Data Science Principles for Interpretable and Explainable AI

Kris Sankaran

May 17, 2024

Abstract

Society's capacity for algorithmic problem-solving has never been greater. Artificial Intelligence is now applied across more domains than ever, a consequence of powerful abstractions, abundant data, and accessible software. As capabilities have expanded, so have risks, with models often deployed without fully understanding their potential impacts. Interpretable and interactive machine learning aims to make complex models more transparent and controllable, enhancing user agency. This review synthesizes key principles from the growing literature in this field.

We first introduce precise vocabulary for discussing interpretability, like the distinction between glass box and explainable algorithms. We then explore connections to classical statistical and design principles, like parsimony and the gulfs of interaction. Basic explainability techniques – including learned embeddings, integrated gradients, and concept bottlenecks – are illustrated with a simple case study. We also review criteria for objectively evaluating interpretability approaches. Throughout, we underscore the importance of considering audience goals when designing interactive algorithmic systems. Finally, we outline open challenges and discuss the potential role of data science in addressing them. Code to reproduce all examples can be found at https://go.wisc.edu/3k1ewe.

1 Introduction

The success of Artificial Intelligence (AI) stems in part from its ability to abstract away details of its context. When an algorithm is applied to predict responses from a collection of features, it matters little whether the response is the shape of a galaxy or the next word in a sentence. Such abstraction has enabled the design of versatile methods with remarkably diverse applications. Nonetheless, algorithms are never used in a vacuum. Models affect human well-being, and it is important to systematically study the human-algorithm interface. How can we ensure that an algorithm behaves acceptably in unforeseen circumstances? How can we prevent harm and unfair treatment[1]? How can align algorithmic behavior with our expectations? The field of interpretable machine learning and explainable AI (XAI) have emerged to address these questions, helping develop techniques that allow algorithmic abstractions to be used wisely within messy, real-world contexts. This literature has engaged researchers from various backgrounds, from theorists seeking to characterize the fundamental limits of explainability techniques (Williamson and Feng, 2020; Bilodeau et al., 2024) to philosophers and psychologists critiquing popular discourses surrounding them (Krishnan, 2019; Nussberger et al., 2022).

What can go wrong without interpretability? Consider these examples:

- Caruana et al. (2015) aimed to help a hospital triage incoming pneumonia patients, ensuring that high-risk patients are admitted while allowing lower-risk patients to receive outpatient care. Using a dataset of past pneumonia patients, their model achieved an AUC of 0.857 in predicting patient mortality based on lab and medical profiles. Surprisingly, an important feature analysis revealed that asthmatic patients had lower predicted probability of death. This arose because the original training data were observational. Historically, asthmatic patients were treated more aggressively than those without asthma, and therefore enjoyed lower mortality. Directly deploying such a model would put

[1]As Recht (2023) writes only half-jokingly, "All models are wrong, but some are dangerous."

asthma patients at risk. Fortunately, their model was editable, so the asthma association could be removed before deployment.

- Gu et al. (2019) designed stickers that, when attached to stop signs, caused object detector algorithms used by self-driving cars to misclassify them as speed limits. This real-world adversarial attack exploits the sensitivity of algorithms to small (and, to humans, seemingly arbitrary) input perturbations. More interpretable models could yield systems with more predictable behavior.
- In Google's first public demo of the generative AI chatbot Bard, it was asked, "What new discoveries from the James Webb Space Telescope (JWST) can I tell my 9 year old about?" The chatbot responded with three seemingly legitimate facts. Only after the demo was shared online did astronomers point out that the third fact, "JWST took the very first pictures of a planet outside of our own solar system," was in fact untrue — the first exoplanet had been imaged in 2004 (Kundaliya, 2023). AI chatbots can "hallucinate" falsehoods, even when their output seems authoritative. Research on XAI could shed light on how these systems generate responses, allowing them to be treated less like an oracle and more like a search engine.

These examples beg the question: What makes an algorithm interpretable? Directly answering this questions is complex, so let's consider a simpler one: What makes a data visualization effective? This question has been studied for decades (Sedlmair et al., 2012; Agrawala et al., 2011; Tufte, 2001; Cleveland, 1993), and every data scientist has firsthand experience with it. A good visualization streamline a taxing cognitive operation into a perceptual one. It helps when the visual encoding – the mapping from sample properties to graphical elements – uses representations that are already familiar or easily learnable. Further, the graphical elements must be legible and well-annotated. We also tend to learn more information-dense visualizations (Tufte, 2001; Oppermann and Munzner, 2022), since they prevent oversummarization and can highlight details for follow-up study. Similarly, an interpretable model can be broken down into relevant components, each of which can be assigned meaning. Instead of information density (showing more of the data), interpretability relies on faithfulness (showing more of the model).

The parallels run deeper. As in a good visualization, the data provenance of a trustworthy model can be traced back to the original measurement mechanism — beautiful design and high-accuracy have little value otherwise. Moreover, like visualization, interpretability must be tailored to its audience and the tasks it is designed to solve (Lipton, 2018). There are different levels of data literacy, and visual representations may be familiar to some audiences but not others. Similarly, AI models are employed across a range of problem domains, necessitating validation in realistic settings (see Section 3). Finally, effective visualizations push readers beyond passive consumption — they inspire deeper exploration of complexity. Likewise, interpretability can support the transition from automated to augmented decision-making (Heer, 2019), enhancing rather than substituting human reason.

Algorithmic interpretability can be approached with the same nuance that is already routine in data visualization. The data visualization literature has identified systematic design approaches and formal evaluation criteria, cautioning against blanket statements about entire approaches[2]. Similarly, discussions of interpretability can go beyond dichotomies about models being either glass or black boxes. To help us navigate the gray areas of this growing field, Section 1.1 introduces relevant vocabulary. Section 2 outlines representative techniques and applies them to a simple simulation example. This section, and the code that accompanies it (https://go.wisc.edu/3k1ewe), can serve as a tutorial on the practical implementation of interpretability techniques. Finally, Sections 3–4 examine the conceptual and technical questions required for effective evaluation and progress as the AI landscape evolves.

1.1 Vocabulary

The terms used in interpretability research can be confusing, because though terms like "explainable" are familiar in a colloquial sense, they have a precise technical meaning in the literature. Indeed, one difficulty

[2]Even information density has been thoughtfully critiqued (Borkin et al., 2013).

is that many people agree that algorithmic interpretability is important — they just have different ideas for what it might look like. Therefore, it helps to have vocabulary that distinguishes key properties while maintaining an appropriate level of abstraction.

An important distinction is between an interpretable model and an explainability technique (Murdoch et al., 2019; Rudin, 2019). An interpretable model is one that, by virtue of its design, is easy to accurately describe and alter. For this reason, they are often called glass boxes. A canonical example is a sparse linear model. For any new example, a prediction can be constructed in a single pass over the features with nonzero coefficients. In contrast, explainability techniques are designed to improve our mental models of arbitrary black box models. A common example is the partial dependence plot (Friedman, 2001). Though originally designed to summarize gradient boosting fits $\hat{f}$, the fact that it only requires $\left(x_i, \hat{f}\left(x_i\right)\right)$ pairs means it applies to any supervised learner. If an interpretable model is a glass box whose inner workings are transparent, then an explainability technique is a systematic way of analyzing the outputs emerging from a black box.

We can relate these approaches to data sciences and design principles; see also Table 1. First, consider interpretable models. When we call a linear model interpretable, we are invoking parsimony and simulatability. Parsimony means that predictions can be traced back to a few model components, each of which comes with a simple story attached. For example, sparse linear models have few nonzero coefficients, and the relationship between each coefficient and the output can be concisely described. A similar principle applies to generalized additive modeling, which have few allowable interactions (Caruana et al., 2015), or in latent variable models, which have few underlying factors (Sankaran and Holmes, 2023). Simulatability formalizes the idea that predictions can be manually reconstructed from model descriptions. For example, in decision trees and falling rules lists, this can be done by answering a sequence of yes-no questions. Finally, since interpretation requires human interaction, it can be affected by the gulfs of interaction (Hutchins et al., 1985). Ideally, an interpretability method should allow users to quickly query properties of the underlying model. Further, the method's outputs should be immediately relevant to the query, requiring no further cognitive processing. To the extent that a method has reached these ideals, it has reduced the gulfs of execution and evaluation.

Unfortunately, even when restricting focus to these glass box models, these definitions are still somewhat ambiguous. These gaps lead to larger questions about evaluation. For example, simulatability alone does not quantify the user effort required to form predictions; it also ignores their accuracy. Linear models become unwieldy when they have many nonzero coefficients, and deep decision trees take time to parse. Similarly, latent variables are only interpretable if they can be related to existing knowledge.

