SLICE: BUILDING MODELS FROM STRUCTURED SINGLE-FEATURE PERTURBATION EFFECTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Single-feature perturbation analysis is a fundamental method for investigating how a predictive model behaves when a single feature is varied while the remaining features are held fixed. Existing visualization approaches analyze these effects after a model has been fitted, typically by aggregating responses across dataset samples or by visualizing individual response curves. However, aggregation can obscure heterogeneous responses across samples, while the individual curves may be numerous, complex, and heavily interleaving, making the overall perturbation behavior difficult to characterize globally. Moreover, these approaches provide no structural guarantee on behavior under other unseen feature values. We reverse this workflow: rather than first fitting a model and subsequently attempting to understand its perturbation effects, we explicitly constrain the form these effects may take and build the model around those constraints. We introduce SLICE (Structured Laminar Individual Conditional Effects), a class of models whose centered single-feature perturbation curves form structured, laminar families. In such families, perturbation curves form an ordered family rather than crossing and interleaving arbitrarily, making the full range of possible single-feature responses easy to characterize globally while still allowing the underlying predictor to capture interactions of arbitrary order. We characterize the prediction functions compatible with these perturbation requirements and develop a differentiable parameterization that satisfies them by construction. Across benchmark datasets, SLICE outperforms additive models and often approaches or matches fully flexible baselines, while preserving globally comprehensible single-feature perturbation behavior. SLICE thus shifts single-feature perturbation analysis from merely post-training inspection to explicit control during model construction.

1 INTRODUCTION

Global understanding of feature-response curves. After training a machine learning model, we often want to analyze it to better understand how it works, to validate it, or to improve its performance. A basic way to analyze a predictive model is to vary one feature while holding the remaining features fixed and observe how the prediction changes. This produces a *feature-response* curve for a particular *context*, defined by the values of the remaining features. Such a curve provides a local view of the model. However, for global model understanding or verification, we might want to understand the effect of perturbing the selected feature for all possible contexts. For instance, we might observe that a prediction of a risk model increases when the patient's pressure is increasing. This raises additional questions. Is it always increasing for all patients? Is the rate of increase always the same? What is the maximum increase it can show when blood pressure rises from 120 to 150? All of those questions require a global understanding of the possible feature-response curves.

Existing approaches. Existing post-hoc approaches address this problem in two main ways. Individual Conditional Expectation (ICE) plots visualize feature-response curves for multiple observations from the dataset (Goldstein et al., 2015). This depicts heterogeneity in feature effects, but introduces a trade-off between coverage and legibility: plotting more contexts provides a more complete empirical picture while making the visualization increasingly crowded, especially when curves cross and interleave. Moreover, even a large ICE plot only describes the sampled contexts and cannot in general characterize the model over all admissible inputs. Centered ICE (c-ICE) and Derivative

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

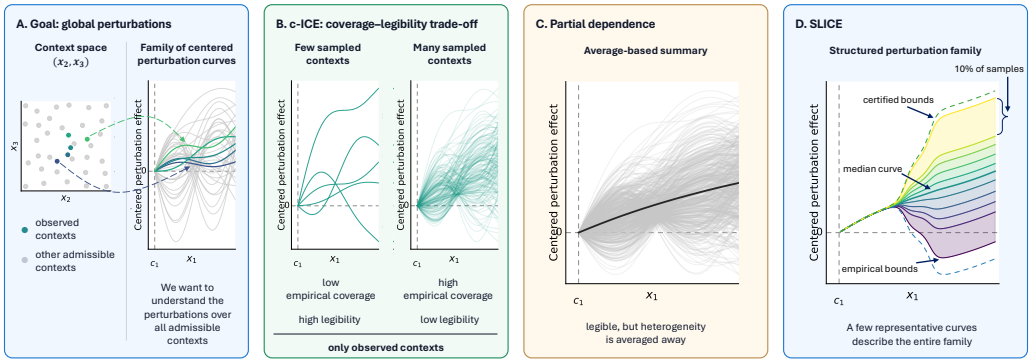

Figure 1: **A.** Given a trained model, we are interested in understanding the effect of perturbing a single feature while other features are kept fixed. We want to understand this effect not only for the observed samples, but for all admissible inputs. **B.** A centered ICE plot shows individual centered perturbation curves for samples from the dataset. It introduces a trade-off between coverage and legibility. In practice, it cannot show all potential inputs. **C.** It is possible to average the perturbation curves to obtain a partial dependence (PD) plot. Although legible, the heterogeneity of perturbation effects is averaged out. **D.** We propose to fit the perturbation effects directly instead of computing them post-hoc. We constrain the family of perturbation effects to be well-behaved, so it is easy to visualize all possible effects for both observed samples and all admissible inputs.

ICE (d-ICE) can make differences between curves easier to compare, but do not remove this fundamental limitation. A complementary approach is to aggregate some sets of feature-response curves. For instance, partial dependence (PD) plots (Friedman, 2001) average feature-response curves across all contexts in the dataset. Other related techniques modify the way the aggregation is performed or construct several regional summaries for different subsets of the feature space. These are very legible (one or few curves per feature) but they inevitably hide some of the heterogeneity of feature effects. As such, they do not allow us to answer the questions about the blood pressure we posed earlier.

Reversing the workflow. These issues stem from the fact that feature-response curves are evaluated post-hoc, after the model has been trained. At that point, we have no control over the complexity and heterogeneity of feature effects. A truly global understanding would require querying the model for all possible inputs and hoping the curves are simple enough to be summarized in some compact way. In this work we reverse this workflow. Rather than first fitting a model and subsequently attempting to explain its perturbation effects, we directly constrain the form that these effects may take and construct the predictive model around those constraints. Since we have direct control over those effects, we can describe them in a way that is legible and complete. We focus on centered feature-response curves. For a feature x_i, we subtract the prediction obtained at a fixed reference value (c_i) while keeping the remaining features unchanged. We refer to the resulting functions as *perturbation curves*. Centering removes context-dependent vertical offsets while preserving how the prediction changes when the feature is perturbed. These are precisely the curves visualized by centered ICE.

Laminar family of perturbation curves. Then the question is how we can make sure that a whole family of perturbation curves is easy to understand. The simplest way would be to require all contexts to produce the same curve. However, this eliminates context-dependent feature effects and leads to an additive model with no feature interactions, and thus limited expressivity. Instead, we introduce the notion of a *laminar* family of perturbation curves. This is inspired by the notion of laminar flow, where the fluid particles follow smooth paths in layers with little or no mixing. Similarly, laminar perturbation curves form an ordered family without arbitrary crossings and interleaving. Moreover they are bounded by two feature-specific boundary curves. This structure makes it possible to characterize the full range of possible perturbation effects using a small number of representative curves together with global bounds. Importantly, these guarantees apply to all admissible contexts, not only to observations contained in the training data.

SLICE. We call models whose single-feature perturbation families are laminar SLICE (Structured Laminar Individual Conditional Effects). We further introduce Affine-SLICE (A-SLICE), a tractable subclass in which the perturbation curves interpolate linearly between the boundaries. We derive the structural form of every A-SLICE model and show that, despite its constrained perturbation behavior, it can contain interactions of arbitrary order. We then develop differentiable parameterizations based on either neural networks or splines, allowing the model to be trained with standard gradient-based optimization. Fig. 1 illustrates the central distinction. Conventional c-ICE plot preserves heterogeneous effects but provides only empirical coverage and becomes increasingly difficult to read as more contexts are shown. Aggregated methods such as a PD plot provide a compact summary but averages away heterogeneity. SLICE instead constrains the perturbation family itself: a small number of representative curves describes the empirical distribution of effects, while model-implied bounds characterize the behavior over all admissible contexts. See Appendix D for a demonstration of the difference on a real dataset.

Contributions. Our main contributions are as follows. First, we formulate global single-feature perturbation structure as a property that can be imposed during model construction rather than analyzed only post hoc, and introduce laminar perturbation families and the corresponding SLICE model class. Second, we characterize Affine-SLICE models and derive a differentiable parameterization that supports interactions of arbitrary order while preserving structured perturbation behavior. Third, we show how the resulting models can be visualized using representative empirical curves together with global perturbation bounds. Finally, across benchmark datasets, A-SLICE consistently improves over additive models and often approaches or matches fully flexible black-box baselines, while retaining globally comprehensible single-feature perturbation behavior.

2 Related works

Post-hoc techniques for analysis of perturbation effects Most methods for analyzing single-feature perturbation effects operate post-hoc on an already trained model. Individual Conditional Expectation (ICE) plots visualize feature-response curves across multiple contexts, typically one for each observation in a dataset (Goldstein et al., 2015). By retaining individual curves, ICE plot exposes heterogeneity in feature effects. However, this comes at the cost of visual complexity: as more contexts are shown, the plot becomes increasingly crowded, particularly when curves cross and interleave. Moreover, ICE characterizes the model only at the selected contexts rather than over all admissible inputs. Centered ICE (c-ICE) and derivative ICE (d-ICE) can improve comparison between curves by removing context-dependent offsets or emphasizing local slopes, respectively, but they do not remove these basic limitations. A different class of approaches obtains a compact global representation by aggregating feature effects across contexts. Partial dependence (PD) plots (Friedman, 2001) average model predictions over the empirical distribution of the remaining features. Marginal plots (Molnar, 2020) instead average conditionally on the feature of interest, while Accumulated Local Effects (ALE) plots (Apley & Zhu, 2020) accumulate local conditional changes in the prediction. As these methods summarize context-dependent effects through an aggregated representation, heterogeneous feature effects may be obscured. Rather than collapsing all contexts into a single global curve, regional feature-effect methods seek an intermediate representation that retains some heterogeneity while remaining interpretable. VINE (Britton, 2019) clusters observations whose ICE curves have similar slopes and constructs shallow decision-tree explanations for cluster membership. REPID (Herbinger et al., 2022) directly partitions the feature space to obtain interpretable regions within which ICE curves are more homogeneous. GADGET (Herbinger et al., 2024) generalizes this principle by searching for interpretable regions where the interaction-related heterogeneity in local feature effects is minimized. Such approaches retain more heterogeneity than a single global average while remaining more compact than a full ICE plot. Rather than fitting an unrestricted predictor and subsequently summarizing its perturbation behavior, SLICE constrains the family of perturbation curves during model construction. This ensures the perturbation curves remain understandable, depict heterogeneity of effects and cover all admissible contexts rather than only those observed in the data.

Models with simple feature effects An alternative to post-hoc analysis is to construct models whose feature effects are simple by design. In a linear model, $f(x) = a_0 + \sum_{i=1}^{n} a_i x_i$, the effect of perturbing a single feature is completely characterized by its coefficient: changing x_i by an amount

Table 1: Comparison of approaches for understanding single-feature perturbation effects. ✓ indicates that a property is fully satisfied, ◐ indicates that it is partially satisfied, and ✗ indicates that it is not satisfied.

Approach	Type	Perturbation representation	Effect heterogeneity	All admissible contexts?	Compact representation
PD	Post-hoc	Average perturbation curve	✗ Averaged out	✗	✓ One curve
M-plot	Post-hoc	Conditional average effect	✗ Averaged out	✗	✓ One curve
ALE	Post-hoc	Accumulated local effect	✗ Averaged out	✗	✓ One curve
ICE / c-ICE	Post-hoc	Individual perturbation curves	✓ Retained	✗	✗ Many curves
Regional effects	Post-hoc	Regional effect curves	◐ Region-level	✗	◐ Several curves
Linear regression	Intrinsic	Coefficient of a linear model	✗ Absent	✓	✓ One coefficient
GAM	Intrinsic	Univariate shape function	✗ Absent	✓	✓ One curve
SLICE	Intrinsic	Structured perturbation family	✓ Retained	✓	✓ Laminar family

λ changes the prediction by λa_i, independently of the remaining features. Generalized additive models (GAMs) (Hastie & Tibshirani, 1986) extend this idea by allowing each feature to have an arbitrary univariate effect while retaining an additive structure, $f(x) = a_0 + \sum_{i=1}^{n} g_i(x_i)$. The effect of perturbing feature x_i is entirely characterized by g_i which does not depend on the context. Equivalently, all perturbation curves for a given feature coincide. Different GAM variants primarily differ in how the component functions g_i are parameterized and estimated. Classical GAMs typically represent the component functions g_i using smooth basis expansions, commonly splines (Servén & Brummitt, 2018). More recent approaches retain the additive structure while using more flexible learning mechanisms for the univariate component functions, including boosted trees (Lou et al., 2012) and neural networks (Agarwal et al., 2021) or neural oblivious decision trees (Chang et al., 2022). Despite differences in parameterization, these models obtain their simple feature effects from the same structural restriction: each feature contributes independently of the context.

3 Structured Laminar Individual Conditional Effects

Notation Let N be a natural number and let $[N] = \{1, \ldots, N\}$. Let us consider a machine learning model $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_N$ and each $\mathcal{X}_i \subset \mathbb{R}$ is a compact set. We use x_{-i} to denote all elements of $\boldsymbol{x}$ apart from x_i and (t, x_{-i}) to denote a vector $\boldsymbol{x}$ with the ith element set to t. We denote the perturbation curve for feature i centered at c_i as $\Delta f_i^{c_i}$ and define it as

$$\Delta f_i^{c_i}(x_i, x_{-i}) = f(x_i, x_{-i}) - f(c_i, x_{-i}). \tag{1}$$

We denote the family of perturbation curves (for feature i, centered at c_i) as $\mathcal{F}_i^{c_i}$ defined as $\{\Delta f_i^{c_i}(\cdot, x_{-i}) \mid x_{-i} \in \mathcal{X}_{-i}\}$. The goal is to characterize when such a family is legible and then construct a machine learning model that guarantees it for all possible inputs. All functions in this paper are assumed to be continuous.

3.1 Simplest family of perturbation curves

The simplest way to ensure that a family $\mathcal{F}_i^{c_i}$ is legible is to restrict it to a single curve $f_i^{c_i}$, i.e., $\mathcal{F}_i^{c_i} = \{f_i^{c_i}\}$. In that case visualizing the entire family is as simple as plotting a single univariate function. Although such model is very legible, it is also very restricted. If there is only a single perturbation curve, then it means that there are no interactions between features and the model is an example of a generalized additive model (GAM). We prove the following proposition in Appendix A.

