Which Explanation Should I Choose? A Function Approximation Perspective to Characterizing Post Hoc Explanations

Tessa Han
Harvard University
Cambridge, MA
than@g.harvard.edu

Suraj Srinivas
Harvard University
Cambridge, MA
ssrinivas@seas.harvard.edu

Himabindu Lakkaraju
Harvard University
Cambridge, MA
hlakkaraju@hbs.edu

Abstract

A critical problem in the literature on post hoc explanations is the lack of a common foundational goal among methods. For example, some methods are motivated by function approximation, some use game theoretic notions such as Shapley-Aumann values, and some are ad hoc, driven by the goal of obtaining clean visualizations. Such fragmentation of goals not only prevents a coherent conceptual understanding of post hoc explainability but also causes the practical challenge of not knowing which method to use when.

In this work, we take the first steps towards bridging these gaps and unify eight popular post hoc explanation methods (LIME, C-LIME, SHAP, Vanilla Gradients, Gradients $\times$ Input, Integrated Gradients, SmoothGrad, Occlusion) by showing that they can all be viewed as performing local function approximation of the black-box model, with methods differing only by the local neighbourhoods and loss functions used to perform the approximation. This unification enables us to (1) state a *no free lunch theorem for explanation methods* which demonstrates that no single method can perform optimally across all neighbourhoods, and (2) provide a *guiding principle* to choose among methods based on the faithfulness of an explanation to the black-box model. We empirically validate our theoretical results using various real-world datasets, model classes, and prediction tasks.

By formalizing a mathematical framework which unifies diverse explanation methods, our work (1) advances the conceptual understanding of these methods, revealing their shared local function approximation objective and characterizing their relation to one another, and (2) guides the use of these methods in practice, providing a principled approach to choose among methods and paving the way for the creation of new ones.

1 Introduction

Machine learning models are increasingly being deployed in critical domains such as healthcare, law, and finance [1–3]. Consequently, there is a growing emphasis on explaining the predictions of these models to decision makers (e.g. doctors, judges) so that they can understand the rationale behind model predictions and determine when to rely on them. To this end, several methods have been proposed in the literature to explain model predictions in a *post hoc* local fashion. While these local explanation methods differ in a variety of ways, they can be broadly classified as *perturbation-based* or *gradient-based* methods based on their approach to estimating each feature's influence on the prediction. Popular perturbation-based methods include `LIME` [4], `SHAP` [5] and `Occlusion` [6], and gradient-based ones include `Vanilla Gradients` [7], `Gradient x Input` [8], `SmoothGrad` [9], and `Integrated Gradients` [10].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Although these methods are all local *post hoc* explanation methods, they have different objectives: for example, some are motivated by function approximation [4], some use game theoretic notions of Shapley-Aumann values [5], while others are driven by the goal of obtaining clean visualizations [9]. Such fragmentation of goals prevents a coherent conceptual understanding of post hoc explainability. Furthermore, the basic properties of each method and the conditions under which each method is effective are not well understood, leading to the practical challenge of not knowing which method to use when. The proper use of explanation methods in practice is particularly critical as different explanation methods have been shown to generate disagreeing, and sometimes even contradictory, explanations [11].

Prior works have taken the first steps towards characterizing explanation methods and establishing connections between them. For example, Agarwal et al. [12] proved that `C-LIME` (a continuous variant of `LIME` [4]) and `SmoothGrad` [9] converge to the same explanation in expectation. In addition, Lundberg and Lee [5] proposed a framework based on Shapley values to unify binary perturbation-based explanations and Covert et al. [13] found that many perturbation-based methods share the property of estimating feature importance based on the change in model behavior upon feature removal. Ancona et al. [14] also analyzed four gradient-based explanation methods and the conditions under which they produce similar explanations. However, these analyses of explanation methods are based on mechanistic properties (such as Shapley values and feature removal), are limited in scope (connecting only two methods, only perturbation-based methods, or only gradient-based methods), and do not inform when one method is preferable to another.

To address the aforementioned gaps, we study explanation methods from a function approximation perspective and formalize an overarching mathematical framework that characterizes a broad set of methods (spanning both perturbation-based and gradient-based methods), and provides a principled approach to choose among these methods. Our work makes the following contributions:

1. We show that eight popular explanation methods (such as `LIME`, `C-LIME`, `KernelSHAP`, `Occlusion`, `Vanilla Gradients`, `Gradient x Input`, `SmoothGrad`, and `Integrated Gradients`) spanning both perturbation-based and gradient-based methods, all perform local function approximation of the underlying model over different neighbourhoods using different loss functions.
2. We introduce a *no free lunch theorem for explanation methods* which demonstrates that no single explanation method can perform local function approximation faithfully across all local neighbourhoods, thereby underscoring the need for a strategy to choose among various explanation methods.
3. To choose among methods, we set forth a *guiding principle* based on the function approximation perspective, deeming a method to be effective if its explanation model recovers the underlying model when the former is in the same model class as the latter (i.e. if the explanation model perfectly approximates the underlying model when it is able to do so).
4. We validate the theoretical results above through extensive empirical evaluation with various real-world datasets, predictive model classes, and prediction tasks.

2 Related Work

Post hoc explanations. Post hoc explanation techniques of pre-trained models can be classified based on their access to the complex model (i.e. black box vs. access to model internals), scope of approximation (i.e. global vs. local), search technique (i.e. perturbation-based vs. gradient-based), and basic units of explanation (i.e. feature importance vs. rule-based). Examples are provided in the previous section. Recent work has also developed *counterfactual explanations* which capture the changes that need to be made to a given instance in order to flip its prediction [15–22]. Such explanations can be leveraged to provide recourse to individuals negatively impacted by algorithmic decisions. An alternate approach is to construct *global explanations*, summarizing the complete behavior of a black-box model by approximating it using an interpretable model [23–25].

Analyzing post hoc explanations. Recent work has shed light on the limitations of post hoc explanation methods. For instance, prior works [26–30] empirically demonstrated that perturbation-based methods, such as `LIME` and `SHAP`, may not be robust, i.e. small changes to the input may lead to drastic changes in the explanations. Adebayo et al. [29] and Srinivas and Fleuret [31] also

demonstrated that gradient-based methods do not always generate explanations that are faithful to the underlying model. In addition, recent research has also theoretically analyzed the robustness [32, 33] and other properties [34] of popular post hoc explanation methods. Finally, Watson et al. [35] show that logical concepts of sufficiency and necessity play a role in several post hoc explanations. However, none of these works provide a framework to characterize post hoc explanation methods and the connections among them or guide the selection of methods, all of which are the focus of our work.

3 Explanations as Local Function Approximations

In this section, we formalize the local function approximation framework and describe its connection to existing explanation methods. We proceed by defining the notation used in the rest of the paper.

Notation: Let $f : \mathcal{X} \to \mathcal{Y}$ be the black-box function we seek to explain in a post hoc manner, with input domain $\mathcal{X}$ (e.g. $\mathcal{X} = \mathbb{R}^d$ or $\{0,1\}^d$) and output domain $\mathcal{Y}$ (e.g. $\mathcal{Y} = \mathbb{R}$ or $[0,1]$). Let $\mathcal{G} = \{g : \mathcal{X} \to \mathcal{Y}\}$ be the class of interpretable models used to generate a local explanation for f by selecting a suitable interpretable model $g \in \mathcal{G}$.

We characterize locality around a point $\mathbf{x}_0 \in \mathcal{X}$ using a noise random variable ξ which is sampled from distribution $\mathcal{Z}$. Let $\mathbf{x}_\xi = \mathbf{x}_0 \oplus \xi$ be a perturbation of $\mathbf{x}_0$ generated by combining $\mathbf{x}_0$ and ξ using a binary operator $\oplus$ (e.g. addition, multiplication). Lastly, let $\ell(f, g, \mathbf{x}_0, \xi) \in \mathbb{R}^+$ be the loss function (e.g. squared error, cross-entropy) measuring the distance between f and g on a noise variable ξ around $\mathbf{x}_0$.

We now define the local function approximation framework.

Definition 1. ***Local function approximation*** *(LFA) of a black-box model f on a neighbourhood distribution $\mathcal{Z}$ around $\mathbf{x}_0$ by an interpretable model family $\mathcal{G}$ and a loss function ℓ is given by*

$$g^* = \arg\min_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{\xi \sim \mathcal{Z}} \ell(f, g, \mathbf{x}_0, \xi) \tag{1}$$

where a valid loss ℓ is such that $\mathbb{E}_{\xi \sim \mathcal{Z}}\, \ell(f, g, \mathbf{x}_0, \xi) = 0 \iff f(\mathbf{x}_\xi) = g(\mathbf{x}_\xi) \quad \forall \xi \sim \mathcal{Z}$

The LFA framework is a formalization of the function approximation perspective first introduced by `LIME` [4] to motivate local explanations. Note that this conceptual framework is distinct from the algorithm introduced by `LIME`. We elaborate on this distinction below.

(1) The LFA framework requires that f and g share the same input domain $\mathcal{X}$ and output domain $\mathcal{Y}$, a fundamental prerequisite for function approximation. This implies, for example, that using an interpretable model g with binary inputs ($\mathcal{X} = \{0,1\}^d$) to approximate a black-box model f with continuous inputs ($\mathcal{X} = \mathbb{R}^d$), as proposed in `LIME`, is not true function approximation.

(2) By imposing a condition on the loss function, the LFA framework ensures model recovery under specific conditions: g^* recovers f (i.e. $g^* = f$) through LFA when f itself is of the interpretable model class G (i.e. $f \in \mathcal{G}$) and perturbations span the input domain of f (i.e. domain$(\mathbf{x}) = \mathcal{X}$). This is a key distinction between the LFA framework and `LIME` (which has no such requirement) and guides the characterization of explanation methods in §4.

(3) Efficiently minimizing Equation 1 requires following standard machine learning methodology of splitting the perturbation data into train / validation / test sets and tuning hyper-parameters on the validation set to ensure generalization. To our knowledge, implementations of `LIME` do not adopt this procedure, making it possible to overfit to a small number of perturbations.

The LFA framework is generic enough to accommodate a variety of explanation methods. In fact, we will show that specific instances of this framework converge to existing methods, as summarized in Table 1. At a high level, existing methods use a linear model g to locally approximate the black-box model f in different input domains (binary or continuous) over different local neighbourhoods specified by noise random variable ξ (where ξ is binary or continuous, drawn from a specified distribution, and combined additively or multiplicatively with point $\mathbf{x}_0$) using different loss functions (squared-error or gradient-matching loss). We discuss the details of these connections in the following sections.

Explanation Method	Local Neighbourhood $\mathcal{Z}$ around $\mathbf{x}_0$	Loss Function ℓ
C-LIME	$\mathbf{x}_0 + \xi;\ \xi(\in \mathbb{R}^d) \sim \text{Normal}(0, \sigma^2)$	Squared Error
SmoothGrad	$\mathbf{x}_0 + \xi;\ \xi(\in \mathbb{R}^d) \sim \text{Normal}(0, \sigma^2)$	Gradient Matching
Vanilla Gradients	$\mathbf{x}_0 + \xi;\ \xi(\in \mathbb{R}^d) \sim \text{Normal}(0, \sigma^2), \sigma \to 0$	Gradient Matching
Integrated Gradients	$\xi\mathbf{x}_0;\ \xi(\in \mathbb{R}) \sim \text{Uniform}(0, 1)$	Gradient Matching
Gradients $\times$ Input	$\xi\mathbf{x}_0;\ \xi(\in \mathbb{R}) \sim \text{Uniform}(a, 1), a \to 1$	Gradient Matching
LIME	$\mathbf{x}_0 \odot \xi;\ \xi(\in \{0,1\}^d) \sim$ Exponential kernel	Squared Error
KernelSHAP	$\mathbf{x}_0 \odot \xi;\ \xi(\in \{0,1\}^d) \sim$ Shapley kernel	Squared Error
Occlusion	$\mathbf{x}_0 \odot \xi;\ \xi(\in \{0,1\}^d) \sim$ Random one-hot vectors	Squared Error

Table 1: Correspondence of existing explanation methods to instances of the LFA framework. Existing methods perform LFA of a black-box model f using the interpretable model class $\mathcal{G}$ of linear models where $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ over a local neighbourhood $\mathcal{Z}$ around point $\mathbf{x}_0$ based on a loss function ℓ. (Note: Exponential and Shapley kernels are defined in Appendix A.1.)

3.1 LFA with Continuous Noise: Gradient-Based Explanation Methods

To connect gradient-based explanation methods to the LFA framework, we leverage the gradient-matching loss function ℓ_{gm}. We define ℓ_{gm} and show that it is a valid loss function for LFA.

$$\ell_{gm}(f, g, \mathbf{x}_0, \xi) = \|\nabla_\xi f(\mathbf{x}_0 \oplus \xi) - \nabla_\xi g(\mathbf{x}_0 \oplus \xi)\|_2^2 \tag{2}$$

This loss function has been previously used in the contexts of generative modeling (where it is dubbed score-matching) [36] and model distillation [31]. However, to our knowledge, its use in interpretability is novel.

Proposition 1. *The gradient-matching loss function ℓ_{gm} is a valid loss function for LFA up to a constant, i.e. $\mathbb{E}_{\xi\sim\mathcal{Z}}\, \ell_{gm}(f, g, \mathbf{x}_0, \xi) = 0 \iff f(\mathbf{x}_\xi) = g(\mathbf{x}_\xi) + C \quad \forall \xi \sim \mathcal{Z}$, where $C \in \mathbb{R}$.*

Proof. If $f(\mathbf{x}_\xi) = g(\mathbf{x}_\xi)$, then $\nabla_\xi f(\mathbf{x}_\xi) = \nabla_\xi g(\mathbf{x}_\xi)$ and it follows from the definition of ℓ_{gm} that $\ell_{gm} = 0$. Integrating $\nabla_\xi f(\mathbf{x}_\xi) = \nabla_\xi g(\mathbf{x}_\xi)$ gives $f(\mathbf{x}_\xi) = g(\mathbf{x}_\xi) + C$. □

Proposition 1 implies that, when using the linear model class $\mathcal{G}$ parameterized by $g(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b$ to approximate f, g^* recovers $\mathbf{w}$ but not b. This can be fixed by setting $b = f(0)$.