Among explainability techniques, the key question is scale, and this leads to the distinction between global and local explanations. Global explanations summarize the overall structure of a model. Partial dependence profiles are an example of a global explanation, because they describe $\hat{f}$ across the range of possible inputs. In contrast, local explanations support reasoning about individual predictions. For example, when a sentiment analysis model classifies a particular hotel review as negative, we might be curious about which phrases were most important in that classification. Global explanations shed light on the high-level properties of a model while local explanations dissect specific instances of algorithmic judgment. Local explanations are built on the observation that perturbations, either of the model or the data, provide valuable context for understanding algorithmic decisions. Sensitivity, a principle with a long history in statistics (Tukey, 1959; Huber, 1964; Holmes, 2017), is a fruitful path to explainability. Nonetheless, as above, we caution that differences between local and global can blur in practice. For example, concept bottleneck models (Section 2.3.4), globally restricts the space of possible prediction functions, forcing them to pass through an interpretable concept bottleneck. However, editing the bottleneck for specific examples and support local interventions, ensuring that concepts like race or gender are not used downstream. In light of these ambiguities, it is helpful to think of this vocabulary as highlighting the main axes of variation in the space of techniques, a checklist that can guide application and evaluation of interpretability in practice.

Principle	**Discussion**
Parsimony	To facilitate interpretation, the total number of relevant model components should be relatively small. For example, in ℓ_1-regularized methods, the number of coefficients is kept small, and decision trees with fewer splits are more interpretable than those with many.
Simulatability	Given a sample and a model description, how easy would it be for a user to manually derive the associated prediction? Regression models with few coefficients and algorithms like rule lists tend to be more simulatable than those that involve more intensive or multistep pooling of evidence.
Sensitivity	How robust are predictions to small changes in either the data or the model? Ideas from robust statistics can inform how we understand variable importance and stability in more complex AI algorithms.
Navigating Scales	Local interpretations concern the role of individual samples in generating predictions, while global interpretations contribute to understanding a model overall. Interpretation at both scales can guide the appropriate real-world model use.
Interaction Gulfs	The "Gulf of Execution" is the time it takes for an interactive program to respond to a user's input, while the "Gulf of Evaluation" is the time it takes for the user to understand the output and specify a new change. Both can influence the practicality of an interpretability method.

Table 1: The core data science principles underlying interpretability. They can guide the development of more trustworthy and transparent AI algorithms.

2 Methods

How is interpretability put into practice? We next review approaches from the direct interpretability and XAI perspectives. Rather than attempting to review all proposals from this rapidly growing literature, we prioritize the simple but timeless ideas that lie behind larger classes of methods. To ground the discussion, we include a simulation example where the generative mechanism is simple to communicate, but complex enough to warrant analysis with AI.

2.1 Simulation design

The simulation is inspired by the types of longitudinal studies often encountered in microbiome data science. Consider following 500 participants for two months, gathering microbiome samples each day. At the study's conclusion, each participant has their health checked, and they are assigned to either healthy or disease groups. For example, the participants may have contracted HIV or developed Type I Diabetes, as in (Gosmann et al., 2017) and (Kostic et al., 2015), respectively. The question of interest is whether there are systematic differences in the microbiome trajectories between groups. These might have diagnostic potential and could shed light on the underlying mechanisms of disease development. For example, does the disease group have lower levels of a protective species? Do they exhibit unstable dynamics for any important microbial subcommunities? For either fundamental biology or diagnostic potential, any models applied to the data be either interpretable or explainable – classification alone has little value unless it can be incorporated into a medically coherent narrative.

In our hypothetical scenario, we model $N = 500$ participants across $T = 50$ time points, with each sample containing abundances of $D = 144$ microbial species. Figure 2.1 illustrates trajectories for a selection of taxa and participants. The generative mechanism is as follows: we create $K = 25$ community trajectories

$C_k \in \mathbb{R}^{T \times D}$, for $k = 1, \dots, K$, by following these steps:

$$\mathbf{C}_{k,d} = \begin{cases} \text{Unif}\,[0, 0.01] & \text{with probability } 0.7 \\ \mathbf{f}_{kd}^{\text{increase}} & \text{with probability } 0.1 \\ \mathbf{f}_{kd}^{\text{decrease}} & \text{with probability } 0.1 \\ \mathbf{f}_{kd}^{\text{bloom}} & \text{with probability } 0.1 \end{cases} \tag{1}$$

The species-wise random trajectories $\mathbf{f}_{kd}$ are defined as follows. For random increases, we compute the cumulative sum of random nonnegative weights. Decreases, as in Fig 2.1b, follow the same process but with a sign reversal. Specifically, the T coordinates of $\mathbf{f}_{kd}^{\text{increase}}$ are defined using,

$$\tilde{f}_{kdt}^{\text{increase}} = \sum_{t'=1}^{t} u_{t'} \tag{2}$$

$$u \sim \text{Dirichlet}\left(\lambda_u \mathbf{1}_T\right), \tag{3}$$

and then renormalizing $\tilde{\mathbf{f}}_{kd}$ to sum to T. We set $\lambda_u = 0.3$ to allow for occasional large jumps. Random blooms $\mathbf{f}_{kd}^{\text{bloom}}$ like those in Fig 2.1a are formed using,

$$\tilde{f}_{kdt}^{\text{bloom}} = \sum_{t^*} K_{r,L}\left(t - t^*\right) \tag{4}$$

$$t_1, \dots, t_{n_{\text{bloom}}} | n_{\text{bloom}} \sim \text{Unif}\,[L, T - L] \tag{5}$$

$$n_{\text{bloom}} \sim \text{Poi}\left(\lambda_{\text{bloom}}\right), \tag{6}$$

where K_r is a Tukey window function (Van Boxtel, G.J.M., et al., 2021) with bandwidth $r = 0.9$ and window size $L = 9$. The trajectory is then renormalized as with $\mathbf{f}_{kd}^{\text{increase}}$. Given dictionaries $\mathbf{C}_k$ of community trajectories, samples $\mathbf{x}_i \in \mathbb{R}^{T \times D}$ for subject i are sampled,

$$\mathbf{x}_i = \sum_{k=1}^{K} \theta_{ik} \mathbf{C}_k \tag{7}$$

$$\theta_i \sim \text{Dir}\left(\lambda_\theta\right). \tag{8}$$

To generate class assignments y_i, we assign the community weights $\boldsymbol{\theta}_i$ to 24 clusters via K means. These clusters are then randomly divided into 12 healthy and 12 disease groups. This approach therefore links subjects with similar underlying community trajectories into similar health outcomes in a way that is highly nonlinear.

In practice, the steps of the generative mechanism would not be known. Visually examining example trajectories in Figure 2.1d does not suggest clear choices about which microbiome features are relevant for disease classification. Each taxon exhibits a complex, nonlinear trend over time, and while some participants show "blooms" at similar times, it's unclear whether these are linked to disease. Further, the sheer number of taxa makes drawing conclusions through manual inspection impractical. Algorithmic approaches are necessary to distinguish disease from healthy trajectories and identify taxonomic or temporal features of interest, which could then be validated in follow-up studies involving direct interventions on features in a controlled setting.

2.2 Directly interpretable models

We can often learn salient, non-obvious characteristics of the data by applying a directly interpretable model. Some have argued that with sufficient ingenuity, such models can rival the accuracy and efficiency of any black box (Rudin, 2019). Even otherwise, the results of directly interpretable analyses can guide the application of more sophisticated models later on. With this in mind, we briefly review sparse logistic regression (Friedman et al., 2010) and decision trees (Loh, 2014), apply them to the simulation, and interpret the results.

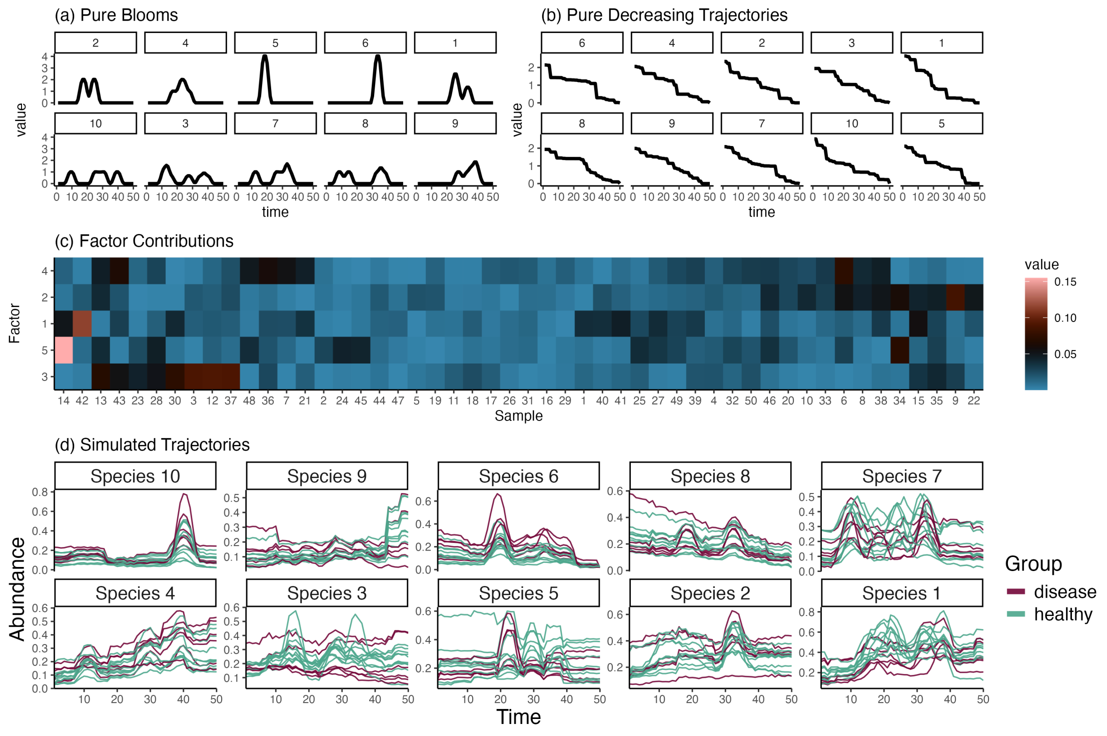

Figure 1: Example simulated trajectories. (a - b). Trajectories for two latent types. Each sample is a mixture of these types. Even for a fixed type, samples may have different variants, e.g., blooms at different times. (c) Observed trajectories are mixtures of these pure underlying trajectory types. The contributions of the five latent trajectory patterns for each sample are shown in this heatmap. (d) Healthy and disease sampled generated in through this process.

