Are Your Explanations Leaking Your Label?

Neil Jethani [1] **Rajesh Ranganath** [2]

Abstract

Feature attribution methods have been used to help understand data, enabling the discovery of biomedical insights. However, recent work has surfaced many flaws with the methods that generate these attributions or explanations. To mitigate these flaws, we propose a set of evaluation metrics that can help users select the best explanation method. The explanation methods are evaluated on their ability to achieve two distinct goals — to either retrospectively explain the degree to which features support a given label or prospectively explain the degree to which features help predict the target. For example, popular methods like SHAP and GradCAM generate retrospective explanations for a particular label. We show that retrospectively using the label to generate an explanation can leak label information, misleading users by hiding predictive features that don't support the label. Further, we introduce a set of new methods, SHAP-KL and FastSHAP-KL, that generate Shapley values that prospectively indicate how predictive each feature is of the target. Finally, we take the metrics we develop and evaluate popular explanation methods, such as LIME and GradCAM, on three real-world clinical datasets. Our results indicate that retrospective and prospective evaluations differ significantly and that methods that provide high-quality retrospective explanations often do not offer high-quality prospective explanations. The code is available at https://tinyurl.com/45htfz8a.

1. Introduction

Explanations in machine learning have had a significant impact. For example, in healthcare, explanations have been used to detect spurious signals[1], discover novel gene expression signatures[2], and identify brain regions that help distinguish between possible sources of dementia[3]. There are many different types of explanations, and the number of explanations methods keep growing[4–14]. However, it is not clear which explanation methods are worthwhile, as numerous studies have identified potential flaws with each method[15–21]. In cases where people already know how to explain the data, one can defer to human judgment, though such cases are often less interesting.

A widely utilized class of explanation methods attribute scores to input features [1–3, 22–25]. For example, a practitioner may use such methods to highlight which chest x-ray regions are suggestive of pneumonia in each sample of data (local interpretability). This property is helpful because, with a chest x-ray, the positioning/size of the organs in the chest differs across individuals.

Unfortunately, it is not apparent which feature attribution method to use, and many different methods are actively used in practice. If one knows how their explanations will be applied, then they should evaluate the success of that application[26]. However, the intended application is often unclear, or the real-world evaluation is expensive to perform. Good, generic feature attribution should have some correspondence with the predictability of the target. Therefore, it is natural to formalize an evaluation by considering the probability distribution of the target as important features are included or excluded from the input. As important features are included in the input, the likelihood of the associated label should increase, while as important features are excluded from the input, the likelihood of the label should decrease. Therefore, we construct a curve of the chosen metric as features are included or excluded and use the area under the curve to aggregate this information.

The proposed evaluation framework can be applied to help answer a set of valuable questions that have not been distinguished from one another (see Figure 2). Firstly, one may be interested in discovering "which features most support a given label?" Alternatively, one may be interested in discovering "which features are most predictive of the target variable?" The former question *retrospectively* considers which features most explain a given label, while the latter *prospectively* interrogates which features best explain how the data is generated. For example, given patient reviews of their experience with their healthcare provider, a retrospective explanation for an overall positive review would present the text supporting a positive experience, whereas a prospective one would present the text that reduces uncertainty about the patients experience. Most feature attribution methods, such as GradCAM, SHAP, and LIME, provide retrospective explanations that explain a given label.

[1]NYU Grossman School of Medicine [2]Courant Institute, NYU. Correspondence to: Neil Jethani <neil.jethani@nyulangone.org>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Retrospective explanation methods can leak information about the chosen label by selectively attributing high importance to features that support the label, adversarially ignoring any features that suggest other possible outcomes. For example, given randomly assigned labels, these explanation methods can fabricate explanations for the labels. In Figure 1 we permuted the labels indicating the severity of diabetic retinopathy associated with retinal fundus images and find that retrospectively the discriminative performance as important features are sequentially included is surprisingly high. Whereas, the prospective evaluation reveals that these labels were generated incorrectly and cannot be predicted using the given features. We expose that, in practice, retrospective methods produce prospective explanations by explaining the predicted class, which may instead disregard features that contradict the predicted class. Therefore, we propose a set of prospective explanation methods, SHAP-KL and FastSHAP-KL, that attribute importance to features based on their ability to predict the target using Shapley values. Often one may prefer a prospective approach as it helps uncover which features reduce uncertainty about the distribution of the target, which is important when such features are unknown, such as in medicine[27, 28], biology[29, 30], and chemistry[31].

With this work, we make several contributions. **(1)** We introduce and explain the difference between retrospective and prospective explanations. **(2)** We propose an evaluation that allows practitioners to select the explanation method that provides the best retrospective or prospective explanations. **(3)** We present SHAP-KL and FastSHAP-KL as a set of methods for estimating Shapley values that provide prospective explanations. **(4)** We then evaluate popular feature attribution methods on a set of real-world clinical tasks involving a variety of high-dimensional data types, including imaging, biosignals, and text, and identify which methods tend to perform well. **(5)** Finally, we show that while some methods provide high-quality retrospective explanations, they may not offer high-quality prospective explanations. Therefore, much like practitioners evaluate their prediction models to select the best one, they should quantitatively evaluate their explanations to choose the best one.

2. Background

2.1. Post-hoc Feature Attribution Methods

Post-hoc feature attribution methods can be divided into removal-based methods and gradient-based methods. Removal-based methods remove different feature subsets in the input to characterize each feature's importance[32]. Many removal methods, such as LIME[7] and SHAP[5], perform the removal procedure separately for each sample of data — a computationally intensive process. SHAP and LIME solve a weighted-least square regression across different subsets sampled from a single sample of data, with the former approximating Shapley values. A few recent removal methods, known as amortized explanation methods, such as L2X[33], REAL-X[21], FastSHAP[34], etc.[35–37], learn an explanation model that outputs explanations for a sample of data with a single forward pass.

Gradient-based methods utilize gradients with respect to the input or intermediate representations of the input to attribute feature importances[38]. SmoothGrad[39], for example, attributes importance based on how sensitive the output is to small changes in the corresponding feature. Meanwhile, IntGrad[6], attributes importance by computing the average gradient to measure the salience of features in the input relative to a reference input. Another popular method, GradCAM[40], differs from many gradient-based methods. It computes the gradient with respect to an intermediate layer of a neural network and can only be used with convolutional neural networks, as the intermediate layer can be directly mapped onto the input. The computation performed by each explanation method is provided in Appendix A.1.

2.2. Issues with Post-hoc Explanation Methods

Many issues with post-hoc feature attribution methods have surfaced, causing some to question their utility[41, 42]. Gradient-based methods may produce explanations that appear invariant to both model and training label randomizations[15]. Other works have shown that explanations are sensitive to small changes or distributional shifts of the input. Adding a constant shift to the input data or adversarially altering the input can dramatically change the explanations produced by gradient-based methods[17, 18]. For many removal-based methods, replacing the removed features with a reference value shifts the input out-of-distribution/off-manifold, which can affect explanation quality and allow for adversarial attack[19, 20]. Further, some amortized explanation methods, such as L2X and INVASE, can learn explanation models that can simply encode the label directly in the explanation[21].

2.3. Related Work

Given all these potential issues, it may be dangerous to make conclusions about the data using post-hoc explanations without evaluating them. Multiple metrics have been proposed for evaluating explanations, each of which measures model performance as features are removed based on their attributions. The first few approaches sequentially perturbed the values of important image pixels and measured the predicted probabilities of the label[43, 44]. Most evaluation metrics follow this approach, choosing to sequentially remove the most important features/least important features[45, 46] or remove a fixed number of features[47]. Hooker et al. [16] noted that removing features shifts the inputs out of the distribution (OOD) on which the predic-

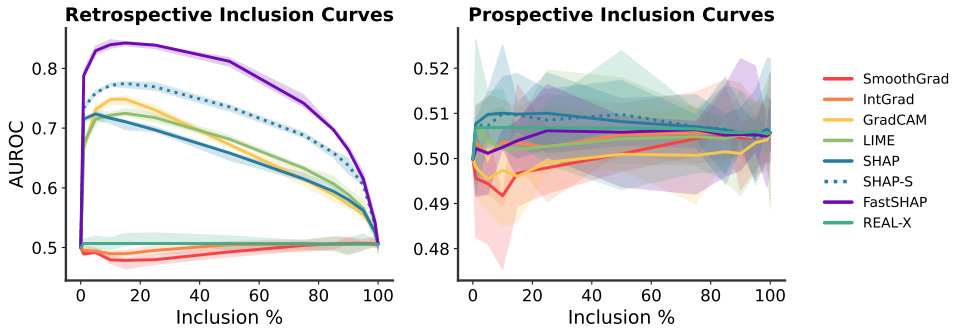


Figure 1. **Retrospective vs prospective evaluation using permuted labels.** The change in the area under the receiver operator curves (AUROC) as an increasing percentage of important features are included. When provided with the permuted label some methods fabricate retrospective explanations consistent with the randomly assigned labels (left). The prospective evaluation (right) reveals that the permuted labels cannot be predicted when the explanations are prevented from leaking the label.

tion model was trained and instead retrains models as the features are removed. However, because the attributions may directly encode the label, this retraining procedure allows the model to learn the label based on the location of the removed features as opposed to their actual values [21]. Recognizing this fact and the computational demands of such a procedure, multiple works have instead proposed evaluation metrics that rely on retraining a single model to approximate the conditional probability of the target given any subset of features[21, 34]. Other evaluations rely on synthetically generating ground truth explanations and measuring each explanation method's ability to attribute importance to these ground truth features[48, 49].