Proposition 1. *Let $f : \mathcal{X} \to \mathbb{R}$ be a machine learning model and let $\mathbf{c} \in \mathcal{X}$. Then for each $i \in [N]$ there exists a function f_i such that $f(\boldsymbol{x}) - f(c_i, x_{-i}) = f_i(x_i)$ for all $x_{-i} \in \mathcal{X}_{-i}$ if and only if f can be represented as $f(\boldsymbol{x}) = \sum_{i=1}^{N} g_i(x_i) + b$ for some univariate functions $g_i : \mathcal{X}_i \to \mathbb{R}$ and a constant $b \in \mathbb{R}$. Moreover, each g_i is equal to f_i up to a constant.*

3.2 Laminar perturbation curves and SLICE

To increase the expressivity of our model, we need to allow for a more diverse family of perturbation curves. In particular, the curves need to change for different contexts. We observe that the main reason behind illegible ICE plots is the fact that the perturbation curves intersect and interleave at

arbitrary points. To address that, we introduce the notion of *laminar* perturbation curves. Laminar curves form a structured, ordered family when the curves do not intersect at arbitrary points.

Definition 1. A family of perturbation curves $\mathcal{F}_i^{c_i}$ is called *laminar* if there exist $l_i : \mathcal{X}_i \to R$, and $v_i : \mathcal{X}_i \to R$ (called boundary functions) with $l_i(c_i) = v_i(c_i) = 0$, $h_i : \mathcal{X}_i \times [0,1] \to [0,1]$ strictly increasing in the second argument, and $r_i : \mathcal{X}_{-i} \to [0,1]$ such that $\Delta f_i^{c_i}(x_i, x_{-i}) = (1 - h_i(x_i, r_i(x_{-i}))l_i(x_i) + h_i(x_i, r_i(x_{-i}))v_i(x_i)$ and $h_i(x_i, 0) = 0$, $h_i(x_i, 1) = 1$ for all $\boldsymbol{x} \in \mathcal{X}$.

Let us unpack this definition. First, as h_i is strictly increasing in the second argument and $h_i(\cdot, 0) = 0$ and $h_i(\cdot, 1) = 1$ implies that $0 \leq h_i \leq 1$. That means that each perturbation curve needs to be between $l_i(x_i)$ and $v_i(x_i)$ for all $\boldsymbol{x} \in \mathcal{X}$. This is very important for global guarantees on the perturbation curves. Moreover, each context $x_{-i} \in \mathcal{X}_{-i}$ determines a corresponding "level" for the perturbation curve described by $r_i(x_{-i})$. Crucially, those levels remain separate and cross only when l_i and v_i themselves cross. See Appendix A for a proof. We call a model whose all perturbation curve families are laminar SLICE (Structured Laminar Individual Conditional Effects).

Definition 2. $f : \mathcal{X} \to \mathbb{R}$ is called SLICE if for each $i \in [N]$ $\mathcal{F}_i^{c_i}$ is laminar.

3.3 AFFINE SLICE

In this section, we are going to focus on a simple type of SLICE model called Affine SLICE (or A-SLICE). It is defined as follows.

Definition 3. $f : \mathcal{X} \to \mathbb{R}$ is called Affine SLICE if for each $i \in [N]$, the perturbation curve family $\mathcal{F}_i^{c_i}$ is laminar and each $h_i(x_i, r) = r$. In other words, for each $i \in [N]$

$$\Delta f_i^{c_i}(x_i, x_{-i}) = (1 - r_i(x_{-i}))l_i(x_i) + r_i(x_{-i})v_i(x_i) \tag{2}$$

That means that each perturbation curve is an affine combination of the two boundary curves. Depending on the setting, slightly different formulations may be more convenient. For that reason, we introduce the width function $w_i : \mathcal{X}_i \to \mathbb{R}$ defined as $w_i(x_i) = v_i(x_i) - l_i(x_i)$, and the middle function $m_i : \mathcal{X}_i \to \mathbb{R}$ defined as $m_i(x_i) = \frac{l_i(x_i)+v_i(x_i)}{2}$. Then

$$\Delta f_i^{c_i}(x_i, x_{-i}) = m_i(x_i) + \delta_i(x_{-i})w_i(x_i) \tag{3}$$

where $\delta_i(x_{-i}) = r_i(x_{-i}) - \frac{1}{2}$ describes the deviation from the middle function and $|\delta_i(x_{-i})| \leq 1/2$.

It turns out that every Affine SLICE can be expressed in the form that we introduce below.

Theorem 1. *A function $f : \mathcal{X} \to \mathbb{R}$ is an Affine SLICE if and only if it can be represented in the following form.*

$$f(\boldsymbol{x}) = \alpha_\emptyset + \sum_{n=1}^{N} g_n(x_n) + \sum_{\substack{S \subseteq [N] \\ |S| \geq 2}} \alpha_S \prod_{k \in S} u_k(x_k) \tag{4}$$

where for any $S \subseteq [N]$, $\alpha_S \in \mathbb{R}$, and each $g_i : \mathcal{X}_i \to \mathbb{R}$ and $u_i : \mathcal{X}_i \to \mathbb{R}$ are univariate functions and $|u_i(x_i)| \leq 1$ for all $x_i \in \mathcal{X}_i$.

The proof of this theorem can be found in Appendix A. We make a few observations about A-SLICE.

The form of the perturbation curves Let us define $A_i = \sum_{S \ni i} |\alpha_S|$. Then for $A_i > 0$

$$\Delta f_i^{c_i}(x_i, x_{-i}) = m_i(x_i) + \delta_i(x_{-i})w_i(x_i) \tag{5}$$

$$m_i(x_i) = g_i(x_i) - g_i(c_i) \tag{6}$$

$$w_i(x_i) = 2A_i(u_i(x_i) - u_i(c_i)) \tag{7}$$

$$\delta_i(x_{-i}) = \frac{1}{2A_i} \sum_{\substack{S \subseteq [N] \\ S \ni i}} \alpha_S \prod_{k \in S \setminus \{i\}} u_k(x_k). \tag{8}$$

Importantly, as we show in Appendix A, $|\delta_i(x_{-i})| \leq \frac{1}{2}$. If we ensure that $g_i(c_i) = 0$ and $u_i(c_i) = 0$ then the equations simplify and we get

$$m_i(x_i) = g_i(x_i), \qquad w_i(x_i) = 2A_i u_i(x_i). \tag{9}$$

It is an extension of GAM We can see that the first two components of the model ($\alpha_\emptyset + \sum_{n=1}^{N} g_n(x_n)$) correspond exactly to a GAM. This makes sense as we saw earlier that a GAM is a model that has a trivial perturbation curve family (containing only one function). A-SLICE then can be interpreted as a GAM with additional interaction terms structured in such a way that the perturbation curve family remains laminar (which cannot be said about GA^{2}M (Lou et al., 2013) which have unconstrained pairwise interactions). As in the case of a GAM, the exact functions g_i are not identifiable (they can be shifted vertically). Interestingly, this does not impact the perturbation curves as they are centered. In our implementation, we remove that additional degree of freedom by ensuring that $g_i(c_i) = 0$. In that case, g_i can be identified with the middle perturbation curve m_i defined earlier. Because of that, we refer to g_i simply as the *middle function*.

Interactions of any order Although A-SLICE puts some structural assumptions on the interaction terms, it allows for interactions of any order! This is what the last term of Eq. (4) describes. Also note even though the number of all possible interaction coefficients is $\sum_{k=2}^{K} \binom{N}{k}$, the number of middle and modulation functions remains linear in the number of features. In practice, we usually limit the maximum interaction order to 2 or 3. We refer to the univariate functions u_i as *modulation functions*, as they modulate the contribution of each feature to every interaction term in which it participates. We additionally enforce that for each $i \in [N]$, $u_i(c_i) = 0$. By doing so, the width function w_i becomes a scaled version of u_i.

4 Implementation

Parametric form of the univariate functions. A-SLICE is parameterized by a set of univariate middle functions $\{g_i\}_{i=1}^{N}$ and modulation functions $\{u_i\}_{i=1}^{N}$. We enforce $g_i(c_i) = u_i(c_i) = 0$ and $|u_i(x_i)| \leq 1$, consistent with the parameterization introduced in Section 3. For numerical features, we use the training-set mean as the anchor c_i in all experiments. Both sets of univariate functions can be represented using different function classes. In this work, we consider two variants: multi-layer perceptrons (MLPs) and cubic splines represented as linear combinations of B-spline basis functions (de Boor, 1968). We denote the resulting models by A-SLICE$_{\text{MLP}}$ and A-SLICE$_{\text{Spline}}$, respectively. Further architectural details are provided in Appendix B.

Optimization and regularization. The middle functions, modulation functions, and interaction coefficients $\{\alpha_S\}$ are optimized jointly using gradient-based optimization. In addition to the predictive loss, we regularize the scale of the perturbation envelopes, with the regularization strength selected during hyperparameter tuning. This regularization encourages tighter perturbation bounds but is not required for their validity. For A-SLICE$_{\text{Spline}}$, we additionally apply second-difference regularization to encourage smooth univariate functions. Further details are provided in Appendix B.

5 Visualization

Visualization is particularly important for our model, since its constraints are designed to make the entire family of perturbation curves globally understandable.

Empirical perturbation curves. For feature i, we first evaluate the context-dependent scalar $\delta_i(x_{-i})$ for every sample in the dataset. We then plot the perturbation curve corresponding to the median δ_i, together with curves at each decile of its empirical distribution, including the empirical minimum and maximum. The median provides a representative effect, while consecutive decile curves delimit bands containing approximately 10% of the observed contexts. Because the shape of a perturbation curve is completely determined by δ_i, every observed curve either coincides with one of the plotted curves or lies between two adjacent ones. Thus, a small number of curves summarizes both the range and the empirical distribution of the full family.

Certified bounds. The empirical curves describe behavior on the observed contexts only, so we additionally visualize an envelope over all admissible inputs. A direct bound follows from the model constraint that each modulation function lies in $[-1, 1]$ and hence $|\delta_i| \leq \frac{1}{2}$, yielding

$$|\Delta f_i^{c_i}(x_i, x_{-i}) - m_i(x_i)| \leq \frac{1}{2}|w_i(x_i)|. \tag{10}$$

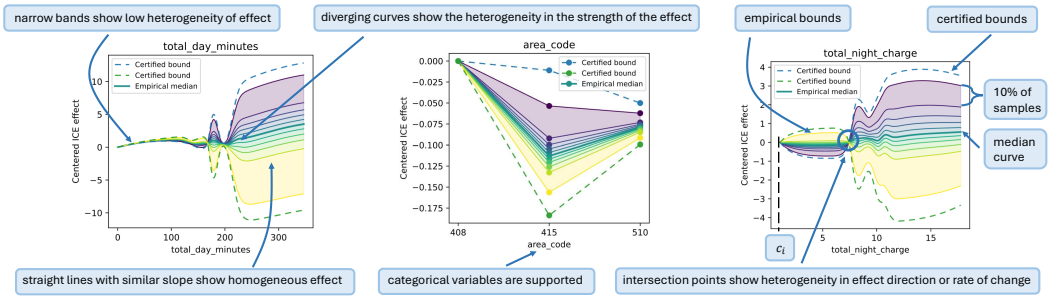

Figure 2: Visualizing perturbation curves in A-SLICE fitted to the churn dataset. The solid line represents the median empirical perturbation curve, while the shaded bands correspond to consecutive deciles of the empirical distribution, each containing approximately 10% of observed contexts. The empirical bounds indicate the range of effects observed in the dataset, whereas the certified bounds enclose all possible perturbation curves over the admissible feature domain. The structured, laminar arrangement ensures that the entire family can be characterized by a small number of representative curves without arbitrary crossings or interleaving.

In practice, these bounds can often be tightened because a fitted modulation function may occupy only a subset of $[-1, 1]$. We can bound each u_i between $\underline{u}_i$ and $\bar{u}_i$, i.e., $\underline{u}_i \leq u_i(x_i) \leq \bar{u}_i \; \forall x_i \in \mathcal{X}_i$. Then we can find asymmetric bounds for δ_i using the method described in Appendix A, giving us $\underline{\delta}_i \leq \delta_i(x_{-i}) \leq \bar{\delta}_i \; \forall x_i \in \mathcal{X}_i$. Then

$$m_i(x_i) + \min(\underline{\delta}_i w_i(x_i), \bar{\delta}_i w_i(x_i)) \leq \Delta f_i^{c_i}(x_i, x_{-i}) \leq m_i(x_i) + \max(\underline{\delta}_i w_i(x_i), \bar{\delta}_i w_i(x_i)). \quad (11)$$

This, however, will depend on the feature range $\mathcal{X}_i$ specified in the visualization. In our visualization we show $m_i(x_i) + \underline{\delta}_i w_i(x_i)$ and $m_i(x_i) + \bar{\delta}_i w_i(x_i)$ directly to emphasize when the bounds cross.

Interpreting the perturbation plots. The visualization provides a direct view of the type and magnitude of effect heterogeneity. A narrow band indicates that the perturbation effect of a feature depends weakly on the remaining features. In the limiting case in which all curves coincide, the feature has no interaction effect and behaves as in an additive model. Diverging curves indicate heterogeneity in effect strength, while differences in shape can reveal more complex interactions. For example, curves that move in opposite directions indicate heterogeneity in effect direction, whereas crossings may also occur when curves move in the same direction but change at different rates.

Discovering heterogeneous subgroups. The plots can also be used to identify and characterize subgroups with distinct perturbation behavior. For observation d and feature i, we can compute $\delta_i^{(d)} = \delta_i\left(x_{-i}^{(d)}\right)$, which describes where that observation lies within the family of possible perturbation curves. Observations can then be grouped according to, for example, the sign or range of $\delta_i^{(d)}$. A simple classifier such as a shallow decision tree can subsequently be fitted using x_{-i} to identify interpretable rules that characterize the resulting subgroups.

Editing the model through its perturbation curves. A further advantage of parameterizing perturbation curves directly is that the visualization can be used to modify the model. For instance, to fix the behavior around a sparsely sampled range or to enforce domain constraints. After training, we can edit the middle and width functions and then fix them and only optimize the parameters α_S. We provide the details in Appendix E.