2.2.1 Sparse logistic regression

Sparse logistic regression solves the optimization problem,

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^D} \sum_{i=1}^{N} \ell\left(y_i | x_i, \beta\right) + \lambda \|\beta\|_1$$

where $\ell\left(y_i|x_i,\beta\right) = \log\left(\left(1 + \exp\left(x_i^T \beta y_i\right)\right)^{-1} \exp\left(x_i^T \beta y_i\right)\right)$. This optimization identifies a sparse coefficient vector $\hat{\beta}$ that maximizes the log-likelihood of pairs (x_i, y_i). The parameter λ controls sparsity: higher values yield simpler but less expressive models. The combination of sparsity and linearity ensures interpretability. Sparsity allows many features within x_i to be ignored during prediction, while linearity ensures that for features that do contribute, they enter the model transparently. Specifically, a one-unit increase in x_d changes the relative risk for class $y = 1$ by $\exp\left(\beta_d\right)$.

2.2.2 Decision trees

In contrast, decision trees do not solve a single optimization problem. Instead, they are defined through a recursive algorithm. Initially, we scan all D dimensions of x_i and determine thresholds t_d such that $\mathbb{I}\left(x_{id} > t_d\right)$ maximizes classification accuracy. From these candidates, we select the optimal d^*, defining the first partition along the axis d^*. The two elements of the partition correspond to two leaves of a tree whose only split according to x_{id^*}. Next, for a tree with L leaves, we iteratively split each element along axis d at a threshold t_d^l, for all pairs of leaves l and dimensions d, selecting the split that maximizes accuracy. This process stops once model complexity outweighs performance on a holdout set.

Interpretability here stems from parsimony and simulatability. The final tree has only as many partition elements as leaves, and with sufficient cost on model complexity, this can be easily shown as a single tree. Further, the axis-aligned splits yield straightforward descriptions for each partition element, recoverable by tracing a path from the root to the leaf. Finally, predictions remain constant within each element, changing only across partition boundaries.

2.2.3 Advances

Sparse and tree-based models are the foundation for many proposals for interpretable machine learning. For example, Zeng et al. (2016) introduced fast decision tree algorithms that match black box algorithms in recidivism prediction, a domain with significant moral ramifications where interpretability is crucial. Similarly, Hazimeh et al. (2020) designed a decision tree variant that competes effectively with deep learning in various operations research tasks. By leveraging advances in combinatorial optimization, their approach allows interpretable trees to be learned from very large datasets without excessive compute demands.

Similarly, ℓ_1 regularization is central to many modern interpretability workflows, even in problems that initially seem distant from traditional statistical learning frameworks. For example, Fridovich-Keil et al. (2022) developed an interpretable approach to the view synthesis problem, where many two-dimensional views of an object are consolidated into a coherent three-dimensional reconstruction. Instead of the typical deep encoder-decoder architecture, they used ℓ_1-regularized regression on the appropriate Fourier representation. This approach achieved better accuracy than competing deep learning approaches with far less sensitivity to hyperparameter selection, and could be trained at a fraction of the compute cost.

2.2.4 Application to the simulation

We next apply these directly interpretable models to the simulated trajectories from Section 2.1. We explore two approaches. First, we concatenate all time points for each participant into a single $T \times D$-dimensional vector. This method can uncover temporally localized predictive signals. Second, we derive time series summaries for each taxon, reducing the number of predictors to $S \times D$ for S different summaries. In our study, we focus solely on linear trends and curvature. For each subject i and taxon d, we fit a linear

Data	**Model**	**In-Sample Accuracy**	**Out-of-Sample Accuracy**	**Training Time (s)**
Original	Lasso	81.2%	78.0%	3.9
	Decision Tree	91.6%	70.2%	95.9
	Transformer	98.7%	83.2%	113.2
	Concept Bottleneck	99.7%	84.0%	120.9
Featurized	Lasso	92.6%	87.8%	0.4
	Decision Tree	74.4%	69.6%	1.8

Table 2: In and out-of-sample accuracies of models applied to the simulation study. For cross-validated models, training time was averaged across folds. Training time was computed on a 2022 MacBook Pro with 16GB RAM and an M2 GPU. Models differ in their inherent interpretability, manual feature curation effort, and generalization performance. Deep learning models can overfit the training sample while still generalizing well. Directly interpretable models can be substantially improved through effective featurization.

model $x_{itd} \sim \mu_{id} + \beta_{id} t$, with time as a predictor. The estimated $\hat{\beta}_{id}$ summarizes the linear trend for that trajectory. Similarly, we use the sum-of-squares of the temporal second differences to summarize curvature: $\frac{1}{T-2} \sum_{t=2}^{T-1} \left(x_{itd+1} - x_{itd-1}\right)^2$. Note that summarization derives potentially relevant features that are not simple linear combinations of the original inputs.

The holdout accuracy for the sparse logistic regression and decision tree models under these two representations of x_i is given in Table 2. Regularization λ and tree complexity parameters were chosen using four fold cross validation. For these data, the sparse logistic regression model outperforms the decision tree. Moreover, manually derived summary statistics yield better performance compared to concatenating all inputs. This suggests a general takeaway: a simple model on the appropriate representation can achieve good performance, and performance is contingent on the representation. The widely invoked "interpretability-accuracy tradeoff" overlooks this nuance. Further, Table 2 includes the training time for the algorithms, illustrating a broad range of runtimes among interpretable models. Simple doesn't always mean fast or scalable.

With respect to interpretability, on concatenated inputs, the sparse logistic regression model with the optimal λ required 37 taxon-by-time point features, while the decision tree needed 13 splits. In contrast, with the summary statistic featurization, the optimal sparse logistic regression model used 50 features – the problem is lower dimensional but the signal is more enriched, a fact suggested by the difference in prediction performance. Surprisingly, this enriched signal is not accessible to the decision tree, which produces a tree with only two splits.

The parsimony of these estimates aligns with our discussion of the characteristics of directly interpretable models. In both approaches, prediction relies only on a small subset of the original inputs. But how much can we trust these features? Fig 2(e - f) shows the estimated $\hat{\beta}$ for nonzero features obtained from separately fitting the sparse logistic regression approaches on independent splits. When using concatenated inputs, only two features overlap, while with summary statistics, this increases to 18. Therefore, while the model on raw time points may be more parsimonious, it is less stable. In this sense, despite its parsimony, the first model does not lend itself naturally to interpretation. Conversely, the second appears more accurate and stable. Moreover, its features, like curvature and trends, match the existing vocabulary of domain experts, making it the more interpretable model in this context. The larger lesson is that interpretability refers to a constellation of model properties – parsimony and simulatability but also accuracy, stability, and contextual relevance – and it is worthwhile to think in these more refined terms rather than simply "interpretable" and "black box" models.