3. Retrospective vs Prospective

This section introduces the notation used throughout the paper and establishes the difference between retrospective and prospective explanations/evaluations. Let $\mathbf{x} \in \mathcal{X}$ be a random vector consisting of d features, or $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_d)$, and let $\mathbf{y} \in \mathcal{Y} = \{1, \ldots, K\}$ be the target variable for a classification problem. We use $\mathbf{s} \in \{0,1\}^d$ to denote subsets of the indices $\{1, \ldots, d\}$. The symbols $\mathbf{x}, \mathbf{y}, \mathbf{s}$ are random variables, and x, y, s denote possible values. Let $F(\mathbf{x}, \mathbf{y})$ be the population distribution from which data is drawn. $F_{\text{model}}(\mathbf{y} \mid \mathbf{x}; \theta) : \mathcal{X} \mapsto \Delta^{K-1}$ is a model that outputs a probability distribution over $\mathbf{y}$ given $\mathbf{x}$.

3.1. Retrospective vs Prospective Explanations

We define two different types of explanations — retrospective explanations and prospective explanations. Consider the following scenario alluded to in Figure 2. One is tasked with predicting whether or not a patient will be readmitted within 30 days based on their discharge summary. If one is told the patient was readmitted, they can review the discharge summary and present only the evidence supporting this outcome. This explanation may be useful when one is performing a root cause analysis, where they are investigating a medical incident to identify the essential factors contributing to an adverse event[50]. This is a retrospective explanation — knowledge of the label informs the explanation. However, suppose one is given a discharge summary and does not know whether or not the patient has been/will be readmitted. In that case, they may instead be interested in understanding which parts of the summary would help predict the outcome. This is a prospective explanation — the explanation identifies information in the input that reduces uncertainty about the target.

In the context of feature attribution, the goal of a retrospective explanation is to elucidate the degree to which features in the input make the observed label the more probable. A retrospective explanation is generated by an explanation method $e(x, y) : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}^d$ that generates an attribution vector $E \in \mathbb{R}^d$ or $E_i \in \mathbb{R}$ for each feature $i = 1, \ldots, d$ for each paired sample of data x, y. Alternately the goal of a prospective explanation is to elucidate the degree to which features in the input help predict the target variable. A prospective explanation is generated by an explanation method $e(x) : \mathcal{X} \mapsto \mathbb{R}^d$ that outputs an attribution vector for a given input x, without using the label y.

3.2. Retrospective Explanations Allow Label Leakage

Given our earlier readmission scenario, a retrospective explanation for a readmitted patient can selectively highlight words in the discharge note indicative of an adverse outcome, adversarially ignoring any words that support another outcome. For example, given a note saying "The pt denies chest pain," the explanation method may ignore the negation and selectively highlight "chest pain." In this way, the explanation can leak information about the label, misleading users by hiding predictive features that don't support the chosen label. To see this in practice, Figure 1 illustrates that, retrospectively, features can be adversarially selected according to their attributions to support even randomly assigned labels.

3.3. Retrospective vs Prospective Evaluation

In this section, we provide quantitative evaluations for either retrospective or prospective explanations — a retrospective

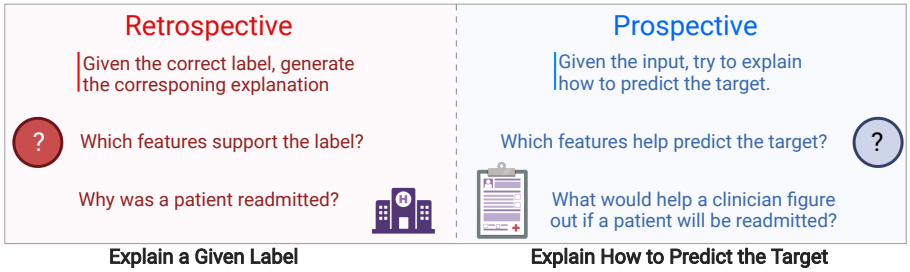


Figure 2. **Comparison of prospective vs retrospective explanations.**

evaluation and a prospective evaluation. For a retrospective evaluation, the explanation methods are given a label and generate the corresponding explanation. For a prospective evaluation, the explanations are generated using the input alone. The results obtained from a retrospective evaluation can differ quite substantially from a prospective evaluation. A retrospective evaluation permits the inclusion of the top $n\%$ of features such that the likelihood of the observed labels exceeds that given the full set of features, whereas a prospective evaluation does not. This phenomenon occurs because retrospective explanations can leak label information. However, during a prospective evaluation, the explanation method is prevented from using label information. The prospective evaluation effectively measures how well the features identified as important help predict the target, where the performance is upper bounded by how well the full feature set predicts the target. We provide proof of these claims in Appendix A.3.

3.3.1. Evaluation Curves

As with prior works[16, 21, 34, 43–47], we utilize evaluation metrics that consider the probability distribution of the target as important features are included or excluded from the input. To measure the probability of the target when a subset of the input is included/exlcuded we train a surrogate *evaluation* model $F_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x}, \mathbf{s}); \beta)$ to approximate its conditional distribution given any subset of features, learned by minimizing the following *Surrogate Objective*[21, 34]:

$$\mathcal{L}(\beta) = \mathop{\mathbb{E}}_{p(\mathbf{x})} \mathop{\mathbb{E}}_{p(\mathbf{s})} D_{\text{KL}}\big(F\left(\mathbf{y} \mid x\right) \,||\, F_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x}, \mathbf{s}); \beta)\big). \quad (1)$$

The surrogate evaluation model $F_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x}, \mathbf{s}); \beta)$ takes as input a vector of masked features $m(x, s)$, where the masking function m replaces features x_i such that $s_i = 0$ with a `[mask]` value that is not in the support of $\mathcal{X}$.

Let

$$\texttt{top}_n(E) = \arg\max_s s^T E,$$

such that $s \in \{0,1\}^d$, $\|s\| = nd$, and $n \in [0, 100]\%$, define a feature selection operation that returns a binary vector that denotes the top $n\%$ of features with the highest attributions $E_i \in \mathbb{R}$. Let $\mathcal{M}$ be a metric that measures the performance of a model, such as the area under the receiver operator curve ($\mathcal{M}_{auroc}$) or the likelihood ($\mathcal{M}_{lh}$) of the observed data. The inclusion curve is generated by progressively increasing n from 0% to 100% to sequentially select the top $n\%$ of features from a dataset using the corresponding explanations and measuring performance of the evaluator model $F_{surr}\Big(\mathbf{y} \mid m\big(x, \texttt{top}_n(E)\big); \beta\Big)$. The exclusion curve instead removes the top $n\%$ of features using the binary vector $\mathbf{1} - \texttt{top}_n(E)$.

To perform a retrospective evaluation of a given explanation method, one provides the explanation methods both x, y as inputs. If a prospective explanation method is being evaluated retrospectively, then the method simply does not use the label — $e(x, y) = e(x)$. To perform a prospective evaluation, one provides the explanation methods only with x as input.

We now define a set of retrospective and prospective evaluation metrics, using the area under the inclusion and exclusion curves to aggregate the information provided by these curves. The the area under the retrospective inclusion curve (iR-AUC$_{\mathcal{M}}$) is given by

$$\mathop{\mathbb{E}}_{t\sim\text{Unif}(0,1)} \mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})} \left[\mathcal{M}\Big(F_{\text{surr}}\Big(y \mid m\big(x, \texttt{top}_n\left(e\left(x, y\right)\right); \beta\big)\Big)\right].$$

The the area under the retrospective exclusion curve (iR-AUC$_{\mathcal{M}}$), replaces $\texttt{top}_n\big(e\left(x, y\right)\big)$ with $\mathbf{1} - \texttt{top}_n\big(e\left(x, y\right)\big)$. The corresponding prospective AUCs (iP-AUC$_{\mathcal{M}}$ and eP-AUC$_{\mathcal{M}}$) replace $e(x, y)$ with $e(x)$. This expectation is empirically estimated by plotting the model's performance on a dataset as the percentage of features included/excluded increases at discrete intervals from 0% to 100% and computing the area under the curve. We supply the full set of equations in Appendix A.2.