Theorem 1. *LFA with gradient-matching loss is equivalent to (1) `SmoothGrad` for additive continuous Gaussian noise, which converges to `Vanilla Gradients` in the limit of a small standard deviation for the Gaussian distribution; and (2) `Integrated Gradients` for multiplicative continuous Uniform noise, which converges to `Gradient x Input` in the limit of a small support for the Uniform distribution.*

Proof Sketch. For `SmoothGrad` and `Integrated Gradients`, the idea is that these methods are exactly the first-order stationary points of the gradient-matching loss function under their respective noise distributions. In other words, the weights of the interpretable model g that minimize the loss function is the explanation returned by each method. For `Vanilla Gradients` and `Gradient x Input`, the result is derived by taking the specified limits and using the Dirac delta function to calculate the limit. In the limit, the weights of the interpretable model g converge to the explanation of each method. Full proofs are in Appendix A.1.

Along with gradient-based methods, `C-LIME` [12] (a perturbation-based method) is an instance of the LFA framework by definition, using the squared-error loss function. The analysis in this section characterizes methods that use continuous noise. It does not extend to binary nor discrete noise methods as gradients nor continuous random variables apply in these domains. In the next section, we discuss binary noise methods.

3.2 LFA with Binary Noise: LIME, KernelSHAP and Occlusion maps

Theorem 2. *LFA with multiplicative binary noise and squared-error loss is equivalent to (1) LIME for noise sampled from an unnormalized exponential kernel over binary vectors; (2) KernelSHAP*

for noise sampled from an unnormalized Shapley kernel; and (3) Occlusion for noise in the form of one-hot vectors.

Proof Sketch. For `LIME` and `KernelSHAP`, the equivalence is mostly by definition: these methods have components that correspond to the interpretable model g and the loss function ℓ of the LFA framework and we need only to determine the local neighbourhood $\mathcal{Z}$. We define the local neighbourhood $\mathcal{Z}$ using each method's weighting kernel. In this setup, the LFA framework yields the respective explanation method in expectation via importance sampling. For `Occlusion`, the equivalence involves enumerating all perturbations, specifying an appropriate loss function, and computing the resulting stationary points of the loss function. Full proofs are in Appendix A.1.

3.3 Which Methods Do Not Perform LFA?

Some popular explanation methods are not instances of the LFA framework due to properties of these methods. These methods include guided backpropagation, DeepLIFT [8], Grad-CAM. For a more detailed discussion, please see Appendix A.2.

4 When Do Explanations Perform Model Recovery?

Having described the LFA framework and its connections to existing explanation methods, we now leverage this framework to analyze the performance of methods under different conditions. We introduce a *no free lunch theorem for explanation methods*, inspired by classical no free lunch theorems in learning theory and optimization. Then, we assess the ability of existing methods to perform *model recovery* based on which we provide recommendations for choosing among methods.

4.1 No Free Lunch Theorem for Explanation Methods

An important implication of the function approximation perspective is that no explanation can be optimal across all neighbourhoods because each explanation is designed to perform LFA in a specific neighbourhood. This is especially true for explanations of non-linear models. We distill this intuitive observation into the following theorem.

Theorem 3 (No Free Lunch for Explanation Methods)**.** *Consider explaining a black-box model f around point $\mathbf{x}_0$ using an interpretable model g from class $\mathcal{G}$ and a valid loss function ℓ where the distance between f and $\mathcal{G}$ is given by $d(f, \mathcal{G}) = \min_{g \in \mathcal{G}} \max_{\mathbf{x} \in \mathcal{X}} \ell(f, g, 0, \mathbf{x})$.*

Then, for any explanation g^* on a neighbourhood distribution $\xi_1 \sim \mathcal{Z}_1$ such that $\max_{\xi_1} \ell(f, g^*, \mathbf{x}_0, \xi_1) \leq \epsilon$, we can always find another neighbourhood $\xi_2 \sim \mathcal{Z}_2$ such that $\max_{\xi_2} \ell(f, g^*, \mathbf{x}_0, \xi_2) \geq d(f, \mathcal{G})$.

Proof Sketch. The idea is that, given an explanation obtained by using g to approximate f over a specific local neighbourhood $\mathcal{Z}$, it is always possible to find a local neighbourhood over which this explanation does not perform well (i.e. does not perform faithful LFA). Thus, no single explanation method can perform well over all local neighbourhoods. The proof entails constructing an "adversarial" input for an explanation g^* such that g^* has a large loss for this input and then creating a neighbourhood that contains this adversarial input which will provably have a large loss. The magnitude of this loss is $d(f, \mathcal{G})$, the distance between f and the model class $\mathcal{G}$, inspired by the Haussdorf distance. The proof is generic and makes no assumptions regarding the forms of ℓ, $\mathcal{G}$ or $\mathcal{Z}_1$. The full proof is in Appendix A.3.

Thus, an explanation on a finite $\mathcal{Z}_1$ necessarily cannot approximate function behaviour at all other points, especially when $\mathcal{G}$ is less expressive than f, which is indicated by a large value of $d(f, \mathcal{G})$. Thus, in the general case, one cannot perform model recovery as $\mathcal{G}$ is less expressive than f.

An important implication of Theorem 3 is that seeking to find the "best" explanation without specifying a corresponding neighbourhood is futile as no universal "best" explanation exists. Furthermore, once the neighbourhood is specified, the best explanation is exactly given by the corresponding instance of the LFA framework.

In the next section, we consider the special case when $d(f, \mathcal{G}) = 0$ (i.e. when $f \in \mathcal{G}$), where Theorem 3 does not apply as the same explanation can be optimal for multiple neighbourhoods and model recovery is thus possible.

4.2 Characterizing Explanation Methods via Model Recovery

We proceed by formally stating the model recovery condition for explanation methods and then use this condition as a guiding principle to choose among methods.

Definition 2 (Model Recovery: Guiding Principle)**.** *Given an instance of the LFA framework with a black-box model f such that $f \in \mathcal{G}$ and a specific noise type (e.g. Gaussian, Uniform), an explanation method performs model recovery if there exists some noise distribution $\mathcal{Z}$ such that LFA returns $g^* = f$.*

In other words, when the black-box model f itself is of the interpretable model class G, there must exist some setting of the noise distribution (within the noise type specified in the instance of the LFA framework) that is able to recover the black-box model. Thus, in this special case, we require *local function approximation* to lead to *global model recovery* over all inputs. This criterion can be thought of as a "sanity check" for explanation methods to ensure that they remain faithful to the black-box model.

Next, we analyze the impact of the choice of perturbation neighbourhood $\mathcal{Z}$, the binary operator $\oplus$, and the interpretable model class $\mathcal{G}$ on an explanation method's ability to satisfy the model recovery guiding principle in different input domains $\mathcal{X}$. Note that while we can choose $\mathcal{Z}$, $\oplus$, and $\mathcal{G}$, we cannot choose $\mathcal{X}$, the input domain.

Which explanation should I choose for continuous $\mathcal{X}$? We now analyze the model recovery properties of existing explanation methods when the input domain is continuous. We consider methods based on additive continuous noise (`SmoothGrad`, `Vanilla Gradients`, and `C-LIME`), multiplicative continuous noise (`Integrated Gradients` and `Gradient x Input`), and multiplicative binary noise (`LIME`, `KernelSHAP`, and `Occlusion`). For these methods, we make the following remark regarding model recovery for the class of linear models.

Remark 1. *For $\mathcal{X} = \mathbb{R}^d$ and linear models f and g where $f(\mathbf{x}) = \mathbf{w}_f^\top \mathbf{x}$ and $g(\mathbf{x}) = \mathbf{w}_g^\top \mathbf{x}$, additive continuous noise methods recover f (i.e. $\boldsymbol{w}_g = \boldsymbol{w}_f$) while multiplicative continuous and multiplicative binary noise methods do not and instead recover $\boldsymbol{w}_g = \boldsymbol{w}_f \odot \boldsymbol{x}$.*

This remark can be verified by directly evaluating the explanations (weights) of linear models, where the gradient exactly corresponds to the weights.

We point out that the inability of multiplicative continuous noise methods to recover the black-box model is not due to the multiplicative nature of the noise, but rather due to the parameterization of the loss function. Specifically, these methods (implicitly) use the loss function $\ell(f, g, \mathbf{x}_0, \xi) = \|\nabla_\xi f(\mathbf{x}_\xi) - \nabla_\xi g(\xi)\|_2^2$. Slightly changing the loss function to $\ell(f, g, \mathbf{x}_0, \xi) = \|\nabla_\xi f(\mathbf{x}_\xi) - \nabla_\xi g(\mathbf{x}_\xi)\|_2^2$, i.e. replacing $g(\xi)$ with $g(\mathbf{x}_\xi)$, would enable g^* to recover f. This would change `Integrated Gradients` to $\int_{\alpha=0}^{1} \nabla_{\alpha\mathbf{x}} f(\alpha\mathbf{x})$ (omitting the input multiplication term) and `Gradient x Input` to `Vanilla Gradients`.

A similar argument can be made for binary noise methods which parameterize the loss function as $\ell(f, g, \mathbf{x}_0, \xi) = \|f(\mathbf{x}_\xi) - g(\xi)\|^2$. By changing the loss function to $\ell(f, g, \mathbf{x}_0, \xi) = \|f(\mathbf{x}_\xi) - g(\mathbf{x}_\xi)\|^2$, binary noise methods can recover f for the case described in Remark 1. However, binary noise methods for continuous domains are unreliable, as there are cases where, despite the modification to ℓ, model recovery is not guaranteed. The following is an example of such a case.

Remark 2. *For $\mathcal{X} = \mathbb{R}^d$, periodic functions f and g where $f(\mathbf{x}) = \sum_{i=1}^{d} \sin(\mathbf{w}_{f_i} \odot \mathbf{x}_i)$ and $g(\mathbf{x}) = \sum_{i=1}^{d} \sin(\mathbf{w}_{g_i} \odot \mathbf{x}_i)$, and an integer n, binary noise methods do not perform model recovery for $|w_{f_i}| \geq \frac{n\pi}{\mathbf{x}_{0_i}}$.*

This is because, for the conditions specified, $\sin(\mathbf{w}_{f_i}\mathbf{x}_{0_i}) = \sin(\pm n\pi) = \sin(0) = 0$, i.e. $\sin(\mathbf{w}_{f_i}\mathbf{x}_{0_i})$ outputs zero for all binary perturbations, thereby preventing model recovery. In this case, the discrete nature of the noise makes model recovery impossible. In general, discrete noise is inadequate for the recovery of models with large frequency components.

Which explanation should I choose for binary $\mathcal{X}$? In the binary domain, continuous noise methods are invalid, restricting us to binary noise methods. By the discussion above, methods with perturbation neighbourhoods characterized by multiplicative binary perturbations, such as `LIME`, `KernelSHAP`, and `Occlusion`, only enable g^* to recover f in the binary domain. Note that the sinusoidal example in Remark 2 does not apply in this regime due to the continuous nature of its domain.

Which explanation should I choose for discrete $\mathcal{X}$? In the discrete domain, continuous noise methods are also invalid. In addition, binary noise methods, such as `LIME`, `KernelSHAP` and `Occlusion`, also cannot be used as model recovery is not guaranteed in the sinusoidal case (Remark 2), following a logic similar to that presented for continuous noise. We notice that none of the existing methods in Table 1 perform general discrete perturbations, suggesting that these methods are not suitable for the discrete domain. Thus, in the discrete domain, a user can apply the LFA framework to define a new explanation method, specifying an appropriate discrete noise type. In the next section, we discuss more broadly about how one can use the LFA framework to create novel explanation methods.

4.3 Designing Novel Explanations with LFA

The LFA framework not only unifies existing explanation methods but also guides the creation of new ones. To explain a given black-box model prediction using the LFA framework, a user must specify the (1) interpretable model class $\mathcal{G}$, (2) neighbourhood distribution $\mathcal{Z}$, (3) loss function ℓ, and (4) binary operator $\oplus$ to combine the input and the noise. Specifying these completely specifies an instance of the LFA framework, thereby generating an explanation method tailored to a given context.

To illustrate this, consider a scenario in which a user seeks to create a sparse variant of `SmoothGrad` that yields non-zero gradients for only a small number of features ("`SparseSmoothGrad`"). Designing `SparseSmoothGrad` only requires the addition of a regularization term to the loss function used in the `SmoothGrad` instance of the LFA framework (e.g. $\ell = \ell_{SmoothGrad} + \|\nabla_\xi g(\mathbf{x}_\xi)\|_0$), at which point, sparse solvers may be employed to solve the problem. Note that, unlike `SmoothGrad`, `SparseSmoothGrad` does not have a closed form solution, but that is not an issue for the LFA framework. More generally, by allowing for the customization of (1), (2), (3), and (4), the LFA framework creates new explanation methods through "variations on a theme".

We summarize Section §4 as a table in Appendix A.4 and discuss the practical implications of Section §4 by providing the following recommendation for choosing among explanation methods.

Recommendation for choosing among explanation methods. In general, choose methods that satisfy the guiding principle of model recovery in the input domain in question. For continuous data, use additive continuous noise methods (e.g. `SmoothGrad`, `Vanilla Gradients`, `C-LIME`) or modified multiplicative continuous noise methods (e.g. `Integrated Gradients`, `Gradient x Input`) as described in Section §4.2. For binary data, use binary noise methods (e.g. `LIME`, `KernelSHAP`, `Occlusion`). Given that methods that use discrete noise do not exist, in case of discrete data, design novel explanation methods using the LFA framework with discrete noise neighbourhoods. Within each input domain, choosing among appropriate methods boils down to determining the perturbation neighbourhood most suitable in the given context.