6 Experiments

We conduct experiments to answer the following questions:

1. Do additional interactions improve performance for a fixed parametric form? (Section 6.1)
2. Can SLICE capture complex high-order interactions? (Section 6.2)
3. Does SLICE outperform other methods on common benchmarks? (Section 6.3)

Table 2: Predictive performance for different A-SLICE interaction orders. Evaluated using five-fold cross-validation. Method rows report R^2; improvement rows report the percentage reduction in RMSE relative to the corresponding additive model. Standard deviations in the brackets. Higher values are better.

Method	airfoil	bike sharing	california	concrete	abalone
MLP backend					
A-SLICE(1)$_{\text{MLP}}$	$0.550_{(.047)}$	$0.683_{(.013)}$	$0.723_{(.003)}$	$0.846_{(.011)}$	$0.559_{(.024)}$
A-SLICE(2)$_{\text{MLP}}$	$0.832_{(.025)}$	$0.815_{(.011)}$	$0.757_{(.007)}$	$0.869_{(.018)}$	$0.569_{(.019)}$
Improvement (RMSE reduction)	$+38.7\%_{(6.6\%)}$	$+23.6\%_{(2.4\%)}$	$+6.3\%_{(1.8\%)}$	$+8.0\%_{(6.5\%)}$	$+1.2\%_{(.6\%)}$
A-SLICE(3)$_{\text{MLP}}$	$0.831_{(.030)}$	$0.816_{(.029)}$	$0.768_{(.011)}$	$0.871_{(.018)}$	$0.565_{(.022)}$
Improvement (RMSE reduction)	$+38.6\%_{(7.6\%)}$	$+24.0\%_{(5.7\%)}$	$+8.4\%_{(2.1\%)}$	$+8.8\%_{(5.3\%)}$	$+0.8\%_{(.2\%)}$
Spline backend					
A-SLICE(1)$_{\text{spline}}$	$0.561_{(.038)}$	$0.673_{(.005)}$	$0.734_{(.006)}$	$0.883_{(.013)}$	$0.547_{(.024)}$
A-SLICE(2)$_{\text{spline}}$	$0.847_{(.017)}$	$0.835_{(.005)}$	$0.780_{(.004)}$	$0.897_{(.007)}$	$0.559_{(.015)}$
Improvement (RMSE reduction)	$+40.9\%_{(3.6\%)}$	$+29.0\%_{(.9\%)}$	$+9.1\%_{(.8\%)}$	$+6.0\%_{(3.3\%)}$	$+1.3\%_{(1.5\%)}$
A-SLICE(3)$_{\text{spline}}$	$0.829_{(.054)}$	$0.847_{(.005)}$	$0.783_{(.005)}$	$0.899_{(.007)}$	$0.553_{(.026)}$
Improvement (RMSE reduction)	$+38.1\%_{(8.3\%)}$	$+31.7\%_{(.9\%)}$	$+9.6\%_{(1.0\%)}$	$+7.1\%_{(3.0\%)}$	$+0.7\%_{(.6\%)}$

Baselines. We compare our approach against two types of baselines: models with simple perturbation effects (i.e., linear regression and GAMs) and fully flexible black-box models that serve as high-capacity performance baselines. Our experiments assess whether SLICE can close the performance gap between GAMs and black-box models while retaining understandable perturbation effects. In particular, we compare against three different variants of GAMs. GAM-Spline uses splines as shape functions (Servén & Brummitt, 2018), Explainable Boosting Machines (EBMs) use boosted decision trees (Lou et al., 2012; Nori et al., 2019) while Neural Additive Models (NAMs) use neural networks (Agarwal et al., 2021). We also include standard ridge linear regression. As black-box baselines, we use XGBoost (Chen & Guestrin, 2016) and Random Forest (Breiman, 2001). Further details about the experimental set-up, including method implementations, study design, hyperparameters and the ranges considered during tuning are provided in Appendix C.

6.1 Do interactions improve performance?

As we demonstrate in Section 3.3, A-SLICE can be interpreted as an extension of a GAM. This naturally raises the question of whether the additional interactions actually improve predictive performance. To investigate this, we fix the parametric family and compare models with no interactions, pairwise interactions, and three-way interactions. We evaluate these models on five regression datasets and report both their predictive performance and relative improvement. The results are shown in Table 2. Across all settings, adding pairwise interactions improves performance. While the improvement from no interactions to pairwise interactions is substantial, adding three-way interactions does not yield consistent additional gains. Thus, pairwise interactions may often be the right compromise between performance and computational cost in real-world settings. We therefore restrict the analysis in Section 6.3 to models with pairwise interactions.

6.2 Can it capture complex high-order interactions?

Although pairwise interactions may be sufficient in some settings, some datasets involve complex interactions with higher-order terms. Fortunately, as discussed earlier, one advantage of A-SLICE is its ability to incorporate interactions of *arbitrary order*. To evaluate this property, we consider synthetic datasets from the Feynman Symbolic Regression Database (Udrescu & Tegmark, 2020). These datasets are generated using well-known equations in physics, primarily from the Feynman Lectures, and therefore capture complex interactions among features. For these experiments, we run A-SLICE with all possible interaction terms. As shown in Table 3, even the strongest GAM baselines achieve only moderate performance because of the highly complex, non-additive structure of these datasets. In contrast, A-SLICE matches and, in some cases, exceeds the performance of the fully flexible black-box models. We additionally study the effect of interaction order while keeping the model architecture fixed in Appendix D.

6.3 Does it perform better than other methods?

In this section, we compare A-SLICE against the baseline methods on commonly used regression and classification datasets. The results can be seen in Table 4. In particular, we demonstrate that

Table 3: Predictive performance across Feynman datasets evaluated using five-fold cross-validation. We report mean R^2 over five folds with standard deviation in the bracket. Higher values are better. Best results within each model category are shown in bold.

Method	I.6.2b $\frac{\exp\left(-\frac{(\theta-\theta_0)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}}$	I.8.14 $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$	I.12.2 $\frac{q_1 q_2}{4\pi\epsilon r^2}$	I.12.11 $q(E_f + Bv\sin\theta)$	I.29.16 $\sqrt{x_1^2+x_2^2-2x_1x_2\cos(\theta_1-\theta_2)}$	I.32.5 $\frac{q^2a^2}{6\pi\epsilon c^3}$	I.40.1 $n_0\exp\left(-\frac{mgx}{k_bT}\right)$	II.2.42 $\frac{\kappa(T_2-T_1)A}{d}$
Black-box models	*High-flexibility models without an interpretable additive structure*							
Random Forest	$0.920_{(.026)}$	$0.870_{(.006)}$	$0.850_{(.075)}$	$0.966_{(.006)}$	$0.853_{(.041)}$	$\mathbf{0.768_{(.107)}}$	$0.808_{(.022)}$	$0.899_{(.015)}$
XGBoost	$\mathbf{0.921_{(.061)}}$	$\mathbf{0.935_{(.017)}}$	$\mathbf{0.854_{(.095)}}$	$\mathbf{0.977_{(.004)}}$	$\mathbf{0.878_{(.014)}}$	$0.681_{(.228)}$	$\mathbf{0.911_{(.007)}}$	$\mathbf{0.932_{(.008)}}$
Generalized additive models	*Models with additive, globally interpretable feature effects*							
GAM-spline	$\mathbf{0.705_{(.078)}}$	$\mathbf{0.191_{(.072)}}$	$\mathbf{0.651_{(.077)}}$	$\mathbf{0.681_{(.030)}}$	$\mathbf{0.269_{(.097)}}$	$\mathbf{0.360_{(.149)}}$	$0.705_{(.036)}$	$0.653_{(.012)}$
Ridge	$0.634_{(.080)}$	$-0.012_{(.016)}$	$0.544_{(.090)}$	$0.652_{(.014)}$	$0.164_{(.048)}$	$0.250_{(.062)}$	$0.669_{(.036)}$	$\mathbf{0.654_{(.008)}}$
NAM	$0.649_{(.056)}$	$0.139_{(.060)}$	$0.593_{(.120)}$	$0.666_{(.026)}$	$0.226_{(.053)}$	$0.340_{(.089)}$	$0.696_{(.032)}$	$0.647_{(.015)}$
EBM	$0.698_{(.070)}$	$0.180_{(.068)}$	$0.646_{(.085)}$	$0.679_{(.021)}$	$0.264_{(.070)}$	$0.340_{(.075)}$	$\mathbf{0.711_{(.024)}}$	$0.641_{(.018)}$
Our approaches	*Models with explicitly controlled single-feature perturbation behavior*							
A-SLICE(all)$_{\text{MLP}}$	$\mathbf{0.954_{(.008)}}$	$0.958_{(.009)}$	$0.933_{(.030)}$	$0.997_{(.002)}$	$0.818_{(.044)}$	$0.917_{(.059)}$	$0.960_{(.008)}$	$0.996_{(.001)}$
A-SLICE(all)$_{\text{spline}}$	$0.926_{(.077)}$	$\mathbf{0.961_{(.008)}}$	$\mathbf{0.980_{(.019)}}$	$\mathbf{1.000_{(.000)}}$	$\mathbf{0.831_{(.047)}}$	$\mathbf{0.968_{(.025)}}$	$\mathbf{0.962_{(.007)}}$	$\mathbf{0.999_{(.001)}}$

Table 4: Predictive performance across regression and binary-classification benchmark datasets. Evaluated using five-fold cross-validation. We report means over 5 folds with standard deviations in the brackets. Higher values are better. Best results within each model category are shown in bold.

	Regression (R^2)				Classification (ROC-AUC)			
Method	**airfoil**	**bike sharing**	**california**	**abalone**	**churn**	**magic**	**phoneme**	**compas**
Black-box models	*High-flexibility models without an interpretable additive structure*							
Random Forest	$0.935_{(.007)}$	$0.945_{(.003)}$	$0.818_{(.004)}$	$\mathbf{0.557_{(.025)}}$	$0.918_{(.019)}$	$0.936_{(.003)}$	$\mathbf{0.962_{(.006)}}$	$0.731_{(.009)}$
XGBoost	$\mathbf{0.951_{(.008)}}$	$\mathbf{0.955_{(.002)}}$	$\mathbf{0.853_{(.003)}}$	$0.557_{(.024)}$	$\mathbf{0.921_{(.017)}}$	$\mathbf{0.936_{(.004)}}$	$0.956_{(.004)}$	$\mathbf{0.733_{(.009)}}$
Generalized additive models	*Models with additive, globally interpretable feature effects*							
GAM-spline	$\mathbf{0.571_{(.044)}}$	$0.697_{(.005)}$	$0.717_{(.025)}$	$\mathbf{0.560_{(.028)}}$	$0.877_{(.018)}$	$0.905_{(.003)}$	$0.897_{(.009)}$	$\mathbf{0.732_{(.007)}}$
Ridge / L2 logistic	$0.506_{(.043)}$	$0.397_{(.012)}$	$0.604_{(.009)}$	$0.523_{(.037)}$	$0.824_{(.018)}$	$0.839_{(.003)}$	$0.812_{(.015)}$	$0.726_{(.008)}$
NAM	$0.496_{(.052)}$	$0.687_{(.009)}$	$0.728_{(.028)}$	$0.521_{(.029)}$	$0.866_{(.020)}$	$0.903_{(.003)}$	$0.892_{(.010)}$	$0.727_{(.013)}$
EBM	$0.569_{(.041)}$	$\mathbf{0.697_{(.005)}}$	$\mathbf{0.774_{(.005)}}$	$0.545_{(.025)}$	$\mathbf{0.882_{(.016)}}$	$\mathbf{0.908_{(.002)}}$	$\mathbf{0.898_{(.010)}}$	$0.732_{(.010)}$
Our approaches	*Models with explicitly controlled single-feature perturbation behavior*							
A-SLICE(2)$_{\text{MLP}}$	$0.837_{(.017)}$	$0.848_{(.064)}$	$0.786_{(.008)}$	$\mathbf{0.571_{(.019)}}$	$0.923_{(.013)}$	$0.924_{(.004)}$	$\mathbf{0.919_{(.006)}}$	$\mathbf{0.733_{(.009)}}$
A-SLICE(2)$_{\text{spline}}$	$\mathbf{0.855_{(.048)}}$	$\mathbf{0.873_{(.009)}}$	$\mathbf{0.796_{(.003)}}$	$0.549_{(.019)}$	$\mathbf{0.925_{(.017)}}$	$\mathbf{0.926_{(.004)}}$	$0.916_{(.008)}$	$0.730_{(.013)}$

A-SLICE can outperform GAMs while, in some settings, matching or exceeding the performance of fully flexible black-box models. At the same time, A-SLICE guarantees that its perturbation curves are laminar and satisfy certified bounds, preserving globally understandable perturbation effects.

7 Discussion and Conclusion

We introduced SLICE, a class of models that makes single-feature perturbation behavior an explicit part of model construction rather than a property examined only after training. By constraining perturbation curves to form structured, laminar families, SLICE enables a compact representation of heterogeneous feature effects with guarantees that extend to all admissible inputs. We characterized the functional form of Affine-SLICE and developed a differentiable parameterization that preserves these guarantees while allowing interactions of arbitrary order. Our experiments demonstrate that these interactions substantially improve predictive performance over additive models and allow SLICE, in selected settings, to approach the performance of fully flexible models. These benefits come with several limitations. First, the structured interaction parameterization restricts the class of functions that SLICE can represent, potentially limiting predictive performance when the underlying relationships are incompatible with laminar perturbation curves. Second, although the univariate components scale linearly with the number of features, the number of interaction coefficients grows combinatorially with the maximum interaction order, making unrestricted high-order interactions impractical in higher dimensions. Finally, while the perturbation bounds characterize the learned model globally, they do not establish whether the corresponding feature combinations are plausible under the data distribution. Future work could explore more flexible laminar parameterizations beyond Affine SLICE, as well as sparse or factorized interaction structures that improve scalability without sacrificing perturbation guarantees. Another promising direction is to develop methods for interactively editing perturbation curves to incorporate domain knowledge or enforce application-specific requirements.