4. Prospective Explanation Methods

4.1. Making Retrospective Explanation Methods Prospective

The majority of explanations methods generate retrospective explanations as a function of both the label y and input x. For example, gradient-based methods, such as Integrated Gradients, SmoothGrad, and GradCAM, compute the gradient of a given class y with respect to the input x or an intermediate representation of the input. Similarly, many removal-based methods, such as LIME, SHAP, and Fast-

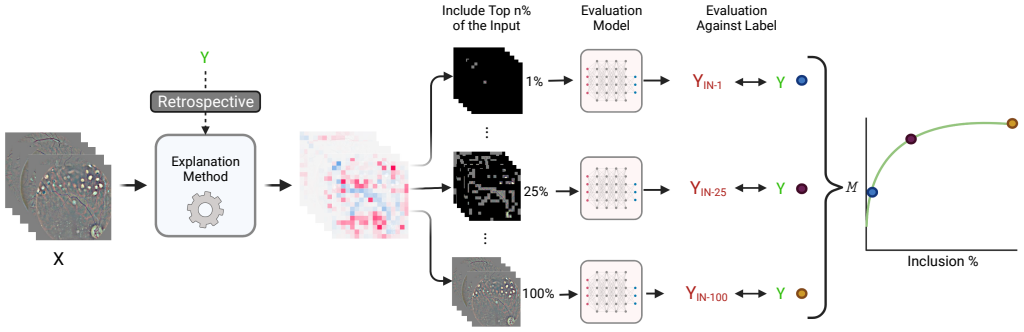


Figure 3. **Illustration of the evaluation framework.** The diagram describes how an inclusion curve is constructed by sequentially including the top features. During a retrospective evaluation the explanation is provided with the label, while during a prospective evaluation it is not.

SHAP, generate retrospective explanations for a particular class. SHAP attributes importance to each feature by averaging the value a feature provides to every possible subset of the input's features. Here the value of a feature subset is given by the model's output for the class y when evaluated on the feature subset, $m_{\text{R}}(x, s)$:

$$v_{x,y}(s) = F_{\text{model}}\left(y \mid m_{\text{R}}(x, s); \theta\right),$$

where $m_{\text{R}}(x, s)$ masks features by replacing them with a reference value (i.e., 0) or with samples from their marginal distributions. Since averaging across all possible feature subsets is prohibitively expensive, SHAP estimates the Shapley values using it's weighted least squares characterisation:

$$e_{\text{SHAP}}(x, y) = \arg\min_{\phi} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[\left(v_{x,y}(s) - s^T\phi - v_{x,y}(\mathbf{0})\right)^2\right] \quad (2)$$

The subset sampling distribution $p(\mathbf{s})$ and additional optimization constraints are presented in Appendix A.1. Appendix A.1 also depicts each of these methods as a function of x and y.

However, a prospective evaluation requires extracting prospective explanations from these retrospective explanation methods. In practice, these retrospective explanation methods have been converted into prospective explanations methods by using the model's prediction for the given input x:

$$e(x) = e\left(x, \arg\max_{y} F_{\text{model}}\left(y \mid x; \theta\right)\right). \quad (3)$$

While retrospective explanations methods can use the predicted class as a label to generate prospective explanations, these explanations may instead leak the predicted class and omit predictive features that don't support the predicted class. This can be a problem when there is uncertainty over what the label will be, as an explanation can make the predicted class appear more likely. Of note, if the label can be deterministically predicted by $F_{\text{model}}(\mathbf{y} \mid \mathbf{x}; \theta)$ such that given the input, the label is always correctly classified, then the explanations are equivalent.

4.2. Constructing Prospective Explanation Methods

Instead, prospective explanation methods should generate explanations by optimizing an objective that measures the degree to which features reduce the uncertainty about the target variable. Such explanation objectives can be crafted by measuring the distance between the $F\left(\mathbf{y} \mid x\right)$ and the distribution of the target when features are removed, $F\left(\mathbf{y} \mid x_s\right)$.

For example, this is the approach that REAL-X takes to learn the parameters ϕ of an explanation model $q_{\text{exp}}(\mathbf{s}|\mathbf{x}; \phi)$ that generates prospective explanations by minimizing the following objective:

$$\begin{aligned}\mathcal{L}_{\text{RX}}(\phi) &= \mathop{\mathbb{E}}_{F(\mathbf{x})} \mathop{\mathbb{E}}_{q_{\text{exp}}(\mathbf{s}|x;\phi)} \left[D_{\text{KL}}\left(F\left(\mathbf{y} \mid x; \theta\right) \,||\, F_{\text{surr}}(\mathbf{y} \mid m(x, s); \beta)\right)\right] \\ &= \mathop{\mathbb{E}}_{F(\mathbf{x})} \mathop{\mathbb{E}}_{q_{\text{exp}}(\mathbf{s}|x;\phi)} \mathop{\mathbb{E}}_{F(y|x;\theta)} \left[-\log F_{\text{surr}}\left(\mathbf{y} \mid m(x, s); \beta\right)\right] + \text{C}.\end{aligned}$$

The prospective explanations are then generated with a single forward pass through the explanation model, $e_{\text{REAL-X}}(x) = q_{\text{exp}}(\mathbf{s} \mid x; \phi)$. Note, $F_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x}, \mathbf{s}); \beta)$, learned by minimizing the Equation (1), is used to approximate replacing the removed features $\mathbf{x}_{\mathbf{1}-s}$ using their conditional distribution.

SHAP-KL and FastSHAP-KL. Using the approach alluded to above, we propose two new prospective explanation methods— SHAP-KL and FastSHAP-KL. We replace the value function for a given subset $v_{x,y}(s)$ with the KL-divergence between the distribution of the target given x and the distribution of the target given a subset of features, $m(x, s)$,

$$v_x(s) = -D_{\text{KL}}\left(F_{\text{surr}}\left(\mathbf{y} \mid x; \beta\right) \,||\, F_{\text{surr}}(\mathbf{y} \mid m(x, s); \beta)\right).$$

Note that the value function is not specific to given y.

SHAP-KL is, therefore, provided by the following objective:

$$e_{\text{SHAP-KL}}(x, y) = \arg\min_{\phi} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[\left(v_x(s) - s^T\phi - v_x(\mathbf{0})\right)^2\right]$$

Analogously, following Jethani et. al.[34], the explanation model for FastSHAP-KL can be learned by minimizing the following objective:

$$\mathcal{L}_{\text{FastSHAP-KL}}(\eta) = \mathop{\mathbb{E}}_{F(\mathbf{x})} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[\left(v_x(s) - \mathbf{s}^\top \phi_{\text{fast-kl}}(x;\eta) - v_x(\mathbf{0}) \right)^2 \right],$$

where prospective explanations are generated in a single forward-pass — $e_{\text{FastSHAP-KL}}(x) = \phi_{\text{fast-kl}}(x;\eta)$. For both objectives, the constraint and subset sampling distribution are the same as for SHAP and are presented in Appendix A.1.

5. Clinical Data Experiments

In healthcare, machine learning has been used to diagnose diseases and predict health outcomes from high-dimensional unstructured data[51–53]. There is an opportunity to explain discoveries that are beyond our current understanding of human physiology, such as the ability to assess for diabetes using ECG data[27]. For example, understanding which regions of the ECG help estimate a patient's likelihood of diabetes may offer insights into the underlying electrophysiological mechanism. Our study examines three high-dimensional medical data types: images, biosignals, and text.

5.1. Data Sources and Modeling

We selected an open-source medical dataset representative of each data type. We chose the PTB-XL dataset[54], where we classified right bundle branch block (RBBB) using ECG data using a 34-layer ResNet model adapted from[55] (see Appendix A.5 for an architecture diagram). For images, we utilized the EyePACs retinal fundus imaging dataset[56] to classify the presence and severity of diabetic retinopathy using a DenseNet121 model pre-trained on ImageNet. Lastly, we choose the MIMIC-III dataset[57] as our text dataset, where we predicted 30-day readmission using discharge summaries using a Bert transformer, implemented using the Hugginface for TensorFlow and pre-trained weights obtained from Alsentzer et al.[58]. The validation and test sets used for training were also used as explanation validation and test sets. For the EyePACs explanations sets, we randomly selected class-balanced subsets of 1000 and 2000 images for the explanation test and validation sets. We describe each dataset, data split, and model training procedure in Appendix A.4.