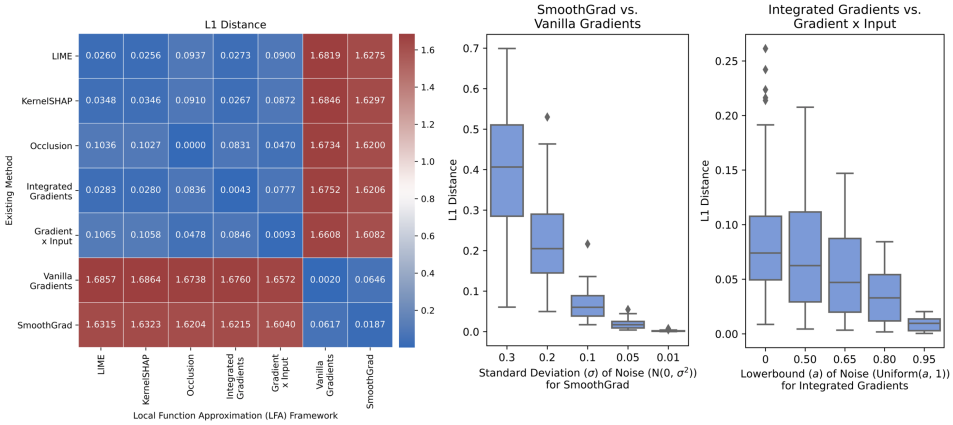
5 Empirical Evaluation

In this section, we present an empirical evaluation of the LFA framework. We first describe the experimental setup and then discuss three experiments and their findings.

5.1 Datasets, Models, and Metrics

Datasets. We experiment with two real-world datasets for two prediction tasks. The first dataset is the life expectancy dataset from the World Health Organization (WHO) [37]. It consists of countries' demographic, economic, and health factors from 2000 to 2015, with 2,938 observations for 20 continuous features. We use this dataset to perform regression, predicting life expectancy. The other dataset is the home equity line of credit (HELOC) dataset from FICO [38]. It consists of information on HELOC applications, with 9,871 observations for 24 continuous features. We use this dataset to perform classification, predicting whether an applicant made payments without being 90 days overdue. Additional dataset details are described in Appendix A.5.

Models. For each dataset, we train four models: a simple model (linear regression for the WHO dataset and logistic regression for the HELOC dataset) that can satisfy conditions of the guiding principle and three more complex models (neural networks of varying complexity) that are more reflective of real-world applications. Model architectures and performance are described in Appendix A.5.



(a) (b) (c)

Figure 1: Correspondence between existing methods and instances of the LFA framework. (**a**) Heatmap of average L1 distance between pairs of explanations. Boxplots of L1 distance between explanations of (**b**) `SmoothGrad` and `Vanilla Gradients` and (**c**) `Integrated Gradients` and `Gradient x Input`. The lower the L1 distance, the more similar two explanations are. Results indicate that existing explanation methods are instances of the LFA framework.

Metrics. To measure the similarity between two vectors (e.g. between two sets of explanations or between an explanation and the true model weights), we use L1 distance and cosine distance. L1 distance ranges between $[0, \infty)$ and is 0 when two vectors are the same. Cosine distance measures the angle between two vectors. It ranges between $[0, 2]$ and is 0 when the angle between two vectors is $0°$ (or $360°$). For both metrics, the lower the value, the more similar two given vectors are.

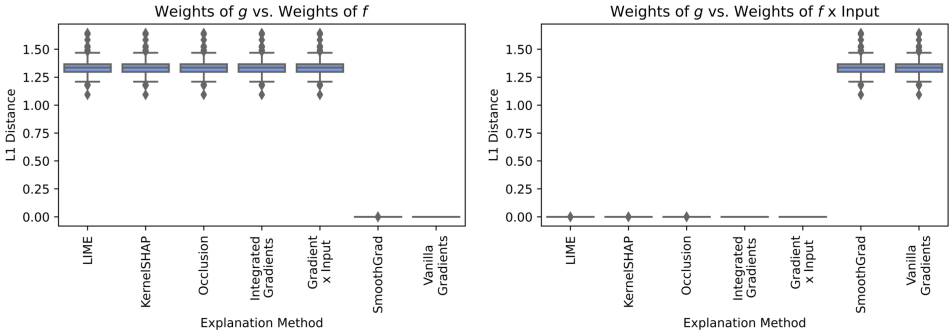
5.2 Experiments

Here, we describe the set-up of the experiments, present experimental results, and discuss their implications.

Experiment 1: Existing explanation methods are instances of the LFA framework. First, we compare existing methods with corresponding instances of the LFA framework to assess whether they generate the same explanations. To this end, we use seven methods to explain the predictions of black-box models on 100 randomly-selected test set points. For each method, explanations are computed using either the existing method (implemented by Meta's Captum library [39]) or the corresponding instance of the LFA framework (Table 1). The similarity of a given pair of explanations is measured using L1 distance or cosine distance.

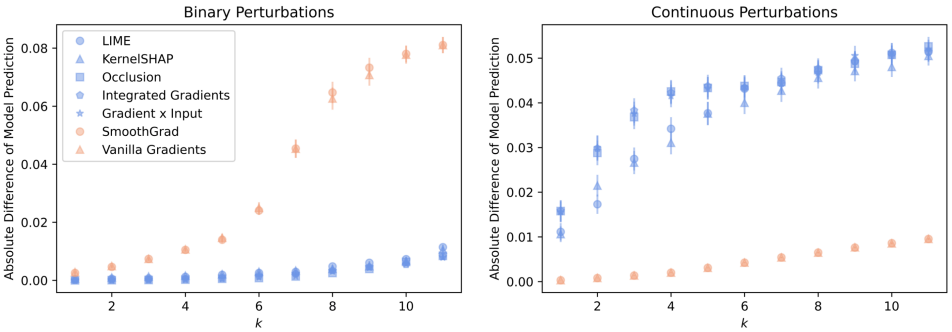
The L1 distance values for a neural network with three hidden layers trained on the WHO dataset are shown in Figure 1. In Figure 1a, lowest L1 distance values appear in the diagonal of the heatmap, indicating that explanations generated by existing methods and corresponding instances of the LFA framework are very similar. Figures 1b and 1c show that explanations generated by instances of the LFA framework corresponding to `SmoothGrad` and `Integrated Gradients` converge to those of `Vanilla Gradients` and `Gradient x Input`, respectively. Together, these results demonstrate that, consistent with the theoretical results derived in Section §3, existing methods are instances of the LFA framework. In addition, the clustering of the methods in Figure 1a indicates that, consistent with the theoretical analysis in Section §4, for continuous data, `SmoothGrad` and `Vanilla Gradients` generate similar explanations while `LIME`, `KernelSHAP`, `Occlusion`, `Integrated Gradients`, and `Gradient x Input` generate similar explanations. We observe similar results across various datasets, models, and metrics (Appendix A.6.1).

Experiment 2: Some methods recover the underlying model while others do not (guiding principle). Next, we empirically assess which existing methods satisfy the guiding principle, i.e. which methods recover the black-box model f when f is of the interpretable model class $\mathcal{G}$. We specify a setting in which f and g are of the same model class, generate explanations using each method, and assess whether g recovers f for each explanation. For the WHO dataset, we set f and g to be linear regression models and generate explanations for 100 randomly-selected test set points. Then, for each point, we compare g's weights with f's gradients or with f's gradients multiplied by



(a) (b)

Figure 2: Analysis of model recovery. The lower the L1 distance, the more similar g's weights are to (**a**) f's weights or (**b**) f's weights multiplied by the input. Results indicate that, for continuous data, additive continuous noise methods recover f's weights, satisfying the guiding principle, while multiplicative binary and multiplicative continuous noise methods do not, recovering f's weights multiplied by the input instead.



(a) (b)

Figure 3: Perturbation tests perturbing bottom k features using (**a**) binary or (**b**) continuous noise. The lower the curve, the better a method identifies unimportant features. Results illustrate the no free lunch theorem: no single method performs best across all neighborhoods.

the input because, based on Section §4, some methods generate explanations on the scale of gradients while others on the scale of gradient-times-input. Note that, for linear regression, f's gradients are f's weights.

Results are shown in Figure 2. Consistent with Section §4, for continuous data, `SmoothGrad` and `Vanilla Gradients` recover the black-box model, thereby satisfying the guiding principle, while `LIME`, `KernelSHAP`, `Occlusion`, `Integrated Gradients`, and `Gradient x Input` do not. We observe similar results for the HELOC dataset using logistic regression models for f and g (Appendix A.6.2).

Experiment 3: No single method performs best across all neighbourhoods (no free lunch theorem). Lastly, we perform a set of experiments to illustrate the no free lunch theorem in Section §4. We generate explanations for black-box model predictions for 100 randomly-selected test set points and evaluate the explanations using perturbation tests based on top-k or bottom-k features. For perturbation tests based on top-k features, the setup is as follows. For a given data point, k, and explanation, we identify the top-k features and either replace them with zero (binary perturbation) or add Gaussian noise to them (continuous perturbation). Then, we calculate the absolute difference in model prediction before and after perturbation. For each point, we generate one binary perturbation (since such perturbations are deterministic) and 100 continuous perturbations (since such perturbations are random), computing the average absolute difference in model prediction for the latter. In this setup, methods that better identify important features yield larger changes in model prediction. For perturbation tests based on bottom-k features, we follow the same procedure but perturb the bottom-k features instead. In this setup, methods that better identify unimportant features yield smaller changes in model prediction.

Results of perturbation tests based on bottom-k features performed on explanations for a neural network with three hidden layers trained on the WHO dataset are displayed in Figure 3. Consistent with the no free lunch theorem, `LIME`, `KernelSHAP`, `Occlusion`, `Integrated Gradients`, and `Gradient x Input` perform best on binary perturbation neighbourhoods (Figure 3a) while `SmoothGrad` and `Vanilla Gradients` perform best on continuous perturbation neighborhoods (Figure 3b). We observe consistent results across perturbation test types (top-k and bottom-k), datasets, and models (Appendix A.6.3). These findings have important implications: one should carefully consider the perturbation neighborhood not only when selecting a method to generate explanations but also when selecting a method to evaluate explanations. In fact, the type of perturbations used to evaluate explanations directly determines explanation method performance.

6 Conclusions and Future Work

In this work, we formalized the *local function approximation (LFA)* framework and demonstrated that eight popular explanation methods can be characterized as instances of this framework with different local neighbourhoods and loss functions. We also introduced the *no free lunch theorem for explanation methods*, showing that no single method can perform optimally across all neighbourhoods, and provided a *guiding principle* for choosing among methods.

The function approximation perspective captures the essence of an explanation – a simplification of the real world (i.e. a black box model) that is nonetheless accurate enough to be useful (i.e. predict outcomes of a set of perturbations). When the real world is "simple", an explanation should completely capture its behaviour, a hallmark expressed precisely by the guiding principle. When the requirements from two explanations are distinct (i.e. they are trained to predict different sets of perturbations), then such explanations disagree and yet are accurate in their own domains, a principle precisely expressed by the no free lunch theorem.

Our work makes several fundamental contributions. We *unify* popular explanation methods, bringing diverse methods into a common framework for the first time. This brings *conceptual coherence and clarity*: diverse explanation methods, even those seemingly unrelated to function approximation, perform LFA but differ in the way they perform it. This also enables *theoretical simplicity*: to study diverse explanation methods, instead of analyzing each method individually, one can simply analyze the LFA framework and apply the findings to each method. An example of this is the no free lunch theorem which holds true for all instances of the LFA framework. Furthermore, our work provides *practical guidance* by offering a principled approach to select among methods and design new ones.

Our work also addresses key open questions in the field. In response to criticism about the lack of agreement in the field regarding the overall goals of post hoc explainability [40], our work points to function approximation as a principled goal. It also addresses the disagreement problem [11] and explains why different methods generate different explanations for the same model prediction. According to the LFA framework, this disagreement occurs because different methods approximate the black box model over different neighborhoods using different loss functions.

To build on this work, future research could consider the following directions. First, we analyzed eight popular post hoc explanation methods and this analysis could be extended to other methods. Second, our work focuses on the faithfulness rather than interpretability of explanations. The latter is encapsulated in the "interpretable" model class $\mathcal{G}$, which includes all the information about human preferences with regards to interpretability. However, it is unclear what constitutes an interpretable explanation and elucidating this takes not only conceptual understanding but also human-computer interaction research such as user studies. These are important directions for future research.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful feedback and the funding agencies listed below for supporting this work. This work is supported in part by the NSF awards #IIS-2008461 and #IIS-2040989, and research awards from Google, JP Morgan, Amazon, Harvard Data Science Initiative, and Dˆ3 Institute at Harvard. H.L. would like to thank Sujatha and Mohan Lakkaraju for their continued support and encouragement. T.H. is supported in part by an NSF GRFP fellowship. The views expressed here are those of the authors and do not reflect the official policy or position of the funding agencies.

References

[1] Kun-Hsing Yu, Andrew L Beam, and Isaac S Kohane. Artificial intelligence in healthcare. *Nature biomedical engineering*, 2(10):719–731, 2018.

[2] Robert Walters and Marko Novak. Artificial intelligence and law. In *Cyber Security, Artificial Intelligence, Data Protection & the Law*, pages 39–69. Springer, 2021.

[3] Longbing Cao. Ai in finance: Challenges, techniques, and opportunities. *ACM Computing Surveys (CSUR)*, 55(3):1–38, 2022.

[4] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[5] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 2017.

[6] Matthew D. Zeiler and Robert Fergus. Visualizing and understanding convolutional networks. *arXiv preprint arXiv:1311.2901*, 2013.

[7] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

[8] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[9] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: Removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[10] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328, 2017.

