models with deep learning, the neural network components remain too opaque for scientists to understand, which leaves the relationships between the latent parameters and input features unclear.

To address these limitations, we propose a *Scientifically-Interpretable Reasoning Network (ScIReN)*, an end-to-end differentiable framework that embeds scientific process-based models into a *fully-transparent* neural model, creating a system that respects scientific knowledge, learns from data in an end-to-end manner, and can discover new scientific relationships. ScIReN contains three main components. First, a learnable encoder takes in environmental features for a given location, and predicts scientifically-meaningful latent parameters. We use a *fully-interpretable network* such as a sparse Kolmogorov-Arnold network [32] to make this portion fully transparent, allowing scientists to understand the relationships between these latent parameters and input features. Second, a novel hard-sigmoid constraint layer projects these parameters into a physically-plausible range set by prior knowledge. Finally, a process-based decoder uses these predicted parameters to simulate the flow of carbon through the soil based on scientific knowledge, and finally predicts the output variables (e.g. amount of soil organic carbon at each depth in the soil). We compare the predicted outputs with ground-truth labels, and backpropagate this loss to train the entire system.

Our main contributions are: (1) We propose a *fully-interpretable* framework for combining scientific reasoning (in the form of process-based modeling) with data-driven learning. The model infers latent mechanisms that are quantified by scientific parameters used in process-based models, and reveals their relationships to input variables in a transparent way. **(2)** We balance smoothness and expressivity in the learned functional relationships by using B-splines with smoothness penalties. **(3)** We propose a novel hard-sigmoid constraint layer to constrain the scientific parameters to fall within physically-plausible ranges. **(4)** We validate our technique on two scientific domains. First, we apply ScIReN on a model of ecosystem respiration; ScIReN discovers the correct relationships between latent parameters and input features, while other methods do not. It also improves out-of-distribution extrapolation compared to black-box models. Second, we test ScIReN on the challenging task of modeling the soil carbon cycle, where we need to simulate carbon flows through 140 soil pools at each location. On a synthetic dataset, ScIReN is able to predict *unlabeled* latent biogeochemical parameters and accurately retrieve their relationships with environmental features. With real data, ScIReN simulates observed soil carbon amounts with accuracy comparable to blackbox methods. We hope our work inspires other researchers to apply and extend ScIReN across diverse scientific tasks, advancing AI's capacity for interpretable scientific discovery.

2 Related work

Knowledge-Guided Machine Learning. There is a rich history of incorporating prior knowledge into neural networks by modifying the loss function, pretraining procedure, or model architecture [25, 58]. A simple approach is to add a loss term that penalizes when physical laws (such as energy conservation) are violated [10, 4, 19]. For example, the density of water is known to increase monotonically with depth; thus, Daw et al. [10] and Jia et al. [19] use a "monotonicity loss" that penalizes when the model's predictions are not monotonic. Physics-informed neural networks assume that the governing equation of a system is known, and penalize when its predictions and gradients violate this equation [44]. Although these approaches encourage the model to comply with physical laws, they cannot guarantee that the model will fully satisfy them [58]. Adding these loss terms makes the loss landscape more complex, making the model difficult to train [26, 45]. These approaches also do not help scientists gain new insights about the relationships between variables.

Scientific process-based models can also be used to generate synthetic data to pretrain the network, as done in lake temperature modeling [20] and agriculture [30]. While this increases the amount of data available to train the network, simply training on process-based model outputs does not provide new insights into physical processes, nor does it guarantee that the predictions satisfy physical constraints.

One can also design model architectures to encode prior knowledge. Convolutional neural networks encode inductive biases such as translation equivariance and locality into the model architecture, reducing the amount of training data required to classify images [28]. In lake temperature modeling, Daw et al. [9] design a monotonicity-preserving LSTM that produces monotonically-increasing intermediate variables (densities) by design. In agriculture, Liu et al. [30] design a hierarchical neural network that incorporates causal relations between different variables. However, it is difficult to design a new architecture for every problem, and these models are still black boxes.

Combining reasoning and learning. A few works reason about constraints and prior knowledge within the network itself. Deep Reasoning Networks use entropy-based losses to encourage the latent space to be interpretable and satisfy constraints (e.g. in Sudoku, each row must contain exactly one of each number) [5]. This approach was used to solve the phase-mapping problem in materials discovery, where the constraints are thermodynamic rules [6]. CLR-DRNets enhanced the reasoning process using a modified LSTM, and used curriculum learning to improve trainability [3]. Physically-informed Graph-based DRNets add a physical decoder that reconstructs X-ray diffraction patterns based on Bragg's law [41]. CS-SUNet uses a smoothness loss to encourage pixels with similar input features to have similar predictions; this inductive bias helps the model predict vegetation productivity at a much finer resolution than the labels [14]. While these approaches have an interpretable latent space, the bulk of the network remains uninterpretable.

Process-based models. Scientists develop *process-based models* to simulate physical processes based on domain knowledge [8]. These models consist of mathematical equations that describe the relationships between various variables. In soil science, *pool-and-flux* models are common, where a matrix equation tracks the amount of carbon at each soil depth and matter type [37]. Transition matrices encode the rate at which carbon is transferred between pools, which are functions of soil and climate properties. Many scientific models can be unified under this matrix form [18, 39].

Unfortunately, despite their sophistication, these models have difficulty matching real observations, and have numerous unknown parameters that are traditionally set in an ad-hoc way. These unknown parameters need to vary across space and (sometimes) time, but it is unclear how to estimate these parameters [36]. A state-of-the-art approach for setting these parameters is PRODA [53]. PRODA first runs Bayesian data assimilation at each location separately to find optimal biogeochemical parameters for each location. Then, a neural network is trained to predict these optimal parameters given environmental covariates. While this approach is effective, it is computationally expensive, and is not always robust since each location's parameters are estimated with only a few observations.

Differentiable Process-Based Models. A few works integrate process-based models into neural networks in an end-to-end differentiable framework; this has been called *differentiable parameter learning* [55], *hybrid modeling* [46], or *differentiable process-based modeling* [50]. For example, [56, 46, 55] used a process-based model as the main backbone, but replaced some components with neural networks when the functional form of the relationship was unknown. By implementing the

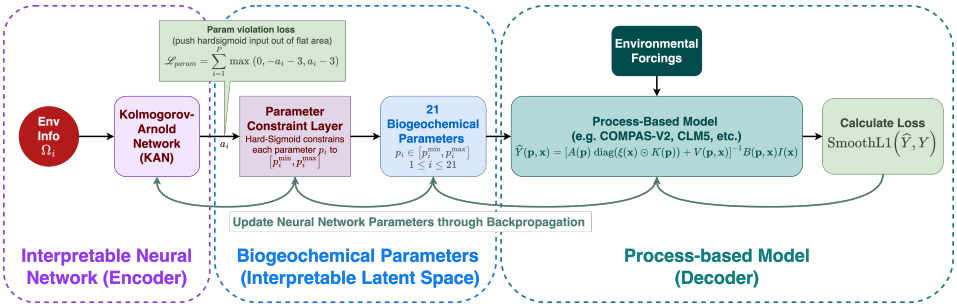

Figure 1: Overview of ScIReN. The encoder is a fully-interpretable model that reveals functional relationships between input features and latent scientific parameters. A constraint layer constrains the latent parameters to be within a pre-specified range, and the process-based decoder uses the predicted latent parameters to simulate the physical process and predict the final labeled output.

process-based model in a differentiable way, the model could be trained end-to-end, and unknown components could be fit using data. Xu et al. [59] scaled this approach up to a more complex soil carbon model with 21 unknown parameters and 140 carbon pools. However, the neural network component is still opaque, making it difficult for scientists to discover new relationships and insights.

Explainable and Interpretable AI. A subfield of machine learning aims to interpret how neural networks make predictions [42]. *Post-hoc feature attribution methods* such as SHAP [35] or Integrated Gradients [51] estimate the impact of each feature on the model's prediction for a given example. *Local surrogate models* such as LIME [47] fit an interpretable surrogate model that approximates the black-box model in a small area, but it is hard to infer global behavior from local approximations. *Marginal effect plots* such as Partial Dependence Plots [16] or Accumulated Local Effects plots [2] visualize how each feature affects the output on average. These methods are mere approximations of a black-box model, and can produce misleading explanations if the approximation is poor [49].