5.2. Explanation Methods

Our study compared ten explanation methods — GradCAM, Integrated Gradients (IntGrad), SmoothGrad, SHAP, SHAP-S, SHAP-KL, LIME, REAL-X, FastSHAP, and FastSHAP-KL. These methods represent many of the most commonly adopted feature attribution methods. We include REAL-X and FastSHAP as our set of amortized removal-based explanations methods that learn an explanation model that outputs explanations with a single forward pass. We also evaluate our methods SHAP-KL and FastSHAP-KL, paying particular attention to their ability to provide prospective explanations. SHAP and LIME replace the removed features with a baseline value (i.e., 0), while SHAP-S, SHAP-KL, REAL-X, FastSHAP, and FastSHAP-KL use $F_{\text{surr}}(\mathbf{y} \mid m(x,s);\beta)$ to simulate removing features using their conditional distributions.

We offer details on how explanations were generated and the set of five hyperparameter configurations tuned for each method in Appendix A.6. Of note, GradCAM can only be used with CNNs models. Therefore, we could not generate GradCAM explanations for the MIMIC-IV dataset using our BERT text classification model. Further, REAL-X failed to optimize on the MIMIC-IV dataset on the five hyperparameters tested. REAL-X uses score-function gradient optimization (often used in reinforcement learning), likely requiring additional tuning for the task.

5.3. Evaluation

Table 1 contains the retrospective and prospective inclusion and exclusion AUCs, where the likelihood of the true label is used as the metric $\mathcal{M}_{lh}$, for each method on all three datasets, and the corresponding curves are depicted in Figure 4. As important features are excluded, the likelihood of the label should decrease; therefore, lower exclusion AUCs (eR-AUC, eP-AUC) are preferable. Analogously, as important features are included, the likelihood of the label should increase; therefore, higher inclusion AUCs (iR-AUC, iP-AUC) are preferable. The likelihood is computed using the evaluation $F_{\text{surr}}(\mathbf{y} \mid m(x,s);\beta)$, trained separately from the surrogate model used to generate explanations to ensure the results aren't specific to a given surrogate.

Retrospective performance values differ from prospective performance values. In general, it is possible to obtain better retrospective performance than prospective performance. For both the EyePACs and MIMIC-IV datasets, this difference is quite pronounced. This is because retrospectively, an explanation method that is provided the label at explanation time can adversarially select only those features that maximize the probability of the corresponding label. This suggests that practitioners should perform the correct evaluation depending on whether they desire a prospective or retrospective understanding of their data. This difference is less drastic in for PTB-XL dataset because the presence of RBBB can be directly perceived from ECGs. When the label is determined entirely by the input, providing the explanation method with the label adds no additional information, $e(x) = e(x,y)$, and the evaluations are equivalent.

Good retrospective performance does not imply good prospective performance. For example, FastSHAP per-

Table 1. **Retrospective and prospective exclusion and inclusion AUCs.** Evaluation of each method on the basis of Exclusion AUC (lower is better) and Inclusion AUC (higher is better) calculated using the mean likelihood of the true label. Parentheses indicate 95% confidence intervals, and the best methods are bolded in each column, including those with overlapping confidence intervals.

		Retrospective		Prospective	
		eR-AUC	iR-AUC	eP-AUC	iP-AUC
PTB-XL: ECGs	GradCAM	0.763 (0.758, 0.769)	0.933 (0.931, 0.936)	0.762 (0.759, 0.765)	**0.930 (0.926, 0.934)**
	IntGrad	0.750 (0.745, 0.754)	0.964 (0.962, 0.966)	0.820 (0.813, 0.825)	0.882 (0.876, 0.888)
	SmoothGrad	0.778 (0.772, 0.783)	0.938 (0.936, 0.940)	0.838 (0.833, 0.848)	0.875 (0.873, 0.878)
	SHAP	0.761 (0.757, 0.767)	0.965 (0.963, 0.967)	0.803 (0.797, 0.811)	0.910 (0.905, 0.913)
	SHAP-S	**0.718 (0.714, 0.723)**	**0.971 (0.970, 0.972)**	0.766 (0.762, 0.770)	0.908 (0.904, 0.911)
	SHAP-KL	0.761 (0.758, 0.766)	0.926 (0.924, 0.931)	**0.761 (0.758, 0.766)**	**0.926 (0.924, 0.931)**
	LIME	0.744 (0.740, 0.749)	0.964 (0.964, 0.965)	0.791 (0.788, 0.793)	0.909 (0.901, 0.918)
	FastSHAP	**0.716 (0.711, 0.719)**	0.966 (0.965, 0.967)	**0.756 (0.754, 0.760)**	0.915 (0.911, 0.919)
	FastSHAP-KL	0.802 (0.794, 0.808)	0.927 (0.925, 0.930)	0.802 (0.794, 0.808)	**0.927 (0.925, 0.930)**
	REAL-X	0.774 (0.765, 0.780)	0.933 (0.929, 0.939)	0.774 (0.765, 0.780)	**0.933 (0.929, 0.939)**
EyePACs: Images	GradCAM	0.088 (0.086, 0.090)	0.283 (0.280, 0.288)	0.183 (0.181, 0.186)	0.140 (0.137, 0.142)
	IntGrad	0.089 (0.088, 0.090)	0.235 (0.234, 0.237)	0.088 (0.087, 0.090)	0.239 (0.236, 0.241)
	SmoothGrad	0.128 (0.127, 0.129)	0.186 (0.182, 0.188)	0.158 (0.156, 0.161)	0.184 (0.182, 0.186)
	SHAP	0.085 (0.084, 0.086)	0.309 (0.304, 0.312)	0.156 (0.155, 0.157)	0.204 (0.201, 0.206)
	SHAP-S	0.044 (0.043, 0.044)	0.339 (0.337, 0.340)	0.122 (0.120, 0.123)	0.198 (0.195, 0.203)
	SHAP-KL	0.074 (0.073, 0.075)	0.203 (0.200, 0.206)	**0.074 (0.073, 0.075)**	0.203 (0.200, 0.206)
	LIME	0.084 (0.082, 0.085)	0.318 (0.316, 0.321)	0.165 (0.162, 0.167)	0.204 (0.201, 0.207)
	FastSHAP	**0.036 (0.035, 0.036)**	**0.357 (0.355, 0.359)**	0.106 (0.105, 0.108)	0.157 (0.155, 0.158)
	FastSHAP-KL	0.096 (0.095, 0.097)	0.247 (0.243, 0.250)	0.096 (0.095, 0.097)	**0.247 (0.243, 0.250)**
	REAL-X	0.141 (0.139, 0.144)	0.170 (0.168, 0.172)	0.141 (0.139, 0.144)	0.170 (0.168, 0.172)
MIMIC-IV: Text	IntGrad	0.505 (0.504, 0.505)	0.531 (0.530, 0.532)	0.505 (0.505, 0.506)	0.530 (0.529, 0.531)
	SmoothGrad	0.506 (0.505, 0.507)	0.530 (0.530, 0.531)	0.506 (0.506, 0.507)	0.530 (0.529, 0.532)
	SHAP	0.372 (0.372, 0.372)	0.662 (0.661, 0.663)	0.471 (0.469, 0.473)	**0.540 (0.538, 0.543)**
	SHAP-S	**0.370 (0.370, 0.371)**	0.664 (0.663, 0.664)	**0.470 (0.469, 0.471)**	**0.541 (0.540, 0.543)**
	SHAP-KL	0.471 (0.470, 0.472)	0.540 (0.538, 0.542)	0.471 (0.470, 0.472)	**0.540 (0.538, 0.542)**
	LIME	0.365 (0.365, 0.366)	**0.667 (0.667, 0.668)**	**0.468 (0.467, 0.469)**	**0.542 (0.540, 0.543)**
	FastSHAP	0.409 (0.408, 0.409)	0.616 (0.615, 0.616)	0.479 (0.478, 0.481)	0.534 (0.534, 0.535)
	FastSHAP-KL	0.494 (0.493, 0.495)	0.530 (0.529, 0.530)	0.494 (0.493, 0.495)	0.530 (0.529, 0.530)