2.3 Deep learning and XAI

The sparse logistic regression with derived summary statistics seems to be our best option in the case study. However, we should be concerned by the fact that we manually define our features – what other potentially predictive features might we be missing out on? Black box algorithms are appealing because they automate

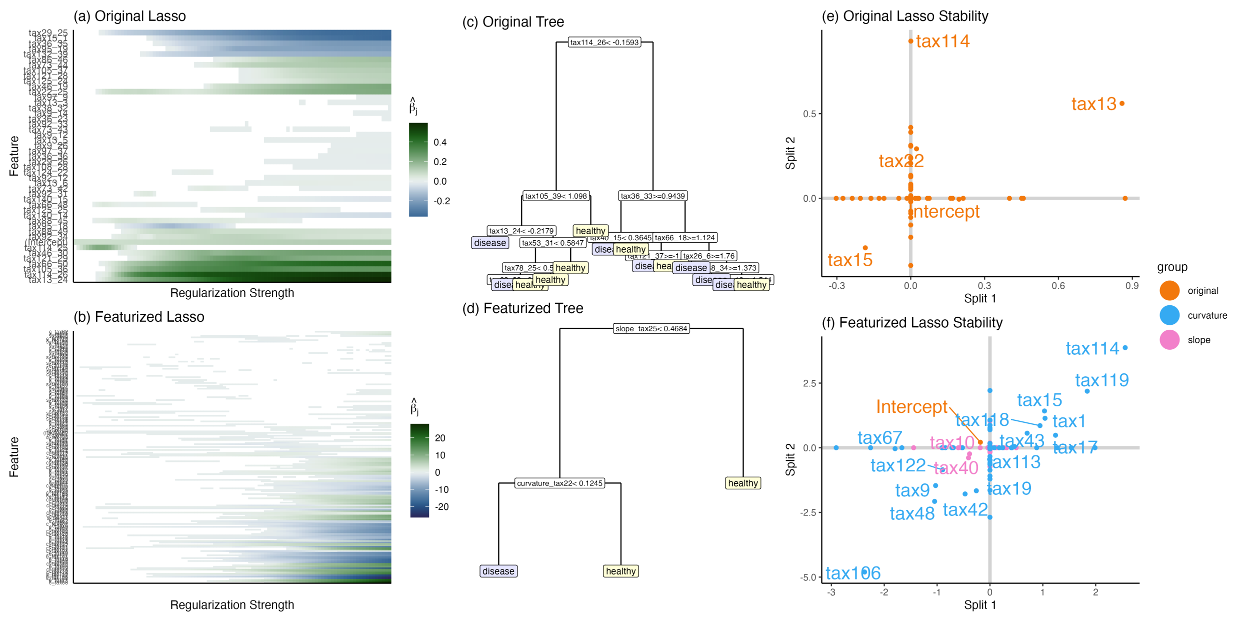

Figure 2: Results from directly interpretable models applied to the simulation study. (a) Lasso coefficient paths on the original, unfeaturized data. ℓ_1-regularization strength decreases from left to right. Only those that are nonzero for at least one penalty λ are displayed. (b) The corresponding lasso paths for the featurized data. Though there are fewer input features overall, more are selected. (c - d) Selected decision trees on the original and the featurized data, respectively. With derived features, a very simple tree achieves surprisingly good performance (compare with Table 2) (e - f) Stability of cross-validation optimized lasso fits on the original and featurized data, respectively. The regression using manually derived features is more stable, which is consistent with the high temporal correlation in the original data (see Fig 2.1a).

the representation learning process (Bengio et al., 2013; Bengio, 2009). For example, in computer vision, deep learning models can identify higher-order, semantically meaningful features from raw pixel inputs (Zeiler and Fergus, 2013; Yosinski et al., 2015; Simonyan et al., 2014). The can learn features that activate on images of eyes simply by being given training data of faces and an appropriate loss function. Applying deep learning model to our microbiome data could reveal predictive temporal and taxonomic features (e.g., the presence of particular subcommunities) that we had not previously considered. The challenge, then, would be to explain these models in a way that advances scientific knowledge and ensures accurate medical diagnoses. To demonstrate the process, let's explore using a transformer encoder model in our simulation.

2.3.1 Transformers

We first review the mechanics of transformer architectures in deep learning. Beyond our case study, transformers are particularly relevant to XAI because they have become a universal building block of modern AI. What used to be multiple specialized architectures – like CNNs for vision, LSTMs for text, and autoregressive networks for audio – have now become consolidated into different forms of transformers. At a high-level, the transformer is an architecture for deriving features from sequential data. It works by recombining representations of individual sequence elements through a series of "attention" operations, which prioritize segments of the sequence relevant to the current element's updated representation. It takes low-level representations of individual sequence elements, like one-hot encodings of individual words, and then modifies them to reflect larger sequential context.

Specifically, consider a sequence of tokens $x_1, \dots, x_T$. These tokens can be embedded into a sequence $z_1, \dots, z_T$ of high-dimensional (e.g., 512) real-valued vectors. In a word-based language model, x_t could be one-hot encodings of the words from the language's vocabulary, while in a vision problem, x_t might represent neighboring patches from an image. The z_t vectors are meant to capture the high-level features of the original t^{th} token (word or patch) within its larger context (sentence or image). Therefore, they are much more valuable for downstream analysis than word identity or pixel values considered in isolation.

Transformer models use a mechanism called self-attention to *transform* the original x_t into the z_t (Vaswani et al., 2017; Lee, 2021). If we stack the x_t into $\mathbf{X} \in \mathbb{R}^{T \times D}$, then self-attention has the form,

$$\frac{1}{\sqrt{D_A}} \left(\mathbf{X}\mathbf{W}_q \mathbf{W}_k^T \mathbf{X}^T \right) \mathbf{X}\mathbf{W}_v$$

for trainable parameters $\mathbf{W}_q \in \mathbb{R}^{D \times D_A}, \mathbf{W}_k \in \mathbb{R}^{D \times D_A}, \mathbf{W}_v \in \mathbb{R}^{D \times D_z}$. The term in parenthesis has dimension $T \times T$ and summarizes the extent to which coordinates t' in the output "attend" to coordinates t in the input. The matrix $\mathbf{W}_v$ allows attention to return a linearly transformed version of $\mathbf{X}$, which can often be helpful for increasing/decreasing output dimensions. Typically, several of these transformations are applied in parallel, concatenated into an object called "multihead attention," and passed through a softmax nonlinearity. The output of this complete process is the matrix of embeddings with rows $z_1, \dots, z_T$.

The embeddings z_t are difficult to interpret because their derivation is neither parsimonious nor simulatable. Empirically, the representations have been shown to encapsulate high-level features associated with each token's context. However, the number of updates, the fact that the update can "attend" to all other tokens simultaneously, and the fact that the attention mechanism is nonlinear all contribute to the difficulty in interpreting these embeddings as well as any decision rules derived from them. Nonetheless, several approaches have emerged for enhancing the interpretability of these types of deep, embedding-based models (Erhan et al., 2010; Wongsuphasawat et al., 2018). We review three approaches below: global explainability, integrated gradients, and concept bottleneck models.

2.3.2 Embedding visualization

Embedding visualizations treat the learned representations z_t as data and apply Exploratory Data Analysis techniques to better understand their properties. Methods like principal components analysis, canonical correlation analysis, or UMAP are often applied to the collection of z_i (Nguyen and Holmes, 2019; Raghu et al., 2021, 2019). Since the embeddings can be studied simultaneously across the full dataset, this is an example of a global explainability technique.

For example (Coenen et al., 2019) used embeddings to analyze the BERT language model (Devlin et al., 2019). In one experiment, they examined sentences containing words that can have multiple meanings depending on context, like, "fair." By computing embeddings for instances of "fair" within these sentences, it was observed that although the word remained the same, its embeddings distinctly clustered in a two-dimensional principal components projection. This clustering reflected the context dependence of the word's meaning, with clusters representing the legal ("fair use"), mathematical ("fair coin"), and civic ("world's fair") uses. This qualitative study can also be accompanied by quantitative analysis. For example, Coenen et al. (2019) and Hewitt and Manning (2019) used linear probes to quantify the extent to which grammatically structure (represented by parse tree representations) could be reconstructed from BERT embeddings.

2.3.3 Integrated gradients

When seeking local explanations, embedding visualization lack sufficient specificity. We often have questions in mind that require more than model-based summaries – for example, what was it about a hotel review that led to its classification as negative sentiment, or how could a loan applicant modify their application so that it can be approved? Addressing these questions requires thinking carefully about the model's sensitivity to perturbation. Many methods formalize this intuition – influence functions (Koh and Liang, 2017), shapely values (Lundberg and Lee, 2017), LIME (Ribeiro et al., 2016), GradCAM (Selvaraju et al., 2017), and Integrated Gradients (IG) (Sundararajan et al., 2017) for example. Here, we dive into the mechanics of IG.

The integrated gradient of a sample x_i with respect to class y defined by:

$$\mathrm{IG}\left(x_i\right) = \left(x_i - x_0\right) \int_{\alpha \in [0,1]} \frac{\partial f_y\left(x_0 + \alpha\left(x_i - x_0\right)\right)}{\partial x_i} d\alpha \tag{9}$$

where x_0 is a reference sample (e.g., an all-black "empty" image) and f_y is the model's predicted probability for the input being assigned to class y. The term $\frac{\partial f_y(u)}{\partial u}$ represents how an infinitesimal change in the i^{th} coordinate of u affects the probability of assigning the input to class y. At first, this appears to be sufficient for local explanation – large gradients correspond to words or pixels with the largest influence on the class assignment. However, this often fails in practice due to saturation. Specifically, deep learning classifiers often use the multiclass logistic function in the final layer, and for large input magnitudes, this function's gradients approach zero, rendering explanations based on $\frac{\partial f_y(u)}{\partial u}$ unsatisfactory.

The solution proposed by IG is to consider versions of the input $x_0 + \alpha\left(x_i - x_0\right)$ that have been shrunk down to x_0, hence avoiding vanishing gradients. Since it might not be clear how much to shrink at first, IG considers scaling along an entire sequence of $\alpha \in [0, 1]$. Integrating gradients over this range gives a more faithful description of the i^{th} coordinate's contribution to classification as y. For example, in Sturmfels et al. (2020), gradients for the object of interest (a bird) are much larger for small α values, ensuring that the final IG for the bird class places appropriate weight on that class.