[11] Satyapriya Krishna*, Tessa Han*, Alex Gu, Javin Pombra, Shahin Jabbari, Steven Wu, and Himabindu Lakkaraju. The disagreement problem in explainable machine learning: A practitioner's perspective. *arXiv preprint arXiv:2202.01602*, 2022.

[12] Sushant Agarwal, Shahin Jabbari, Chirag Agarwal, Sohini Upadhyay, Steven Wu, and Himabindu Lakkaraju. Towards the unification and robustness of perturbation and gradient based explanations. In *International Conference on Machine Learning*, pages 110–119, 2021.

[13] Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

[14] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018.

[15] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31:841, 2017.

[16] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *ACM Conference on Fairness, Accountability, and Transparency*, pages 10–19, 2019.

[17] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions, 2019.

[18] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. FACE: Feasible and actionable counterfactual explanations. In *AAAI/ACM Conference on AI, Ethics, and Society*, page 344–350, 2020.

[19] Arnaud Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *CoRR, abs/ 1907.02584*, 2019.

[20] Solon Barocas, Andrew Selbst, and Manish Raghavan. The hidden assumptions behind counterfactual explanations and principal reasons. In *ACM Conference on Fairness, Accountability, and Transparency*, pages 80–89, 2020.

[21] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse: from counterfactual explanations to interventions. *CoRR, abs/2002.06278*, 2020.

[22] Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. *CoRR, abs/2006.06831*, 2020.

[23] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Faithful and customizable explanations of black box models. In *AAAI Conference on Artificial Intelligence, Ethics, and Society*, pages 131–138, 2019.

[24] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpretability via model extraction. *CoRR, abs/1706.09773*, 2017.

[25] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning*, 2018.

[26] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.

[27] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. How can we fool LIME and SHAP? adversarial attacks on post hoc explanation methods. In *AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[28] Ann-Kathrin Dombrowski, Maximilian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *CoRR, abs/1906.07983*, 2019.

[29] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9505–9515, 2018.

[30] David Alvarez-Melis and Tommi Jaakkola. On the robustness of interpretability methods. *CoRR*, abs/1806.08049, 2018.

[31] Suraj Srinivas and François Fleuret. Knowledge transfer with jacobian matching. In *International Conference on Machine Learning*, pages 4723–4731, 2018.

[32] Alexander Levine, Sahil Singla, and Soheil Feizi. Certifiably robust interpretation in deep learning. *CoRR, abs/1905.12105*, 2019.

[33] Prasad Chalasani, Jiefeng Chen, Amrita Roy Chowdhury, Xi Wu, and Somesh Jha. Concise explanations of neural networks using adversarial training. In *International Conference on Machine Learning*, pages 1383–1391, 2020.

[34] Damien Garreau and Ulrike von Luxburg. Looking deeper into LIME. *CoRR*, abs/2008.11092, 2020.

[35] David S Watson, Limor Gultchin, Ankur Taly, and Luciano Floridi. Local explanations via necessity and sufficiency: Unifying theory and practice. In *Uncertainty in Artificial Intelligence*, pages 1382–1392. PMLR, 2021.

[36] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. URL `http://jmlr.org/papers/v6/hyvarinen05a.html`.

[37] World Health Organization (WHO). Life expectancy dataset. *Global Health Observatory Data Repository*, 2018.

[38] FICO. Home equity line of credit (heloc) dataset. *Explainable Machine Learning Challenge*, 2019.

[39] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.

[40] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[41] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2015.

[42] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *arXiv preprint arXiv:1505.04366*, 2015.

[43] Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, pages 3809–3818. PMLR, 2018.

[44] Suraj Srinivas and Francois Fleuret. Rethinking the role of gradient-based attribution methods for model interpretability. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=dYeAHXnpWJ4`.

[45] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[46] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[47] Suraj Srinivas and François Fleuret. Full-gradient representation for neural network visualization. *Advances in neural information processing systems*, 32, 2019.

[48] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract, Section §1.
   (b) Did you describe the limitations of your work? [Yes] See Section §6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section §6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sections §3, §4, and Appendix.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Sections §3, §4, and Appendix.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We plan to include a zip file with the code in the supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figure 3 and Appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We do not directly obtain data from individuals.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] To our knowledge, the data contains no such information nor content. See Appendix.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not use crowdsourcing nor did we conduct research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We did not use crowdsourcing nor did we conduct research with human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We did not use crowdsourcing nor did we conduct research with human subjects.

A Appendix

A.1 Proofs: Existing Methods are Instances of the LFA Framework (Section 3)

A.1.1 LIME

The instance of the LFA framework with (1) interpretable model class $\mathcal{G}$ as the class of linear models where $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, (2) perturbations of the form $\mathbf{x}_\xi = \mathbf{x}_0 \odot \xi$ where $\xi \, (\in \{0,1\}^d) \sim \pi_{\mathbf{x}_0}$ with $\pi_{\mathbf{x}_0}$ being the exponential kernel (defined below), and (3) loss function as squared-error loss given by $\ell(f, g, \mathbf{x}_0, \xi) = (f(\mathbf{x}_\xi) - g(\xi))^2$ is equivalent to `LIME`.

As defined in [4] (Section 3.4), the exponential kernel $\pi_{\mathbf{x}_0}(\xi) \propto exp\{-\frac{D(\mathbf{x}_0, \mathbf{x}_\xi)}{\sigma^2}\}$ with distance function D (such as cosine distance or L2 distance) and width σ.

Proof. For this instance of the LFA framework, by definition, the interpretable model g is given by:

$$\begin{aligned} g^* &= \arg\min_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{\xi \sim \pi_{\mathbf{x}_0}} \ell(f, g, \mathbf{x}_0, \xi) \\ &= \arg\min_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{\xi \sim p} [\ell(f, g, \mathbf{x}_0, \xi) \cdot \pi_{\mathbf{x}_0}(\xi)] \text{ where } p \text{ is the Bernouilli(0.5) distribution} \end{aligned}$$

Through importance sampling using a Bernouilli(0.5) proposal distribution (i.e. a Uniform(0,1) distribution over the space of binary inputs), the optimization setting of the LFA framework is that described for `LIME` by Ribeiro et al. [4] (Equations 1 and 2). □

A.1.2 KernelSHAP

The instance of the LFA framework with (1) interpretable model class $\mathcal{G}$ as the class of linear models where $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, (2) perturbations of the form $\mathbf{x}_\xi = \mathbf{x}_0 \odot \xi$ where $\xi \, (\in \{0,1\}^d) \sim \pi$ with π being the Shapley kernel (defined below), and (3) loss function as squared-error loss given by $\ell(f, g, \mathbf{x}_0, \xi) = (f(\mathbf{x}_\xi) - g(\xi))^2$ is equivalent to `KernelSHAP`.

As defined in [5] (Theorem 2), the Shapley kernel $\pi(\xi) \propto \frac{M-1}{\binom{M}{k} \cdot k \cdot (M-k)}$ where M is the total number of elements in ξ and k is the number of ones in ξ.

Proof. For this instance of the LFA framework, by definition, the interpretable model g is given by:

$$\begin{aligned} g^* &= \arg\min_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{\xi \sim \pi} \ell(f, g, \mathbf{x}_0, \xi) \\ &= \arg\min_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{\xi \sim p} [\ell(f, g, \mathbf{x}_0, \xi) \cdot \pi(\xi)] \text{ where } p \text{ is the Bernouilli(0.5) distribution} \end{aligned}$$

Through importance sampling using a Bernouilli(0.5) proposal distribution (i.e. a Uniform(0,1) distribution over the space of binary inputs), the optimization setting of the LFA framework is that described for `KernelSHAP` by Lundberg and Lee [5] (Equation 2 and Theorem 2). □

A.1.3 Occlusion

The instance of the LFA framework with (1) interpretable model class $\mathcal{G}$ as the class of linear models where $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, (2) perturbations of the form $\mathbf{x}_\xi = \mathbf{x}_0 \odot \xi$ where $\xi \, (\in \{0,1\}^d)$ is a random one-hot vector, and (3) loss function as squared-error loss given by $\ell(f, g, \mathbf{x}_0, \xi) = (\Delta f - g(\xi))^2$ where $\Delta f = f(\mathbf{x}_0) - f(\mathbf{x}_0(1 - \xi))$ converges to `Occlusion`.

Proof. This instance of the LFA framework optimizes $g(\xi)$ to approximate Δf. For ξ_i (a one-hot vector with element i equal to 1), $g(\xi_i) = w_i$ and Δf_i is the difference in the model prediction when feature i takes the original value versus when feature i is set to zero. Δf is the definition of explanations generated by `Occlusion`. Thus, in this instance of the LFA framework, the weights of g recover the explanations of `Occlusion`. □

A.1.4 C-LIME

The instance of the LFA framework with (1) interpretable model class $\mathcal{G}$ as the class of linear models where $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, (2) perturbations of the form $\mathbf{x}_\xi = \mathbf{x}_0 + \xi$ where $\xi\ (\in \mathbb{R}^d) \sim \text{Normal}(0, \sigma^2)$, and (3) loss function as squared-error loss given by $\ell(f, g, \mathbf{x}_0, \xi) = (f(\mathbf{x}_\xi) - g(\xi))^2$ is equivalent to `C-LIME`.

Proof. This instance of the LFA framework is equivalent to `C-LIME` by definition of `C-LIME`. □

A.1.5 SmoothGrad

In this section, we provide two derivations showing the connection between the LFA framework and `SmoothGrad`. When using gradient-matching loss, the instance of the LFA framework is exactly equivalent to `SmoothGrad` given the same n perturbations. When using squared-error loss, the instance of the LFA framework is equivalent to `SmoothGrad` asymptotically for a large number of perturbations.

Gradient-matching loss function

This instance of the LFA framework with (1) interpretable model class $\mathcal{G}$ as the class of linear models where $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, (2) perturbations of the form $\mathbf{x}_\xi = \mathbf{x}_0 + \xi$ where $\xi\ (\in \mathbb{R}^d) \sim \text{Normal}(0, \sigma^2)$, and (3) loss function as gradient-matching loss given by $\ell_{gm}(f, g, \mathbf{x}_0, \xi) = \|\nabla_\xi f(\mathbf{x}_\xi) - \nabla_\xi g(\xi)\|_2^2$ is equivalent to `SmoothGrad`. In other words, for the same n perturbations, this instance of the LFA framework and `SmoothGrad` yield the same explanation.

Proof. For this instance of the LFA framework, by definition, the interpretable model g is given by $g^* = \arg\min_{g \in \mathcal{G}} L$ where:

$$
\begin{aligned}
L &= \mathbb{E}_\xi \ell(f, g, \mathbf{x}_0, \xi) \\
&= \frac{1}{n} \sum_n \|\nabla_\xi f(\mathbf{x}_\xi) - \nabla_\xi g(\xi)\|_2^2 \\
&= \frac{1}{n} \sum_n \|\nabla_{\mathbf{x}_0} f(\mathbf{x}_\xi) - \mathbf{w}\|_2^2
\end{aligned}
$$

To derive the solution for $\mathbf{w}$, take the partial derivative of L w.r.t. to $\mathbf{w}$, set the partial derivative to zero, and solve for $\mathbf{w}$.

$$
\nabla_{\mathbf{w}} L = 0 \Rightarrow (-2)\frac{1}{n} \sum_n [\nabla_{\mathbf{x}_0} f(\mathbf{x}_\xi) - \mathbf{w}] = 0 \Rightarrow \mathbf{w} = \frac{1}{n} \sum_n \nabla_{\mathbf{x}_0} f(\mathbf{x}_\xi)
$$

Therefore, for the same n perturbations, the weights $\mathbf{w}$ of the interpretable model g are equivalent to the `SmoothGrad` explanations. □

Squared-error loss function

Consider the instance of the LFA framework corresponding to `SmoothGrad` described above, except with loss function as squared-error loss given by $\ell(f, g, \mathbf{x}_0, \xi) = (f(\mathbf{x}_\xi) - g(\xi))^2$. This instance of the LFA framework converges to `SmoothGrad` in expectation. Note that this instance of the LFA framework is `C-LIME` and its convergence to `SmoothGrad` in expectation is consistent with the results of Agarwal et al. [12] who previously proved the same convergence (using a different approach).

Proof. For this instance of the LFA framework, by definition, the interpretable model g is given by $g^* = \arg\min_{g \in \mathcal{G}} L$ where:

$$
\begin{aligned}
L &= \mathbb{E}_\xi \ell(f, g, \mathbf{x}_0, \xi) \\
&= \mathbb{E}_\xi [(f(\mathbf{x}_\xi) - g(\xi))^2] \\
&= \mathbb{E}_\xi [(f(\mathbf{x}_\xi) - \mathbf{w}^\top \xi)^2]
\end{aligned}
$$

To derive the solution for $\mathbf{w}$, take the partial derivative of L w.r.t. to $\mathbf{w}$, set the partial derivative to zero, and solve for $\mathbf{w}$.

$$
\begin{aligned}
\nabla_{\mathbf{w}} L &= 0 \\
-2\mathbb{E}_{\xi}[(f(\mathbf{x}_{\xi}) - \mathbf{w}^{\top}\xi)\xi^{\top}] &= 0 \\
\mathbb{E}_{\xi}[f(\mathbf{x}_{\xi})\xi^{\top} - \mathbf{w}^{\top}\xi\xi^{\top}] &= 0 \\
\mathbb{E}_{\xi}[f(\mathbf{x}_{\xi})\xi^{\top}] - \mathbf{w}^{\top}\mathbb{E}[\xi\xi^{\top}] &= 0 \\
\sigma^2\mathbb{E}_{\xi}[\nabla_{\mathbf{x}_{\xi}} f(\mathbf{x}_{\xi})^{\top}] - \sigma^2\mathbf{w}^{\top} &= 0 \text{ by Stein's Lemma} \\
\sigma^2\mathbb{E}_{\xi}[\nabla_{\mathbf{x}_0} f(\mathbf{x}_{\xi})^{\top}] - \sigma^2\mathbf{w}^{\top} &= 0 \\
\mathbf{w} &= \mathbb{E}_{\xi}[\nabla_{\mathbf{x}_0} f(\mathbf{x}_{\xi})]
\end{aligned}
$$

Therefore, the weights $\mathbf{w}$ of the interpretable model g converge to `SmoothGrad` explanations in expectation. □

A.1.6 Vanilla gradients

Consider the instance of the LFA framework corresponding to `SmoothGrad` described above (with loss function as either squared-error loss or gradient-matching loss). As $\sigma \to 0$, this instance of the LFA framework converges to `Vanilla Gradients`.