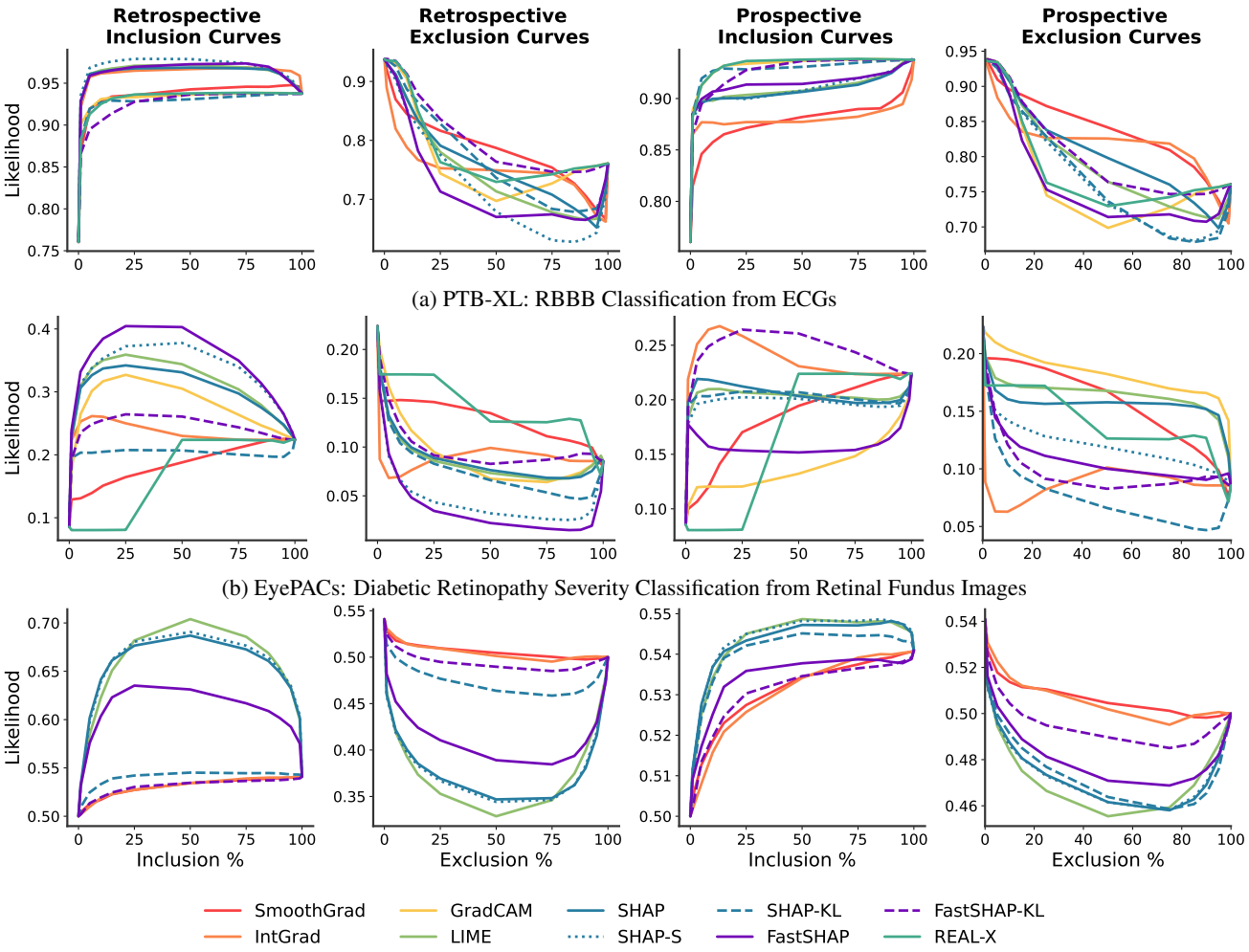
forms very well on the EyePACs dataset upon retrospective evaluation; however, it performs relatively poorly upon prospective evaluation. This suggests that even if a user validates that their attributions provide good retrospective explanations, they should not use these explanations to explain how to predict the target without further evaluation.

SHAP-S and FastSHAP tend to perform well retrospectively, and SHAP-KL and FastSHAP-KL tend to perform well prospectively. Across both metrics and on at least two of the three datasets both SHAP-S and FastSHAP achieve the best retrospective performance, while SHAP-KL and FastSHAP-KL achieve the best retrospective performance. FastSHAP is the amortized counterpart of SHAP-S. They both generate attributions by optimizing the same objective. They are both retrospective removal-based methods that explain the output for a specific class using Shapley values. They both remove features using a supervised surrogate model that simulates replacing the removed features using their conditional distributions.

SHAP-KL and FastSHAP-KL are the prospective counterparts of SHAP-S and FastSHAP. They value removed features based on their effect on the KL-divergence between the resulting distribution over the target variable $\mathbf{y}$ and the distribution given the full feature set. This suggests that if one desires prospective explanations, then a prospective explanation method that measures how well the target can be predicted should be evaluated. Meanwhile, converted retrospective methods may preferentially attribute high importance to features that support the model's prediction, which can hide predictive features that don't support the predicted class. Retrospective removal-based explanation methods perform may also perform well prospectiely and should be evaluated.

Gradient-based methods can occasionally perform well. GradCAM achieves a relatively high iP-AUC on the PTB-XL dataset, and IntGrad achieves the second-best iP-AUC on the EyePACs dataset. While in Section 2.2 we discuss the many problems that gradient-based methods may present with, our results indicate that these issues are not necessarily always present. An evaluation is required to ensure that this is the case.

No one method is always the best; an evaluation is required to select the best method. None of the methods perform the best across all datasets for any metric. Therefore, an evaluation is required for choosing the best explanation method. Much like practitioners evaluate their prediction models to select the best one, they should evaluate their explanations methods to choose the best one.



(a) PTB-XL: RBBB Classification from ECGs

(b) EyePACs: Diabetic Retinopathy Severity Classification from Retinal Fundus Images

(c) MIMIC-III: 30 Day Readmission Prediction from Discharge Summaries

Figure 4. **Inclusion and Exclusion Curves.** The likelihood of the label as an increasing percentage of the features estimated to be important either retrospectively or prospectively are excluded/included.

Explanation Efficiency. We include wall clock times for each explanation method on the explanation test set in Appendix A.10. Gradient-based methods are quite efficient, especially GradCAM, which only requires a single gradient estimate per explanation. Removal-based methods such as SHAP and LIME are slow. Meanwhile, the amortized methods incur a fixed training cost, which is often made up for by its meager marginal cost for generating each explanation. In some cases, the amortization also improves explanation quality. However, this may not always be the case, as the model may over-fit the data (i.e. FastSHAP on MIMIC-IV) or optimize poorly.

Limitations. Our evaluation relies on learning a model to approximate the distribution of the target given any subset of the input. In some cases, the shape of our prospective evaluation curve indicates that the model may optimize some subsets of a given size better than others. Interestingly, we found that the evaluation model does not underperform the original model on the full feature set (see Appendix A.9), where the training procedure may augment that data to improve model performance, such as on the EyePACs dataset. Additionally, we propose aggregating the results of the inclusion and exclusion curves using the AUC, however users may instead find that the performance at a specific point on the curve is more applicable for their use case (i.e., maximum or n=5%).

6. Conclusion

In this work, we differentiate between explanations that retrospectively explain how well features support a particular label or prospectively explain how well features predict the target and evaluate popular methods accordingly to illustrate this difference. While sometimes one may prefer retrospective explanations, we show they can leak label information, adversarially ignoring features that contradict the label. We also introduce SHAP-KL and FastSHAP-KL to attribute Shapley prospectively values and empirically verify that they identify features that help predict the target.

References

[1] Marcus A Badgeley, John R Zech, Luke Oakden-Rayner, Benjamin S Glicksberg, Manway Liu, William Gale, Michael V McConnell, Bethany Percha, Thomas M Snyder, and Joel T Dudley. Deep learning predicts hip fracture using confounding patient and healthcare variables. *NPJ digital medicine*, 2(1):1–10, 2019.

[2] Joseph D Janizek, Ayse Berceste Dincer, Safiye Celik, Hugh Chen, William Chen, Kamila Naxerova, and Su-In Lee. Uncovering expression signatures of synergistic drug response using an ensemble of explainable ai models. *bioRxiv*, 2021.

[3] Tomomichi Iizuka, Makoto Fukasawa, and Masashi Kameyama. Deep-learning-based imaging-classification identified cingulate island sign in dementia with lewy bodies. *Scientific reports*, 9(1):1–9, 2019.

[4] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[5] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

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

[7] 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.

[8] Been Kim, Cynthia Rudin, and Julie Shah. The bayesian case model: A generative approach for case-based reasoning and prototype classification. arxiv e-prints, art. *arXiv preprint arXiv:1503.01161*, 2015.

[9] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.

[10] Karthik S Gurumoorthy, Amit Dhurandhar, Guillermo Cecchi, and Charu Aggarwal. Efficient data representation by selecting prototypes with importance weights. arxiv e-prints, art. *arXiv preprint arXiv:1707.01212*, 2017.

[11] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[12] Jonathan Crabbé, Zhaozhi Qian, Fergus Imrie, and Mihaela van der Schaar. Explaining latent representations with a corpus of examples. *Advances in Neural Information Processing Systems*, 34, 2021.

[13] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[14] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

[15] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[16] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *Advances in neural information processing systems*, 32, 2019.

[17] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[18] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3681–3688, 2019.

[19] Christopher Frye, Damien de Mijolla, Tom Begley, Laurence Cowton, Megan Stanley, and Ilya Feige. Shapley explainability on the data manifold. In *International Conference on Learning Representations*, 2021.

[20] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[21] Neil Jethani, Mukund Sudarshan, Yindalon Aphinyanaphongs, and Rajesh Ranganath. Have we learned to explain?: How interpretability methods can

learn to encode predictions in their interpretations. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1459–1467. PMLR, 13–15 Apr 2021.