Integrated gradients and its local explanation counterparts are widely used in XAI, but they are also among the most controversial. Both theoretical and empirical results suggest that explanations do not necessarily respect expectations. For example, Adebayo et al. (2018) observed that saliency maps, including those from IG, derived from randomly initialized neural network classifiers look closely resemble those from properly trained ones. A theoretical study by Bilodeau et al. (2024) found that locally similar functions could nonetheless be assigned different explanations at their query points, indicating that global properties can influence local explanations. These alarming findings have renewed interest in the interpretability community to develop local explainability techniques with formal guarantees.

2.3.4 Concept bottleneck models

Both embedding analysis and integrated gradients rely solely on the model to be explained and the original training data. This is both a strength and a limitation. On the one hand, it ensures generality, the same explanation method can be effective across various applications. On the other, it limits the incorporation of domain-specific language into the explanation. Concept Bottleneck Models (CBMs) (Koh et al., 2020) offer an abstraction for infusing domain-specific language into models. CBM "concepts" bridge model and human representations of relevant data features. The overarching goal of this class of models is to create representations that can be manipulated by both the model (as high-dimensional, numerical vectors) and experts (as community-derived concepts).

CBMs leverage expert knowledge to form annotations useful for both model training and explanation. During training, each sample has the form (x_i, y_i, c_i). x_i and y_i are the usual supervised model input and output for sample i. $c_i \in \{0,1\}^K$ represents a dense concept annotation. For example, x_i could be the image of a bird, y_i its species label, and c_i a summary of interpretable features like wing color or beak length. The CBM learns a model $x_i \to c_i \to y_i$, compressing low-level measurements in x_i into concept annotations c_i before making a prediction y_i. Koh et al. (2020) propose several strategies for implementing this bottleneck, including joint training of functions f and g to minimize the weighted sum of concept L_1 and label L_2 losses:

$$\hat{f}, \hat{g} := \arg\min_{f,g} \sum_i L_1\left(c_i, f\left(x_i\right)\right) + \lambda L_2\left(y_i, g\left(c_i\right)\right) \tag{10}$$

for some $\lambda > 0$.

On a test sample x^*, predictions are made using $\hat{c} = f\left(x^*\right)$ and $\hat{y} = g\left(\hat{c}\right) = g\left(f\left(x^*\right)\right)$. Note that concept labels c_i are not needed at this stage. Typically, a simple model g, like multiclass logistic regression, is used, This model essentially automates human-interpretable feature extraction, bridging manual feature engineering with black box representation learning. While manually designing a "wing color" feature extractor would be tedious, it is helpful to know that the model f learns a proxy for such a feature, and its output can be used like any human engineered feature. Moreover, this approach streamlines counterfactual reasoning. For example, we can ask what the model would have predicted had the wing been yellow instead of red, simply by intervening on the c_i associated with wing color. This is important in fairness or recourse analysis. For example, if we know that a feature should not be used for prediction (e.g., the race of a loan applicant), then we can zero out the coefficient from that concept in the model g.

Listing 1: The analogous implementation of a Concept Bottleneck Model. `c` represents the concept labels, which are transformed into class predictions using a shallow multilayer perceptron.

```
class ConceptBottleneck(nn.Module):
 """
 Learn Concepts and Classes for Sequences
 """
 def __init__(self, n_embd=144, n_positions=50, n_layer=6, n_concept=25, n_class=2):
  super(ConceptBottleneck, self).__init__()
  self.n_concept = n_concept
  self.n_class = n_class
  config = GPT2Config(n_embd=n_embd, n_positions=n_positions, n_layer=n_layer)
  self.backbone = GPT2Model(config)
  self.concept = nn.Linear(n_embd * n_positions, self.n_concept)
  self.mlp = nn.Sequential(
    nn.Linear(self.n_concept, self.n_concept),
    nn.ReLU(),
    nn.Linear(self.n_concept, self.n_concept),
    nn.ReLU(),
    nn.Linear(self.n_concept, self.n_concept),
    nn.ReLU(),
    nn.Linear(self.n_concept, self.n_class - 1)
  )

 def forward(self, x):
  z = self.backbone(inputs_embeds=x)
  c = self.concept(z.last_hidden_state.view(x.shape[0], -1))
  return c, self.mlp(c)
```

2.3.5 Application to the simulation

With these XAI techniques, we can apply black box models to our microbiome case study and still draw scientific insight from them. The simulated dataset is well-suited to a transformer encoder architecture, where each sample corresponds to a word and the full trajectory to a sentence. The final healthy vs. disease classification is analogous to text sentiment classification. Guided by these parallels, we trained a GPT-2 architecture with a random sample of 375 simulated participants; the remaining 125 are reserved for validation. We used default hyperparameters from a public implementation with the two modifications. First, we changed the input dimension to 250 to one-hot encode the species. Second, we reduced the number of layers to 6 from the default of 12, since the sample size is relatively small. After training this model for 70 epochs, the model achieves a holdout accuracy of 83.2%. This performance is noteworthy considering that it makes progress towards the best model with handcrafted features (cf. Table 2), which were designed with the true generative mechanism in mind. Indeed, considering the close alignment between handcrafted features and the generative mechanism, we suspect that the Bayes error for this problem is close to 13%. The ability to nearly match a handcrafted model underscores the appeal of modern deep learning. It is not necessarily about achieving better performance than interpretable, handcrafted counterparts on all problems; rather, it comes from the potential to bypasses feature engineering – a traditionally labor-intensive and time-consuming step – while achieving comparable performance.

The remaining challenge lies in understanding how the model achieved this performance. We consider each of the techniques discussed above. Fig 3 offers two global views of the data. Fig 3a, gives the sparse PCA projections derived from the scaled n subjects $\times$ (D taxa $\times$ T timepoints) matrix of community profiles over time, while Fig 3b shows the analog for the N subjects $\times$ L embeddings transformer representations. The embeddings in (b) more clearly distinguish healthy and diseased subjects, capturing more of the total variance within the top two dimensions. This is as expected, since the final classification uses a linear hyperplane, encouraging the transformer to learn a representation that ensures linearly separability. To better understand the features encoded by the transformer, we focus on the subset of embeddings associated with species 21. Fig 3c shows sparse PCA applied to solely these embeddings. We have highlighted samples that lie near the linear interpolation between samples 110 and 378 in the original embedding space, and their original trajectories are shown in panel (d). Along this interpolation, the peaks and valleys become less pronounced, suggesting that the model has learned to differentiate curves with different trajectory. This is a subtle feature, and one that we know by design is related to the response. In contrast, some visually prominent features unrelated to disease, like the maximum value of the series, are appropriately disregarded.

We next study local explanations using IG. Based on the input structure, IG returns a value for each time point-by-taxon-by-taxon combination. Example values are given in Fig 4. In each panel, we can identify time points that informed the final classification and the direction in which perturbations change the target class label. "+" denotes that an increased abundance at that time point increases the probability of correct classification; "-" suggests the opposite. Larger symbols imply larger changes in probability. For example, for subject 6, the first ten time points and the valley surrounding time 20 were most important. Increases in either of these intervals increase the probability of correctness. In contrast, for subject 63, an increase in the valley surrounding time 20 decreases this probability. Since these subjects are from opposite classes, deeper valleys around time 20 seem to be related to health. However, beware that these conclusions are essentially local, and this pattern need not hold across all samples for IG to declare it important. In general, IG effectively localizes the responsibility for a classified instance down to a subset of time points, aiding error analysis and highlighting subsets of features that could flip the class of a prediction.

Next, we take a closer look at the use of CBMs in our case study. Recall that concepts are a form of dense annotation for each sample. We extend our original generative mechanism to provide sample-level concepts. For each θ_i^k, we define a binary vector by thresholding: $c_i = \mathbf{1}\{\theta_i^k > t\}$. Since each θ_i^k corresponds to a latent trajectory b_k, this acts as a coarse label for sample i, indicating which of the k underlying trajectories it is most influenced by. Next, we adapt our transformer to predicts both the binary pattern of c_i and the final disease status, adding a concept-level loss that computes a logistic loss coordinatewise (see Listing 1). The results are displayed in Table 2. Notably, using this concept definition, the model performs better than the original, unconstrained transformer, suggesting closer alignment to the original generative