AI USE STATEMENT

In this work, we used generative AI tools to assist with mathematical derivations and proofs, including suggesting and exploring structural forms of the prediction functions compatible with the proposed constraints. Generative AI was also used to assist with implementing the proposed algorithm, developing the benchmarking framework, and implementing baseline methods. Additionally, we used generative AI tools for searching and summarizing relevant literature, drafting parts of the manuscript, and editing the text to improve clarity and readability. Throughout this process, AI tools were used interactively under the authors' direction, with the authors guiding the research, critically evaluating generated suggestions, and making all final methodological and editorial decisions. The core research ideas, including the proposal to learn single-feature perturbation behavior directly rather than analyze it post-hoc, the definition of laminar perturbation curves, and the formulation of the Affine-SLICE model, originated from the authors. All AI-assisted mathematical arguments and proofs were reviewed and verified, generated code was inspected and tested for correctness, and AI-assisted literature summaries were checked against the original sources. The authors take full responsibility for the final content of this work, including all theoretical claims, experimental results, code, and text produced with the assistance of generative AI.

ETHICS STATEMENT

Although SLICE is designed to improve the interpretability of machine learning models, its explanations should not be treated as guarantees of model reliability, fairness, or correctness. In particular, the structural guarantees provided by SLICE concern the behavior of the learned prediction function and do not establish causal relationships or ensure that the model behaves appropriately in real-world applications. As with other explainability methods, there is a risk that seemingly simple and comprehensible explanations may create an unjustified sense of trust in a model, potentially facilitating the deployment of biased or unreliable systems, particularly in high-stakes domains. We therefore emphasize that SLICE should complement, rather than replace, rigorous model validation, fairness assessments, and domain-specific evaluation.

REPRODUCIBILITY STATEMENT

We have taken several steps to facilitate the reproducibility of our theoretical and experimental results. Section 3 provides formal definitions and theoretical characterization of SLICE, with complete proofs provided in Appendix A. Section 4 describes the model parameterization and optimization procedure, with additional architectural and implementation details in Appendix B. Section 6 presents the experimental setup and results, while Appendix C documents the datasets, preprocessing procedures, hyperparameter search spaces, and training protocols. The source code implementing SLICE, together with scripts for reproducing the experiments and generating the reported results, is available in an anonymized repository at `https://anonymous.4open.science/r/SLICE-ICLR/`.

REFERENCES

Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. Neural Additive Models: Interpretable Machine Learning with Neural Nets. In *Advances in Neural Information Processing Systems*, volume 34, pp. 4699–4711. Curran Associates, Inc., 2021.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Daniel W. Apley and Jingyu 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, September 2020. ISSN 1369-7412, 1467-9868. doi: 10.1111/rssb.12377.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324.

Matthew Britton. VINE: Visualizing Statistical Interactions in Black Box Models, April 2019.

Chun-Hao Chang, Rich Caruana, and Anna Goldenberg. NODE-GAM: Neural Generalized Additive Model for Interpretable Deep Learning, March 2022.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785.

Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng, Yutian Li, Jiaming Yuan, and David Cortes. *Xgboost: Extreme Gradient Boosting*.

Carl de Boor. On uniform approximation by splines. *Journal of Approximation Theory*, 1(2):219–235, September 1968. ISSN 00219045. doi: 10.1016/0021-9045(68)90026-9.

Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *journal of Computational and Graphical Statistics*, 24(1):44–65, 2015.

Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1(3):297–318, 1986.

Julia Herbinger, Bernd Bischl, and Giuseppe Casalicchio. REPID: Regional Effect Plots with implicit Interaction Detection. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, pp. 10209–10233. PMLR, May 2022.

Julia Herbinger, Marvin N. Wright, Thomas Nagler, Bernd Bischl, and Giuseppe Casalicchio. Decomposing Global Feature Effects Based on Feature Interactions. *Journal of Machine Learning Research*, 25(381):1–65, 2024. ISSN 1533-7928.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI machine learning repository.

Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019.

Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pp. 150–158, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 978-1-4503-1462-6. doi: 10.1145/2339530.2339556.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pp. 623–631, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 978-1-4503-2174-7. doi: 10.1145/2487575.2487579.

Christoph Molnar. *Interpretable Machine Learning*. 2020.

Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. InterpretML: A Unified Framework for Machine Learning Interpretability, September 2019.

Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. PMLB: A large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 10(1):36, December 2017. ISSN 1756-0381. doi: 10.1186/s13040-017-0154-4.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Daniel Servén and Charlie Brummitt. pyGAM: Generalized additive models in python, March 2018.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, April 2020. ISSN 2375-2548. doi: 10.1126/sciadv.aay2631.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198.

A THEORETICAL RESULTS

A.1 PROOF OF THE AFFINE SLICE FORM

First, let us restate the theorem below.

Theorem. *A function $f : \mathcal{X} \to \mathbb{R}$ is an Affine SLICE if and only if it can be represented in the following form.*

$$f(\boldsymbol{x}) = \alpha_\emptyset + \sum_{n=1}^{N} g_n(x_n) + \sum_{\substack{S\subseteq[N]\\|S|\geq 2}} \alpha_S \prod_{k\in S} u_k(x_k) \tag{12}$$

where for any $S \subseteq [N]$, $\alpha_S \in \mathbb{R}$, and each $g_i : \mathcal{X}_i \to \mathbb{R}$ and $u_i : \mathcal{X}_i \to \mathbb{R}$ are univariate functions and $|u_i(x_i)| \leq 1$ for all $x_i \in \mathcal{X}_i$.

To prove this theorem. We first need to prove two lemmas.

A.1.1 FIRST LEMMA

Lemma 1 (Factorization through the coordinate functions)**.** *Let f be an Affine SLICE (i.e., for each $i \in [N]$ Eq. (2) holds) and let*

$$F(\boldsymbol{x}) := f(\boldsymbol{x}) - \sum_{i=1}^{N} m_i(x_i). \tag{13}$$

Then there exists a well-defined function

$$G : w_1(X_1) \times \cdots \times w_N(X_N) \to \mathbb{R} \tag{14}$$

such that, for every $\boldsymbol{x} \in X$,

$$F(\boldsymbol{x}) = G(w_1(x_1), \ldots, w_N(x_N)). \tag{15}$$

Moreover, for each $i \in \{1, \ldots, N\}$, there exists a function

$$\rho_i : w_1(\mathcal{X}_1) \times \cdots \times w_{i-1}(\mathcal{X}_{i-1}) \times w_{i+1}(\mathcal{X}_{i+1}) \times \cdots \times w_N(\mathcal{X}_N) \to \mathbb{R} \tag{16}$$

such that, for every

$$(z_1, \ldots, z_N) \in w_1(X_1) \times \cdots \times w_N(X_N), \tag{17}$$

we have

$$G(\boldsymbol{z}) - G(0, z_{-i}) = \rho_i(z_{-i}) z_i. \tag{18}$$

Proof. By definition of F,

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) - \sum_{j=1}^{N} m_j(x_j). \tag{19}$$

Using the assumed identity

$$f(\boldsymbol{x}) - f(c_i, x_{-i}) = m_i(x_i) + \delta_i(x_{-i}) w_i(x_i), \tag{20}$$

together with $m_i(c_i) = 0$, we obtain

$$\begin{aligned} F(\boldsymbol{x}) - F(c_i, x_{-i}) &= \left(f(\boldsymbol{x}) - \sum_{j=1}^{N} m_j(x_j) \right) - \left(f(c_i, x_{-i}) - \sum_{j\neq i} m_j(x_j) - m_i(c_i) \right) && (21)\\ &= f(\boldsymbol{x}) - f(c_i, x_{-i}) - m_i(x_i) + m_i(c_i) && (22)\\ &= f(\boldsymbol{x}) - f(c_i, x_{-i}) - m_i(x_i) && (23)\\ &= \delta_i(x_{-i}) w_i(x_i). && (24) \end{aligned}$$

Thus, for every i,

$$F(\boldsymbol{x}) = F(c_i, x_{-i}) + \delta_i(x_{-i}) w_i(x_i). \tag{25}$$

We can see that F depends on x_i only through $w_i(x_i)$.

Now let $\boldsymbol{x}, \boldsymbol{y} \in X$ satisfy

$$w_i(x_i) = w_i(y_i) \tag{26}$$

for every $i \in \{1, \dots, N\}$. Define intermediate points $\boldsymbol{z}^{(0)}, \boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(N)} \in X$ by

$$\boldsymbol{z}^{(0)} := \boldsymbol{x}, \tag{27}$$

and, for $j = 1, \dots, N$,

$$\boldsymbol{z}^{(j)} := (y_1, \dots, y_j, x_{j+1}, \dots, x_N). \tag{28}$$

Then $\boldsymbol{z}^{(N)} = \boldsymbol{y}$. Moreover, $\boldsymbol{z}^{(j-1)}$ and $\boldsymbol{z}^{(j)}$ differ only in their j-th coordinate, and the corresponding w_j-values are equal. Therefore,

$$F(\boldsymbol{z}^{(j-1)}) = F(\boldsymbol{z}^{(j)}) \tag{29}$$

for every $j = 1, \dots, N$. Hence

$$F(\boldsymbol{x}) = F(\boldsymbol{y}). \tag{30}$$

Therefore F is constant on the level sets of the map

$$\boldsymbol{x} \mapsto (w_1(x_1), \dots, w_N(x_N)). \tag{31}$$

Define

$$\mathcal{Z}_i := w_i(\mathcal{X}_i). \tag{32}$$

Since $w_i(c_i) = 0$, we have $0 \in \mathcal{Z}_i$. The preceding argument implies that the following definition is independent of the chosen representative: for

$$(z_1, \dots, z_N) \in \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_N, \tag{33}$$

choose any $\boldsymbol{x} \in \mathcal{X}$ such that

$$w_i(x_i) = z_i \tag{34}$$

for every i, and define

$$G(z_1, \dots, z_N) := F(\boldsymbol{x}). \tag{35}$$

Thus G is well-defined and satisfies

$$F(\boldsymbol{x}) = G(w_1(x_1), \dots, w_N(x_N)). \tag{36}$$

It remains to show that G is affine in each coordinate separately. Fix i. If $w_i \equiv 0$, then $\mathcal{Z}_i = \{0\}$, and the coordinate is inactive. Otherwise, since w_i is not identically zero, there exists $x_i^\star \in \mathcal{X}_i$ such that

$$w_i(x_i^\star) \neq 0. \tag{37}$$

Define

$$z_i^\star := w_i(x_i^\star). \tag{38}$$

For $z_{-i} \in \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_{i-1} \times Z_{i+1} \times \cdots \times Z_N$, define

$$\rho_i(z_{-i}) := \frac{G(z_i^\star, z_{-i}) - G(0, z_{-i})}{z_i^\star}. \tag{39}$$

This is well-defined because G is well-defined and $z_i^\star \neq 0$.

Now take any $(z_1, \dots, z_N) \in Z_1 \times \cdots \times Z_N$. Choose $\boldsymbol{x} \in X$ such that

$$w_i(x_i) = z_i \tag{40}$$

for every j. Then

$$\begin{aligned} G(z_1, \dots, z_N) - G(0, z_{-i}) &= F(\boldsymbol{x}) - F(c_i, x_{-i}) && (41)\\ &= \delta_i(x_{-i}) w_i(x_i) && (42)\\ &= \delta_i(x_{-i}) z_i. && (43) \end{aligned}$$

Similarly,

$$\begin{aligned} G(z_i^\star, z_{-i}) - G(0, z_{-i}) &= F(x_i^\star, x_{-i}) - F(c_i, x_{-i}) && (44)\\ &= \delta_i(x_{-i}) w_i(x_i^\star) && (45)\\ &= \delta_i(x_{-i}) z_i^\star. && (46) \end{aligned}$$

Dividing by $z_i^\star$ gives

$$\rho_i(z_{-i}) = \delta_i(x_{-i}). \tag{47}$$

Hence

$$G(\boldsymbol{z}) - G(0, z_{-i}) = \rho_i(z_{-i}) z_i. \tag{48}$$

Since n was arbitrary, the claim follows. □

A.1.2 SECOND LEMMA

Lemma 2 (Separately affine functions are multi-affine)**.** *Let $M \in \mathbb{N}$. For each $j \in \{1, \dots, M\}$, let $\mathcal{Z}_j \subset \mathbb{R}$ satisfy $0 \in \mathcal{Z}_j$ and contain at least one nonzero element. Let*

$$G : \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_M \to \mathbb{R} \tag{49}$$

satisfy the following property: for every $j \in \{1, \dots, M\}$, there exists a function

$$\rho_j : \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_{j-1} \times \mathcal{Z}_{j+1} \times \cdots \times Z_M \to \mathbb{R} \tag{50}$$

such that, for all $(z_1, \dots, z_M) \in \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_M$, G is

$$G(z_1, \dots, z_M) - G(0, z_{-j}) = \rho_j(z_{-j}) z_j. \tag{51}$$

We say that G is separately affine in each of the M variables.

Then there exist constants

$$\alpha_S \in \mathbb{R}, \qquad S \subseteq \{1, \dots, M\}, \tag{52}$$

such that, for all $(z_1, \dots, z_M) \in \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_M$,

$$G(z_1, \dots, z_M) = \sum_{S \subseteq \{1, \dots, M\}} \alpha_S \prod_{k \in S} z_k. \tag{53}$$

Proof. We prove the result by induction on M.