[22] Scott M Lundberg, Bala Nair, Monica S Vavilala, Mayumi Horibe, Michael J Eisses, Trevor Adams, David E Liston, Daniel King-Wai Low, Shu-Fang Newman, Jerry Kim, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature biomedical engineering*, 2(10):749–760, 2018.

[23] Dongdong Zhang, Samuel Yang, Xiaohui Yuan, and Ping Zhang. Interpretable deep learning for automatic diagnosis of 12-lead electrocardiogram. *Iscience*, 24(4):102373, 2021.

[24] Hisaki Makimoto, Moritz Höckmann, Tina Lin, David Glöckner, Shqipe Gerguri, Lukas Clasen, Jan Schmidt, Athena Assadi-Schmidt, Alexandru Bejinariu, Patrick Müller, et al. Performance of a convolutional neural network derived from an ecg database in recognizing myocardial infarction. *Scientific reports*, 10(1):1–9, 2020.

[25] Simon Meyer Lauritsen, Mads Kristensen, Mathias Vassard Olsen, Morten Skaarup Larsen, Katrine Meyer Lauritsen, Marianne Johansson Jørgensen, Jeppe Lange, and Bo Thiesson. Explainable artificial intelligence model to predict acute critical illness from electronic health records. *Nature communications*, 11(1):1–11, 2020.

[26] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[27] Neil Jethani, Aahlad Puli, Hao Zhang, Leonid Garber, Lior Jankelson, Yindalon Aphinyanaphongs, and Rajesh Ranganath. New-onset diabetes assessment using artificial intelligence-enhanced electrocardiography. *arXiv*, 2022.

[28] Sushravya Raghunath, Alvaro E Ulloa Cerna, Linyuan Jing, David P VanMaanen, Joshua Stough, Dustin N Hartzel, Joseph B Leader, H Lester Kirchner, Martin C Stumpe, Ashraf Hafez, et al. Prediction of mortality from 12-lead electrocardiogram voltage data using a deep neural network. *Nature medicine*, 26(6):886–891, 2020.

[29] Shilu Zhang, Deborah Chasman, Sara Knaack, and Sushmita Roy. In silico prediction of high-resolution hi-c interaction matrices. *Nature communications*, 10(1):1–18, 2019.

[30] Remzi Celebi, Oliver Bear Don't Walk, Rajiv Movva, Semih Alpsoy, and Michel Dumontier. In-silico prediction of synergistic anti-cancer drug combinations using multi-omics data. *Scientific Reports*, 9(1):1–10, 2019.

[31] Dávid Péter Kovács, William McCorkindale, and Alpha A Lee. Quantitative interpretation explains machine learning models for chemical reaction prediction and uncovers bias. *Nature communications*, 12(1):1–9, 2021.

[32] 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.

[33] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 883–892. PMLR, 2018.

[34] Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: Real-time shapley value estimation. In *International Conference on Learning Representations*, 2022.

[35] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision*, pages 3429–3437, 2017.

[36] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. Invase: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2018.

[37] Patrick Schwab and Walter Karlen. Cxplain: Causal explanations for model interpretation under uncertainty. *arXiv preprint arXiv:1910.12336*, 2019.

[38] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Gradient-based attribution methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 169–191. Springer, 2019.

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

[40] RR Selvaraju, M Cogswell, A Das, R Vedantam, D Parikh, and D Batra. Grad-cam: visual explanations from deep networks via gradient-based localization. arxiv. *Preprint posted online*, 7, 2016.

[41] Cynthia Rudin. Stop explaining black box machine

learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[42] Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L Beam. The false hope of current approaches to explainable artificial intelligence in health care. *The Lancet Digital Health*, 3(11):e745–e750, 2021.

[43] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 2017.

[44] 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.

[45] Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*, 2017.

[46] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[47] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

[48] Mengjiao Yang and Been Kim. Benchmarking attribution methods with relative feature importance. *arXiv preprint arXiv:1907.09701*, 2019.

[49] Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. Do feature attribution methods correctly attribute features? *arXiv preprint arXiv:2104.14403*, 2021.

[50] Juan A Sanchez, Kevin W Lobdell, Susan D Moffatt-Bruce, and James I Fann. Investigating the causes of adverse events. *The Annals of Thoracic Surgery*, 103(6):1693–1699, 2017.

[51] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[52] Eric J Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature medicine*, 25(1):44–56, 2019.

[53] Andre Esteva, Katherine Chou, Serena Yeung, Nikhil Naik, Ali Madani, Ali Mottaghi, Yun Liu, Eric Topol, Jeff Dean, and Richard Socher. Deep learning-enabled medical computer vision. *NPJ digital medicine*, 4(1):1–9, 2021.

[54] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I Lunze, Wojciech Samek, and Tobias Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific Data*, 7(1):1–15, 2020.

[55] Awni Y Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H Tison, Codie Bourn, Mintu P Turakhia, and Andrew Y Ng. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature medicine*, 25(1):65–69, 2019.

[56] Ben Graham. Kaggle diabetic retinopathy detection competition report. *University of Warwick*, 2015.

[57] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[58] Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

[59] 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.

[60] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[61] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*, 2019.

[62] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

A. Appendix

A.1. Review of Explanation Methods:

A.1.1. Gradient-based Methods

Gradient-based methods have been widely adopted due to the ease and efficiency with which gradients can be computed. They compute gradients with respect to the input or intermediate representations of the input to attribute feature importances[38]. Gradients explain how sensitive the model's output is to changes in the input (or intermediate representation). Such method popular include GradCAM[40], Integrated Gradients[6], and SmoothGrad[39], among others[44, 59]. SmoothGrad[39], for example, attributes importance based on how sensitive the output is to small changes in the corresponding feature, where the output is smoothed through the introduction of Gaussian noise as follows:

$$e_{\text{SG}}(x, y) = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial F_{\text{model}}\left(y \mid x + \mathcal{N}(0, \sigma^2); \theta\right)}{\partial x}.$$

Meanwhile, IntGrad[6] attributes importance by computing the average gradient to measure the salience of features in the input relative to a reference input $\bar{x}$ as follows:

$$e_{\text{IG}}(x, y) = (x - \bar{x}) \odot \frac{1}{n} \sum_{i=1}^{n} \frac{\partial F_{\text{model}}\left(y \mid \bar{x} + \frac{i}{n}(x - \bar{x}); \theta\right)}{\partial \left(\bar{x} + \frac{i}{n}(x - \bar{x})\right)}.$$

Another popular method, GradCAM[40], differs from many gradient-based methods. It computes the gradient with respect to an intermediate representation of the input learned by the model $A(x; \theta)$. This method can only be used with convolutional neural networks (CNN), as the structure of CNNs uniquely allow the representation to be directly mapped onto the input. GradCAM computes explanations as follows:

$$e_{\text{GradCAM}}(x, y) = \text{ReLu}\left[\sum_{k}^{c} \left(\frac{1}{hw} \sum_{i-1}^{h} \sum_{j=1}^{w} \frac{\partial F_{\text{model}}\left(y \mid x; \theta\right)}{\partial A(x; \theta)_{i,j}^{k}}\right) A(x; \theta)^{k}\right],$$

where in this case $A(x; \theta)$ is a c-channel, h by w, two-dimensional convolutional layer. Gradients computed for an intermediate representation of the input provide an understanding of the sensitivity of the output to changes in this latent space.

A.1.2. Removal-based Methods

Popular removal-based explanation methods include LIME[7] and SHAP[5], among many others[4, 35]. Both LIME and SHAP work by solving independent optimization problems for each sample of data. They solve a weighted-least square regression across different subsets sampled from a single sample of data, with the former approximating Shapley values. SHAP computes Shapley values in the following way:

$$e_{\text{SHAP}}(x, y) = \arg\min_{\phi} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[\left(F_{\text{model}}\left(y \mid m(x, s); \theta\right) - s^T \phi - F(y)\right)^2\right]$$

$$p(s) \propto \frac{d - 1}{\binom{d}{\mathbf{1}^\top s} \cdot \mathbf{1}^\top s \cdot (d - \mathbf{1}^\top s)} \qquad \text{(Shapley kernel)}$$

Similarly, LIME computes it's feature attributions using a measure of distance $\mathcal{D}$ and attribution complexity Ω as follows:

$$e_{\text{LIME}}(x, y) = \arg\min_{\phi} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[\left(F_{\text{model}}\left(y \mid m(x, s); \theta\right) - s^T \phi - F(y)\right)^2\right] + \Omega(\phi)$$

$$p(s) \propto \mathcal{D}(m(x, s), x). \qquad \text{(LIME kernel)}$$

These objectives make a few things clear. Firstly, the optimization is performed for a single sample of data x, y. Secondly, the computation requires sampling numerous subsets from $p(s)$ and removing features in the input using a masking function $m(x, s)$, which replaces the removed features with a reference value for high-dimensional data.