On the other hand, *inherently-interpretable models* aim to make the entire model transparent by design [49]. In linear regression, the coefficients directly reveal how each input affects the prediction. Unfortunately, linear models are not expressive enough for many applications. Neural additive models add expressivity by modeling the output variable as the sum of single-variable functions (one for each input) [1]. However, they cannot represent complex interactions between variables. Kolmogorov-Arnold networks stack multiple additive models into layers, and provably have the ability to approximate any function [32], but are harder to interpret. All of these approaches are typically applied in supervised settings – they have not been combined with process-based models or scientific knowledge to predict *unlabeled* variables.

3 Methods

To combine scientific knowledge and data-driven learning into a fully-transparent model, ScIReN contains three main components. First, a neural network encoder f_{NN} (with learnable weights θ) takes in input features $\mathbf{x} \in \mathbb{R}^D$ (e.g. soil and climate variables at a given location), and outputs unconstrained latent parameters $\mathbf{a} \in \mathbb{R}^P$: $\mathbf{a} = f_{NN}(\mathbf{x}; \theta)$. In ScIReN, f_{NN} should be fully-transparent, which can be achieved by making it a neural additive model or sparse Kolmogorov-Arnold network. The latent parameters are scientifically-meaningful variables that govern the underlying physical process, yet cannot be observed directly. Secondly, a constraint layer maps the unconstrained parameters to a scientifically-plausible prior range for each parameter set by prior knowledge; $\mathbf{p} = \mathrm{Proj}(\mathbf{a})$. Finally, the constrained parameters are passed through a *fixed, deterministic* process-based decoder g_{PBM}, which simulates the system and produces the final predicted output $\hat{y} = g_{PBM}(\mathbf{p})$. We compare this with the true label and backpropagate. The framework is summarized in Figure 1, and we elaborate on each component below.

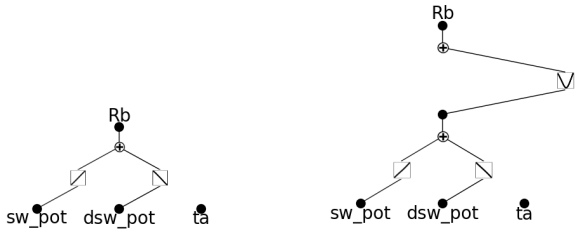

Figure 2: Learned encoder examples: 1-layer KAN (left) and 2-layer KAN (right)

3.1 Encoder: Learned Interpretable Relationships

We want to learn a function f_{NN} mapping observed input features $\mathbf{x}$ to latent scientific parameters $\mathbf{p}$. In prior work, this function is typically a fully-connected neural network [59, 56, 46, 55]. However, soil scientists want to understand how biogeochemical parameters (e.g. transfer rates between pools) depend on input features (e.g. temperature), but this is difficult to read out from a neural network.

Recently, a line of work has aimed to produce neural networks that are inherently interpretable while being expressive. Neural additive models (NAM) [1] model the output as the sum of single-variable functions of each input feature. Specifically, they learn a neural network $\phi_i : \mathbb{R} \to \mathbb{R}$ (with one input and one output) for each feature x_i, and sum contributions from each feature into the output:

$$NAM(\mathbf{x}) = b + \sum_{i=1}^{D} \phi_i(x_i; \theta_i) \tag{1}$$

where $(b, \{\theta_i\}_{i=1}^{D})$ are learnable parameters trained via backpropagation. While NAM is quite expressive, it cannot model non-additive feature interactions, which are important in soil science [11].

To increase the expressivity of neural additive models, we can generate intermediate variables using neural additive model of the inputs, then further use a neural additive model on the intermediate variables to generate the output. Specifically, generate intermediate variables $\mathbf{z} = \{z_1, \ldots, z_H\}$ as

$$z_j = NAM_j(\mathbf{x}) = b_j + \sum_{i=1}^{D} \phi_{j,i}(x_i), \quad \forall j \in [1, H] \tag{2}$$

Now define each output variable a_p as a neural additive model over the intermediate variables

$$a_p = NAM_p(\mathbf{z}) = b_p + \sum_{j=1}^{H} \Phi_{p,j}(z_j) = b'_p + \sum_{j=1}^{H} \Phi_{p,j} \left(\sum_{i=1}^{D} \phi_{j,i}(x_i) \right) \tag{3}$$

where all the bias terms are collected into b'_p. This is now the same form as a two-layer Kolmogorov-Arnold network (KAN, see equation 2.1 in [32]). By the Kolmogorov-Arnold theorem, this multi-layer stack of neural additive models can approximate any multivariate continuous function [32].

Two examples of learned encoders are shown in Figure 2. On the left, a 1-layer KAN models the latent parameter $Rb \approx w_1 \cdot sw_pot + (-w_2) \cdot dsw_pot$. One can immediately see how each feature influences Rb. On the right, a 2-layer KAN is needed to capture this nonlinear relationship, where an intermediate variable $Rb' \approx w_1 \cdot sw_pot + (-w_2) \cdot dsw_pot$, and then $Rb \approx |Rb'|$.

3.2 Sparsity and Smoothness Regularization

Kolmogorov-Arnold networks are still hard to interpret if each output depends on many inputs. Liu et al. [31] propose entropy regularization to sparsify the network. Specifically, we compute an importance score for each edge, as the mean absolute deviation of the output activations from the edge, weighted by their eventual contribution to the final output variables [31]. We denote this score for edge $(i \to j)$ in layer l as $E^l_{i,j}$. We encourage the entropy of the edge importance distribution to be low (making the network choose a few important edges and push others towards zero). We also encourage the absolute deviation of each edge's outputs to be low via an L1 penalty:

$$e^l_{i,j} = \frac{E^l_{i,j}}{\sum_{i,j} E^l_{i,j}} \quad \text{(normalize edge importance to sum to 1)} \tag{4}$$

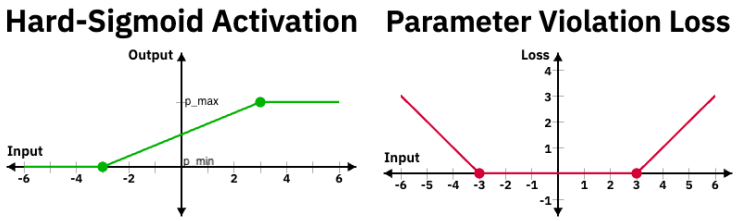

Figure 3: **Left:** The hard-sigmoid function constrains parameters to $[p_{min}, p_{max}]$, without adding nonlinearity. **Right:** parameter violation loss pushes the hard-sigmoid input away from flat regions.

$$\mathcal{L}_{entropy} = -\sum_{l}\sum_{i,j} e^{l}_{i,j} \log e^{l}_{i,j}; \quad \mathcal{L}_{L1} = -\sum_{l}\sum_{i,j} |E^{l}_{i,j}| \tag{5}$$

Note that KANs parameterize the learnable edge activation functions Φ, ϕ using B-splines instead of neural networks. B-splines represent a curve as a weighted sum of basis functions, each of which peaks at a different point on the x-axis (see [12] for details). We can apply a second-order difference penalty on the spline coefficients – this encourages the coefficients to change in a linear way, making the function more linear [13]. This allows us to increase the number of basis functions (knots) and the expressivity of the function while maintaining smoothness and preventing overfitting. If $c_1 \dots c_G$ are the coefficients, the penalty is

$$\mathcal{L}_{smooth} = \sum_{i=1}^{G-2} ((c_{i+2} - c_{i+1}) - (c_{i+1} - c_i))^2 \tag{6}$$

3.3 Linear Parameter Constraint Layer

For some latent scientific parameters p_i, there is a known prior range $[p_i^{min}, p_i^{max}]$ based on prior knowledge and physical plausibility. Xu et al. [59] applied a sigmoid function to the encoder output a_i to force the predicted parameter into the prior range: $p_i = \sigma(a_i)$. However, this adds nonlinearity and harms interpretability. For example, suppose a parameter p_i is actually a linear function of input variable x_j, $p_i = wx_j$. If the parameter was constrained using a sigmoid, the unconstrained encoder would have to learn $a_i = \sigma^{-1}(wx_j)$, so that after the sigmoid function the parameter becomes $p_i = \sigma(\sigma^{-1}(wx_j)) = wx_j$. The additional inverse sigmoid makes the function less interpretable.

Instead, we use a piecewise linear *hard-sigmoid* function (figure 3 left) to constrain the parameters:

$$p_i = \text{Hardsigmoid}(a_i) = \begin{cases} p_i^{min} & \text{if } a_i \leq -3, \\ p_i^{max} & \text{if } a_i \geq +3, \\ \frac{1}{6\tau}(p_i^{max} - p_i^{min}) \cdot a_i + \frac{1}{2}(p_i^{max} + p_i^{min}) & \text{otherwise} \end{cases} \tag{7}$$

where τ is a hyperparameter that influences how extreme the predicted parameters are at initialization.