First suppose $M = 1$. The assumption says that there exists a constant ρ_1 such that, for every $z_1 \in Z_1$,

$$G(z_1) - G(0) = \rho_1 z_1. \tag{54}$$

Therefore

$$G(z_1) = G(0) + \rho_1 z_1. \tag{55}$$

Thus the claim holds with

$$\alpha_\emptyset = G(0), \qquad \alpha_{\{1\}} = \rho_1. \tag{56}$$

Now assume that the claim holds for $M - 1$ variables. Let

$$G : \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_M \to \mathbb{R} \tag{57}$$

satisfy the stated separate-affinity property. Since $\mathcal{Z}_M$ contains a nonzero element, choose $a_M \in Z_M$ such that

$$a_M \neq 0. \tag{58}$$

Define

$$A(z_1, \dots, z_{M-1}) := G(z_1, \dots, z_{M-1}, 0), \tag{59}$$

and

$$B(z_1, \dots, z_{M-1}) := \frac{G(z_1, \dots, z_{M-1}, a_M) - G(z_1, \dots, z_{M-1}, 0)}{a_M}. \tag{60}$$

From the assumptions, for all $(z_1, \dots, z_M)$

$$G(z_1, \dots, z_M) = G(z_1, \dots, z_{M-1}, 0) + \rho_M(z_1, \dots, z_{M-1}) z_M \tag{61}$$

In particular, this is true when $z_M = a_M$, thus

$$\rho_M(z_1, \dots, z_{M-1}) a_M = G(z_1, \dots, a_M) - G(z_1, \dots, z_{M-1}, 0). \tag{62}$$

As $a_M \neq 0$,

$$\rho_M(z_1, \dots, z_{M-1}) = \frac{G(z_1, \dots, a_M) - G(z_1, \dots, z_{M-1}, 0)}{a_M} = B(z_1, \dots, z_{M-1}). \tag{63}$$

Thus we can write

$$G(z_1, \dots, z_M) = A(z_1, \dots, z_{M-1}) + B(z_1, \dots, z_{M-1}) z_M. \tag{64}$$

We now show that A and B satisfy the same separate-affinity property in the first $M-1$ variables.

Fix $j \in \{1, \dots, M-1\}$. Since G is separately affine in the j-th coordinate,

$$G(z_1, \dots, z_M) - G(z_1, \dots, z_{j-1}, 0, z_{j+1}, \dots, z_M) = \rho_j(z_{-j}) z_j. \tag{65}$$

Setting $z_M = 0$ gives

$$A(z_1, \dots, z_{M-1}) - A(z_1, \dots, z_{j-1}, 0, z_{j+1}, \dots, z_{M-1}) \tag{66}$$

$$= \rho_j(z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_{M-1}, 0) z_j. \tag{67}$$

Hence A is separately affine in the j-th coordinate. Since j was arbitrary, A is separately affine in the first $M-1$ variables.

Next, again fix $j \in \{1, \dots, M-1\}$. We have

$$B(z_1, \dots, z_{M-1}) - B(z_1, \dots, z_{j-1}, 0, z_{j+1}, \dots, z_{M-1}) \tag{68}$$

$$= \frac{G(z_1, \dots, z_{M-1}, a_M) - G(z_1, \dots, z_{M-1}, 0)}{a_M} \tag{69}$$

$$- \frac{G(z_1, \dots, z_{j-1}, 0, z_{j+1}, \dots, z_{M-1}, a_M) - G(z_1, \dots, z_{j-1}, 0, z_{j+1}, \dots, z_{M-1}, 0)}{a_M} \tag{70}$$

$$= \frac{G(z_1, \dots, z_{M-1}, a_M) - G(z_1, \dots, z_{j-1}, 0, z_{j+1}, \dots, z_{M-1}, a_M)}{a_M} \tag{71}$$

$$- \frac{G(z_1, \dots, z_{M-1}, 0) - G(z_1, \dots, z_{j-1}, 0, z_{j+1}, \dots, z_{M-1}, 0)}{a_M}. \tag{72}$$

Using separate affinity of G in the j-th coordinate, the first numerator equals

$$\rho_j(z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_{M-1}, a_M) z_j, \tag{73}$$

and the second numerator equals

$$\rho_j(z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_{M-1}, 0) z_j. \tag{74}$$

Therefore

$$B(z_1, \dots, z_{M-1}) - B(z_1, \dots, z_{j-1}, 0, z_{j+1}, \dots, z_{M-1}) \tag{75}$$

$$= \frac{\rho_j(z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_{M-1}, a_M) - \rho_j(z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_{M-1}, 0)}{a_M} z_j. \tag{76}$$

Hence B is separately affine in the j-th coordinate. Since j was arbitrary, B is separately affine in the first $M-1$ variables.

By the induction hypothesis, there exist constants

$$\beta_S \in \mathbb{R}, \qquad S \subseteq \{1, \dots, M-1\}, \tag{77}$$

such that

$$A(z_1, \dots, z_{M-1}) = \sum_{S \subseteq \{1, \dots, M-1\}} \beta_S \prod_{k \in S} z_k. \tag{78}$$

Similarly, there exist constants

$$\gamma_S \in \mathbb{R}, \qquad S \subseteq \{1, \dots, M-1\}, \tag{79}$$

such that

$$B(z_1, \dots, z_{M-1}) = \sum_{S \subseteq \{1, \dots, M-1\}} \gamma_S \prod_{k \in S} z_k. \tag{80}$$

Thus

$$G(z_1, \dots, z_M) = A(z_1, \dots, z_{M-1}) + B(z_1, \dots, z_{M-1}) z_M \tag{81}$$

$$= \sum_{S \subseteq \{1, \dots, M-1\}} \beta_S \prod_{k \in S} z_k + z_M \sum_{S \subseteq \{1, \dots, M-1\}} \gamma_S \prod_{k \in S} z_k \tag{82}$$

$$= \sum_{S \subseteq \{1, \dots, M-1\}} \beta_S \prod_{k \in S} z_k + \sum_{S \subseteq \{1, \dots, M-1\}} \gamma_S \prod_{k \in S \cup \{M\}} z_k. \tag{83}$$

Define coefficients α_T, for $T \subseteq \{1, \ldots, M\}$, by

$$\alpha_T := \beta_T \tag{84}$$

whenever $T \subseteq \{1, \ldots, M-1\}$, and by

$$\alpha_{S \cup \{M\}} := \gamma_S \tag{85}$$

whenever $S \subseteq \{1, \ldots, M-1\}$. Then

$$G(z_1, \ldots, z_M) = \sum_{T \subseteq \{1,\ldots,M\}} \alpha_T \prod_{k \in T} z_k. \tag{86}$$

This completes the induction step, and therefore the proof. □

A.1.3 Proof of the theorem

Proof. Let us assume that f is an Affine SLICE. Let us define

$$F(\boldsymbol{x}) := f(\boldsymbol{x}) - \sum_{i=1}^{N} m_i(x_i). \tag{87}$$

By the first lemma, there exists a well-defined function

$$G : \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_N \to \mathbb{R}, \qquad Z_i := w_i(\mathcal{X}_i), \tag{88}$$

such that

$$F(\boldsymbol{x}) = G(w_1(x_1), \ldots, w_N(x_N)). \tag{89}$$

Moreover, G is separately affine in the sense that, for every i, there exists a function ρ_i such that

$$G(z_1, \ldots, z_N) - G(z_1, \ldots, z_{i-1}, 0, z_{i+1}, \ldots, z_N) = \rho_i(z_{-i}) z_i. \tag{90}$$

By the second lemma, there exist constants

$$\alpha_S \in \mathbb{R}, \qquad S \subseteq \{1, \ldots, N\}, \tag{91}$$

such that

$$G(z_1, \ldots, z_N) = \sum_{S \subseteq \{1,\ldots,N\}} \alpha_S \prod_{k \in S} z_k. \tag{92}$$

Substituting

$$z_k = w_k(x_k) \tag{93}$$

gives

$$F(\boldsymbol{x}) = \sum_{S \subseteq \{1,\ldots,N\}} \alpha_S \prod_{k \in S} w_k(x_k). \tag{94}$$

Since

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) - \sum_{i=1}^{N} m_i(x_i), \tag{95}$$

we obtain

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} m_i(x_i) + \sum_{S \subseteq \{1,\ldots,N\}} \alpha_S \prod_{k \in S} w_k(x_k). \tag{96}$$

Separating the empty-set term and singleton set terms yields

$$f(\boldsymbol{x}) = \alpha_\emptyset + \sum_{n=1}^{N} (m_i(x_i) + \alpha_{\{i\}} w_i(x_i)) + \sum_{\substack{S \subseteq \{1,\ldots,N\} \\ |S| \geq 2}} \alpha_S \prod_{k \in S} w_k(x_k). \tag{97}$$

We define

$$g_i(x_i) = m_i(x_i) + \alpha_{\{i\}} w_i(x_i) \tag{98}$$

Also we define

$$W_i = \max_{x_i \in \mathcal{X}_i} |w_i(x_i)|, \tag{99}$$

$$\gamma_S = \alpha_S \prod_{k \in S} W_k, \tag{100}$$

and for $W_i > 0$ we set

$$u_i(x_i) = \frac{w_i(x_i)}{W_i}. \tag{101}$$

Otherwise we set

$$u_i(x_i) = 0. \tag{102}$$

Then

$$|u_i(x_i)| \leq 1 \tag{103}$$

and

$$f(\boldsymbol{x}) = \alpha_\emptyset + \sum_{n=1}^{N} g_i(x_i) + \sum_{\substack{S \subseteq \{1,\ldots,N\} \\ |S| \geq 2}} \gamma_S \prod_{k \in S} u_k(x_k). \tag{104}$$

as needed.

To prove the other direction, we assume that f is in the form given in Eq. (4). Then

$$\Delta f_i^{c_i}(x_i, x_{-i}) = g_i(x_i) - g_i(c_i) + \sum_{S \ni i} \alpha_S (u_i(x_i) - u_i(c_i)) \prod_{k \in S \setminus \{i\}} u_k(x_k) \tag{105}$$

Let

$$m_i(x_i) = g_i(x_i) - g_i(c_i) \tag{106}$$

and

$$A_i = \sum_{i \ni S} |\alpha_S|. \tag{107}$$

If $A_i = 0$ then it means that all $\alpha_S = 0$ for $S \ni i$. Then

$$\Delta f_i^{c_i}(x_i, x_{-i}) = m_i(x_i) \tag{108}$$

So we can just set $\delta_i(x_{-i}) = 0$. Let us consider a case when $A_i > 0$. Then we can define

$$w_i(x_i) = 2A_i(u_i(x_i) - u_i(c_i)) \tag{109}$$

and

$$\delta_i(x_{-i}) = \frac{1}{2A_i} \sum_{\substack{S \subseteq [N] \\ S \ni i}} \alpha_S \prod_{k \in S \setminus \{i\}} u_k(x_k). \tag{110}$$

Then we can write

$$\Delta f_i^{c_i}(x_i, x_{-i}) = m_i(x_i) + \delta_i(x_{-i}) w_i(x_i). \tag{111}$$

Moreover, using triangle inequality and the fact that $|u_i(x_i)| \leq 1$, we get

$$\begin{aligned} |\delta_i(x_{-i})| &= \frac{1}{2|A_i|} \left| \sum_{\substack{S \subseteq [N] \\ S \ni i}} \alpha_S \prod_{k \in S \setminus \{i\}} u_k(x_k) \right| \\ &\leq \frac{1}{2A_i} \sum_{\substack{S \subseteq [N] \\ S \ni i}} |\alpha_S| \prod_{k \in S \setminus \{i\}} |u_k(x_k)| \\ &\leq \frac{1}{2A_i} \sum_{\substack{S \subseteq [N] \\ S \ni i}} |\alpha_S| = \frac{A_i}{2A_i} = \frac{1}{2} \end{aligned} \tag{112}$$

Thus $|\delta_i(x_{-i})| \leq \frac{1}{2}$. This proves the converse and completes the proof. □

A.2 TRIVIAL PERTURBATION CURVE FAMILIES ARE GAMS

In this section, we prove Proposition 1. Let us restate it first.

Proposition. *Let $f : \mathcal{X} \to \mathbb{R}$ be a machine learning model and let $\boldsymbol{c} \in \mathcal{X}$. Then for each $i \in [N]$ there exists a function f_i such that $f(\boldsymbol{x}) - f(c_i, x_{-i}) = f_i(x_i)$ for all $x_{-i} \in \mathcal{X}_{-i}$ if and only if f can be represented as $f(\boldsymbol{x}) = \sum_{i=1}^{N} g_i(x_i) + b$ for some univariate functions $g_i : \mathcal{X}_i \to \mathbb{R}$ and a constant $b \in \mathbb{R}$. Moreover, each g_i is equal to f_i up to a constant.*

Proof. Let us assume that

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} g_i(x_i) + b. \tag{113}$$

Then

$$f(\boldsymbol{x}) - f(c_i, x_{-i}) = g_i(x_i) - g_i(c_i) \tag{114}$$

So we can set

$$f_i(x_i) = g_i(x_i) - g_i(c_i) \tag{115}$$

and the implication is satisfied (with $g_i(c_i)$ being the constant).

To prove the converse, we assume that for each $i \in [N]$

$$f(\boldsymbol{x}) - f(c_i, x_{-i}) = f_i(x_i) \tag{116}$$

for some f_i.

For $i = 1$, this gives us

$$f(\boldsymbol{x}) = f_1(x_1) + h_1(x_{-1}), \tag{117}$$

where $h_1(x_{-1}) = f(c_i, x_{-1})$. That implies for $i = 2$ that

$$f_2(x_2) = f(\boldsymbol{x}) - f(c_2, x_{-2}) = h_1(x_{-1}) - h_1(c_2, x_{-1,-2}). \tag{118}$$

This means that we can write

$$h_1(x_{-1}) = f_2(x_2) + h_2(x_{-1,-2}), \tag{119}$$

where $h_2(x_{-1,-2}) = h_1(c_2, x_{-1,-2})$. That means that

$$f(\boldsymbol{x}) = f_1(x_1) + f_2(x_2) + h_2(x_{-1,-2}). \tag{120}$$

By repeating this process, we will finally obtain that

$$f(\boldsymbol{x}) = f_1(x_1) + \ldots + f_{N-1}(x_{N-1}) + h_{N-1}(x_N) \tag{121}$$

Finally considering $i = N$,

$$f_N(x_N) = f(\boldsymbol{x}) - f(c_N, x_{-N}) = h_{N-1}(x_N) - h_{N-1}(c_N). \tag{122}$$

Thus

$$h_{N-1}(x_N) = f_N(x_N) + h_{N-1}(c_N). \tag{123}$$

That means that

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(x_i) + h_{N-1}(c_N) \tag{124}$$

which proves the implication (for $b = h_{N-1}(c_N)$). □

A.3 BOUNDING δ_i

In this section we show how to find lower and upper bound for δ_i given bounds for each modulation function u_i (i.e., $\underline{u}_i \leq u_i(x_i) \leq \bar{u}_i$). It turns out that it is enough to calculate the set

$$\left\{ \sum_{\substack{s \subseteq [N] \\ S \ni i}} \alpha_S \prod_{k \in S \setminus \{i\}} \hat{u}_k \;\middle|\; \hat{u}_j \in \{\underline{u}_j, \bar{u}_j\} \; \forall j \in [N] \right\} \tag{125}$$

and then take it minimum and maximum value to get $\underline{\delta}_i$ and $\bar{\delta}_i$.