Care should be taken when using removal-based methods, specifically concerning how features are removed. Replacing the removed features with a reference value shifts the input out-of-distribution/off-manifold, which can affect explanation quality

and allow for adversarial attack[19, 20]. To address this issue, SHAP-S has been introduced to to approximate replacing the removed features $\mathbf{x}_{\mathbf{1}-s}$ using their conditional distribution using a surrogate learned by minimizing the Equation (1). SHAP-S computes Shapley values in the following way:

$$e_{\text{SHAP-S}}(x, y) = \arg\min_{\phi} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[\left(F_{\text{surr}}\left(y \mid m(x, s); \beta\right) - s^T \phi - F(y) \right)^2 \right]$$

$$p(s) \propto \frac{d-1}{\binom{d}{\mathbf{1}^\top s} \cdot \mathbf{1}^\top s \cdot (d - \mathbf{1}^\top s)}.$$

This is the same objective as SHAP except F_{model} is replaced with F_{surr}.

A.1.3. Amortized Removal-based Explanation Methods:

The removal-based methods mentioned above perform their optimization procedure separately for each sample of data. This procedure can become computational expensive, especially when explaining a dataset with many samples of data. The methods that we will discuss in this section amortize the cost of providing explanations by learning an explanation model that efficiently attributes importance to each feature in an instance of data with a single forward pass. Amortized explanation methods learn an explanation model that share parameters across samples of data when optimizing an objective that measures explanation quality. FastSHAP, for example, builds upon the SHAP-S objective to learn an explanation model that outputs Shapley values with a single forward pass. The follow objective is used to train the explanation model:

$$\mathcal{L}_{\text{FastSHAP}}(\eta) = \mathop{\mathbb{E}}_{F(\mathbf{x})} \mathop{\mathbb{E}}_{\text{Unif}(\mathbf{y})} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[\left(F_{\text{surr}}\left(y \mid m(x, s); \beta\right) - s^T \phi_{\text{explanation}}(x, y; \eta) - F(y) \right)^2 \right]$$

$$p(s) \propto \frac{d-1}{\binom{d}{\mathbf{1}^\top s} \cdot \mathbf{1}^\top s \cdot (d - \mathbf{1}^\top s)}.$$

Explanations can then be computed for a given sample of data as follows:

$$e_{\text{FastSHAP}}(x, y) = \phi_{\text{explanation}}(x, y; \eta).$$

All of the aforementioned explanation methods produce and explanation retrospectively as a function of x and y, whereas REAL-X, another amortized explanation method, produces explanations prospectively as a function of only x. For a given sample of data the goal of REAL-X is to return a sufficiently small subset of features in a given input that best predicts the target and reduces the uncertainty about the target variable. This objective is amortized by learning an explanation model that outputs a distribution over subset selections by sharing parameters across sample of data as follows:

$$\begin{aligned}
\mathcal{L}_{\text{REAL-X}}(\phi) &= \mathop{\mathbb{E}}_{F(\mathbf{x})} \mathop{\mathbb{E}}_{q_{\text{exp}}(\mathbf{s}|x;\phi)} \left[D_{\text{KL}}\big(F\left(\mathbf{y} \mid x; \theta\right) \,||\, F_{\text{surr}}(\mathbf{y} \mid m(x, s); \beta)\big] \right. \\
&= \mathop{\mathbb{E}}_{F(\mathbf{x})} \mathop{\mathbb{E}}_{q_{\text{exp}}(\mathbf{s}|x;\phi)} \mathop{\mathbb{E}}_{F(y|x;\theta)} \left[-\log F_{\text{surr}}\left(\mathbf{y} \mid m(x, s); \beta\right) \right] + \text{Const.}
\end{aligned}$$

Explanations can then be computed for a given sample of data as follows:

$$e_{\text{REAL-X}}(x) = q_{\text{exp}}(\mathbf{s} \mid x; \phi).$$

From this equation, it is clear that REAL-X explanations are generated using a function of x alone.

A.2. PR AUCs

The the areas under the retrospective inclusion curve (iR-AUC$_{\mathcal{M}}$) and exclusion curve (eR-AUC$_{\mathcal{M}}$) are computed using the following equations:

$$\text{iR-AUC}_{\mathcal{M}} = \mathop{\mathbb{E}}_{t\sim\text{Unif}(0,1)} \mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})} \left[\mathcal{M}\left(F_{\text{surr}}\Big(y \mid m\big(x, \texttt{top}_n\left(e\left(x,y\right)\right)\big); \beta \Big) \right) \right] \tag{4}$$

$$\text{eR-AUC}_{\mathcal{M}} = \mathop{\mathbb{E}}_{t\sim\text{Unif}(0,1)} \mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})} \left[\mathcal{M}\left(F_{\text{surr}}\Big(y \mid m\big(x, \mathbf{1} - \texttt{top}_n\left(e\left(x,y\right)\right)\big); \beta \Big) \right) \right]. \tag{5}$$

Whereas, the areas under the prospective inclusion curve (iP-AUC$_{\mathcal{M}}$) and exclusion curve (eP-AUC$_{\mathcal{M}}$) are computed using the following equations:

$$\text{iP-AUC}_{\mathcal{M}} = \mathop{\mathbb{E}}_{t\sim\text{Unif}(0,1)} \mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})} \left[\mathcal{M}\left(F_{\text{surr}}\Big(y \mid m\big(x, \texttt{top}_n\left(e\left(x\right)\right)\big); \beta \Big) \right) \right] \tag{6}$$

$$\text{eP-AUC}_{\mathcal{M}} = \mathop{\mathbb{E}}_{t\sim\text{Unif}(0,1)} \mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})} \left[\mathcal{M}\left(F_{\text{surr}}\Big(y \mid m\big(x, \mathbf{1} - \texttt{top}_n\left(e\left(x\right)\right)\big); \beta \Big) \right) \right]. \tag{7}$$

The metrics are identical aside from the type of explanation method used: $e(x)$ or $e(x, y)$.

A.3. Proof of Label Leakage with a Retrospective Evaluation

The results obtained from a retrospective evaluation can differ quite substantially from a prospective evaluation. This difference occurs because a retrospective explanation can leak information about the label. The following lemma states that under a retrospective evaluation, the likelihood of the labels when the top n% of features are included can exceed the likelihood when all the features are included, whereas this cannot occur during a prospective evaluation.

Lemma 1. *If $F_{surr}(\mathbf{y} \mid m(\mathbf{x}, s); \beta)\big)$ is equal in distribution to $F(\mathbf{y} \mid \mathbf{x}_s)$ for all $s \in \{0,1\}^d$, then there exists a retrospective explanation method $e(x, y)$ and data generating distribution $x, y \sim F(x, y)$ such that*

$$\mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F_{surr}\Big(y \mid m\Big(x, \mathtt{top}_n\big(e\left(x, y\right)\big)\Big); \beta\Big)\right] > \mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F\left(y \mid x\right)\right]$$

for some $n \in [0, 100]\%$. However, there does not exists any prospective explanation method $e(x)$ where this condition holds for any $F(x, y)$.

Proof. There are two parts to this proof. The first part focuses on providing an example where a retrospective method identifies a subset of features for which the likelihood of label exceeds that of the full feature set. Consider the the following data generating process for the input $\mathbf{x} := \{\mathbf{x}_1, \mathbf{x}_2\}$ and target $\mathbf{y}$:

$$x_1 \sim \text{Uniform}(0, 1), \quad x_2 = \frac{1}{2}, \quad y \sim \text{Bernoulli}\left(\frac{\mathbf{x}_1 + \mathbf{x}_2}{2}\right),$$

and the following retrospective explanation method:

$$e(x, y) = \begin{cases} \begin{bmatrix} 0 & 1 \end{bmatrix} & x_1 < 0.5, y = 1 \\ \begin{bmatrix} 1 & 0 \end{bmatrix} & x_1 \geq 0.5, y = 1 \\ \begin{bmatrix} 1 & 0 \end{bmatrix} & x_1 \leq 0.5, y = 0 \\ \begin{bmatrix} 0 & 1 \end{bmatrix} & x_1 > 0.5, y = 0 \end{cases}$$

Then for $n = 50\%$ (inclusion of a single feature),

$$F_{\text{surr}}\Big(y \mid m\Big(x, \mathtt{top}_{50\%}\big(e\left(x, y\right)\big)\Big); \beta\Big) = \begin{cases} \frac{1}{2} & x_1 < 0.5, y = 1 \\ \frac{x_1}{2} + \frac{1}{4} & x_1 \geq 0.5, y = 1 \\ \frac{3}{4} - \frac{x_1}{2} & x_1 \leq 0.5, y = 0 \\ \frac{1}{2} & x_1 > 0.5, y = 0 \end{cases}$$