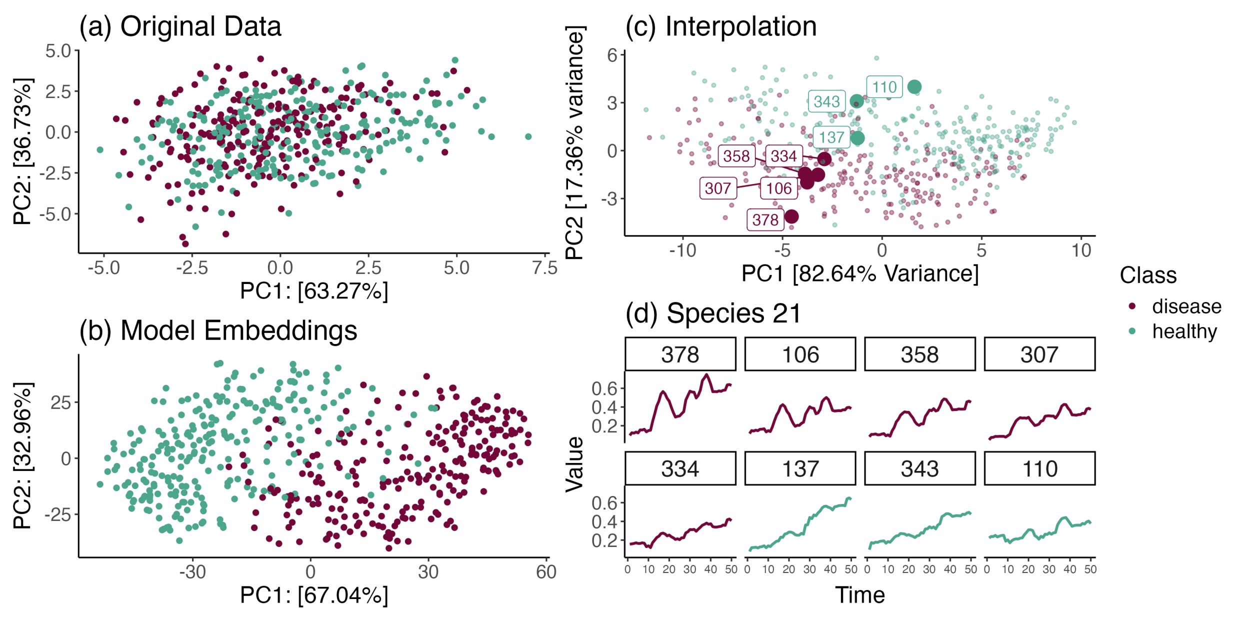

Figure 3: Global explanation of the simulation's transformer model via embeddings. (a) Healthy and disease classes generally overlap in the original data. Each point gives the projection of a subject onto the top two sparse PCA directions. (b) Transformer representations more clearly separate the two classes. (c) A sequence of points along the linear interpolation between samples 110 and 378 in embedding space restricted to features from species 21. (d) Trajectories associated with each labeled point in (c).

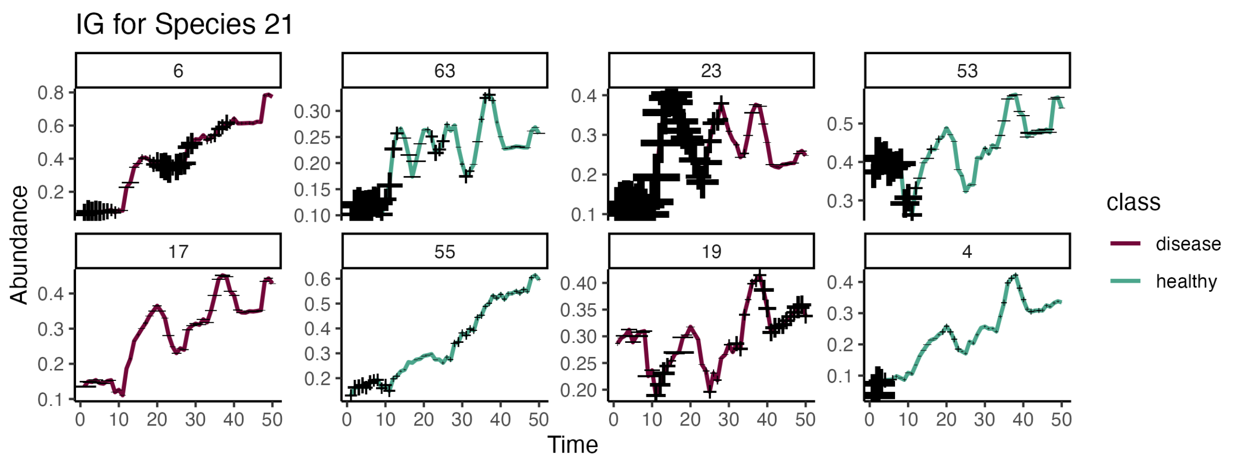

Figure 4: Integrated gradients for the transformer model overlaid on a subset of samples from the simulation study. Each curve represents the trajectory for species 21, with ± symbols indicating the size and sign of the integrated gradient at that sample and time point. The subset of relevant time points can vary from sample to sample.

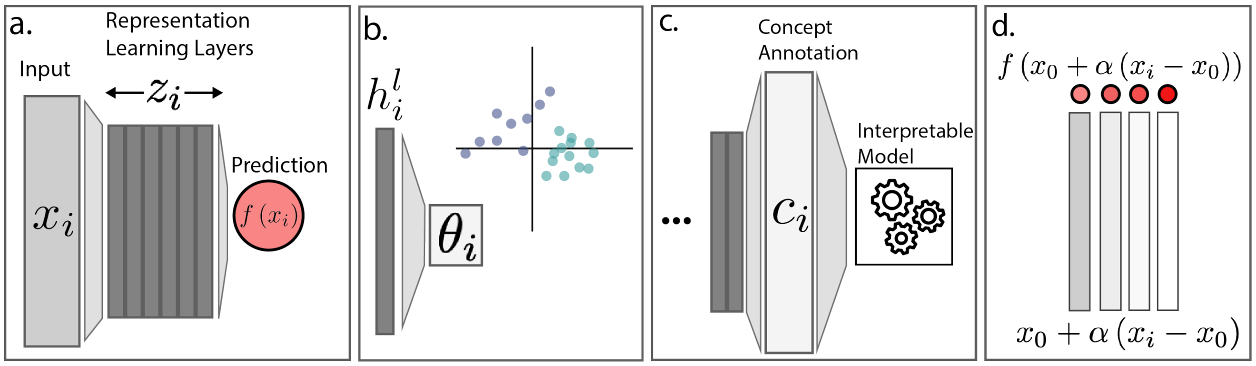

Figure 5: Summary of XAI techniques. (a) Deep learning models transform input x_i into higher-level semantic representations. The final layer is used for prediction. (b) Global embeddings represent a layer of interest with lower-dimensional vectors z_i, which can be interpreted alongside sample-level metadata, like point color in this example. (c) CBMs only use the black box to predict dense, concept-level annotations, not the final response. An interpretable model translates predicted concepts into the prediction (d) In IG, a sample is subject to a sequence of perturbations, and the impact on the final prediction is recorded to identify important coordinates.

mechanism. In general, better CBMs require denser concept annotation. Note, however, that recent work has proposed alternatives that discover concepts de novo or identify cheap concept proxies following model training (Yuksekgonul et al., 2023; Ghorbani et al., 2019).

3 Evaluation

Developing effective benchmarks for interpretability methods is an active area of research, critical for objectively comparing proposals. However, designing effective criteria faces several challenges,

- Operationalization: Interpretability encompasses several technical definitions, like parsimony and simulatability, making it difficult to capture precisely with a single metric.
- Output Types: Different classes of methods return different types of outputs, like attribution maps and concepts, which are hard to compare directly.
- Context Dependence: Method performance must be gauged relative to the audience and tasks they were designed for. Nonetheless, a good benchmark should generalize across tasks.
- Ambiguous Ground Truth: Unlike supervised learning, establish ground truth interpretability output (e.g., the "correct" feature attribution map) is often impossible.
- Human-Computer Interaction: The success of an interpretability method depends on how it interfaces with users. Simplified experiments may not reflect real-world complexity, while real-world observations may be confounded by factors that don't generalize across applications.

These challenges mirror those encountered in data visualization, which has developed various evaluation protocols to guide systematic progress. Unlike traditional machine learning, where isolated computational benchmarks have led to significant progress, we expect evaluation in interpretability will depend on triangulating multiple sources of experimental evidence, like in data visualization.

Interpretability evaluation methods fall into two general categories: tailored benchmark datasets and user studies with controlled tasks. Within each class, there are variations that account for the challenges above in different ways. For dataset benchmarks, one approach is ablation (Hooker et al., 2019; Wang et al., 2022).

Specifically, for methods that output feature importance attributions, this involves retraining models versions of the data where purported important regions are masked out. Deterioration in model performance suggests that the attributions accurately reflect regions that the original model relied upon. Note that retraining is necessary. The ablated images are not drawn from the same distribution as the training data, so deterioration in performance of the original model could be due to either removal of important features (the criteria of interest) or domain shift (a confounding factor). Another data-driven benchmarking approach is to design synthetic datasets. These datasets are generated using known mechanisms, allowing for later evaluation of explanations. For example, the funnybirds benchmark creates imaginary cartoon birds where "species" classes are derived from attributes, like wing, beak, and tail type, that are controlled during data generation (Hesse et al., 2023). Given a specific instance, an ideal explanation should focus on the parts of the bird that distinguish it from related classes. To make these benchmarks more realistic, variation in viewpoint and illumination can be introduced.