We demonstrate this below.

Proposition 2 (Exact bounds on the context parameter). *For feature i with $A_i > 0$, define*

$$\underline{u}_k = \min_{x_k \in \mathcal{X}_k} u_k(x_k), \qquad \overline{u}_k = \max_{x_k \in \mathcal{X}_k} u_k(x_k),$$

and

$$\mathcal{V}_{-i} = \prod_{k \neq i} \{\underline{u}_k, \overline{u}_k\}.$$

Then

$$\min_{x_{-i} \in \mathcal{X}_{-i}} \delta_i(x_{-i}) = \min_{v \in \mathcal{V}_{-i}} \phi_i(v),$$

and

$$\max_{x_{-i} \in \mathcal{X}_{-i}} \delta_i(x_{-i}) = \max_{v \in \mathcal{V}_{-i}} \phi_i(v),$$

where

$$\phi_i(z_{-i}) = \frac{1}{2A_i} \sum_{S \ni i} \alpha_S \prod_{k \in S \setminus \{i\}} z_k.$$

Proof. We prove the result for the maximum; the argument for the minimum is analogous. Since ϕ_i is continuous on the compact box

$$\prod_{k \neq i} [\underline{u}_k, \overline{u}_k],$$

it attains its maximum at some point $z^\star$.

Suppose that $z^\star$ is not a vertex. Then there exists a coordinate $j \neq i$ such that

$$\underline{u}_j < z_j^\star < \overline{u}_j.$$

With all other coordinates fixed, ϕ_i is affine in z_j, and can therefore be written as

$$\phi_i(z_{-i}) = a + b z_j$$

for some constants a and b.

If $b > 0$, replacing $z_j^\star$ by $\overline{u}_j$ would increase ϕ_i, contradicting the maximality of $z^\star$. Similarly, if $b < 0$, replacing $z_j^\star$ by $\underline{u}_j$ would increase ϕ_i. Hence $b = 0$, so replacing $z_j^\star$ by either endpoint leaves the value of ϕ_i unchanged.

Applying the same argument to each coordinate that is not at an endpoint yields a vertex $v \in \mathcal{V}_{-i}$ satisfying

$$\phi_i(v) = \phi_i(z^\star).$$

Thus the maximum over the box is attained at a vertex.

Finally, every vertex is realizable by choosing each x_k independently at a point where u_k attains the corresponding minimum or maximum. Hence the vertex maximum is exactly the maximum of $\delta_i(x_{-i})$ over all admissible contexts. The same argument applies to the minimum. □

B Implementation details

We implement A-SLICE in PyTorch (Paszke et al., 2019) and optimize it with Lightning. The public estimators follow the scikit-learn API. The implementation supports regression and binary classification.

B.1 Data representation and preprocessing

All preprocessing statistics are estimated on the training set. A numerical feature is standardized as

$$\widetilde{x}_i = \frac{x_i - \mu_i}{\sigma_i}, \tag{126}$$

where μ_i and σ_i are the training-set mean and population standard deviation. We replace a zero standard deviation by one. The center c_i is first chosen in the original feature space and is then

transformed using the same statistics. Regression targets are standardized in the same way. Predictions and displayed effects are transformed back to the original target scale.

Categorical features are represented by integer codes and are not standardized. The codes index feature-specific embedding tables. Category labels and their codes are determined from the data supplied to `fit`. An unseen category is rejected at prediction time.

B.2 Univariate functions

B.2.1 Enforcing constraints

Middle functions. For each feature, let h_i be an unconstrained scalar function. We define the middle function as

$$g_i(x_i) = h_i(x_i) - h_i(c_i). \tag{127}$$

Hence $g_i(c_i) = 0$ by construction. This removes the additive degree of freedom and makes g_i the middle perturbation curve centered at c_i.

Modulation functions. Let q_i be a second unconstrained scalar function and let

$$d_i(x_i) = q_i(x_i) - q_i(c_i). \tag{128}$$

The signed modulation used in the experiments is

$$u_i(x_i) = \frac{d_i(x_i)}{\sqrt{\tau^2 + d_i(x_i)^2}}, \tag{129}$$

where $\tau > 0$ is a temperature. Thus $u_i(c_i) = 0$ and $-1 < u_i(x_i) < 1$. The package also provides a non-negative variant,

$$u_i(x_i) = \frac{d_i(x_i)^2}{\tau^2 + d_i(x_i)^2}, \tag{130}$$

for which $0 \le u_i(x_i) < 1$. The signed parameterization is not identifiable up to orientation: changing the sign of u_i and of every interaction coefficient containing feature i leaves the predictor unchanged. This has no effect on the perturbation envelope.

B.2.2 MLP

Architecture. The functions h_i and q_i are independent scalar MLPs with the same architecture. Each has L hidden layers of width H, an activation after each hidden affine layer, and a linear scalar output. Parameters are not shared across features or between h_i and q_i. We use Kaiming-uniform initialization for the weights and the corresponding uniform initialization for biases.

For a categorical feature, h_i and q_i are replaced by two learned scalar embeddings per category. We apply the same centering and modulation transforms as in Eq. (127) and Eq. (129).

Batched processing. To improve the computation time, we vectorize predictions of all univariate functions into several tensor operations. At each layer, weights have separate axes for channel, feature, input unit, and output unit. The two channels correspond to h_i and q_i. Feature networks remain independent, but their matrix operations are batched. We append the vector of centers to each input batch, evaluate inputs and centers together, and subtract the final center outputs. Categorical embeddings are evaluated in a separate batched bank and then restored to their original feature positions.

Interaction subsets are grouped by order. For each group, the implementation gathers the required modulation values, takes their product, and multiplies the resulting design matrix by the corresponding interaction coefficients. This is mathematically equivalent to evaluating every univariate function and interaction term separately.

B.2.3 Spline

For the spline backend, the two latent functions are

$$h_i(x_i) = \boldsymbol{B}_i(x_i)^\top \boldsymbol{\beta}_i, \qquad q_i(x_i) = \boldsymbol{B}_i(x_i)^\top \boldsymbol{\eta}_i, \tag{131}$$

where $\boldsymbol{B}_i$ is a feature-specific B-spline basis. Centering and the bounded transform are then applied exactly as above. Basis matrices are constructed once before optimization and passed to the model in mini-batches.

Knot selection. We use cubic, clamped B-splines. If J denotes the number of interior knots and p the spline degree, each feature has $J + p + 1$ basis functions. Boundary knots are the minimum and maximum of the standardized training values, each repeated $p + 1$ times. Interior knots are either equally spaced or placed at equally spaced empirical quantiles. Tied quantiles are replaced deterministically by distinct interior points. At prediction and inspection time, numerical values are clipped to the fitted training range before the basis is evaluated. This prevents uncontrolled extrapolation.

Smoothness regularization. We use a second-difference penalty for each coefficient vector. Writing Δ^2 for the second forward difference, the two penalties are

$$\mathcal{R}_g = \mathrm{mean}_{i,j}\left[(\Delta^2 \beta_{ij})^2\right], \qquad \mathcal{R}_u = \mathrm{mean}_{i,j}\left[(\Delta^2 \eta_{ij})^2\right]. \tag{132}$$

They regularize the latent middle and modulation expansions separately. The middle coefficients are initialized to zero. Modulation coefficients are drawn from $\mathcal{N}(0, 0.1^2)$.

B.3 Objective and optimization

Let $\mathcal{I}_K$ contain all feature subsets of orders two through K. The implemented predictor is

$$f(\boldsymbol{x}) = \alpha_\varnothing + \sum_{i=1}^{N} g_i(x_i) + \sum_{S \in \mathcal{I}_K} \alpha_S \prod_{k \in S} u_k(x_k). \tag{133}$$

For feature i, define the interaction budget

$$A_i = \sum_{\substack{S \in \mathcal{I}_K \\ i \in S}} |\alpha_S| \tag{134}$$

and the numerically stabilized envelope scale

$$B_i = \mathrm{hypot}(2A_i, \varepsilon), \qquad \varepsilon = 10^{-6}. \tag{135}$$

Loss function. For a regression mini-batch of size n, the objective is

$$\mathcal{L} = \frac{1}{n} \sum_{r=1}^{n} \left(f(\boldsymbol{x}^{(r)}) - y^{(r)} \right)^2 + \lambda_B \frac{1}{N} \sum_{i=1}^{N} B_i^2 + \lambda_g \mathcal{R}_g + \lambda_u \mathcal{R}_u. \tag{136}$$

The last two terms are present only for the spline backend. The envelope penalty encourages tighter certified bounds. It is not needed for their validity. It is zero for an additive model because there are no interaction terms. Validation loss is unregularized mean squared error.

The implementation also supports binary classification. In that case, the squared-error term is replaced by binary cross-entropy with logits. Optional inverse-frequency class weights are applied per observation. Perturbation effects are then measured on the log-odds scale.

Optimization. We jointly optimize all univariate parameters, the intercept, and the interaction coefficients with AdamW (Loshchilov & Hutter, 2019). We do not use a learning-rate schedule. MLP weight decay is tuned. Spline models use zero weight decay because smoothness is controlled explicitly by Eq. (132). The intercept is initialized to the mean standardized target, which is zero up to numerical precision.

The benchmark fits for at most 200 epochs. Fifteen percent of the supplied training data is reserved for early stopping. Training batches are shuffled. We stop after 15 epochs without an improvement in validation loss and restore the best parameter state in memory. Preprocessing statistics, centers, and spline knots are fitted without using the held-out validation rows.

Table 5: Hyperparameter search spaces for A-SLICE.

Backend	Hyperparameter	Search space
Both	Mini-batch size	$\{64, 128, 256, 512, 1024\}$
Both	Temperature τ	$\{0.5, 1, 2\}$
Both	Learning rate	$[10^{-4}, 10^{-1}]$
Both	Envelope penalty λ_B	$[10^{-5}, 10^{-2}]$
MLP	Hidden layers L	$\{1, 2, 3\}$
MLP	Hidden width H	$\{16, 32, 64\}$
MLP	Activation	{ELU, SiLU, leaky ReLU}
MLP	AdamW weight decay	$[10^{-5}, 10^{-2}]$
Spline	Interior knots J	$\{4, 6, 8, 12, 16\}$
Spline	Knot placement	{quantile, uniform}
Spline	Middle smoothness λ_g	$[10^{-5}, 10^{-1}]$
Spline	Modulation smoothness λ_u	$[10^{-5}, 10^{-1}]$

B.4 HYPERPARAMETERS

Hyperparameter ranges for tuning. Table 5 gives the search spaces used for the full benchmark comparisons. Continuous ranges are sampled log-uniformly. The same spaces are used for all interaction orders. In the controlled comparisons of backends and interaction orders, all settings except the learning rate are fixed at their default values. The learning rate is tuned over the same range.

For the full benchmark, we use 50 trials from a multivariate TPE sampler and also evaluate the default configuration. The controlled comparisons use 10 TPE trials, again with an additional evaluation of the default configuration. The default is: batch size 128, $\tau = 1$, learning rate 10^{-3}, and $\lambda_B = 10^{-3}$. The MLP default has two hidden layers, width 32, ELU activations, and weight decay 10^{-3}. The spline default has eight interior knots, quantile placement, and $\lambda_g = \lambda_u = 10^{-3}$. The degree is fixed at three. Signed modulation and the training-set mean center are fixed in all reported A-SLICE experiments.

B.5 CENTERS

The training center serves two purposes. First, subtracting the value at c_i fixes the otherwise arbitrary vertical shift of g_i. Second, it makes every interaction containing feature i vanish when $x_i = c_i$. The additive component g_i can therefore be read directly as the middle perturbation curve at the training center. We use the training-set mean for every numerical feature in the experiments. For a categorical feature, the implementation uses the first category in its deterministic ordering unless an explicit category is supplied.

The center used for a visualization need not equal the training center. Let a_i be any reference value and define

$$g_i^{(a)}(t) = g_i(t) - g_i(a_i), \qquad s_i^{(a)}(t) = 2A_i\big(u_i(t) - u_i(a_i)\big). \tag{137}$$

Then the exact reference-centered perturbation is

$$f(t, \boldsymbol{x}_{-i}) - f(a_i, \boldsymbol{x}_{-i}) = g_i^{(a)}(t) + \delta_i(\boldsymbol{x}_{-i}) s_i^{(a)}(t). \tag{138}$$

The same context-dependent scalar δ_i appears as for the training center. Thus a valid envelope for any display center is

$$g_i^{(a)}(t) - \frac{|s_i^{(a)}(t)|}{2} \le f(t, \boldsymbol{x}_{-i}) - f(a_i, \boldsymbol{x}_{-i}) \le g_i^{(a)}(t) + \frac{|s_i^{(a)}(t)|}{2}. \tag{139}$$

Changing a_i only re-centers the displayed curves and bounds. It does not retrain or alter the fitted predictor. The implementation accepts the training minimum, mean, median, or an explicit valid value as the visualization reference.

C EXPERIMENTAL DETAILS

Study design. All results in this section are evaluated using five-fold cross-validation. We shuffle observations before forming the folds. We use ordinary folds for regression and stratified folds for binary classification. For each outer fold, we split the development data (i.e., the four folds excluding the test fold) into a validation set used for hyperparameter tuning and a training set used for model fitting. The validation set contains 20% of the full dataset, equivalently 25% of the development set. The training, validation, and test partitions therefore contain 60%, 20%, and 20% of the observations, respectively.

Most methods draw a second validation set from the training data for early stopping or regularization. We fix its size to 15% of the data supplied to the method. This internal split is distinct from the validation partition used to select hyperparameters. A-SLICE, EBM, NAM, and XGBoost use an internal validation set. The linear models, GAM-spline, and Random Forest do not.

We perform 50 hyperparameter-tuning trials for each method unless stated otherwise. We use 10 trials when the parametric family of A-SLICE is fixed and only the learning rate is tuned. This applies to Table 2 and Fig. 3. In addition to these trials, we always evaluate the default configuration. The default is therefore eligible for selection.