While the gradient $\frac{\partial p_i}{\partial a_i}$ is zero when a_i is outside the range $[-3, 3]$, we can add another loss that places a penalty when a_i is in the flat area (figure 3 right).

$$\mathcal{L}_{param} = \sum_{i=1}^{P} \max(0, -a_i - 3, a_i - 3) \tag{8}$$

This provides a gradient that pushes a_i towards the linear range $[-3, 3]$ when it is in the flat range.

For the ecosystem respiration model, we only know that the latent parameter is nonnegative, so we use ReLU to impose the constraint, and a flipped ReLU loss to push inputs out of the flat region.

$$p_i = \max(a_i, 0); \quad \mathcal{L}_{param} = \max(-a_i, 0) \tag{9}$$

3.4 Differentiable Process-Based Decoder

ScIReN uses a process-based model that expresses output variables as a *fixed, differentiable function* of scientific parameters and input variables: $\hat{y} = g_{PBM}(\mathbf{p}, \mathbf{x})$. Two examples are described below.

Ecosystem respiration. Consider the model of ecosystem respiration in [46]. Based on scientific knowledge, we write the output variable R_{eco} (ecosystem respiration) as a differentiable function of two latent parameters, base respiration R_b and temperature sensitivity Q_{10}. Specifically:

$$R_{eco} = g_{PBM}(\mathbf{p}, \mathbf{x}) = R_b(\mathbf{x}) \cdot Q_{10}^{\frac{t_a - T_{ref}}{10}} \tag{10}$$

where the latent parameters are $\mathbf{p} = \{R_b, Q_{10}\}$, the input features are $\mathbf{x} = \{sw_pot, dsw_pot, t_a\}$, and $T_{ref} = 15$. R_b has an unknown relationship with the input features (learned in the encoder), while Q_{10} is a learnable constant.

Soil carbon modeling. As a more complex process, we use the soil organic carbon module from Community Land Model 5 (CLM5) [34], which tracks the amount of soil organic carbon (SOC) in 140 pools in the soil (20 depths and 7 material types per depth). Denote the amount of carbon in pool i as Y_i. The core of the model is a mass conservation equation for each pool, where the change in carbon equals inflow (from plants and other pools) minus outflow (to other pools or the atmosphere):

$$\frac{dY_i(t)}{dt} = \text{inflow to pool } i - \text{outflow from pool } i \tag{11}$$

The equations for each pool can be combined into a single matrix equation. If we assume steady state ($\frac{dY_i(t)}{dt} = 0$), we can write Y as a function of biogeochemical parameters $\mathbf{p}$ and input features $\mathbf{x}$:

$$\hat{Y}(\mathbf{p}, \mathbf{x}) = \left[A(\mathbf{p})\ \text{diag}(\xi(\mathbf{x}) \odot K(\mathbf{p})) + V(\mathbf{p}, \mathbf{x})\right]^{-1} B(\mathbf{p}, \mathbf{x}) I(\mathbf{x}) \tag{12}$$

The details of this equation are explained in the Appendix A.4. For now, it is sufficient to note that the process-based model takes in 21 latent biogeochemical parameters $\mathbf{p}$, uses the parameters to construct matrices describing carbon fluxes and decomposition, and finally predicts the amount of carbon in 140 pools (20 layers and 7 pools), $\hat{Y}$. Each operation (including matrix inversion) is differentiable and can be implemented in PyTorch. Note that our labeled data only contains aggregate SOC amounts at specific depths (which may not match the 20 fixed layers). Thus, we sum up the SOC pools at each layer, and linearly interpolate to predict SOC at the observed depths.

3.5 Final Loss

The final loss contains a smooth L1 loss between the predicted and ground-truth output variables, as well as the parameter regularization loss and KAN regularization losses:

$$\mathcal{L} = \sum_{i=1}^{N} \left[SmoothL1(\hat{Y}_i, Y_i) \right] + \lambda_{param}\mathcal{L}_{param} + \lambda_{L1}\mathcal{L}_{L1} + \lambda_{entropy}\mathcal{L}_{entropy} + \lambda_{smooth}\mathcal{L}_{smooth}$$

Since the entire network is differentiable, we can backpropagate the loss through the process-based model to optimize the latent parameters and the learnable weights of the neural network. The loss weights λ are tuned on a validation set for each domain; they are relatively intuitive to tune since we can visualize whether the KAN is too sparse/dense and whether the functions are too jagged/smooth.

4 Experiments

We test our approach on two domains: the model of ecosystem respiration in [46], and the CLM5 soil carbon cycle model on sites in the contiguous United States [59].

4.1 Evaluation Metrics and Baselines

For each dataset, we evaluate the test-set accuracy (R^2) of various methods in predicting observed variables. For experiments using synthetic labels, we also have ground-truth latent parameter values and functional relationships, so we can evaluate how accurately each method recovers these parameters and relationships. To evaluate functional relationship quality, for both ground-truth (synthetic) relationships and our learned models, we first compute the fraction of variance in the output that is explained by each input feature. For 1-layer KAN, we can simply compute the variance of each edge's post-activation outputs, and divide by the total variance in the output. For other models, this is non-trivial; we use Partial Dependence Variance [17] to estimate how much variance in the

Table 1: Ecosystem respiration, linear R_b. Mean and standard deviation across 5 seeds.

Method	R^2 (observed, ↑)	R^2 (latent, ↑)	KL, functional relationships (↓)
Pure-NN	0.968 ± 0.004	N/A	N/A
Blackbox-Hybrid, nonlinear constraint	0.972 ± 0.003	0.818 ± 0.052	0.284 ± 0.041
Blackbox-Hybrid, linear constraint	0.976 ± 0.001	0.941 ± 0.031	0.161 ± 0.068
ScIReN, nonlinear constraint (1-layer KAN)	0.975 ± 0.000	0.995 ± 0.000	0.002 ± 0.001
ScIReN, linear constraint (1-layer KAN)	$\mathbf{0.976} \pm 0.000$	$\mathbf{1.000} \pm 0.000$	$\mathbf{0.001} \pm 0.001$

Table 2: Ecosystem respiration, nonlinear R_b. Mean and standard deviation across 5 seeds.

Method	R^2 (observed, ↑)	R^2 (latent, ↑)	KL, functional relationships (↓)
Pure-NN	0.946 ± 0.004	N/A	N/A
Blackbox-Hybrid, nonlinear constraint	0.948 ± 0.008	-0.265 ± 0.355	0.937 ± 0.168
Blackbox-Hybrid, linear constraint	0.961 ± 0.001	0.949 ± 0.059	0.199 ± 0.094
ScIReN, linear constraint (1-layer KAN)	0.651 ± 0.014	-1.225 ± 0.668	1.182 ± 0.085
ScIReN, linear constraint (2-layer KAN)	$\mathbf{0.961} \pm 0.001$	$\mathbf{0.993} \pm 0.005$	$\mathbf{0.078} \pm 0.045$

output is explained by each input feature. Once we have feature importance distributions, we use KL divergence to measure how far the model's learned feature importances are from the ground-truth.

For baselines, we compare against a pure neural network that only predicts observed variables (and cannot infer latent variables), and a blackbox hybrid model [46, 59] where latent parameters are predicted by a neural network. We run ScIReN and Blackbox-Hybrid using a nonlinear constraint (sigmoid or softplus) and a linear constraint (hard-sigmoid or ReLU).

4.2 Ecosystem Respiration

For ecosystem respiration, we used the same dataset and splits as [46], except we removed the 20% highest-temperature examples from the train set, forcing the model to extrapolate to higher temperatures than seen during training. We created two sets of latent R_b (base respiration) values. First, we model R_b as a linear function of 2 features sw_pot, dsw_pot (following [46]):

$$R_b = 0.0075 \cdot sw_pot - 0.00375 \cdot dsw_pot + 1.03506858 \tag{13}$$

Second, to create a setting where 2-layer KAN is needed, we add an absolute value.

$$R'_b = 0.0075 \cdot sw_pot - 0.00375 \cdot dsw_pot; \quad R_b = \left| \frac{R'_b - mean(R'_b)}{stdev(R'_b)} \right| + 0.1 \tag{14}$$

We then generated the observed variable R_{eco} according to the process-based model with multiplicative noise, as in [7]:

$$R_{eco} = R_b \cdot Q_{10}^{\frac{t_a - T_{ref}}{10}} \cdot (1 + \epsilon), \quad \epsilon \sim N(0, 0.1), \text{truncated to } [-0.95, 0.95] \tag{15}$$

Table 1 shows results for the first setting (linear R_b). For predicting the observed variable R_{eco}, Blackbox-Hybrid and ScIReN outperform pure-NN as the process-based model provides prior knowledge that helps the model extrapolate out-of-distribution. For inferring the latent variable R_b and functional relationships, ScIReN with linear constraint does best; it correctly learns that R_b only depends on sw_pot and dsw_pot, not t_a (see Figure 2 left). This is difficult because the irrelevant feature t_a (air temperature) is highly correlated with feature sw_pot. Also, if the model learns the wrong Q_{10} value, it can make R_b depend on t_a to compensate. ScIReN's entropy loss pushes it to eliminate as many variables as possible, and the smoothness loss (with the linear constraint) makes the relationship as linear as possible. Other methods learn complex relationships that perform worse.