In user studies, one approach is to have participants make pairwise preference judgments between explanations (Jeyakumar et al., 2020). However, subjective preference may not align with objective task performance. To this end, more specific experimental tasks can be designed, evaluating user decisions with and without support from interpretability outputs (Ma et al., 2024; Colin et al., 2022). For example, to assess simulatability, participants could be asked to guess a model's output on an example before and after viewing an explanation. Alternatively, they could be asked to guess how predictions might change under perturbations to the input – effective explanations should help in making accurate guesses. Finally, integrating interpretability methods into workflow-specific interfaces makes it possible to study their performance within tailored tasks. For example, Wu et al. (2021) designed an interface to aid auditors in generating counterfactual examples that flip model classifications, helping to characterize sensitivity to out-of-distribution shifts. Influential examples that participants discovered could be incorporated into training to improve generalization. Different choices in the frontend interface and backend interpretability engine led to differential success in discovering surprising counterfactuals (those with small perturbations in input that led to large change in class predictions). In this way, interface-oriented user studies can more precisely gauge "human-AI collaboration." Indeed, these experiments have uncovered counterintuitive phenomena, like instances where task performance worsens with AI explanations — human "collaborators" can become less critical when shown an explanation (Bansal et al., 2021). Finally, a key challenge in user studies is selecting an appropriate participant pool with representative proxy tasks (Buçinca et al., 2020). For example, an interpretability method in healthcare may be more effective for patients than for doctors. Narrow tasks can more directly guide deployment while broader ones are easier to run at scale.

There is often overlap between benchmark datasets and user study evaluation. For example, Casper et al. (2023) created a dataset to understand the effectiveness of explanations in aiding model debugging. The dataset introduced distractor patches to a small fraction of images, leading to failures on the test set. For example, a small strawberry was pasted into 1 in 3000 images in both training and test sets, and in each "poisoned" training image, the class was switched to "goose," leading to failures at test time. Participants were presented with misclassified test images, along with various explanations. Their accuracy and speed in identifying the poisoning mechanism served to quantify the utility of these explanations for debugging.

Evidently, while there is no one way to evaluate interpretability methods, there is a natural hierarchy of tests to which proposals can be subjected. General-purpose proposals should first pass ablation and synthetic data benchmarks. Success in these criteria justifies investing resources in user studies with more realistic audience and task designs. Future interpretability methods may be evaluated similarly to new therapies in medicine: in silico associations (ablation and synthetic data studies) can guide in vitro experiments (user studies with generic audiences) which, if successful, lead to follow-up clinical trials (user studies with representative audiences).

4 Discussion

This review has surveyed the significant advances in interpretability over recent decades. In closing, we highlight key ongoing shifts in the AI landscape and their implications for interpretability. One important trend is the rise of multimodal models, which are trained on mixed data modalities (Ngiam et al., 2011; Chen et al., 2021). This has facilitated novel applications, like automatic open vocabulary segmentation (Zou et al., 2023), where, rather than annotating each image pixel with one of a preset collection of class labels, the system can generate segmentation masks for arbitrary text strings (e.g., "The pedestrian that is crossing the street."). While single modality methods remain applicable to individual model components (e.g,. analyzing the word embeddings in a vision language model), there is a pressing need for interpretability techniques that offer an integrated view of how these models merge information across sources. This could help developers train models that effectively balance evidence from complementary views, enhancing overall system robustness. Statistical methods for data integration (Ding et al., 2022) could offer a principled approach for analyzing models with these more complex, interacting components.

Another significant trend is the emergence of foundation models (Bommasani et al., 2021). These models are trained on corpora spanning petabytes of data, demonstrate a capacity for in-context learning. They can immediately solve tasks that previously required additional labeled data and fine-tuning. In this sense, foundation models are generalists, allowing users across diverse domains to benefit from a single, albeit intensive, modeling investment. Therefore, understanding how to interpret these models and their role in downstream specialized tasks is an important area for study. It is no longer wise to explain AI systems as if they were trained in isolation; they must be considered part of a broader AI ecosystem. Further, Xie et al. (2022) and Teh (2019) have pointed out connections between in-context learning and Bayesian posterior updating. This suggests that statistical thinking may resolve attribution questions when data are drawn from heterogeneous sources, which is important for resolving data provenance and ownership.

Third, as new legal and social norms around algorithmic decision-making are established, we will likely encounter scenarios where even comprehensive explanation are unsatisfactory, and where high-performing glass boxes become critical. In certain applications, reliability and interpretability will outweigh performance alone. For example, in healthcare and legal contexts, simple decision-making criteria are ideal, and in science, generalizable insight holds greater value than out-of-sample performance alone. Though recent progress in deep learning has sparked the community's imagination about potentially more human-like or human-surpassing systems, we should perhaps consider futures where systems only marginally outperform current ones but are significantly safer and transparent.

Finally, we ask: how can we make interpretability a more interactive process? Interactivity is critical for ensuring that our systems augment intelligence, empowering users to become creative and critical problem solvers. Neglecting this problem may lead to an overreliance on oracles, even among experts. For example, doctors should feel comfortable rejecting AI recommendations, even if there is a chance their decision turns out to be incorrect. A prerequisite for such interactivity is shared language (Heer, 2019; Crabbe et al., 2021; Kim, 2022) – it is difficult to interact without communication. CBMs are already a step in this direction, where concepts serve as a representation linking both human and machine reasoning. The goal of a shared language isn't as fantastical as it might seem; programming, after all, is an exercise in communicating with machines, with the programming language as the shared representation.

Looking ahead, we imagine a dynamic where interpretability is central to the interactive design and development of AI systems. Returning to our visualization analogy, it is already common for national newspapers to feature sophisticated interactive visualizations, and college freshman across various disciplines are taught data literacy and visual exploration. In the future, interpretability can help AI models be communicated and developed in a similarly democratic way, making it easier for data to shape human judgment and creativity.

Acknowledgement

This work was supported by award R01GM152744 from the National Institute of General Medical Sciences of the National Institutes of Health.

We also thank the organizers and participants of the BIRS Workshop, "Statistical Aspects of Trustworthy Machine Learning" for creative and critical discussion.

References

Adebayo J, Gilmer J, Muelly M, Goodfellow I, Hardt M, Kim B (2018). Sanity checks for saliency maps. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 9525–9536. Curran Associates Inc., Red Hook, NY, USA.

Agrawala M, Li W, Berthouzoz F (2011). Design principles for visual communication. *Commun. ACM*, 54(4): 60–69.

Bansal G, Wu T, Zhou J, Fok R, Nushi B, Kamar E, et al. (2021). Does the whole exceed its parts? the effect of ai explanations on complementary team performance. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21. ACM.

Bengio Y (2009). *Learning Deep Architectures for AI.* NOW.

Bengio Y, Courville A, Vincent P (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8): 1798–1828.

Bilodeau B, Jaques N, Koh PW, Kim B (2024). Impossibility theorems for feature attribution. *Proceedings of the National Academy of Sciences*, 121(2).

Bommasani R, Hudson DA, Adeli E, Altman R, Arora S, von Arx S, et al. (2021). On the opportunities and risks of foundation models.

Borkin MA, Vo AA, Bylinskii Z, Isola P, Sunkavalli S, Oliva A, et al. (2013). What makes a visualization memorable? *IEEE Transactions on Visualization and Computer Graphics*, 19(12): 2306–2315.

Buçinca Z, Lin P, Gajos KZ, Glassman EL (2020). Proxy tasks and subjective measures can be misleading in evaluating explainable ai systems. In: *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20. ACM.

Caruana R, Lou Y, Gehrke J, Koch P, Sturm M, Elhadad N (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, 1721–1730. Association for Computing Machinery, New York, NY, USA.

Casper S, Bu T, Li Y, Li J, Zhang K, Hariharan K, et al. (2023). Red Teaming Deep Neural Networks with Feature Synthesis Tools. In: *Advances in Neural Information Processing Systems* (A Oh, T Naumann, A Globerson, K Saenko, M Hardt, S Levine, eds.), volume 36, 80470–80516. Curran Associates, Inc.

Chen S, Guhur PL, Schmid C, Laptev I (2021). History Aware Multimodal Transformer for Vision-and-Language Navigation. In: *Advances in Neural Information Processing Systems* (M Ranzato, A Beygelzimer, Y Dauphin, PS Liang, JW Vaughan, eds.), volume 34, 5834–5847. Curran Associates, Inc.

Cleveland WS (1993). *Visualizing Data.* Hobart Press.

Coenen A, Reif E, Yuan A, Kim B, Pearce A, Viégas F, et al. (2019). *Visualizing and measuring the geometry of BERT.* Curran Associates Inc., Red Hook, NY, USA.

Colin J, FEL T, Cadene R, Serre T (2022). What I Cannot Predict, I Do Not Understand: A Human-Centered Evaluation Framework for Explainability Methods. In: *Advances in Neural Information Processing Systems* (S Koyejo, S Mohamed, A Agarwal, D Belgrave, K Cho, A Oh, eds.), volume 35, 2832–2845. Curran Associates, Inc.