Hyperparameter optimization and refitting. We optimize hyperparameters independently within each outer fold. We use the multivariate tree-structured Parzen estimator (TPE) (Bergstra et al., 2011) implemented in Optuna (Akiba et al., 2019). The sampler and model seeds are fixed within a fold. Each candidate is therefore evaluated on the same split and with the same model seed. We do not prune trials. We select regression candidates by minimum validation RMSE and classification candidates by maximum validation AUROC. We then rebuild the selected model and fit it on the complete development fold. Methods with early stopping create a new 15% internal validation split during this refit. The outer test fold is evaluated once.

Evaluation. For regression, we record RMSE, MAE, and R^2. For binary classification, we record log loss, AUROC, and balanced accuracy. The main benchmark reports R^2 for regression and AUROC for classification. We report the mean and sample standard deviation over the five outer folds. No test-fold metric is used for model or hyperparameter selection.

Dataset handling. Rows with a missing target or at least one missing feature are removed before splitting. Binary targets are encoded as 0 and 1. All learned preprocessing is fitted without access to the tuning-validation or test partition. For datasets with categorical features, we require every category in a held-out partition to occur in its corresponding training partition. The splitter makes at most 20 deterministic attempts to satisfy this condition. A dataset is rejected if no valid split is found.

A-SLICE. The A-SLICE implementation and its optimization objective are described in Appendix B. The full A-SLICE search spaces are given in Table 5. In the controlled comparison of interaction orders and backends, all parameters except the learning rate are fixed. The learning rate is sampled log-uniformly from $[10^{-4}, 10^{-1}]$. A-SLICE(K) contains all interaction terms of order at most K. A-SLICE(1) is additive. "A-SLICE(all)" contains every interaction order supported by the number of features.

C.1 BASELINES

Tables 6 and 7 give every tuned hyperparameter. A continuous interval marked "log" is sampled log-uniformly. Other continuous intervals are sampled uniformly. Integer intervals include both endpoints. All parameters not listed in the search tables remain fixed at their stated default values.

C.1.1 LINEAR MODELS

Implementation. For regression, the linear baseline is ridge regression. For classification, it is logistic regression with an L2 penalty and the `lbfgs` solver (Liu & Nocedal, 1989). We fit both

through scikit-learn (Pedregosa et al., 2011) pipelines. Numerical features are standardized. Categorical features are one-hot encoded. Unknown categories are ignored at prediction time. The preprocessing pipeline is fitted on the method's training partition only.

Default parameters and hyperparameter range. Ridge uses $\alpha = 1$. Logistic regression uses $C = 1$, where C is the inverse regularization strength. We tune only the regularization strength. The ranges are shown in Table 6.

C.1.2 GAM-SPLINE

Implementation. We use `LinearGAM` for regression and `LogisticGAM` for classification from pyGAM package (Servén & Brummitt, 2018). Each numerical source feature is represented by a cubic P-spline term. Each categorical source feature is represented by a factor term. A common smoothing penalty is used across terms. Category codes are learned from the training partition. We use direct model fitting rather than pyGAM's internal grid search.

Default parameters and hyperparameter ranges. The default has 20 splines per numerical term and smoothing penalty $\lambda = 0.6$. The spline order is fixed at three. We tune the number of splines and the shared smoothing penalty as shown in Table 6.

C.1.3 EBM

Implementation. We use the Explainable Boosting Machine implementation from InterpretML (Nori et al., 2019) Because we are interested in GAMs, we set the number of interactions to zero. Regression uses the RMSE objective. Classification uses log loss.

Default parameters. The default uses 1024 main-effect bins, 64 interaction bins, 14 outer bags, no inner bags, and two-leaf trees. The maximum number of boosting rounds is 50,000. Early stopping uses patience 100 and tolerance 10^{-5}. The default learning rate is 0.04 for regression and 0.015 for classification. The default number of smoothing rounds is 500 and 75, respectively. Other explicit defaults follow InterpretML 0.7.8.

Hyperparameter ranges. We use the task-specific spaces in Table 6.

C.1.4 NAM

Implementation. We use the official Google Research Neural Additive Model (Agarwal et al., 2021) graph builder. Numerical features and one-hot-encoded categorical features are jointly scaled to $[-1, 1]$. The encoder and scaler are fitted on the internal training partition. Regression checkpoints are selected by internal-validation RMSE. Classification checkpoints are selected by internal-validation AUROC. Outer hyperparameter selection still follows the metrics stated above.

Default parameters. The default uses batch size 1024, learning rate 10^{-2}, decay rate 0.995, dropout 0.5, no feature dropout, no output or L2 regularization, 1000 basis functions, an ExU activation, and a units multiplier of two. Training is capped at 100 epochs. This cap and the 15% internal validation split are benchmark settings. The remaining defaults match the official training flags.

Hyperparameter ranges. The project-specific search space is shown in Table 6. It contains every official default.

C.1.5 XGBOOST

Implementation. We use XGBoost implementation form xgboost package (Chen et al.). Classification uses the binary logistic objective. Categorical features are one-hot encoded. The preprocessor is fitted before the internal split is transformed and never uses held-out rows.

Default parameters. The default has 100 trees, learning rate 0.3, maximum depth six, minimum child weight one, full row and column subsampling, and L2 penalty one. It uses depth-wise growth,

Table 6: Hyperparameter search spaces for the linear, GAM-spline, EBM, and NAM baselines. "R/C" denotes regression/classification.

Method	Hyperparameter	Default	Search space
Ridge	L2 penalty α	1	$[10^{-4}, 10^{4}]$ (log)
L2 logistic	inverse L2 penalty C	1	$[10^{-4}, 10^{4}]$ (log)
GAM-spline	number of splines	20	$\{5, 10, 20, 40\}$
GAM-spline	smoothing penalty λ	0.6	$[10^{-3}, 10^{3}]$ (log)
EBM	leaves per tree	2	$\{2, 3\}$
EBM	smoothing rounds (R)	500	$\{100, 200, 500, 1000, 2000\}$
EBM	smoothing rounds (C)	75	$\{0, 25, 50, 75, 100, 150, 200\}$
EBM	learning rate	0.04/0.015	$[0.0025, 0.2]$ (log)
EBM	minimum samples per leaf	4	$\{2, 3, 4, 5, 10, 20\}$
NAM	batch size	1024	$\{64, 128, 256, 512, 1024\}$
NAM	learning rate	10^{-2}	$[10^{-4}, 10^{-1}]$ (log)
NAM	output regularization	0	$\{0, 0.01, 0.1\}$
NAM	L2 regularization	0	$\{0, 0.01, 0.1\}$
NAM	decay rate	0.995	$\{0.99, 0.995\}$
NAM	dropout	0.5	$\{0.3, 0.5, 0.7\}$
NAM	feature dropout	0	$\{0, 0.1, 0.3\}$
NAM	basis functions	1000	$\{500, 1000, 2000\}$
NAM	units multiplier	2	$\{2, 4, 8\}$
NAM	activation	ExU	{ExU, ReLU}

Table 7: Hyperparameter search spaces for XGBoost and Random Forest.

Method	Hyperparameter	Default	Search space
XGBoost	trees	100	$\{50, 100, 300, 1000, 2000\}$
XGBoost	maximum depth	6	$\{2, \ldots, 10\}$
XGBoost	minimum child weight	1	$\{1, 2, 5, 10\}$
XGBoost	learning rate	0.3	$[10^{-2}, 1]$ (log)
XGBoost	row subsample	1	$[0.6, 1]$
XGBoost	column subsample	1	$[0.6, 1]$
XGBoost	L2 penalty	1	$[10^{-3}, 10]$ (log)
Random Forest	trees	100	$\{100, 300, 1000\}$
Random Forest	maximum depth	none	$\{$none$, 4, 8, 16, 32\}$
Random Forest	minimum samples to split	2	$\{2, 5, 10, 20\}$
Random Forest	minimum samples per leaf	1	$\{1, 2, 4, 8\}$
Random Forest	features per split	1	$\{\sqrt{p}, 0.5p, p\}$
Random Forest	bootstrap sample fraction	all	$\{$all$, 0.5, 0.75\}$

automatic tree construction, and at most 256 histogram bins. All other explicit settings match XGBoost 3.2 tree-booster defaults. We stop after 50 rounds without improvement on the internal validation set.

Hyperparameter ranges. We tune the seven parameters in Table 7.

C.1.6 Random Forest

Implementation. We use the task-specific scikit-learn (Pedregosa et al., 2011) Random Forest estimator. Categorical features are one-hot encoded.

Default parameters. The default has 100 trees, no depth limit, a minimum of two samples to split, one sample per leaf, all features available at each split, bootstrap sampling, and no limit on the bootstrap sample size. The split criterion retains the task-specific scikit-learn default.

Hyperparameter ranges. We tune the six parameters in Table 7.

Table 8: Mean computation time per experimental fold in seconds (with standard deviation in the bracket). That includes 50 tuning trials, fitting with default and best parameters.

Method	airfoil	bike sharing	Regression california	concrete	abalone	churn	Classification magic	phoneme	compas
Black-box models	*High-flexibility models without an interpretable additive structure*								
Random Forest	$34.5_{(15.9)}$	$462_{(156)}$	$593_{(168)}$	$46.3_{(15)}$	$76.4_{(28.5)}$	$123_{(40.9)}$	$613_{(242)}$	$104_{(42.8)}$	$76.8_{(21.4)}$
XGBoost	$14.3_{(5.26)}$	$46.4_{(17.2)}$	$92.3_{(25.8)}$	$15.3_{(6.15)}$	$10.6_{(2.37)}$	$13.5_{(5.08)}$	$40.6_{(12.2)}$	$12.9_{(5.3)}$	$7.62_{(1.1)}$
Generalized additive models	*Models with additive, globally interpretable feature effects*								
GAM-spline	$44.2_{(14.9)}$	$24.9_{(.703)}$	$31.6_{(2.64)}$	$12.4_{(2.24)}$	$7.59_{(1.29)}$	$198_{(33.3)}$	$350_{(99)}$	$121_{(44.3)}$	$118_{(48.5)}$
Ridge / L2 logistic	$2.12_{(.138)}$	$3.75_{(.0303)}$	$2.22_{(.21)}$	$2.09_{(.148)}$	$2.5_{(.179)}$	$5.25_{(.251)}$	$5.25_{(.25)}$	$2.67_{(.171)}$	$5.35_{(.129)}$
NAM	$328_{(36.8)}$	$5.37e+03_{(1.29e+03)}$	$5.65e+03_{(1.77e+03)}$	$642_{(100)}$	$1.24e+03_{(107)}$	$2.14e+03_{(166)}$	$5.68e+03_{(1.67e+03)}$	$1.21e+03_{(225)}$	$1.56e+03_{(153)}$
EBM	$141_{(64.7)}$	$1.96e+03_{(2.21e+03)}$	$1.61e+03_{(455)}$	$721_{(535)}$	$643_{(276)}$	$224_{(70.4)}$	$555_{(151)}$	$277_{(191)}$	$234_{(88.4)}$
Our approaches	*Models with explicitly controlled single-feature perturbation behavior*								
A-SLICE(2)$_{\text{MLP}}$	$167_{(46.2)}$	$3.81e+03_{(1.15e+03)}$	$2.33e+03_{(840)}$	$145_{(42)}$	$371_{(124)}$	$1.29e+03_{(780)}$	$1.78e+03_{(409)}$	$517_{(138)}$	$361_{(164)}$
A-SLICE(2)$_{\text{spline}}$	$251_{(58.5)}$	$4.98e+03_{(1.36e+03)}$	$2.28e+03_{(379)}$	$172_{(28.4)}$	$759_{(129)}$	$1.63e+03_{(453)}$	$2.63e+03_{(812)}$	$614_{(167)}$	$737_{(252)}$

Table 9: Mean computation time per experimental fold in seconds (with standard deviation in the bracket). That includes 50 tuning trials, fitting with default and best parameters.

Method	I.6.2b $\frac{\exp\left(-\frac{(\theta-\theta_0)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}}$	I.8.14 $d=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$	I.12.2 $\frac{q1q2}{4\pi\epsilon r^2}$	I.12.11 $q(E_f+Bv\sin\theta)$	I.29.16 $\sqrt{\frac{x_1^2+x_2^2-}{2x_1x_2\cos(\theta_1-\theta_2)}}$	I.32.5 $\frac{q^2a^2}{6\pi\epsilon c^3}$	I.40.1 $n_0\exp\left(-\frac{mgx}{k_bT}\right)$	II.2.42 $\frac{\kappa(T_2-T_1)A}{d}$
Black-box models	*High-flexibility models without an interpretable additive structure*							
Random Forest	$27.6_{(12.1)}$	$35.1_{(16.3)}$	$40.8_{(15.5)}$	$34.9_{(15.8)}$	$38.7_{(16.5)}$	$39.6_{(14.2)}$	$39.1_{(17.4)}$	$32.6_{(12.4)}$
XGBoost	$7.85_{(2.47)}$	$14_{(3.18)}$	$9.22_{(2.81)}$	$16_{(2.76)}$	$12.6_{(4.54)}$	$8.89_{(2.48)}$	$18_{(3.81)}$	$16.9_{(3.81)}$
Our approaches	*Models with explicitly controlled single-feature perturbation behavior*							
A-SLICE(all)$_{\text{MLP}}$	$138_{(42.2)}$	$171_{(23.1)}$	$178_{(54.2)}$	$246_{(50.1)}$	$135_{(34.8)}$	$105_{(31)}$	$247_{(50.7)}$	$274_{(14.6)}$
A-SLICE(all)$_{\text{spline}}$	$161_{(39.5)}$	$173_{(34.4)}$	$223_{(20.8)}$	$302_{(95.2)}$	$145_{(38.7)}$	$174_{(24.4)}$	$304_{(66.3)}$	$279_{(59)}$
Other models								
ebm_additive	$106_{(31)}$	$108_{(32.7)}$	$148_{(37.1)}$	$111_{(12.9)}$	$108_{(33.7)}$	$81.9_{(10.3)}$	$170_{(46.6)}$	$119_{(17.3)}$
nam	$178_{(12.5)}$	$202_{(9.63)}$	$214_{(20.6)}$	$330_{(71.2)}$	$216_{(3.79)}$	$311_{(28.1)}$	$343_{(18.8)}$	$293_{(44.3)}$
pygam	$4.9_{(.665)}$	$3.31_{(.182)}$	$4.4_{(1.05)}$	$4.07_{(.377)}$	$3.46_{(.202)}$	$3.49_{(.221)}$	$4.11_{(.138)}$	$3.98_{(.215)}$
ridge	$2.08_{(.161)}$	$2.08_{(.147)}$	$2.04_{(.151)}$	$2.03_{(.0226)}$	$2.01_{(.0166)}$	$2.07_{(.147)}$	$2.1_{(.133)}$	$2.07_{(.16)}$

C.2 Reproducibility and software

Every split, Optuna sampler, and fitted estimator receives a deterministic seed. The seed used for model fitting is separated from the seed used to construct the data split. Each run stores the resolved configuration, dataset source, feature schema, preprocessing notes, Python and platform versions, Git revision, and a hash of the dependency lock file.