Table 2 shows results for nonlinear R_b, where a 2-layer KAN is needed to model the complex relationship (1-layer KAN is insufficient). ScIReN predicts the observed variable, latent variable, and functional relationships almost perfectly, significantly outperforming pure-NN and Blackbox-Hybrid. Qualitatively, figure 2 (right) shows that ScIReN learned the true relationship.

Table 3: Soil carbon cycle (synthetic parameters). Mean and standard deviation across 5 splits/seeds.

Method	R^2 (observed, ↑)	R^2 (latent, ↑)	KL, functional relationships (↓)
Pure NN	0.944 ± 0.026	N/A	N/A
Blackbox-Hybrid, nonlinear constraint	0.997 ± 0.001	0.491 ± 0.155	0.878 ± 0.127
Blackbox-Hybrid, linear constraint	0.997 ± 0.001	0.558 ± 0.142	0.864 ± 0.141
ScIReN, linear constraint (1-layer KAN)	**1.000** ± 0.001	**0.999** ± 0.000	**0.046** ± 0.018

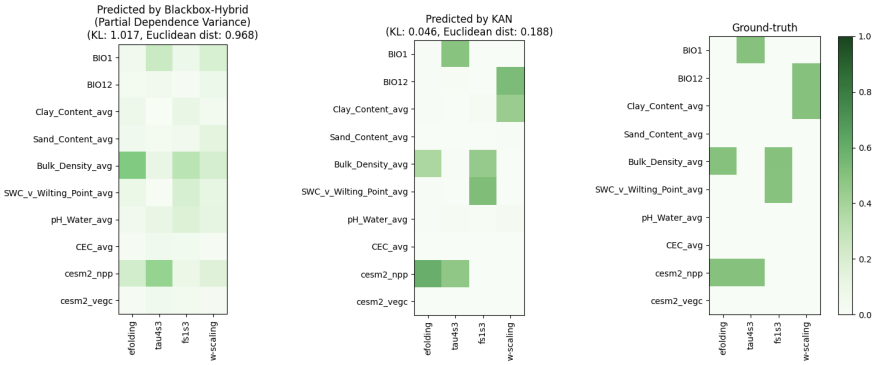

Figure 4: Functional relationships learned by Blackbox-Hybrid (left) and ScIReN (center) vs. truth (right), on synthetic labels. ScIReN recovers the true relationships much more accurately.

Table 4: Soil carbon cycle (real labels). Mean and standard deviation across 5 splits/seeds.

Method	R^2(↑)	MAE (↓)	Pearson correlation (↑)
Pure NN	**0.596** ± 0.127	**4561.2** ± 473.6	**0.790** ± 0.055
Blackbox-Hybrid, nonlinear constraint	0.584 ± 0.082	4726.2 ± 727.3	0.776 ± 0.048
Blackbox-Hybrid, linear constraint	0.589 ± 0.070	4849.7 ± 650.3	0.774 ± 0.040
ScIReN, linear constraint (1-layer KAN)	0.582 ± 0.080	4708.2 ± 673.1	0.769 ± 0.049
ScIReN, linear constraint (2-layer KAN)	0.571 ± 0.094	4707.3 ± 826.3	0.765 ± 0.052

4.3 Soil carbon cycle

For soil carbon, similar to [57] we split the US into 2×2 degree blocks and randomly assign the blocks to five folds. We average across five data splits – each split uses one fold for testing, one fold for validation, and the other folds for training. Each split also uses its own seed for initialization.

First, we synthetically generate functional relationships between the 10 input features and 4 most sensitive biogeochemical parameters from [59]. Of the $10 \times 4 = 40$ possible relationships, we select 20% and randomly assign each to (linear, quadratic, log, exp, abs) with random affine shifts. We set the other parameters to default values as they are poorly constrained by data [59]; this *equifinality* is an inherent limitation of process-based models, which we mitigate by only predicting the 4 most sensitive parameters. We then use the CLM5 process-based model to generate synthetic SOC labels from these parameters. Table 3 shows how various methods perform in recovering these functional relationships. ScIReN (1-layer KAN) recovers the ground-truth relationships (see Figure 4) and observed/latent variables almost perfectly, while Blackbox-Hybrid is not incentivized to produce sparse relationships and mixes correlated features in. Finally, we train all methods on real carbon labels in Table 4. ScIReN's accuracy in predicting SOC amounts is comparable to blackbox hybrid models and pure neural networks, which do not reveal functional relationships. This indicates that we can obtain full interpretability without significantly sacrificing predictive accuracy. Note that a 1-layer KAN is enough to achieve good accuracy, making the encoder even easier to interpret.

5 Conclusion

We have proposed ScIReN, an end-to-end framework that combines data-driven learning with established scientific knowledge (in the form of process-based reasoning) to discover interpretable

relationships between latent scientific parameters and inputs. The entropy loss helps the model discover robust relationships and disregard irrelevant variables, while the smoothness loss enables expressivity without overfitting. A limitation of ScIReN is that the KAN encoder has many hyperparameters (including loss weights) which significantly impact performance; future work could pursue making KANs easier to train. Nevertheless, ScIReN demonstrates excellent performance at retrieving true functional relationships, can extrapolate out-of-distribution, and contains useful biases towards simplicity. Most importantly, ScIReN is fully transparent – it respects existing scientific knowledge while being able to generate novel insights, thus facilitating data-driven scientific discovery.

References

[1] R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton. Neural additive models: Interpretable machine learning with neural nets. *Advances in neural information processing systems*, 34:4699–4711, 2021.

[2] D. W. Apley and J. Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82 (4):1059–1086, 2020.

[3] Y. Bai, D. Chen, and C. P. Gomes. Clr-drnets: Curriculum learning with restarts to solve visual combinatorial games. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, pages 17–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021.

[4] T. Beucler, S. Rasp, M. Pritchard, and P. Gentine. Achieving conservation of energy in neural network emulators for climate modeling. *arXiv preprint arXiv:1906.06622*, 2019.

[5] D. Chen, Y. Bai, W. Zhao, S. Ament, J. Gregoire, and C. Gomes. Deep reasoning networks for unsupervised pattern de-mixing with constraint reasoning. In *International Conference on Machine Learning*, pages 1500–1509. PMLR, 2020.

[6] D. Chen, Y. Bai, S. Ament, W. Zhao, D. Guevarra, L. Zhou, B. Selman, R. B. van Dover, J. M. Gregoire, and C. P. Gomes. Automating crystal-structure phase mapping by combining deep learning with constraint reasoning. *Nature Machine Intelligence*, 3(9):812–822, 2021.

[7] K.-H. Cohrs, G. Varando, N. Carvalhais, M. Reichstein, and G. Camps-Valls. Causal hybrid modeling with double machine learning—applications in carbon flux modeling. *Machine Learning: Science and Technology*, 5(3):035021, 2024.

[8] K. Cuddington, M.-J. Fortin, L. Gerber, A. Hastings, A. Liebhold, M. O'connor, and C. Ray. Process-based models are required to manage ecological systems in a changing world. *Ecosphere*, 4(2):1–12, 2013.

[9] A. Daw, R. Q. Thomas, C. C. Carey, J. S. Read, A. P. Appling, and A. Karpatne. Physics-guided architecture (pga) of neural networks for quantifying uncertainty in lake temperature modeling. In *Proceedings of the 2020 siam international conference on data mining*, pages 532–540. SIAM, 2020.

[10] A. Daw, A. Karpatne, W. D. Watkins, J. S. Read, and V. Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. In *Knowledge guided machine learning*, pages 353–372. Chapman and Hall/CRC, 2022.