Crabbe J, Qian Z, Imrie F, van der Schaar M (2021). Explaining Latent Representations with a Corpus of Examples. In: *Advances in Neural Information Processing Systems* (M Ranzato, A Beygelzimer, Y Dauphin, PS Liang, JW Vaughan, eds.), volume 34, 12154–12166. Curran Associates, Inc.

Devlin J, Chang MW, Lee K, Toutanova K (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (J Burstein, C Doran, T Solorio, eds.), 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota.

Ding DY, Li S, Narasimhan B, Tibshirani R (2022). Cooperative learning for multiview analysis. *Proceedings of the National Academy of Sciences*, 119(38).

Erhan D, Bengio Y, Courville A, Manzagol PA, Vincent P, Bengio S (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(19): 625–660.

Fridovich-Keil S, Yu A, Tancik M, Chen Q, Recht B, Kanazawa A (2022). Plenoxels: Radiance fields without neural networks. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Friedman J, Hastie T, Tibshirani R (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1).

Friedman JH (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5).

Ghorbani A, Wexler J, Zou J, Kim B (2019). *Towards automatic concept-based explanations.* Curran Associates Inc., Red Hook, NY, USA.

Gosmann C, Anahtar MN, Handley SA, Farcasanu M, Abu-Ali G, Bowman BA, et al. (2017). Lactobacillus-deficient cervicovaginal bacterial communities are associated with increased hiv acquisition in young south african women. *Immunity*, 46(1): 29–37.

Gu T, Liu K, Dolan-Gavitt B, Garg S (2019). Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7: 47230–47244.

Hazimeh H, Ponomareva N, Mol P, Tan Z, Mazumder R (2020). The tree ensemble layer: differentiability meets conditional computation. In: *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Heer J (2019). Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences*, 116(6): 1844–1850.

Hesse R, Schaub-Meyer S, Roth S (2023). FunnyBirds: A synthetic vision dataset for a part-based analysis of explainable AI methods. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE.

Hewitt J, Manning CD (2019). A structural probe for finding syntax in word representations. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (J Burstein, C Doran, T Solorio, eds.), 4129–4138. Association for Computational Linguistics, Minneapolis, Minnesota.

Holmes S (2017). Statistical proof? the problem of irreproducibility. *Bulletin of the American Mathematical Society*, 55(1): 31–55.

Hooker S, Erhan D, Kindermans PJ, Kim B (2019). A Benchmark for Interpretability Methods in Deep Neural Networks. In: *Advances in Neural Information Processing Systems* (H Wallach, H Larochelle, A Beygelzimer, Fd Alché-Buc, E Fox, R Garnett, eds.), volume 32. Curran Associates, Inc.

Huber PJ (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1): 73–101.

Hutchins EL, Hollan JD, Norman DA (1985). Direct manipulation interfaces. *Human–Computer Interaction*, 1(4): 311–338.

Jeyakumar JV, Noor J, Cheng YH, Garcia L, Srivastava M (2020). How Can I Explain This to You? An Empirical Study of Deep Neural Network Explanation Methods. In: *Advances in Neural Information Processing Systems*, volume 33, 4211–4222. Curran Associates, Inc.

Kim B (2022). Beyond Interpretability: Developing a Language to Shape Our Relationships with AI.

Koh PW, Liang P (2017). Understanding black-box predictions via influence functions. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 1885–1894. JMLR.org.

Koh PW, Nguyen T, Tang YS, Mussmann S, Pierson E, Kim B, et al. (2020). Concept bottleneck models.

Kostic AD, Gevers D, Siljander H, Vatanen T, Hyötyläinen T, Hämäläinen AM, et al. (2015). The dynamics of the human infant gut microbiome in development and in progression toward type 1 diabetes. *Cell Host and Microbe*, 17(2): 260–273.

Krishnan M (2019). Against interpretability: a critical examination of the interpretability problem in machine learning. *Philosophy and Technology*, 33(3): 487–502.

Kundaliya D (2023). Computing - incisive media: Google ai chatbot bard gives wrong answer in its first demo. *Computing*. Nom - OpenAI; Copyright - Copyright Newstex Feb 9, 2023; Dernière mise à jour - 2023-11-30.

Lee JS (2021). Transformers: a Primer.

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

Loh W (2014). Fifty years of classification and regression trees. *International Statistical Review*, 82(3): 329–348.

Lundberg SM, Lee SI (2017). A unified approach to interpreting model predictions. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 4768–4777. Curran Associates Inc., Red Hook, NY, USA.

Ma J, Lai V, Zhang Y, Chen C, Hamilton P, Ljubenkov D, et al. (2024). OpenHEXAI: An open-source framework for human-centered evaluation of explainable machine learning.

Murdoch WJ, Singh C, Kumbier K, Abbasi-Asl R, Yu B (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44): 22071–22080.

Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY (2011). Multimodal deep learning. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, 689–696. Omnipress, Madison, WI, USA.

Nguyen LH, Holmes S (2019). Ten quick tips for effective dimensionality reduction. *PLOS Computational Biology*, 15(6): e1006907.

Nussberger AM, Luo L, Celis LE, Crockett MJ (2022). Public attitudes value interpretability but prioritize accuracy in artificial intelligence. *Nature Communications*, 13(1).

Oppermann M, Munzner T (2022). Vizsnippets: Compressing visualization bundles into representative previews for browsing visualization collections. *IEEE Transactions on Visualization and Computer Graphics*, 28(1): 747–757.

Raghu M, Unterthiner T, Kornblith S, Zhang C, Dosovitskiy A (2021). Do Vision Transformers See Like Convolutional Neural Networks? In: *Advances in Neural Information Processing Systems* (M Ranzato, A Beygelzimer, Y Dauphin, PS Liang, JW Vaughan, eds.), volume 34, 12116–12128. Curran Associates, Inc.

Raghu M, Zhang C, Kleinberg J, Bengio S (2019). *Transfusion: understanding transfer learning for medical imaging*. Curran Associates Inc., Red Hook, NY, USA.

Recht B (2023). All models are wrong, but some are dangerous.

Ribeiro MT, Singh S, Guestrin C (2016). "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*, KDD '16, 1135–1144. Association for Computing Machinery, New York, NY, USA.

Rudin C (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215.

Sankaran K, Holmes SP (2023). Generative models: An interdisciplinary perspective. *Annual Review of Statistics and Its Application*, 10(1): 325–352.

Sedlmair M, Meyer M, Munzner T (2012). Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12): 2431–2440.

Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, 618–626.

Simonyan K, Vedaldi A, Zisserman A (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In: *Workshop at International Conference on Learning Representations*.

Sturmfels P, Lundberg S, Lee SI (2020). Visualizing the impact of feature attribution baselines. *Distill*, 5(1).

Sundararajan M, Taly A, Yan Q (2017). Axiomatic attribution for deep networks. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 3319–3328. JMLR.org.

Teh YW (2019). On Statistical Thinking in Deep Learning A Blog Post. *IMS Medallion Lecture*.

Tufte ER (2001). *The visual display of quantitative information*. Graphics Press, Cheshire, CT, 2 edition.

Tukey JW (1959). A survey of sampling from contaminated distributions. (33): 57.

Van Boxtel, GJM, et al (2021). *gsignal: Signal processing*.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. (2017). Attention is all you need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6000–6010. Curran Associates Inc., Red Hook, NY, USA.

Wang L, Shen Y, Peng S, Zhang S, Xiao X, Liu H, et al. (2022). A fine-grained interpretability evaluation benchmark for neural NLP. In: *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics, Stroudsburg, PA, USA.

Williamson B, Feng J (2020). Efficient nonparametric statistical inference on population feature importance using shapley values. In: *Proceedings of the 37th International Conference on Machine Learning* (HD III, A Singh, eds.), volume 119 of *Proceedings of Machine Learning Research*, 10282–10291. PMLR.

Wongsuphasawat K, Smilkov D, Wexler J, Wilson J, Mané D, Fritz D, et al. (2018). Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE Transactions on Visualization and Computer Graphics*, 24: 1–12.

Wu T, Ribeiro MT, Heer J, Weld D (2021). Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics.

Xie SM, Raghunathan A, Liang P, Ma T (2022). An explanation of in-context learning as implicit bayesian inference. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yosinski J, Clune J, Nguyen AM, Fuchs TJ, Lipson H (2015). Understanding neural networks through deep visualization. *ArXiv*, abs/1506.06579.

Yuksekgonul M, Wang M, Zou J (2023). Post-hoc concept bottleneck models. In: *The Eleventh International Conference on Learning Representations*.

Zeiler MD, Fergus R (2013). Visualizing and understanding convolutional networks. *ArXiv*, abs/1311.2901.

Zeng J, Ustun B, Rudin C (2016). Interpretable classification models for recidivism prediction. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 180(3): 689–722.

Zou X, Yang J, Zhang H, Li F, Li L, Wang J, et al. (2023). Segment Everything Everywhere All at Once. In: *Advances in Neural Information Processing Systems* (A Oh, T Naumann, A Globerson, K Saenko, M Hardt, S Levine, eds.), volume 36, 19769–19782. Curran Associates, Inc.