The locked environment uses scikit-learn 1.9.0, Optuna 4.9.0, InterpretML 0.7.8, pyGAM 0.12.0, XGBoost 3.2.0, TensorFlow 2.20.0, PyTorch 2.13.0, and Lightning 2.6.5. A-SLICE uses one GPU in the full benchmark configurations. EBM, XGBoost, and Random Forest use one CPU worker per fit.

C.3 Computation time

All experiments were run on two machines with Intel Core i9-10980XE limited to 8 cores each. One with NVIDIA RTX A4000 GPU and the other with NVIDIA GeForce RTX 3080 GPU. The complete benchmark in Table 4 required 81 hours (around 40h per machine). The mean times for a single fold (involving 50 tuning trials, fitting with default parameters and final refitting) for each method and dataset are shown in Table 8.

The experiments in Table 3 took 10.6 hours (around 5.3h per machine). Similarly the mean times for a single fold are shown in Table 9.

The experiments in Table 2 took 23.6 hours (around 11.8h per machine). The mean times for a single fold (involving 10 tuning trials, fitting with default parameters and final refitting) are shown in Table 10.

C.4 Datasets

We use the nine real-world datasets listed in Table 11 from UCI (Kelly et al.) and OpenML (Vanschoren et al., 2013) repositories. The regression datasets in Table 2 are Airfoil, Bike Sharing, California Housing, Concrete, and Abalone. The main benchmark additionally uses Churn, MAGIC

Table 10: Mean computation time per experimental fold in seconds for A-SLICE interaction orders (with standard deviation in the bracket). That includes 10 tuning trials, fitting with default and best parameters.

Method	airfoil	bike sharing	california	concrete	abalone
MLP backend					
A-SLICE(1)$_{\text{MLP}}$	$23.6_{(6.73)}$	$493_{(75.2)}$	$419_{(60.8)}$	$23.8_{(5.51)}$	$69_{(19.1)}$
A-SLICE(2)$_{\text{MLP}}$	$74.6_{(10.9)}$	$1.13e+03_{(87.4)}$	$744_{(120)}$	$41.7_{(8.85)}$	$104_{(30)}$
A-SLICE(3)$_{\text{MLP}}$	$87.7_{(13.8)}$	$3.74e+03_{(476)}$	$1.2e+03_{(155)}$	$71.3_{(17.4)}$	$188_{(47.6)}$
Spline backend					
A-SLICE(1)$_{\text{spline}}$	$38_{(2.96)}$	$400_{(32)}$	$470_{(49.5)}$	$36.7_{(1.9)}$	$127_{(7.25)}$
A-SLICE(2)$_{\text{spline}}$	$66.6_{(4.97)}$	$948_{(101)}$	$875_{(56.5)}$	$63.2_{(6.47)}$	$207_{(22.2)}$
A-SLICE(3)$_{\text{spline}}$	$84.9_{(3.6)}$	$3.23e+03_{(238)}$	$1.62e+03_{(202)}$	$108_{(17.8)}$	$358_{(25.8)}$

Table 11: Real-world datasets used in the reported experiments. Numerical and categorical feature counts refer to the source-level representation supplied to each method, before method-specific encoding.

Dataset	Source	Task	Samples	Num.	Cat.
Airfoil	UCI	Regression	1,503	5	0
Bike Sharing	OpenML 42712, version 2	Regression	17,379	6	6
California Housing	scikit-learn	Regression	20,640	8	0
Concrete	UCI	Regression	1,030	8	0
Abalone	UCI	Regression	4,177	7	1
Churn	OpenML 40701	Binary classification	5,000	15	3
MAGIC Gamma Telescope	OpenML 1120	Binary classification	19,020	10	0
Phoneme	OpenML 1489	Binary classification	5,404	5	0
COMPAS	OpenML 42192	Binary classification	5,278	5	8

Gamma Telescope, Phoneme, and COMPAS for binary classification. Dataset sizes in the table are measured after the dataset-specific processing described below.

Real-world dataset processing. We retain the source feature values except for explicit schema decisions. For Bike Sharing, month and hour are numerical. Season, year, holiday, weekday, working-day status, and weather are categorical. Abalone sex is categorical. For Churn, we remove state and phone number because they are identifiers or high-cardinality nominal labels. Area code, international-plan status, and voice-mail-plan status are categorical. The COMPAS source contains five numerical count or age features and eight binary categorical indicators. Rows with any missing feature or target are removed before splitting.

We encode binary targets as 0 and 1. Churn and COMPAS retain their source $0/1$ ordering. For MAGIC Gamma Telescope, source labels `g` and `h` map to 0 and 1. For Phoneme, source labels 1 and 2 map to 0 and 1. This mapping fixes the positive class used for AUROC. Method-specific preprocessing is described in the corresponding baseline or A-SLICE section.

Feynman symbolic-regression datasets. To test high-order interactions, we use the eight equations in Table 12. We obtain them from the PMLB (Olson et al., 2017) Feynman Symbolic Regression Database. The original PMLB datasets contain 100,000 observations. We draw one fixed sample of 1,000 observations without replacement using seed 42 and reuse it for every method and fold. All features and targets are numerical.

Table 12: Feynman symbolic-regression datasets used in the reported experiments. Every dataset contains 1,000 sampled observations.

Equation	Target function	Features
I.6.2b	$\exp\left[-(\theta-\theta_0)^2/(2\sigma^2)\right]/\sqrt{2\pi\sigma^2}$	3
I.8.14	$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$	4
I.12.2	$q_1 q_2/(4\pi\epsilon r^2)$	4
I.12.11	$q(E_f + Bv\sin\theta)$	5
I.29.16	$\sqrt{x_1^2+x_2^2-2x_1x_2\cos(\theta_1-\theta_2)}$	4
I.32.5	$q^2a^2/(6\pi\epsilon c^3)$	4
I.40.1	$n_0\exp[-mgx/(k_bT)]$	6
II.2.42	$\kappa(T_2-T_1)A/d$	5

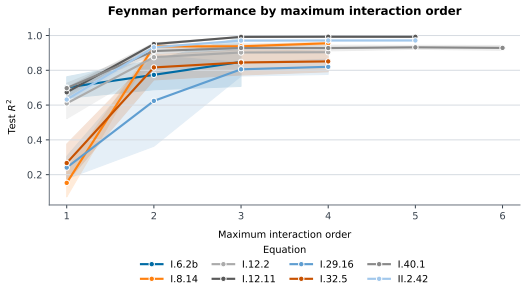

Figure 3: Impact of the maximum interaction order on performance on Feynman datasets.

D Additional results

D.1 Effect on maximum interaction order

We perform a series of experiments on the Feynman dataset, where we fix the architecture and change the maximum interaction order. In our case we choose A-SLICE(K)$_{\text{spline}}$ The results can be seen in Fig. 3. We see that adding pairwise and three-way interactions results in significant improvements while higher-order interaction show diminishing returns.

D.2 Perturbation curves - bike sharing dataset

In Fig. 4, we show SLICE fitted on the bike sharing dataset. In Fig. 5, we show c-ICE plots for an XGBoost model. We plot c-ICE curves both for the samples from the dataset as well as for other admissible contexts. We argue that it is much easier to understand the perturbation curves for SLICE compared to standard c-ICE curves calculated post-hoc from unconstrained model.

E Additional discussion

E.1 Model editing

To edit the model, we can fix the middle and width functions and optimize α_S parameters of the following function.

$$f(\boldsymbol{x}) = \alpha_\emptyset + \sum_{i=1}^{N} m_i(x_i) + \sum_{\substack{S\subseteq[N]\\|S|\geq 2}} \alpha_S \prod_{k\in S} w_k(x_k) \tag{140}$$

The only difference is that because each $|w_i|$ is not bounded by 1 then we need to enforce a constraint on α_S to ensure that we can have $|\delta_i(x_{-i})| \leq 1$. The easiest way to do it is to introduce

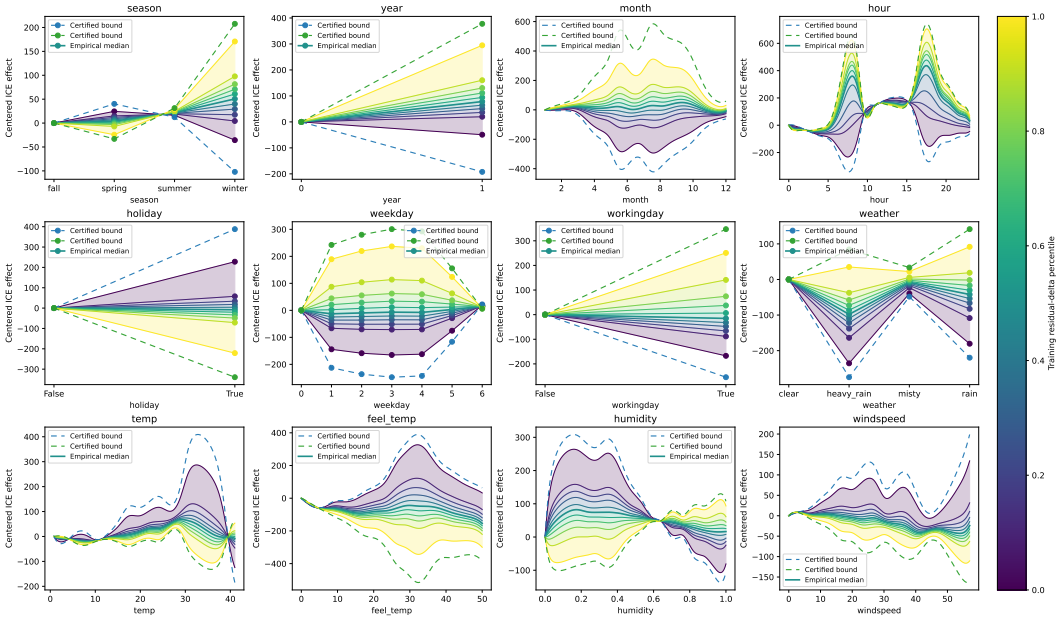

Figure 4: A-SLICE(2)$_{\text{spline}}$ fitted to the bike sharing dataset.

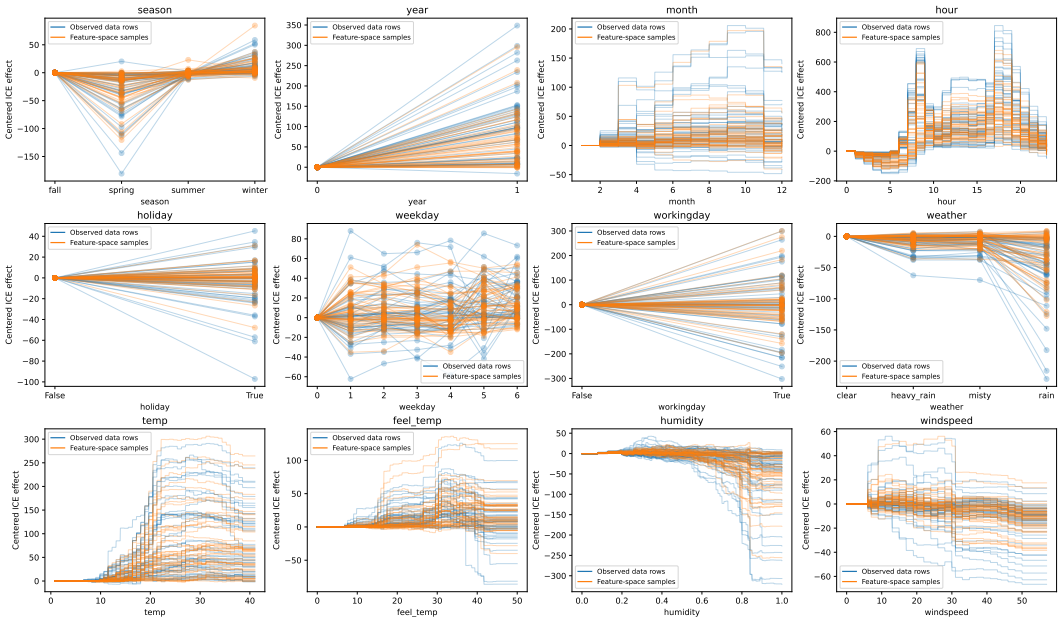

Figure 5: c-ICE plots of an XGBoost model fitted to the bike sharing dataset.

unconstrained parameters γ_S and introduce

$$W_i = \max_{x_i \in \mathcal{X}_i} |w_i(x_i)|, \tag{141}$$

$$D_i(\gamma) = \sum_{\substack{|S| \geq 2 \\ S \ni i}} |\gamma_S| \prod_{k \in S \setminus \{i\}} W_k \tag{142}$$

$$D(\gamma) = \max_{i \in [N]} D_i(\gamma) \tag{143}$$

Then we can set

$$\alpha_S = \frac{\gamma_S}{2(1 + D(\gamma))} \tag{144}$$

and the resulting model is

$$f(\boldsymbol{x}) = \alpha_\emptyset + \sum_{i=1}^{N} m_i(x_i) + \sum_{\substack{S \subseteq [N] \\ |S| \geq 2}} \frac{\gamma_S}{2(1 + D(\gamma))} \prod_{k \in S} w_k(x_k). \tag{145}$$

This guarantees that $|\delta_i(x_{-i})| \leq 1/2$.