[11] W. I. Dieleman, S. Vicca, F. A. Dijkstra, F. Hagedorn, M. J. Hovenden, K. S. Larsen, J. A. Morgan, A. Volder, C. Beier, J. S. Dukes, et al. Simple additive effects are rare: a quantitative review of plant biomass and soil process responses to combined manipulations of co2 and temperature. *Global change biology*, 18(9):2681–2693, 2012.

[12] P. H. Eilers and B. D. Marx. Flexible smoothing with b-splines and penalties. *Statistical science*, 11(2):89–121, 1996.

[13] P. H. Eilers and B. D. Marx. Splines, knots, and penalties. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(6):637–653, 2010.

[14] J. Fan, D. Chen, J. Wen, Y. Sun, and C. Gomes. Monitoring vegetation from space at extremely fine resolutions via coarsely-supervised smooth u-net. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 5066–5072, 2022.

[15] D. Feng, J. Liu, K. Lawson, and C. Shen. Differentiable, learnable, regionalized process-based models with multiphysical outputs can approach state-of-the-art hydrologic prediction accuracy. *Water Resources Research*, 58(10):e2022WR032404, 2022.

[16] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[17] B. M. Greenwell, B. C. Boehmke, and A. J. McCarthy. A simple and effective model-based variable importance measure. *arXiv preprint arXiv:1805.04755*, 2018.

[18] Y. Huang, X. Lu, Z. Shi, D. Lawrence, C. D. Koven, J. Xia, Z. Du, E. Kluzek, and Y. Luo. Matrix approach to land carbon cycle modeling: A case study with the community land model. *Global Change Biology*, 24(3):1394–1404, 2018.

[19] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In *Proceedings of the 2019 SIAM international conference on data mining*, pages 558–566. SIAM, 2019.

[20] X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. Steinbach, and V. Kumar. Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles. *ACM/IMS Transactions on Data Science*, 2(3):1–26, 2021.

[21] E. G. Jobbágy and R. B. Jackson. The vertical distribution of soil organic carbon and its relation to climate and vegetation. *Ecological applications*, 10(2):423–436, 2000.

[22] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[23] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering*, 29(10):2318–2331, 2017.

[24] A. Karpatne, R. Kannan, and V. Kumar. *Knowledge guided machine learning: Accelerating discovery using scientific knowledge and data*. CRC Press, 2022.

[25] A. Karpatne, X. Jia, and V. Kumar. Knowledge-guided machine learning: Current trends and future prospects. *arXiv preprint arXiv:2403.15989*, 2024.

[26] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

[27] R. Lal, W. Negassa, and K. Lorenz. Carbon sequestration in soil. *Current Opinion in Environmental Sustainability*, 15:79–86, 2015.

[28] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[29] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[30] L. Liu, W. Zhou, K. Guan, B. Peng, S. Xu, J. Tang, Q. Zhu, J. Till, X. Jia, C. Jiang, et al. Knowledge-guided machine learning can improve carbon cycle quantification in agroecosystems. *Nature communications*, 15(1):357, 2024.

[31] Z. Liu, P. Ma, Y. Wang, W. Matusik, and M. Tegmark. Kan 2.0: Kolmogorov-arnold networks meet science. *arXiv preprint arXiv:2408.10205*, 2024.

[32] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[33] X. Lu, Y.-P. Wang, Y. Luo, and L. Jiang. Ecosystem carbon transit versus turnover times in response to climate warming and rising atmospheric co2 concentration. *Biogeosciences*, 15(21): 6559–6572, 2018. doi: 10.5194/bg-15-6559-2018. URL `https://bg.copernicus.org/articles/15/6559/2018/`.

[34] X. Lu, Z. Du, Y. Huang, D. Lawrence, E. Kluzek, N. Collier, D. Lombardozzi, N. Sobhani, E. A. Schuur, and Y. Luo. Full implementation of matrix approach to biogeochemistry module of clm5. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002105, 2020.

[35] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[36] Y. Luo and E. A. Schuur. Model parameterization to represent processes at unresolved scales and changing properties of evolving systems. *Global Change Biology*, 26(3):1109–1117, 2020.

[37] Y. Luo and B. Smith. *Land carbon cycle modeling: Matrix approach, data assimilation, & ecological forecasting*. CRC Press, 2022.

[38] Y. Luo, A. Ahlström, S. D. Allison, N. H. Batjes, V. Brovkin, N. Carvalhais, A. Chappell, P. Ciais, E. A. Davidson, A. Finzi, et al. Toward more realistic projections of soil carbon dynamics by earth system models. *Global Biogeochemical Cycles*, 30(1):40–56, 2016.

[39] Y. Luo, Y. Huang, C. A. Sierra, J. Xia, A. Ahlström, Y. Chen, O. Hararuk, E. Hou, L. Jiang, C. Liao, et al. Matrix approach to land carbon cycle modeling. *Journal of Advances in Modeling Earth Systems*, 14(7):e2022MS003008, 2022.

[40] R. Marcinkevičs and J. E. Vogt. Interpretable and explainable machine learning: A methods-centric overview with concrete examples. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(3):e1493, 2023.

[41] Y. Min, M.-C. Chang, S. Kong, J. M. Gregoire, R. B. van Dover, M. O. Thompson, and C. P. Gomes. Physically informed graph-based deep reasoning net for efficient combinatorial phase mapping. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 392–399. IEEE, 2023.

[42] C. Molnar. *Interpretable Machine Learning*. 3 edition, 2025. ISBN 978-3-911578-03-5. URL `https://christophm.github.io/interpretable-ml-book`.

[43] G. Novakovsky, N. Dexter, M. W. Libbrecht, W. W. Wasserman, and S. Mostafavi. Obtaining genetics insights from deep learning via explainable artificial intelligence. *Nature Reviews Genetics*, 24(2):125–137, 2023.

[44] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[45] P. Rathore, W. Lei, Z. Frangella, L. Lu, and M. Udell. Challenges in training pinns: A loss landscape perspective. *arXiv preprint arXiv:2402.01868*, 2024.

[46] M. Reichstein, B. Ahrens, B. Kraft, G. Camps-Valls, N. Carvalhais, F. Gans, P. Gentine, and A. J. Winkler. Combining system modeling and machine learning into hybrid ecosystem modeling. In *Knowledge guided machine learning*, pages 327–352. Chapman and Hall/CRC, 2022.

[47] M. T. Ribeiro, S. Singh, and C. Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[48] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, et al. Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, 55(2):1–96, 2022.

[49] C. 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.

[50] C. Shen, A. P. Appling, P. Gentine, T. Bandai, H. Gupta, A. Tartakovsky, M. Baity-Jesi, F. Fenicia, D. Kifer, L. Li, et al. Differentiable modelling to unify machine learning and physical models for geosciences. *Nature Reviews Earth & Environment*, 4(8):552–567, 2023.

[51] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[52] T. Tan, G. Genova, G. B. Heuvelink, J. Lehmann, L. Poggio, D. Woolf, and F. You. Importance of terrain and climate for predicting soil organic carbon is highly variable across local to continental scales. *Environmental science & technology*, 58(26):11492–11503, 2024.

[53] F. Tao, Z. Zhou, Y. Huang, Q. Li, X. Lu, S. Ma, X. Huang, Y. Liang, G. Hugelius, L. Jiang, et al. Deep learning optimizes data-driven representation of soil organic carbon in earth system model over the conterminous united states. *Frontiers in Big Data*, 3:17, 2020.

[54] F. Tao, Y. Huang, B. A. Hungate, S. Manzoni, S. D. Frey, M. W. I. Schmidt, M. Reichstein, N. Carvalhais, P. Ciais, L. Jiang, et al. Microbial carbon use efficiency promotes global soil carbon storage. *Nature (London)*, 618(7967), 05 2023. ISSN ISSN 0028-0836. doi: 10.1038/s41586-023-06042-3. URL `https://www.osti.gov/biblio/2212450`.

[55] W.-P. Tsai, D. Feng, M. Pan, H. Beck, K. Lawson, Y. Yang, J. Liu, and C. Shen. From calibration to parameter learning: Harnessing the scaling effects of big data in geoscientific modeling. *Nature communications*, 12(1):5988, 2021.

[56] R. van Bree, D. Marcos, and I. Athanasiadis. Hybrid phenology modeling for predicting temperature effects on tree dormancy. *arXiv preprint arXiv:2501.16848*, 2025.

[57] S. Wang, W. Chen, S. M. Xie, G. Azzari, and D. B. Lobell. Weakly supervised deep learning for segmentation of remote sensing imagery. *Remote Sensing*, 12(2):207, 2020.

[58] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Computing Surveys*, 55(4): 1–37, 2022.