Proof. Starting with the solution for $\mathbf{w}$ derived for `SmoothGrad`, take the limit of $\mathbf{w}$ as $\sigma \to 0^+$.

$$
\begin{aligned}
\lim_{\sigma \to 0^+} \mathbf{w} &= \lim_{\sigma \to 0^+} \mathbb{E}_{\xi}[\nabla_{\mathbf{x}_0} f(\mathbf{x}_{\xi})] \\
&= \lim_{\sigma \to 0^+} \int_{-\infty}^{\infty} \nabla_{\mathbf{x}_0} f(\mathbf{x}_{\xi})\, p(\xi; 0, \sigma)\, d\xi \\
&= \lim_{\sigma \to 0^+} \int_{-\infty}^{\infty} \nabla_{\mathbf{x}_0} f(\mathbf{x}_0 + \xi)\, \eta_{\xi}(\xi)\, d\xi \text{ where } \eta_{\xi}(\xi) = p(\xi; 0, \sigma) \\
&= \nabla_{\mathbf{x}_0} f(\mathbf{x}_0) \text{ by property of the Dirac delta distribution}
\end{aligned}
$$

To derive the third line from the second line, we view the Normal density function $p(\xi; 0, \sigma)$ as a nascent delta function $\eta_{\xi}(\xi)$ (which is defined such that $\lim_{\sigma \to 0^+} \int_{-\infty}^{\infty} p(\xi; 0, \sigma) = \delta(\xi)$, where δ is the Dirac delta distribution) and by assuming that $\nabla_{\mathbf{x}_0} f(\mathbf{x}_{\xi})$ has a compact support.

Therefore, the weights $\mathbf{w}$ of the interpretable model g converge to `Vanilla Gradients` explanations. □

A.1.7 Integrated Gradients

This instance of the LFA framework with (1) interpretable model class $\mathcal{G}$ as the class of linear models where $g(\mathbf{x}) = \mathbf{w}^{\top}\mathbf{x}$, (2) perturbations of the form $\mathbf{x}_{\xi} = \mathbf{x}_0 \odot \xi$ where $\xi\ (\in \mathbb{R}^d) \sim \text{Uniform}(0, 1)$, and (3) loss function as gradient-matching loss given by $\ell_{gm}(f, g, \mathbf{x}_0, \xi) = \|\nabla_{\xi} f(\mathbf{x}_{\xi}) - \nabla_{\xi} g(\xi)\|_2^2$ is equivalent to `Integrated Gradients`. In other words, for the same n perturbations, this instance of the LFA framework and `Integrated Gradients` yield the same explanation.

Proof. For this instance of the LFA framework, by definition, the interpretable model g is given by $g^* = \arg\min_{g \in \mathcal{G}} L$ where:

$$
\begin{aligned}
L &= \mathbb{E}_{\xi}\ell(f, g, \mathbf{x}_0, \xi) \\
&= \mathbb{E}_{\xi}\|\nabla_{\xi} f(\mathbf{x}_{\xi}) - \nabla_{\xi} g(\xi)\|_2^2 \\
&= \mathbb{E}_{\xi}\|\nabla_{\mathbf{x}_0} f(\mathbf{x}_{\xi}) \odot \mathbf{x}_0 - \mathbf{w}\|_2^2
\end{aligned}
$$

Note that, by the chain rule, $\nabla_{\xi} f(\mathbf{x}_{\xi}) = \nabla_{\xi} f(\mathbf{x}_0 \odot \xi) = \nabla_{\mathbf{x}_{\xi}} f(\mathbf{x}_{\xi}) \odot \nabla_{\xi}\mathbf{x}_{\xi} = \nabla_{\mathbf{x}_0} f(\mathbf{x}_{\xi}) \odot \mathbf{x}_0$.

To derive the solution for $\mathbf{w}$, take the partial derivative of L w.r.t. to $\mathbf{w}$, set the partial derivative to zero, and solve for $\mathbf{w}$.

$$
\begin{aligned}
\nabla_{\mathbf{w}} L &= 0 \\
-2\mathbb{E}_{\xi}[\nabla_{\mathbf{x}_0} f(\mathbf{x}_{\xi}) \odot \mathbf{x}_0 - \mathbf{w}] &= 0 \\
\mathbf{w} &= \mathbf{x}_0 \odot \mathbb{E}_{\xi}[\nabla_{\mathbf{x}_0} f(\mathbf{x}_{\xi})]
\end{aligned}
$$

Therefore, the weights $\mathbf{w}$ of the interpretable model g converge to `Integrated Gradients` explanations in expectation. □

A.1.8 Gradient × Input

Consider the instance of the LFA framework corresponding to `Integrated Gradients` described above, except with $\xi\,(\in \mathbb{R}^d) \sim \text{Uniform}(a, 1)$. As $a \to 1$, this instance of the LFA framework converges to `Gradient x Input`.

Proof. As $a \to 1$, $\xi \to \vec{1}$, and $\mathbf{w} \to \mathbf{x}_0 \odot \nabla_{\mathbf{x}_0} f(\mathbf{x}_0)$. Therefore, the weights $\mathbf{w}$ of the interpretable model g converge to `Gradient x Input` explanations. □

A.2 Which Explanations Are Not Function Approximators?

In this section, we briefly discuss explanation methods that cannot be viewed as instances of the LFA framework. In the cases listed below, the lack of connection to the LFA framework is mainly due to a property of the explanation method.

Model-independent methods: Some explanation methods are known to produce attributions that are independent from the model they intend to explain. These methods cannot be cast in the LFA framework in any meaningful way due to the model recovery conditions we impose. Such model-independent methods include guided backpropagation [41] and DeconvNet [42], following theory by Nie et al. [43], as well as logit-gradient based methods [44] such as Grad-CAM [45], Grad-CAM++ [46], and FullGrad [47].

Modified-backpropagation methods: Some explanation methods such as DeepLIFT [8], guided backpropagation [**?**], DeconvNet [42], and layer-wise relevance propagation [48] work by modifying the backpropagation equations and propagating attributions using finite-difference-like methods. Such methods break an important property called "implementation invariance", first identified by Sundararajan et al. [10], which states that two functionally identical models can have different attributions due to the lack of a chain rule for modified backpropagation methods. This property ensures that such methods cannot be function approximators, as the attribution changes based on the function implementation.

Unsigned-gradient methods: Some gradient-based methods return unsigned attribution values instead of the full signed values. Such methods can be written in the LFA framework using the following loss function $\ell(f, g, \mathbf{x}_0, \xi) = \||\nabla_\xi f(\mathbf{x}_0 \oplus \xi)| - \mathbf{w}_g\|^2$ where $\mathbf{w}_g$ consists of the weights of the interpretable model g. Using this loss function with different choices for neighborhoods gives unsigned versions of different gradient methods. However, this loss function is not a valid loss function, i.e., $\ell = 0 \implies f = g$. Using this loss function, $\mathbf{w}_g$ is always positive and thus cannot recover an underlying model's negative weights.

A.3 Proof: No Free Lunch Theorem ((Section 4))

Theorem 4. *Assume a black-box model f and an interpretable model class $\mathcal{G}$, and distance between them given by $d(f, \mathcal{G}) = \min_{g \in \mathcal{G}} \max_{\mathbf{x} \in \mathcal{X}} \ell(f, g, 0, \mathbf{x})$, and that we are required to explain f around input $\mathbf{x}_0$, and the loss ℓ is a valid distance metric.*

Then, for any explanation g^* on a neighborhood distribution $\xi_1 \sim \mathcal{Z}_1$ such that $\max_{\xi_1} \ell(f, g_1^*, \mathbf{x}_0, \xi_1) \leq \epsilon$, we can always find another neighborhood $\xi_2 \sim \mathcal{Z}_2$ such that $\max_{\xi_2} \ell(f, g_1^*, \mathbf{x}_0, \xi_2) \geq d(f, \mathcal{G})$.

Proof. Given an explanation g^*, we can find an "adversarial" input $\mathbf{x}_{adv}$ such that $\mathbf{x}_{adv} = \arg\max_{\mathbf{x} \in \mathcal{X}} \ell(f, g^*, 0, \mathbf{x})$ has a large error ℓ. Construct perturbation $\mathbf{x}_2 = \mathbf{x}_0 + \xi_2$ such that $p(\xi_2) = \text{Uniform}(0, \mathbf{x}_{adv} - \mathbf{x}_0)$, which implies $p(\mathbf{x}_2) = \text{Uniform}(\mathbf{x}_0, \mathbf{x}_{adv})$.

By definition $\max_{\xi_2} \ell(f, g^*, \mathbf{x}_0, \xi_2) = \ell(f, g^*, \mathbf{x}_0, \mathbf{x}_{adv} - \mathbf{x}_0) = \max_{\mathbf{x} \in \mathcal{X}} \ell(f, g^*, 0, \mathbf{x}) \geq \min_{g \in \mathcal{G}} \max_{\mathbf{x} \in \mathcal{X}} \ell(f, g, 0, \mathbf{x}) = d(f, \mathcal{G})$ □

A salient feature of this proof is that it makes no assumptions about the form of model, input or output domains. This implies that the result applies equally to discrete and continuous domains, regression and classification tasks, and for any model type.

A.4 Summary of Properties of Existing Explanation Methods

Method	**Characteristics of ξ**	**g recovers f?**	**Scale of g's weights when $\mathcal{X} \in \mathbb{R}^d$**
C-LIME	Continuous, Additive	When $\mathcal{X} \in \mathbb{R}^d$	Gradient
SmoothGrad	Continuous, Additive	When $\mathcal{X} \in \mathbb{R}^d$	Gradient
Vanilla Gradients	Continuous, Additive	When $\mathcal{X} \in \mathbb{R}^d$	Gradient
Integrated Gradients	Continuous, Multiplicative	No	Gradient $\times$ Input
Gradients $\times$ Input	Continuous, Multiplicative	No	Gradient $\times$ Input
LIME	Binary, Multiplicative	When $\mathcal{X} \in \{0,1\}^d$	Gradient $\times$ Input
KernelSHAP	Binary, Multiplicative	When $\mathcal{X} \in \{0,1\}^d$	Gradient $\times$ Input
Occlusion	Binary, Multiplicative	When $\mathcal{X} \in \{0,1\}^d$	Gradient $\times$ Input

Table 2: Summary of properties of existing explanation methods in relation to the LFA framework. In this table, we consider the scale of g's weights when $\mathcal{X} \in \mathbb{R}^d$.

A.5 Setup of Experiments

Datasets. The first dataset is the life expectancy dataset from the Global Health Observatory data repository of the World Health Organization (WHO) [37]. The WHO dataset consists of demographic, economic, and health factors of 193 countries from 2000 to 2015 such as a country's population, gross domestic product, health expenditure, human development index, infant mortality rate, hepatitis B immunization rate, and life expectancy. The other dataset is the home equity line of credit (HELOC) dataset from the Explainable Machine Learning Challenge organized by FICO [38]. The HELOC dataset contains information on HELOC applications made by homeowners, such as an applicant's installment balance, number of trades, longest delinquency period, and risk category (whether an applicant made payments without being 90 days overdue). To our knowledge, these datasets do not contain personally identifiable information nor offensive content.

For the WHO dataset, missing values were imputed using kNN imputation with $k = 5$. For the HELOC dataset, missing values were dropped. For both datasets, continuous features were mean-centered and then normalized to [0, 1] range.

Models. For the WHO dataset, we train four models: a linear regression model (train MSE: 9.39×10^{-5}; test MSE: 9.80×10^{-5}) and three feed-forward neural networks. The neural networks have 8-node hidden layers with tanh activation and a linear output layer. The first neural network has 3 hidden layers (train MSE: 7.83×10^{-5}; test MSE: 8.23×10^{-5}), the second has 5 hidden layers (train MSE: 7.76×10^{-5}; test MSE: 8.11×10^{-5}), and the third has 8 hidden layers (train MSE: 7.78×10^{-5}; test MSE: 8.20×10^{-5}). The neural networks will be referred to as NN1, NN2, and NN3, respectively.

For the HELOC dataset, we train four models: a logistic regression model (train accuracy: 0.73; test accuracy: 0.74) and three feed-forward neural networks. The neural networks have 8-node hidden layers with relu activation and an output layer with sigmoid activation. The first neural network has 3 hidden layers (train accuracy: 0.75; test accuracy: 0.75), the second has 5 hidden layers (train accuracy: 0.75; test accuracy: 0.75), and the third has 8 hidden layers (train accuracy: 0.75; test accuracy: 0.75). The neural networks will be referred to as NNA, NNB, and NNC, respectively.

Models were trained based on an 80/20 train/test split using stochastic gradient descent. Hyperparameters were selected to reach decent model performance. The emphasis is on generating explanations for individual model predictions, not on high model performance. Thus, we do not focus on tuning model hyperparameters. Linear and logistic regression models trained for 100 epochs while neural network models trained for 300 epochs. All models used a batch size of 64 and a cosine annealing scheduler for the learning rate. Hyperparameters for all models are included in the code accompanying this paper.