[59] H. Xu, J. Fan, F. Tao, L. Jiang, F. You, B. Z. Houlton, Y. Sun, C. P. Gomes, and Y. Luo. Biogeochemistry-informed neural network (binn) for improving accuracy of model prediction and scientific understanding of soil organic carbon. *arXiv preprint arXiv:2502.00672*, 2025.

Acknowledgements

This research is supported by AI-LEAF: "AI Institute for Land, Economy, Agriculture, and Forestry", funded by the USDA National Institute of Food and Agriculture (NIFA) and the NSF National AI Research Institutes Competitive Award. This research is also partially supported by Schmidt Sciences programs, an AI2050 Senior Fellowship and an Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship; the National Science Foundation; the US Department of Energy; the CALS Moonshot Seed Grant program; the "NYS Connects: Climate Smart Farms & Forestry" project, funded by the USDA, the New York State Department of Environmental Conservation, and the New York State Department of Agriculture and Markets; an NSF Research Traineeship (NRT) fellowship in Digital Plant Science; and the Air Force Office of Scientific Research.

A Technical Appendices

A.1 Ecosystem respiration: data and process-based model

For the ecosystem respiration experiments, we used the process-based model and dataset from [46]. We have labels for output variable R_{eco} (ecosystem respiration). The process-based model specifies that R_{eco} is a differentiable function of two latent parameters, base respiration R_b and temperature sensitivity Q_{10}. Specifically:

$$R_{eco} = g_{PBM}(\mathbf{p}, \mathbf{x}) = R_b(\mathbf{x}) \cdot Q_{10}^{\frac{t_a - T_{ref}}{10}} \tag{16}$$

where the latent parameters are $\mathbf{p} = \{R_b, Q_{10}\}$, the input features are $\mathbf{x} = \{sw_pot, dsw_pot, t_a\}$ (potential shortwave radiation, derivative of potential shortwave radiation, air temperature), and $T_{ref} = 15$ is a constant. In this example, R_b is a function of input features, while Q_{10} is assumed to be independent of the input features and is a global learnable constant.

Note that this model is somewhat ill-posed, as the input feature temperature (t_a) can influence R_{eco} either through changing the base respiration R_b, or through the Q_{10} exponential term. For example, as explained in [7], if the model predicted the wrong value of Q_{10}, it can adjust its predicted R_b to compensate and still match the labels for R_{eco}, as follows:

$$R_{eco} = \underbrace{R_b^{TRUE}(\mathbf{x}) \cdot \left(\frac{Q_{10}^{TRUE}}{Q_{10}^{PRED}}\right)^{\frac{t_a - T_{ref}}{10}}}_{\text{Predicted } R_b} \cdot \left(Q_{10}^{PRED}\right)^{\frac{t_a - T_{ref}}{10}} \tag{17}$$

where the predicted R_b function soaked up additional temperature dependencies that should have been included in the Q_{10} exponential. Thus the task is difficult, but ScIReN's bias towards linear and sparse functions can allow it to recover the true R_b.

The dataset and splits are the same as used in [46, 7], except that we remove datapoints with top-20% t_a (temperature) values from the training set (and not the validation/test sets), to test the model's ability to generalize out-of-distribution. This results in 56102 training, 17520 validation, and 17568 test datapoints. The input features are taken from values observed by a EC tower in Neustift, Austria. The Q_{10} parameter (latent) was set to 1.5. We created two sets of latent R_b (base respiration) values:

1. First, we model R_b as a linear function of 2 features sw_pot, dsw_pot (following [46]):

$$R_b = 0.0075 \cdot sw_pot - 0.00375 \cdot dsw_pot + 1.03506858 \tag{18}$$

   where the constant 1.03506858 was selected to make all values non-negative.

2. Second, to create a setting where 2-layer KAN is needed, we add an absolute value.

$$R_b' = 0.0075 \cdot sw_pot - 0.00375 \cdot dsw_pot; \quad R_b = \left|\frac{R_b' - mean(R_b')}{stdev(R_b')}\right| + 0.1 \tag{19}$$

We then generated the observed variable R_{eco} according to the process-based model with multiplicative noise, as in [7]:

$$R_{eco} = R_b \cdot Q_{10}^{\frac{t_a - T_{ref}}{10}} \cdot (1 + \epsilon), \quad \epsilon \sim N(0, 0.1), \text{truncated to } [-0.95, 0.95] \tag{20}$$

A.2 Ecosystem respiration: hyperparameters

We initialized Q_{10} to 0.5 for all models (so it should increase during training), and give Q_{10} a learning rate that is 100 times as high as the other learnable parameters in the model. We fixed λ_{param} to 1 if using the ReLU constraint. For Pure-NN and Blackbox-Hybrid, we used an MLP with 2 layers and 16 hidden units. For ScIReN, we used KANs with cubic splines, 30 grid points, and a margin of 2 times the input range.

For each method, we tuned hyperparameters using a grid search, selecting the hyperparameters that had the lowest validation mean-squared-error loss. For Pure-NN and Blackbox-Hybrid, we considered learning rates from $[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$, and weight decay from $[0, 10^{-4}, 10^{-3}]$. For the KAN-based methods, we considered learning rates from $[10^{-3}, 10^{-2}, 10^{-1}]$, weight decay from $[0, 10^{-4}]$, and considered $\lambda_{entropy}, \lambda_{L1}, \lambda_{smooth}$ from $[10^{-3}, 10^{-2}, 10^{-1}, 1]$. (We generally set $\lambda_{entropy} = \lambda_{L1}$. Due to the number of hyperparameters in KAN, we did not try all configurations.)

The hyperparameters selected for Table 1 (linear R_b) are shown below in Table 5. We also report approximate runtimes per run on a single V100 GPU.

Table 5: Hyperparameters for Table 1.

Method	LR	Weight decay	$\lambda_{entropy}, \lambda_{L1}$	λ_{smooth}	Runtime
Pure-NN	0.1	0	-	-	15min
Blackbox-Hybrid, nonlinear constraint	0.1	0	-	-	24min
Blackbox-Hybrid, linear constraint	0.1	0	-	-	24min
ScIReN, nonlinear constraint (1-layer KAN)	0.1	10^{-4}	0.01	0.1	15min
ScIReN, linear constraint (1-layer KAN)	0.01	10^{-4}	0.01	1.0	15min

The hyperparameters selected for Table 2 (nonlinear R_b) are shown below in Table 6, along witih approximate runtimes per run on a single V100 GPU. Note that for 2-layer KAN, we used 8 hidden units, and we used a ‘zero‘ base function (instead of identity).

Table 6: Hyperparameters for Table 2.

Method	LR	Weight decay	$\lambda_{entropy}, \lambda_{L1}$	λ_{smooth}	Runtime
Pure-NN	0.1	0	-	-	25min
Blackbox-Hybrid, nonlinear constraint	10^{-3}	10^{-3}	-	-	25min
Blackbox-Hybrid, linear constraint	10^{-3}	10^{-4}	-	-	25min
ScIReN, linear constraint (1-layer KAN)	0.1	10^{-4}	0.01	0.1	20min
ScIReN, linear constraint (2-layer KAN)	0.01	10^{-4}	10^{-3}	1.0	38min

A.3 Ecosystem respiration: Additional figures

For Linear R_b we present additional figures demonstrating the qualitative performance of various methods (Figures 5, 6, 7, 8).

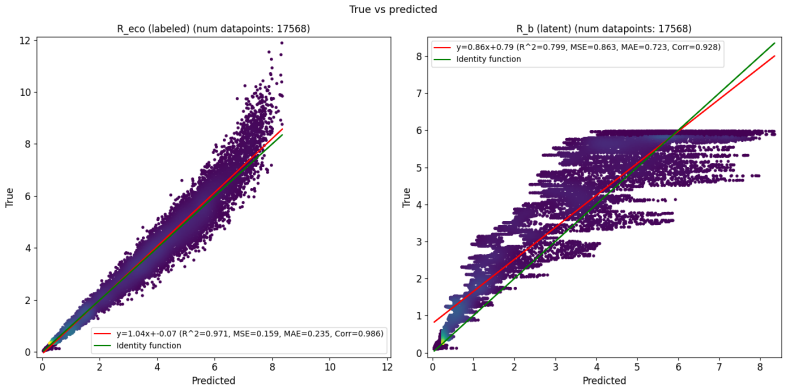

Figure 5: Pure NN. Because the model does not have access to scientific knowledge (process-based model), it struggles to extrapolate on the higher end of ecosystem respiration / temperature (left). It is also unable to predict the latent variable (right).

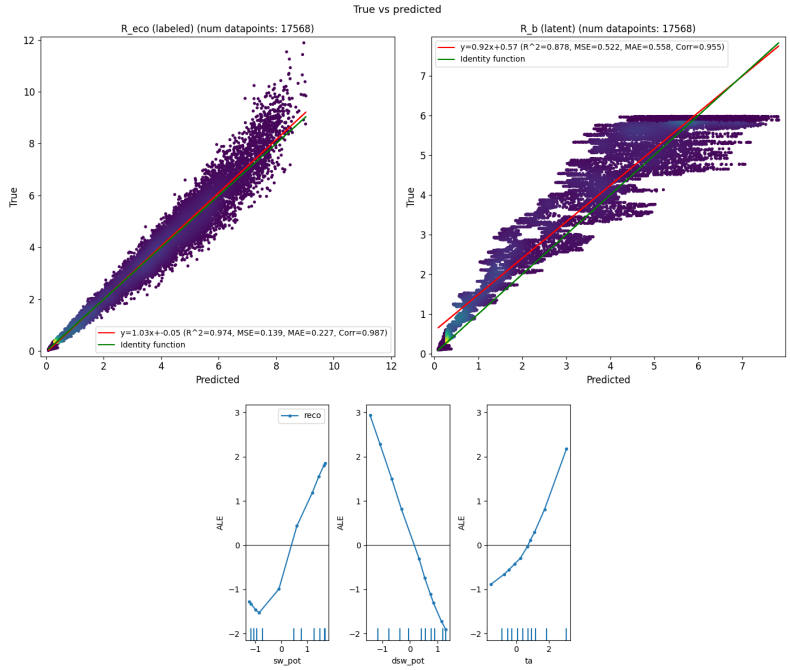

Figure 6: Blackbox-Hybrid, nonlinear constraint. The process-based model helps with extrapolation (top left). However the latent variable prediction is inaccurate (top right). According to the Accumulated Local Effects plots (which estimate the impact of each input feature; bottom), the model thinks that both sw_pot and t_a influence the latent variable, but only sw_pot is actually relevant.

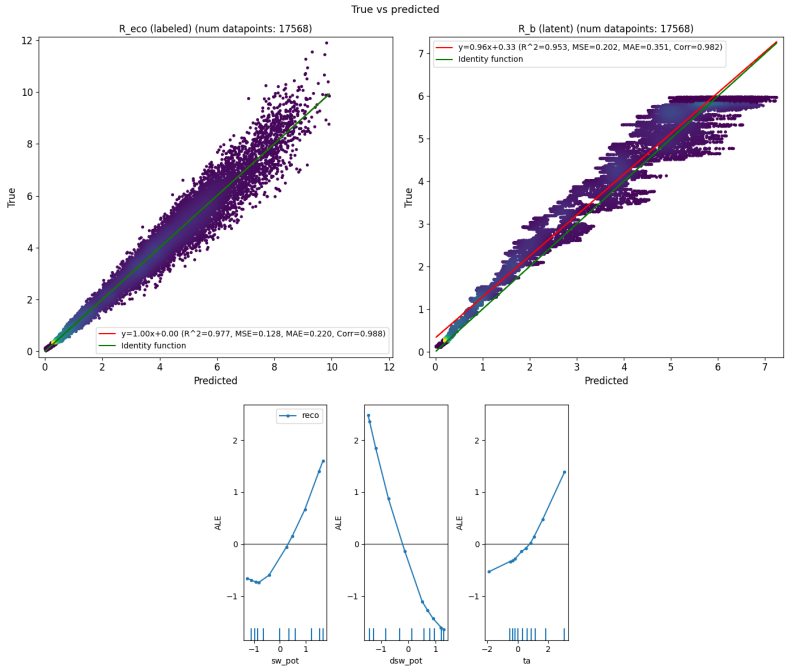

Figure 7: Blackbox-Hybrid, linear constraint. Changing to linear constraint improves the latent variable prediction (top right) since the model now only needs to learn a simple linear relationship between R_b and input features. However, the model is still using t_a somewhat even though it is actually useless (bottom).

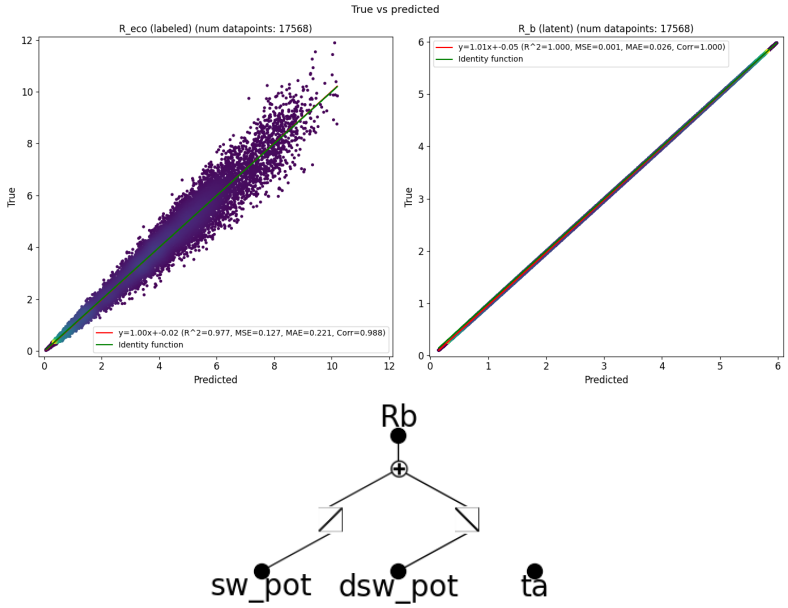

Figure 8: ScIReN, linear constraint. ScIReN's entropy loss encourages sparsity, forcing it to choose the best among correlated features. The smoothness loss also encourages a near-linear function. Thus, it is able to predict the observed variable (top left), latent variable (top right), and functional relationships (bottom) correctly.

A.4 Soil carbon cycle: CLM5 process-based model and datasets

The textbook by Luo and Smith [37] provides a detailed introduction to land carbon cycle models, but we summarize some key details here. As stated in the main text, we can write the change in carbon for each pool as the difference between inputs and outputs:

$$\frac{dY_i(t)}{dt} = \text{inflow to pool } i - \text{outflow from pool } i \tag{21}$$

Since there are 140 equations (one for each pool), we can stack them into a single matrix equation as follows:

$$\frac{dY(t)}{dt} = B(\mathbf{p}, t)I(t) - A(\mathbf{p})\ \text{diag}(\xi(t) \odot K(\mathbf{p}))\ Y(t) - V(\mathbf{p}, t)Y(t) \tag{22}$$

Here, $Y(t) \in \mathbb{R}^{140}$ is the amount of carbon in each of the 140 pools at time t. $I(t) \in \mathbb{R}$ is the total carbon input to the system from vegetation at time t, and $B(t) \in \mathbb{R}^{140}$ is the allocation of input to different pools. $A(\mathbf{p}, t) \in \mathbb{R}^{140\times140}$ quantifies carbon transfers between pools at the same depth. $K(\mathbf{p}) \in \mathbb{R}^{140}$ contains the intrinsic decomposition rate of each carbon pool, which is the same for each pool across 20 layers. $\xi(t) \in \mathbb{R}^{140}$ is computed based on monthly climate data and indicates how the environment modifies the intrinsic decomposition rate. ξ and K are multiplied elementwise and then converted into a 140×140 diagonal matrix. $V(\mathbf{p}, t) \in \mathbb{R}^{140\times140}$ quantifies vertical carbon transfers between adjacent depths. $\odot$ denotes elementwise multiplication, while diag indicates we construct a diagonal matrix from the given vector. The t indicates that the parameter varies with time, while $\mathbf{p}$ indicates that the matrix is constructed in a deterministic way from latent biogeochemical parameters predicted by the encoder. In this study we do not consider temporal variations, but any parameters that depend on t are constructed using *monthly-average* climate forcing data.

In this work, we assume steady state ($\frac{dY}{dt} = 0$), which is a reasonable assumption since previous research has shown that recent disequilibrium effects from climate change and human activities are minor in comparison to the total SOC storage that has developed over thousands of years [54, 33]. Thus, we can solve for Y analytically as a function of $\mathbf{p}$:

$$\hat{Y}(t) = \left[A(\mathbf{p})\ \text{diag}(\xi(t) \odot K(\mathbf{p})) + V(\mathbf{p}, t)\right]^{-1} B(\mathbf{p}, t)I(t)$$

Note that $\hat{Y}(t)$ contains predicted carbon amounts at 20 fixed layers, and 7 pools per layer. However, our labels are only aggregate SOC amounts at specific depths (which may not match the 20 fixed layers). Thus, we sum up the SOC pools at each layer, and linearly interpolate to predict SOC at the observed depths.