Explanation Methods. Each explanation method is implemented using (1) the existing method and (2) the LFA framework. For (1), we used Meta's Captum library [39]. When using Captum, methods with number of perturbations as a parameter (i.e. LIME, KernelSHAP, SmoothGrad, and Integrated Gradients) used 1000 perturbations, a number of perturbations at which explanations for the method converged. For (2), we implemented the LFA framework, instantiating each method based on Table 1. For each method, the number of perturbations is set to 1000 for the same reason above. The interpretable model g is optimized using stochastic gradient descent. The perturbations are split into a train and test set (80/20 split) and g^* is optimized based on test set performance.

Analyses were performed on GPUs. The total amount of compute is approximately 54 GPU-hours.

A.6 Full Results for Experiments

A.6.1 Experiment 1: Existing Methods Are Instances of the LFA Framework

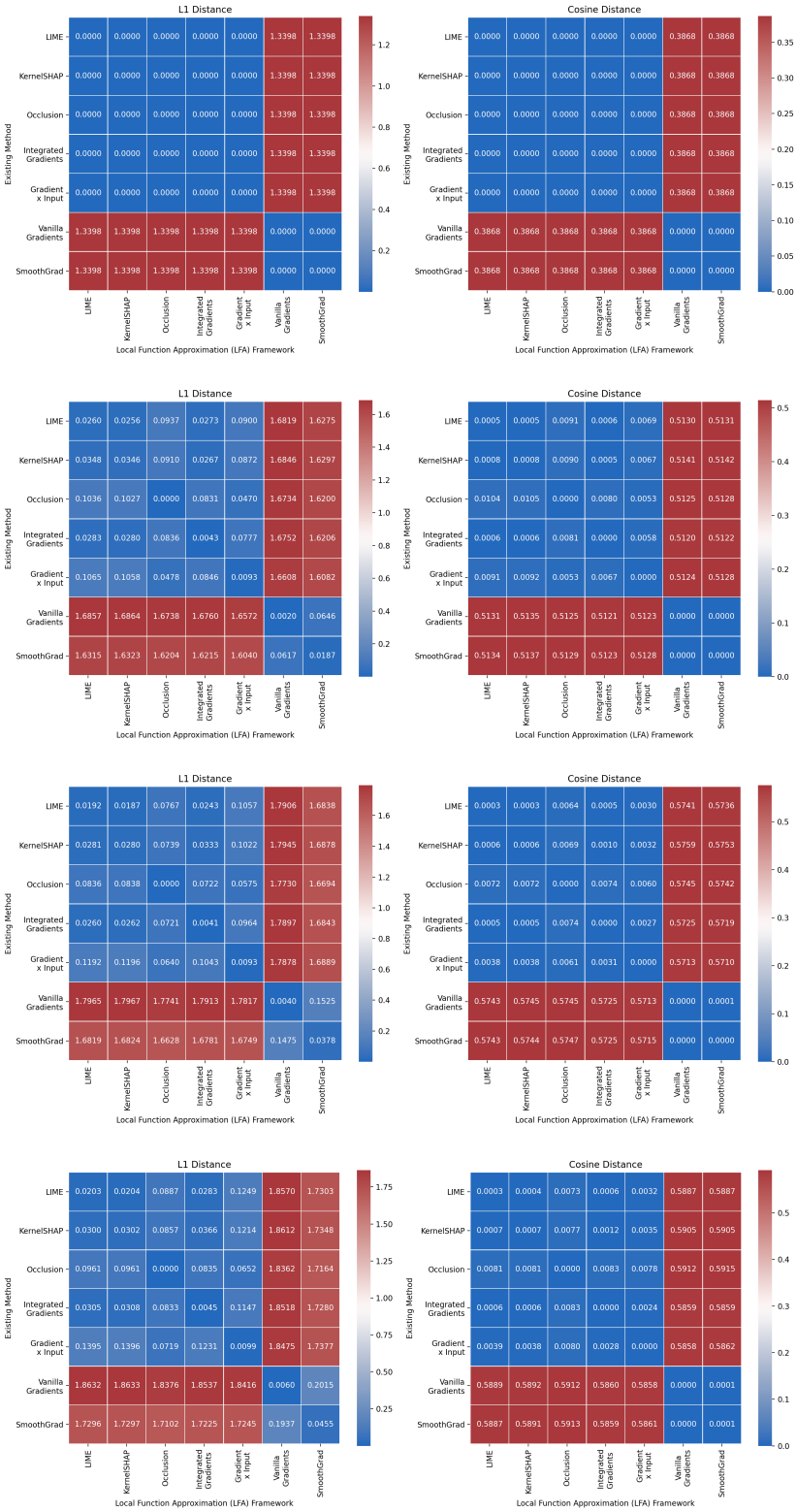


Figure 4: Correspondence of existing methods to instances of the LFA framework. Experiments performed on the WHO dataset for linear regression (Row 1), NN1 (Row 2), NN2 (Row 3), and NN3 (Row 4). The similarity of pairs of explanations are measured based on L1 norm (left column) and cosine distance (right column).

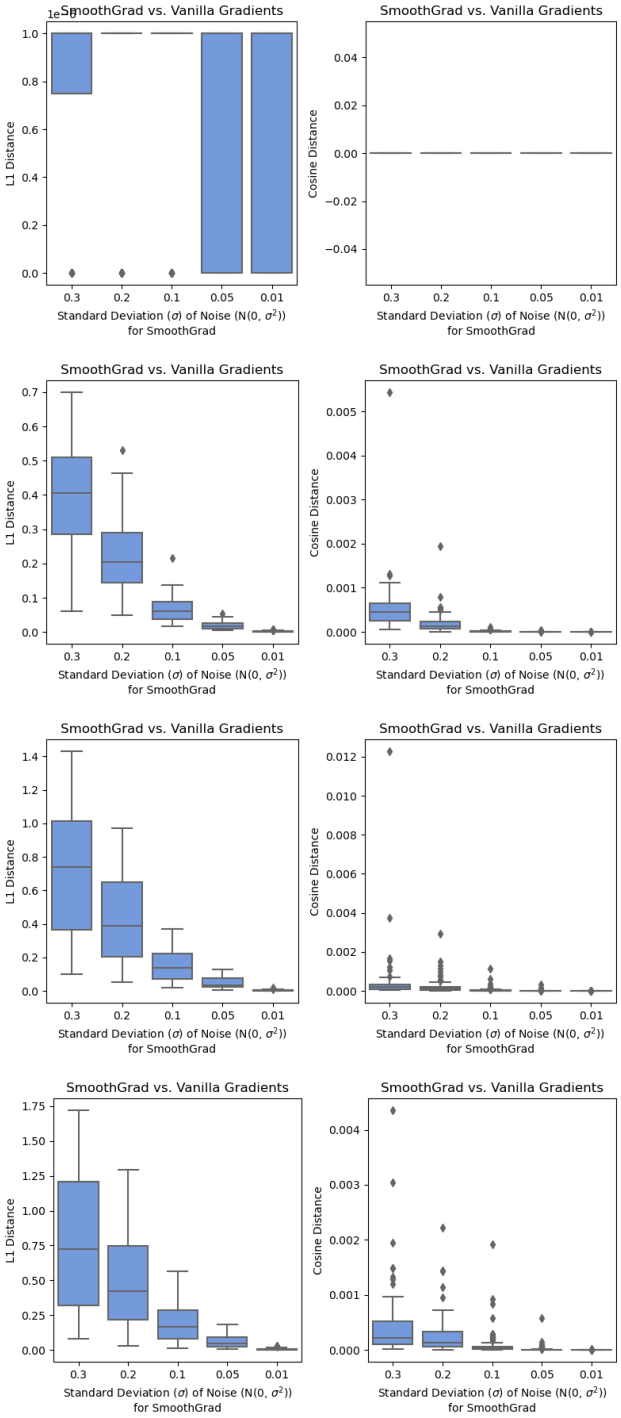


Figure 5: Using the LFA framework, explanations generated by SmoothGrad converge to those generated by Vanilla Gradients. Experiments performed on the WHO dataset for linear regression (Row 1), NN1 (Row 2), NN2 (Row 3), and NN3 (Row 4). The similarity of pairs of explanations are measured based on L1 norm (left column) and cosine distance (right column).

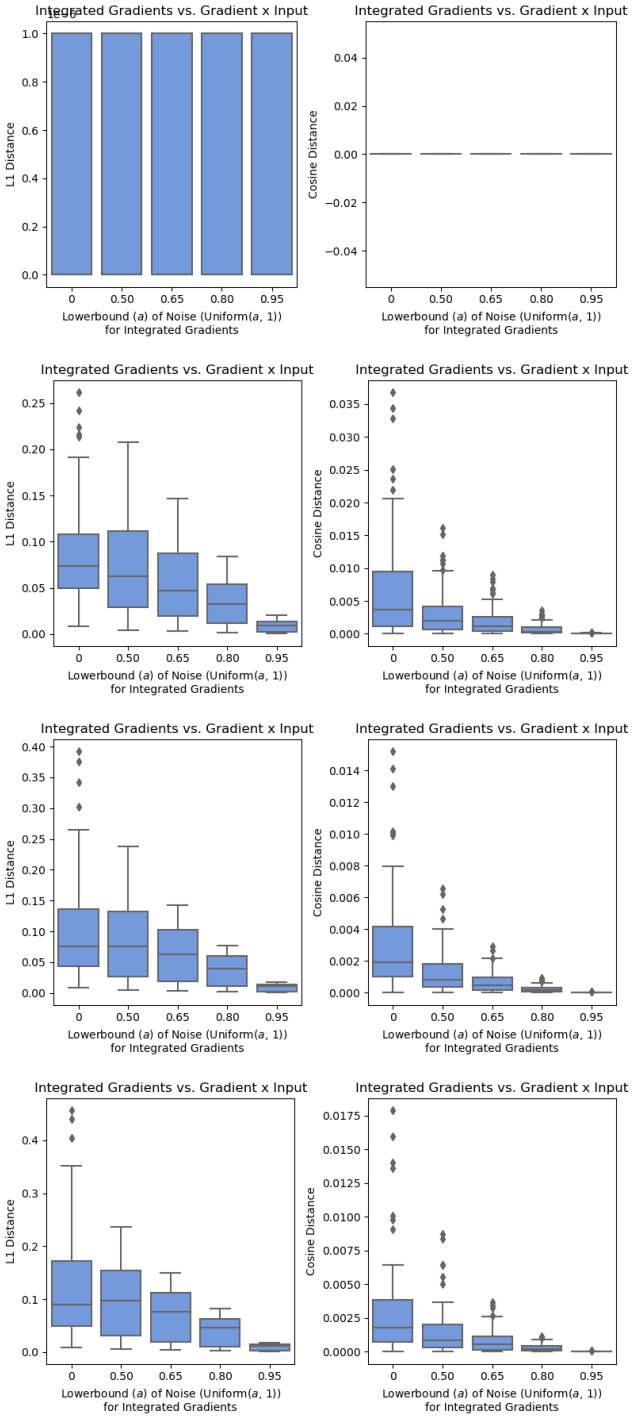


Figure 6: Using the LFA framework, explanations generated by Integrated Gradients converge to those generated by Gradient $\times$ Input. Experiments performed on the WHO dataset for linear regression (Row 1), NN1 (Row 2), NN2 (Row 3), and NN3 (Row 4). The similarity of pairs of explanations are measured based on L1 norm (left column) and cosine distance (right column).

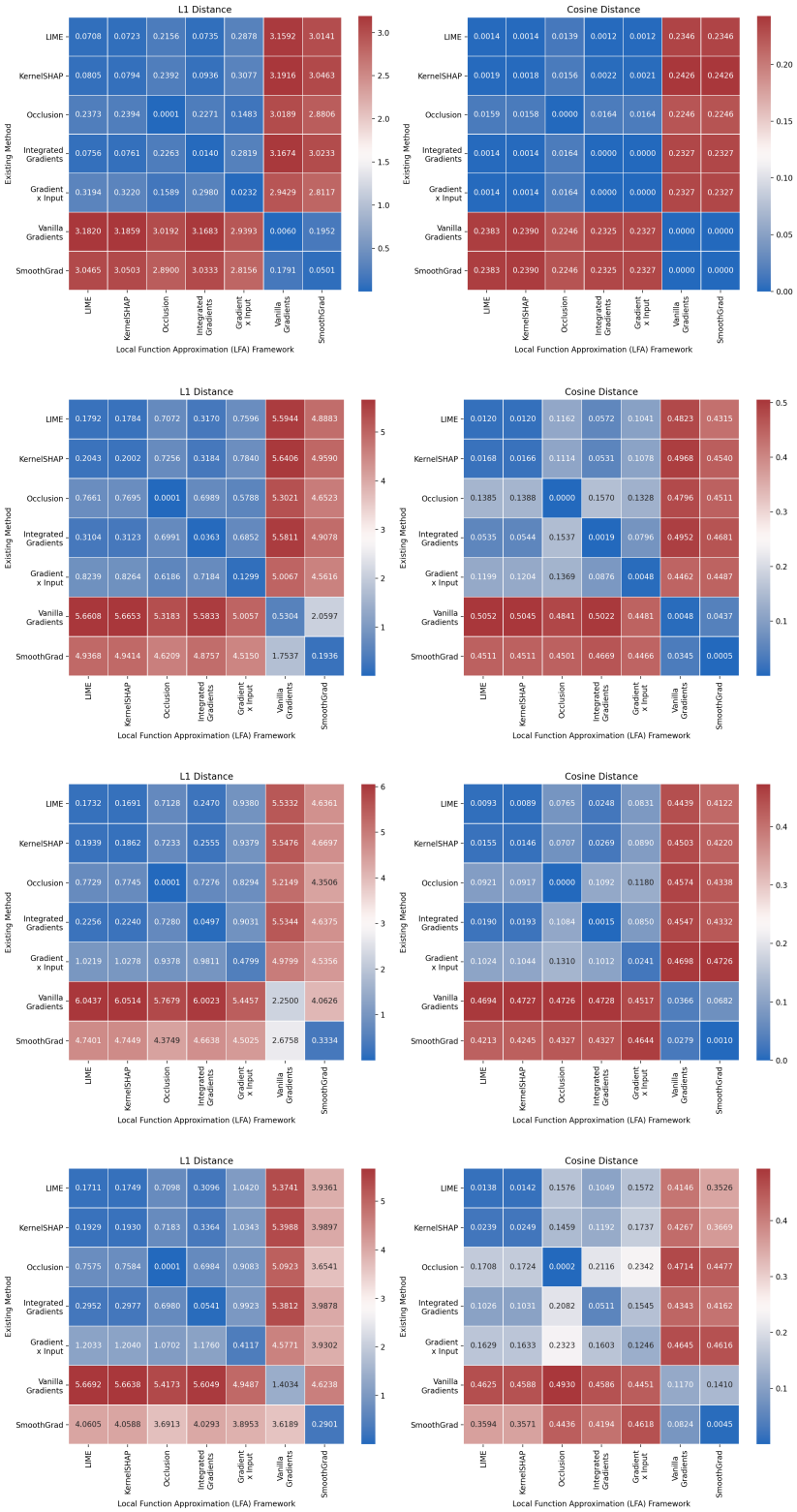


Figure 7: Correspondence of existing methods to instances of the LFA framework. Experiments performed on the HELOC dataset for logistic regression (Row 1), NNA (Row 2), NNB (Row 3), and NNC (Row 4). The similarity of pairs of explanations are measured based on L1 norm (left column) and cosine distance (right column).

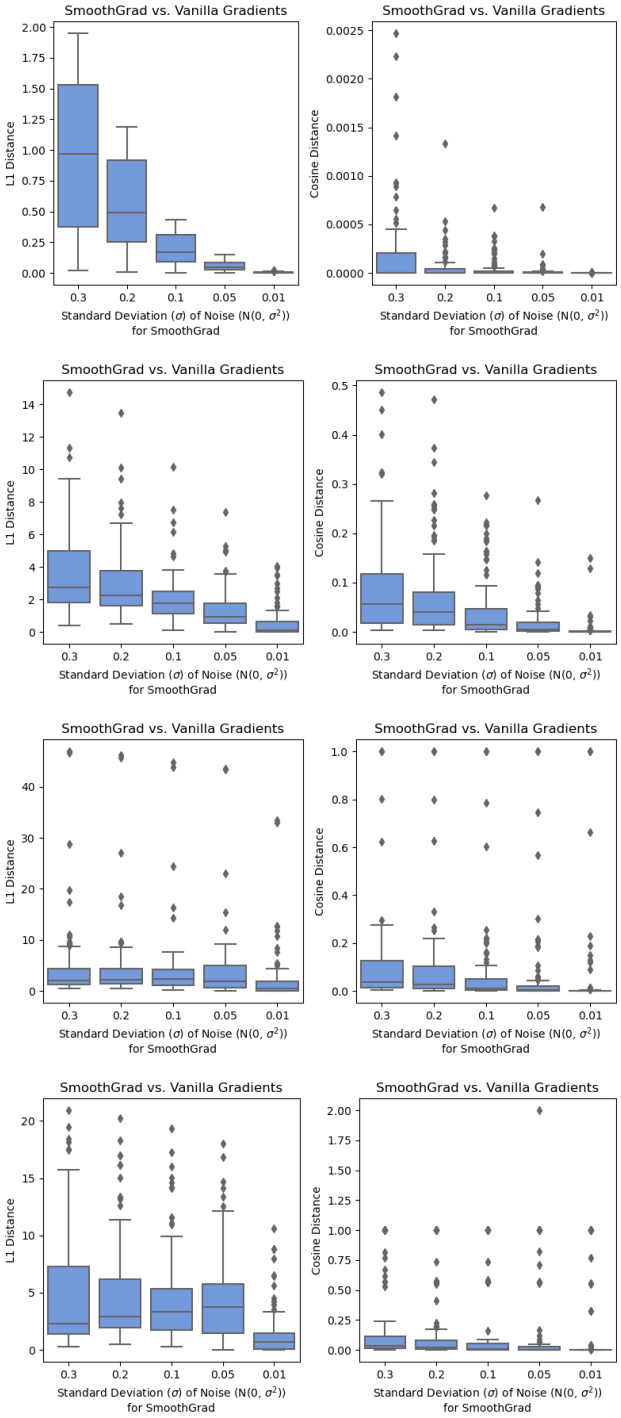


Figure 8: Using the LFA framework, explanations generated by SmoothGrad converge to those generated by Vanilla Gradients. Experiments performed on the HELOC dataset for logistic regression (Row 1), NNA (Row 2), NNB (Row 3), and NNC (Row 4). The similarity of pairs of explanations are measured based on L1 norm (left column) and cosine distance (right column).

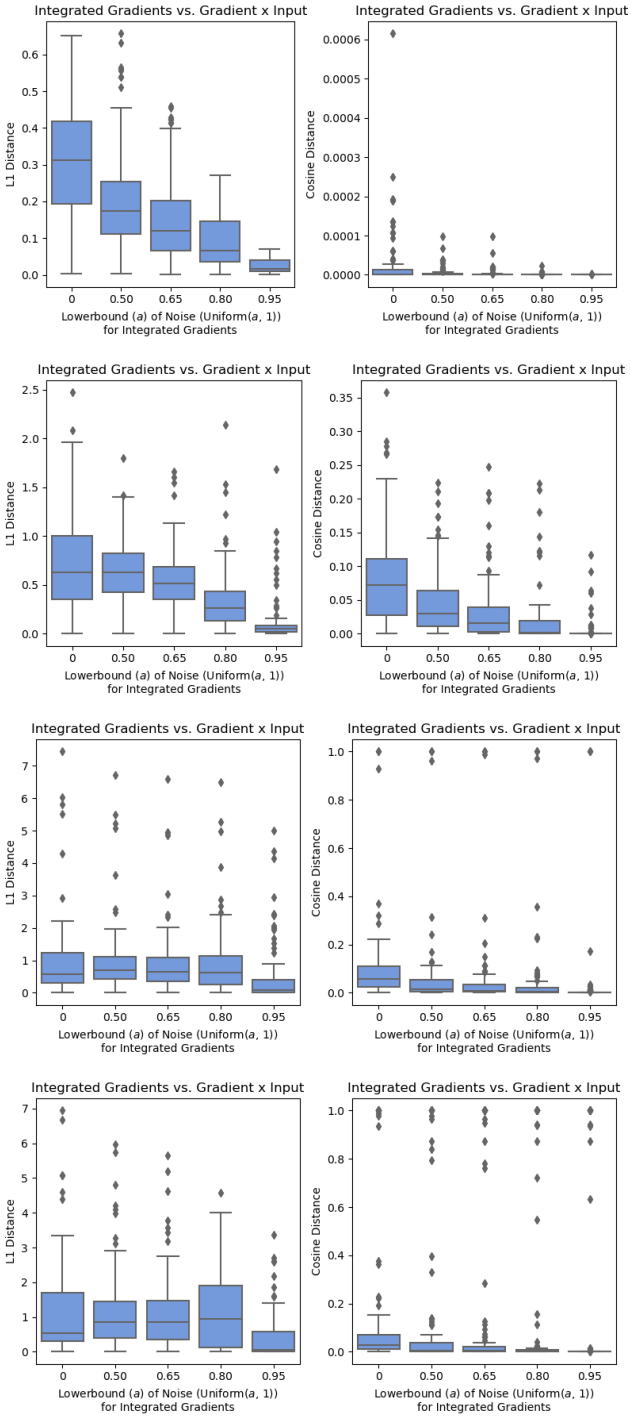


Figure 9: Using the LFA framework, explanations generated by Integrated Gradients converge to those generated by Gradient $\times$ Input. Experiments performed on the HELOC dataset for logistic regression (Row 1), NNA (Row 2), NNB (Row 3), and NNC (Row 4). The similarity of pairs of explanations are measured based on L1 norm (left column) and cosine distance (right column).

A.6.2 Experiment 2: g's recovery of f

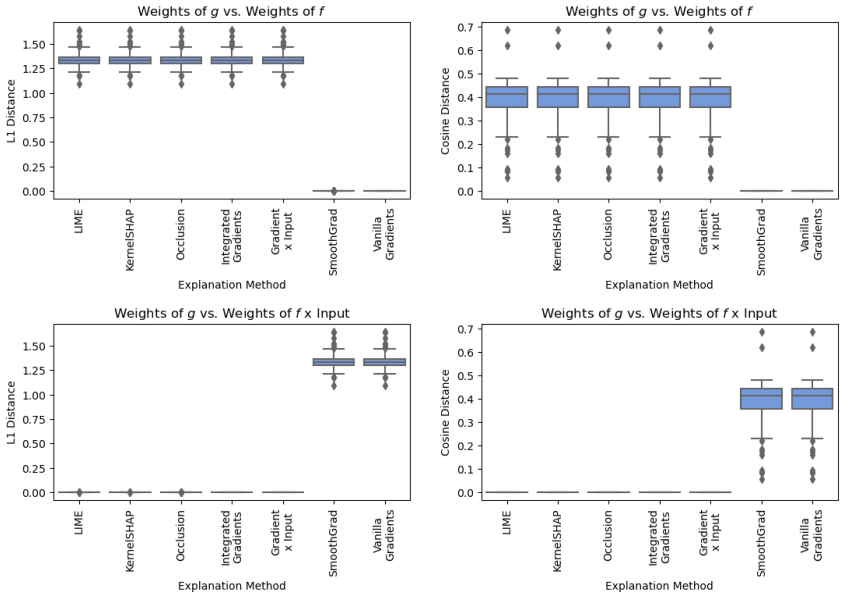


Figure 10: Analysis of g's recovery of f using a linear regression model trained on the WHO dataset. g's weights are compared with f's weights (top row) or f's weights multiplied by the input (bottom row) based on L1 norm (left column) or cosine distance (right column).

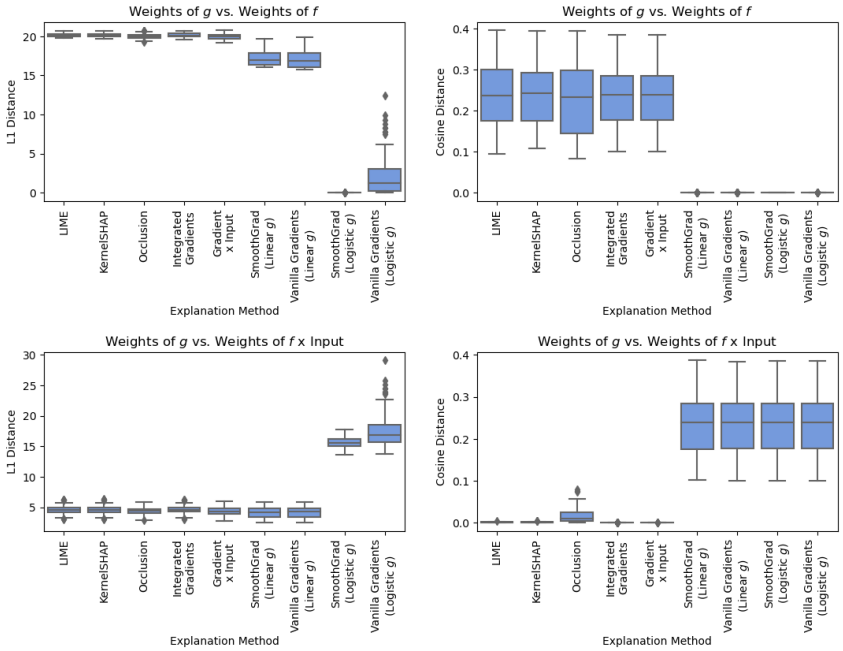


Figure 11: Analysis of g's recovery of f using a logistic regression model trained on the HELOC dataset. g's weights are compared with f's weights (top row) or f's weights multiplied by the input (bottom row) based on L1 norm (left column) or cosine distance (right column).

A.6.3 Experiment 3: Perturbation Tests

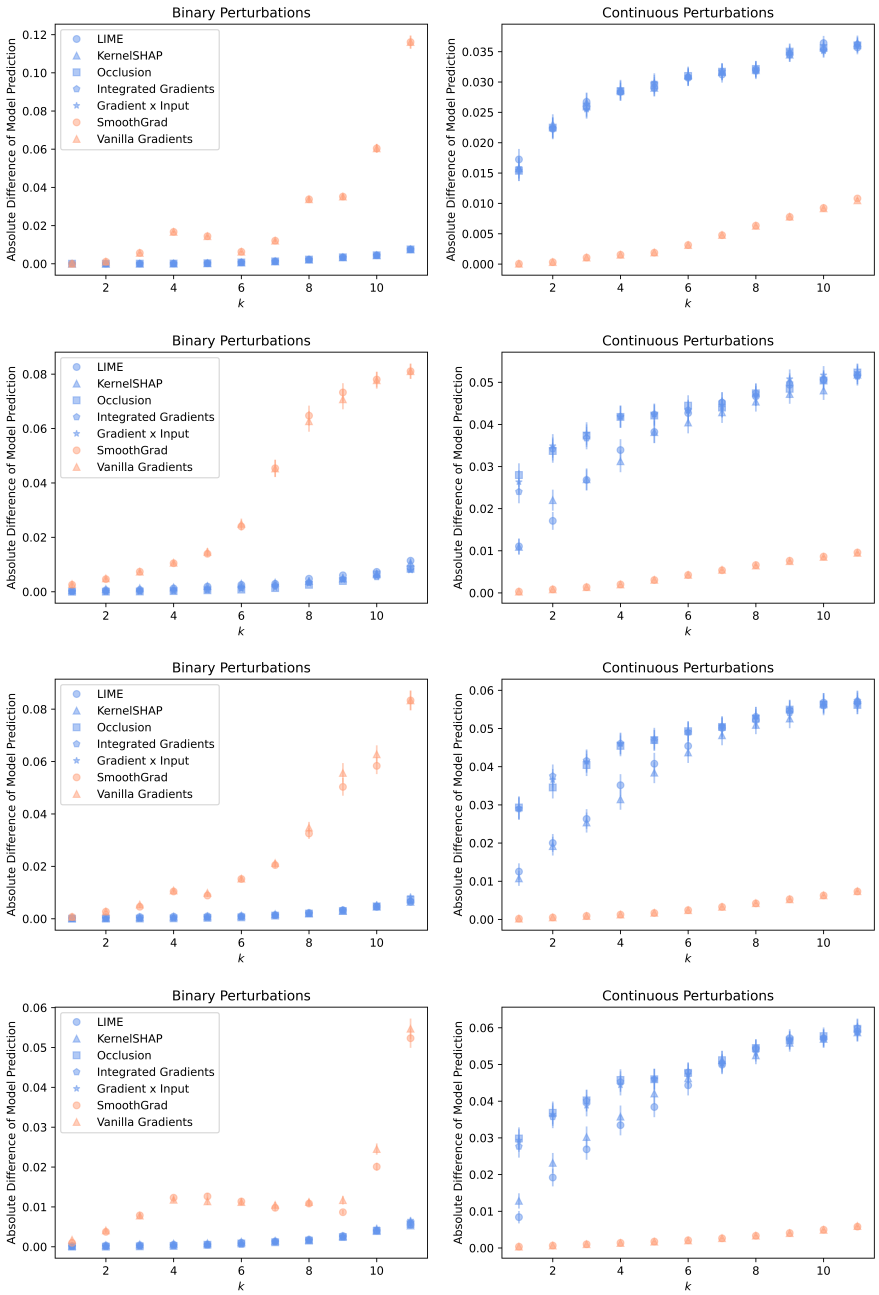


Figure 12: Perturbation tests based on bottom-k features using binary noise (left column) or continuous noise (right column) performed on the WHO dataset for linear regression (Row 1), NN1 (Row 2), NN2 (Row 3), and NN3 (Row 4). The lower the curve, the better a method identifies unimportant features. (Note: Row 2 is a duplicate of Figure 3).