The datasets used in these experiments are documented in [53, 59]. We restricted to ten input features for interpretability:

1. Annual mean temperature
2. Annual precipitation
3. Clay content
4. Sand content
5. Bulk density
6. Soil water capacity
7. Soil pH in H2O
8. Cation Exchange Capacity
9. NPP (Net Primary Productivity)
10. Vegetation Carbon Stock

To test ScIReN's ability on small datasets, we only use 1018 examples in our dataset total, which are representative locations in the conterminous United States. We split the dataset spatially into 2×2-degree blocks, and assign each block to one of 5 folds. We run experiments across 5 splits, each time holding out a different test fold (and validation fold).

For the synthetic functional relationship experiment, we only considered the four most sensitive biogeochemical parameters identified in [59], as the other parameters are poorly constrained by data.

We randomly selected 20% of the possible input-output relationships to exist. For each relationship, we randomly draw a function (linear, quadratic, exponential, logarithmic, absolute value), and add random affine shifts. We then calculate the value of each output parameter as the sum of functions of each relevant input, e.g.

$$p_i = f_{i,2}(x_2) + f_{i,6}(x_6) = \log(ax_2 + b) + (cx_6^2 + d) \tag{23}$$

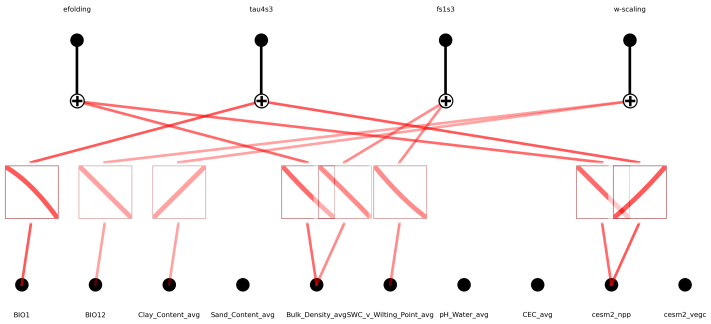

Figure 9: Prescribed functional relationships.

An example of the true prescribed functional relationships is shown in Figure 9.

A.5 Soil carbon cycle: Hyperparameters

For soil carbon cycle: for neural networks, we used a 3-layer MLP with 128 hidden units, residual connection, batch normalization, and leaky ReLU activation. We set the temperature $\tau = 1$. For KANs, we used cubic splines, 30 grid points, and a margin of 2 times the input range; we also used 128 hidden units for the 2-layer KAN (probably too many). We set $\lambda_{param} = 1000$ if using hardsigmoid (linear constraint).

We tuned the learning rate from $[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$, weight decay from $[0, 10^{-4}, 10^{-3}]$, $\lambda_{entropy}, \lambda_{L1}$ from $[0.1, 1, 10]$, and λ_{smooth} from $[10, 100, 1000]$. The weights are higher than for ecosystem respiration because the magnitude of the supervised loss is 100-1000 times larger.

Table 7 contains hyperparameters used for Table 3. These were trained using a single CPU.

Table 7: Hyperparameters for Table 3.

Method	LR	Weight decay	$\lambda_{entropy}, \lambda_{L1}$	λ_{smooth}	Runtime
Pure-NN	10^{-3}	0	-	-	15min
Blackbox-Hybrid, nonlinear constraint	10^{-4}	10^{-4}	-	-	180min
Blackbox-Hybrid, linear constraint	10^{-3}	10^{-4}	-	-	180min
ScIReN, linear constraint (1-layer KAN)	10^{-2}	0	10	100	190min

Table 8 contains hyperparameters used for Table 4. These were trained using PyTorch Distributed-DataParallel on 8 CPUs.

Table 8: Hyperparameters for Table 4.

Method	LR	Weight decay	$\lambda_{entropy}, \lambda_{L1}$	λ_{smooth}	Runtime
Pure-NN	0.1	0	-	-	2min
Blackbox-Hybrid, nonlinear constraint	10^{-3}	0	-	-	25min
Blackbox-Hybrid, linear constraint	10^{-2}	10^{-4}	-	-	25min
ScIReN, linear constraint (1-layer KAN)	10^{-2}	0	1	100	40min
ScIReN, linear constraint (2-layer KAN)	10^{-2}	0	1	100	80min

A.6 Computational Resources

For ecosystem respiration, experiments were run on a cluster, using a single NVIDIA V100 GPU. For CLM5, experiments were run on a cluster using 1 or 8 CPUs (no GPUs). (For the synthetic experiment in Table 3, using 1 CPU performed better than 8 CPUs, possibly because distributed training with 8 CPUs made it hard for each worker's gradient updates to agree on which conections to choose.) For the real label experiments used 8 CPUs with distributed training. Some runs were also performed on a laptop with M3 chip.

We did not log the total amount of compute required for all failed experiments. The total amount of cluster usage during this period was about $17,000$ CPU-hours; this included a small amount of time spent on other projects, but excluded runs on a local laptop, which may be roughly equivalent.

A.7 Ablations

To justify the components of ScIReN, in Table 9 we show the results of removing each loss term, on Table 2 (ecosystem respiration, nonlinear R_b, 2-layer KAN)

Table 9: Ecosystem respiration, nonlinear R_b. Ablations.

Method	R^2 (observed, $\uparrow$)	R^2 (latent, $\uparrow$)	KL, functional relationships ($\downarrow$)
Original	**0.9619**	0.9989	**0.1490**
Remove smoothness loss	0.9600	0.9720	0.4973
Remove L1 (absolute deviation) loss	0.9618	**0.9993**	0.2016
Remove entropy loss	0.9608	0.8754	0.6132
Reduce to 3 grid cells	0.9558	0.9863	0.8942

Most methods do similarly on the observed variable. On the latent variable, removing L1 (absolute deviation) loss actually did slightly better; the L1 loss and entropy loss are somewhat interchangable as they both aim to shrink the variance explained by most features to zero. In predicting functional relationships, all losses seem quite valuable. Removing the entropy loss caused a large drop in performance, as the model would pay too much attention to correlated features instead of the true causal one. Finally, reducing to 3 grid cells makes it impossible for the network to learn sharp transitions like the absolute value function.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are well-supported by experimental evidence in the paper. Experiments show that ScIReN is able to correctly recover interpretable relationships between input features and latent processes in synthetic datasets (Section 4.2, 4.3), and embedding the process-based model in the model helps extrapolation in ecosystem respiration (Section 4.2).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We point out several limitations of our method. ScIReN has a large number of loss weights and KAN hyperparameters to tune (Section 3.5), and performance is quite sensitive to them (Section 5). Additionally, process-based models may suffer from equifinality (data is not always enough to constrain latent parameters) (Section 4.3).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results. The only theoretical claim is that stacking multiple layers of neural additive models yields a Kolmogorov-Arnold Network, which is derived in detail in the text (Section 3.1).

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All code and data needed to reproduce the experiments are provided in the supplementary material and will be posted on GitHub if the paper is accepted. The README contains exact commands to reproduce the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All code and data is provided in the supplementary material, along with a README that explains exactly what commands to run to reproduce the results.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Full hyperparameter lists are provided in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Standard deviations across 5 seeds are provided for each method. For CLM5 (soil carbon cycle), each seed also uses a different train/validation/test split.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, computational details are specified in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research did not involve human subjects, all datasets were already publicly-available, and we do not forsee any negative societal consequences of the work.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: The paper discusses how the method can accelerate scientific discovery by revealing new previously-unknown relationships between latent processes and observed variables. As mentioned in the introduction, the soil carbon cycle application is highly relevant to sustainability since understanding carbon flow through soil is key to designing approaches to mitigate climate change. In Section 4.3 we mention the issue of equifinality in the process-based model, which could cause the model to draw misleading scientific conclusions if not mitigated.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not forsee major risk of misuse of our approach. While accelerated scientific discovery could conceivably make it faster to develop harmful technologies (such as bioweapons), this is a very remote possibility as our work simply discovers relationships that can be inferred from existing datasets and process-based models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Our work draws from several previous publications, public GitHub repositories, and datasets, which are cited in the paper. The licenses and GitHub repositories will be mentioned in the README of the code.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The codebase of the paper is well-documented and explained in the README.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This research does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This research does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: This work did not use LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.