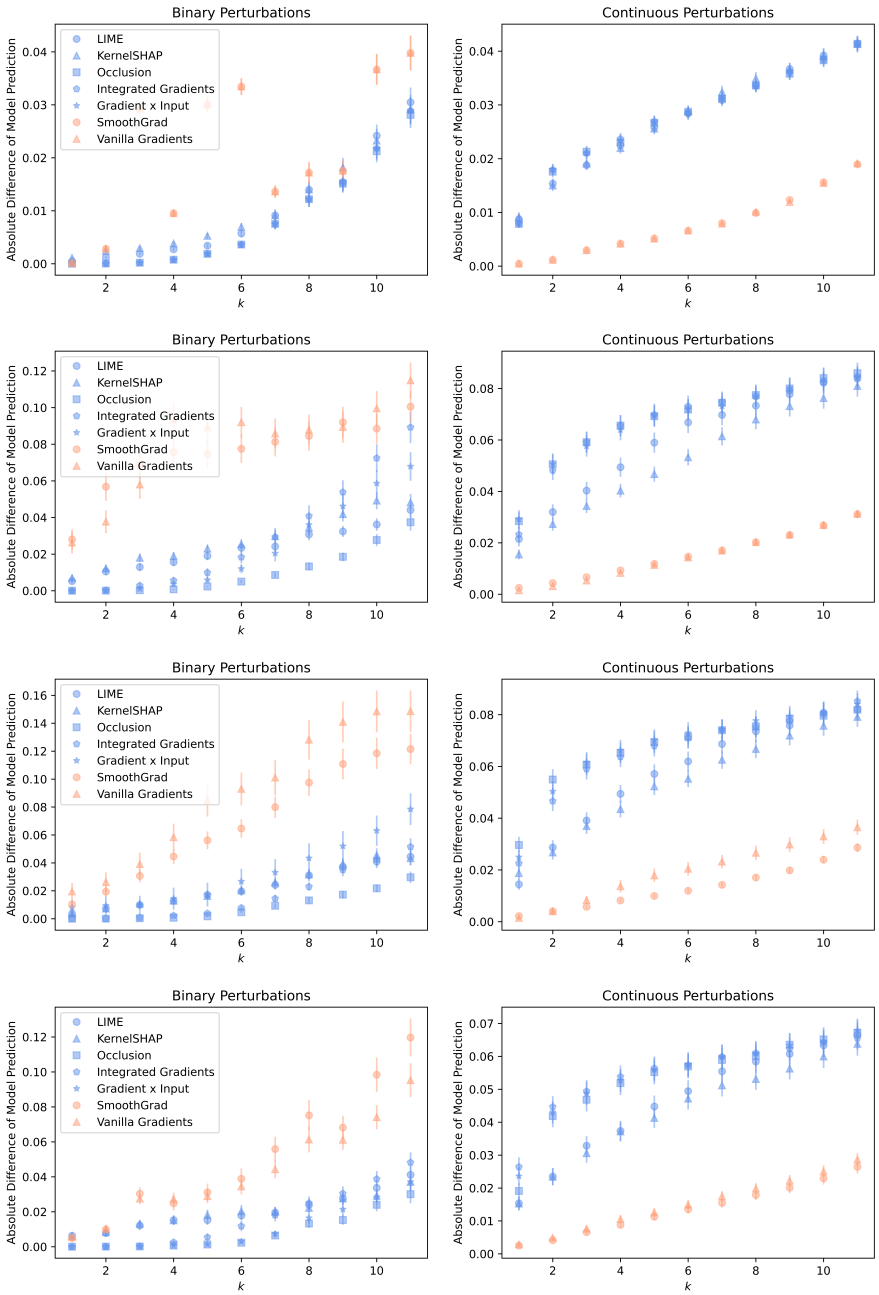


Figure 13: Perturbation tests based on bottom-k features using binary noise (left column) or continuous noise (right column) performed on the HELOC dataset for logistic regression (Row 1), NNA (Row 2), NNB (Row 3), and NNC (Row 4). The lower the curve, the better a method identifies unimportant features.

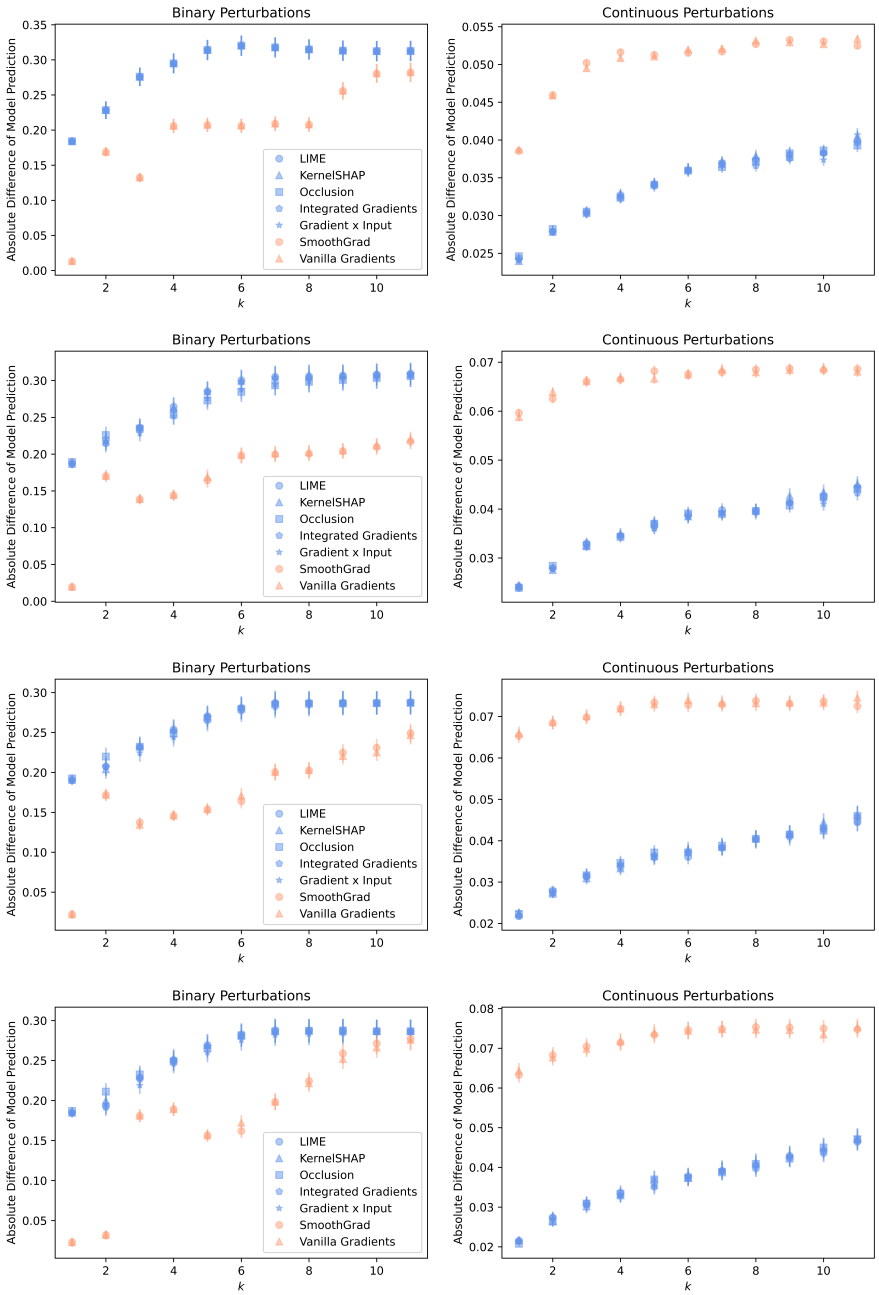


Figure 14: Perturbation tests based on top-k features using binary noise (left column) or continuous noise (right column) performed on the WHO dataset for linear regression (Row 1), NN1 (Row 2), NN2 (Row 3), and NN3 (Row 4). The higher the curve, the better a method identifies important features.

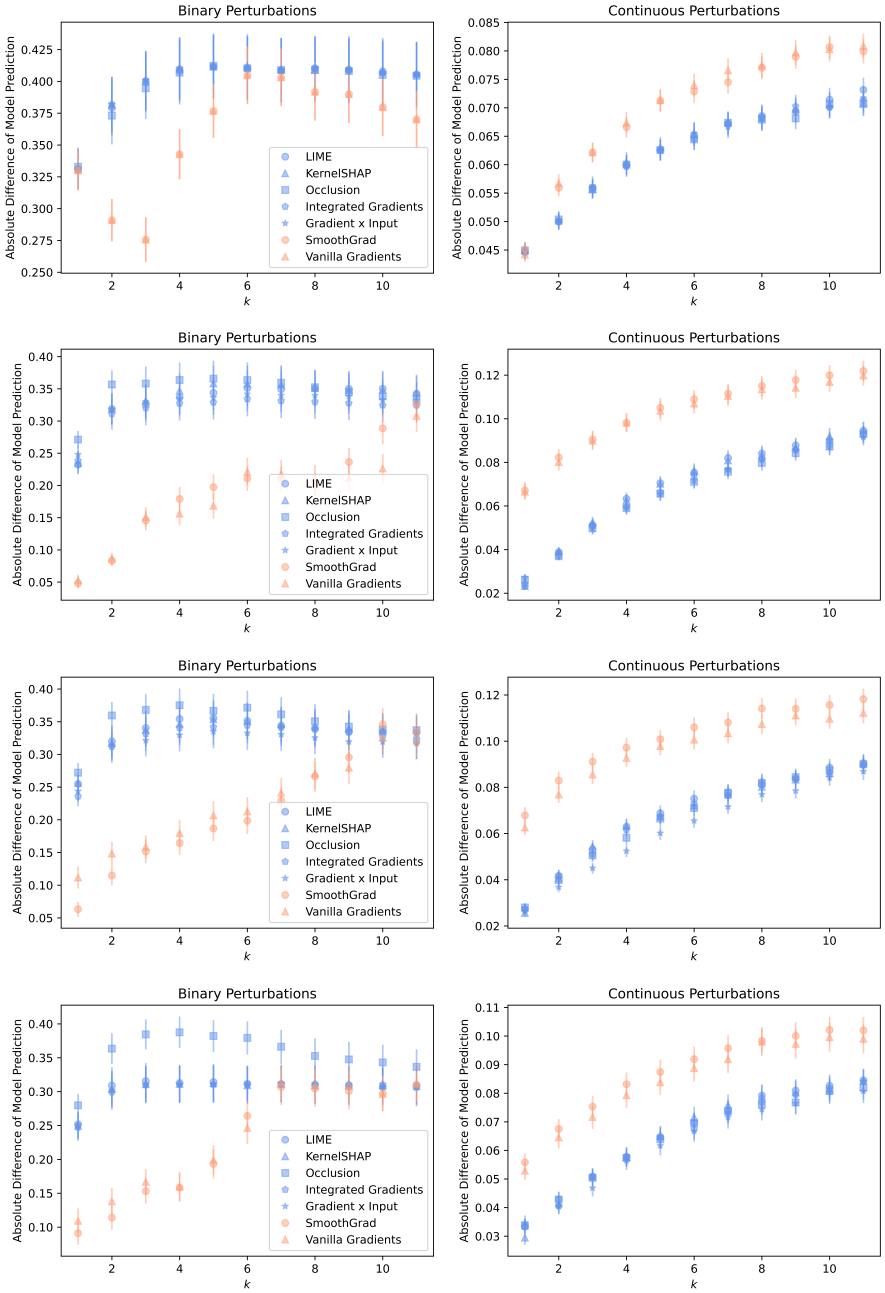


Figure 15: Perturbation tests based on top-k features using binary noise (left column) or continuous noise (right column) performed on the HELOC dataset for logistic regression (Row 1), NNA (Row 2), NNB (Row 3), and NNC (Row 4). The higher the curve, the better a method identifies important features.