Whereas, for the full feature set,

$$F\left(y \mid x\right) = \begin{cases} \frac{x_1}{2} + \frac{1}{4} & y = 1 \\ \frac{3}{4} - \frac{x_1}{2} & y = 0 \end{cases}$$

Therefore,

$$\begin{aligned} F_{\text{surr}}\Big(y \mid m\big(x, \mathtt{top}_{50\%}\left(e\left(x, y\right)\right)\big); \beta\Big) &> F\left(y \mid x\right) \quad \text{if} \quad x_1 < 0.5, y = 1 \\ F_{\text{surr}}\Big(y \mid m\big(x, \mathtt{top}_{50\%}\left(e\left(x, y\right)\right)\big); \beta\Big) &= F\left(y \mid x\right) \quad \text{if} \quad x_1 \geq 0.5, y = 1 \\ F_{\text{surr}}\Big(y \mid m\big(x, \mathtt{top}_{50\%}\left(e\left(x, y\right)\right)\big); \beta\Big) &= F\left(y \mid x\right) \quad \text{if} \quad x_1 \leq 0.5, y = 0 \\ F_{\text{surr}}\Big(y \mid m\big(x, \mathtt{top}_{50\%}\left(e\left(x, y\right)\right)\big); \beta\Big) &> F\left(y \mid x\right) \quad \text{if} \quad x_1 > 0.5, y = 0 \end{aligned}.$$

Since the in all cases $F_{\text{surr}}\left(y \mid m\left(x, \texttt{top}_{50\%}\left(e\left(x, y\right)\right)\right)\right) \geq F\left(y \mid x\right)$ and the events $x_1 < 0.5, y = 1$ and $x_1 > 0.5, y = 0$ occur with non-zero probability,

$$\mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F_{\text{surr}}\Big(y \mid m\Big(x, \texttt{top}_{50\%}\big(e\left(x, y\right)\big)\Big); \beta\Big)\right] > \mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F\left(y \mid x\right)\right].$$

The second part of this proof focuses on proving that during a prospective evaluation that assesses the quality of prospective explanations $e(x)$ the likelihood given the top n% of features cannot exceed that given then full feature. This is because prospective explanations prevent the leakage of label information. More formally, the generative process by which the explanations are created can be expressed via the following markov chain:

$$\mathbf{y} \rightarrow \mathbf{x} \rightarrow m\Big(\mathbf{x}, \texttt{top}_n\big(e\left(\mathbf{x}\right)\big)\Big),$$

where the target generates the input, which then generates the explanation that is used to mask/include features. Then according to the data processing inequality,

$$I(\mathbf{x}; \mathbf{y}) \geq I(m\left(\mathbf{x}, \texttt{top}_n\left(e\left(\mathbf{x}\right)\right)\right); \mathbf{y}) \quad \forall n \in [0, 100]\%,$$

which states that the mutual information content between an input and the target cannot be increased by processing the input. Rewriting the mutual information in terms of the conditional entropy produces

$$H(\mathbf{y} \mid \mathbf{x}) \leq H(\mathbf{y} \mid m\left(\mathbf{x}, \texttt{top}_n\left(e\left(\mathbf{x}\right)\right)\right)) \quad \forall n \in [0, 100]\%.$$

Then, using the definition of conditional entropy the inequality can be written as an expectation.

$$\begin{aligned}
\mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F\left(y \mid x\right)\right] &\geq \mathop{\mathbb{E}}_{F\left(m\left(\mathbf{x}, \texttt{top}_n\left(e\left(\mathbf{x}\right)\right)\right), \mathbf{y}\right)}\left[F\Big(y \mid m\Big(x, \texttt{top}_n\big(e\left(x\right)\big)\Big)\Big)\right] \\
\mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F\left(y \mid x\right)\right] &\geq \mathop{\mathbb{E}}_{F\left(m\left(\mathbf{x}, \texttt{top}_n\left(e\left(\mathbf{x}\right)\right)\right), \mathbf{y}\right)} \mathop{\mathbb{E}}_{F\left(\mathbf{x} \mid m\left(\mathbf{x}, \texttt{top}_n\left(e\left(\mathbf{x}\right)\right)\right), \mathbf{y}\right)}\left[F\Big(y \mid m\Big(x, \texttt{top}_n\big(e\left(x\right)\big)\Big)\Big)\right] \\
\mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F\left(y \mid x\right)\right] &\geq \mathop{\mathbb{E}}_{F\left(\mathbf{x}, m\left(\mathbf{x}, \texttt{top}_n\left(e\left(\mathbf{x}\right)\right)\right), \mathbf{y}\right)}\left[F\Big(y \mid m\Big(x, \texttt{top}_n\big(e\left(x\right)\big)\Big)\Big)\right] \\
\mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F\left(y \mid x\right)\right] &\geq \mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})} \mathop{\mathbb{E}}_{F\left(m\left(\mathbf{x}, \texttt{top}_n\left(e\left(\mathbf{x}\right)\right)\right) \mid \mathbf{x}, \mathbf{y}\right)}\left[F\Big(y \mid m\Big(x, \texttt{top}_n\big(e\left(x\right)\big)\Big)\Big)\right].
\end{aligned}$$

Finally, under our assumption that $F_{\text{surr}}\big(\mathbf{y} \mid m(\mathbf{x}, s); \beta\big)$ is equal in distribution to $F(\mathbf{y} \mid \mathbf{x}_s)$ for all $s \in \{0, 1\}^d$ the conditional entropy can be writing in terms of an expectation over the data generating distribution $F(\mathbf{x}, \mathbf{y})$, yielding

$$\mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F\left(y \mid x\right)\right] \geq \mathop{\mathbb{E}}_{F(\mathbf{x},\mathbf{y})}\left[F_{\text{surr}}\Big(y \mid m\Big(x, \texttt{top}_n\big(e\left(x\right)\big)\Big)\Big)\right] \quad \forall n \in [0, 100]\%.$$

Therefore, the prospective evaluation effectively measures how well the features identified as important help to predict the target, where the performance is upper bounded by how well the full feature set predicts the target.

A.4. Detailed Data Sources and Modeling

The PTB-XL dataset consists of $21,430$ 12-Lead/10s ECGs. We sought to classify the presence of right bundle branch block (RBBB) from lead VI of the ECG. There was a 7.75% prevalence of RBBB in the dataset. The dataset was split into training, validation, and test sets according to an $8:1:1$ ratio. The validation and test sets were directly used as the explanation validation and test sets ($2,163$ ECGs). We trained a 34-layer ResNet model adapted from[55] to classify the presence of a RBBB (see Appendix A.5 for an architecture diagram). The PTB-XL data is made available under the Creative Commons Attribution 4.0 International Public License.

The EyePACs dataset consists of $88,702$ retinal fundus images. The task is formulated as a multi-class classification labeled with the presence and severity of diabetic retinopathy. The label distribution was 73.5% normal, 7% mild, 15% moderate, 2.5% severe, and 2% proliferative. The dataset was split into training, validation, and test sets according to a $4:1:5$ ratio. The dataset was downloaded from Kaggle[1] and processed with TensorFlow Datasets[60] according to the 2015 Kaggle competition winner[56] to generate 544 by 544 pixel images. Class balanced explanation validation (1000 images) and test sets (2500 images) were randomly sampled from the validation and test sets, respectively. We trained a DenseNet121 model, pre-trained on ImageNet, to classify the severity of diabetic retinopathy. The EyePACs dataset is made available under a set of rules found here: https://www.kaggle.com/competitions/diabetic-retinopathy-detection/rules.

We processed the MIMIC-IV dataset according to Huang et al.[61], yielding a cohort of $34,560$ patient admissions with $2,963$ positive 30-day readmission labels and $42,358$ negative labels. As detailed by Huang et al.[61], the dataset was balanced to yield a final dataset of $5,926$ discharge summaries and split into training, validation, and test sets according to an $8:1:1$ ratio. The validation and test sets were directly used as the explanation validation and test sets (584 discharge summaries). The discharge summaries were split into 128 token segments and tokenized using the Huggingface[62] Bert tokenizer. We trained Bert transformer, implemented using the Hugginface for TensorFlow and pre-trained weights obtained from Alsentzer et al.[58], to predict 30-day readmission. The MIMIC-IV data is made available under the PhysioNet Credentialed Health Data License 1.5.0.

Both the DenseNet121 and ResNet models were trained for 100 epochs using Adam with a learning rate of 10^{-3} and a batch size of 32. The BERT model was trained for 50 epochs using Adam with a learning rate of $2 * 10^{-5}$ and a batch size of 16. We used a learning rate scheduler that multiplied the learning rate by a factor of 0.95 after three epochs of no validation loss improvement. Early stopping was triggered after the validation loss ceased to improve for ten epochs.

[1] https://www.kaggle.com/c/diabetic-retinopathy-detection/data

A.5. ECG Model Architecture

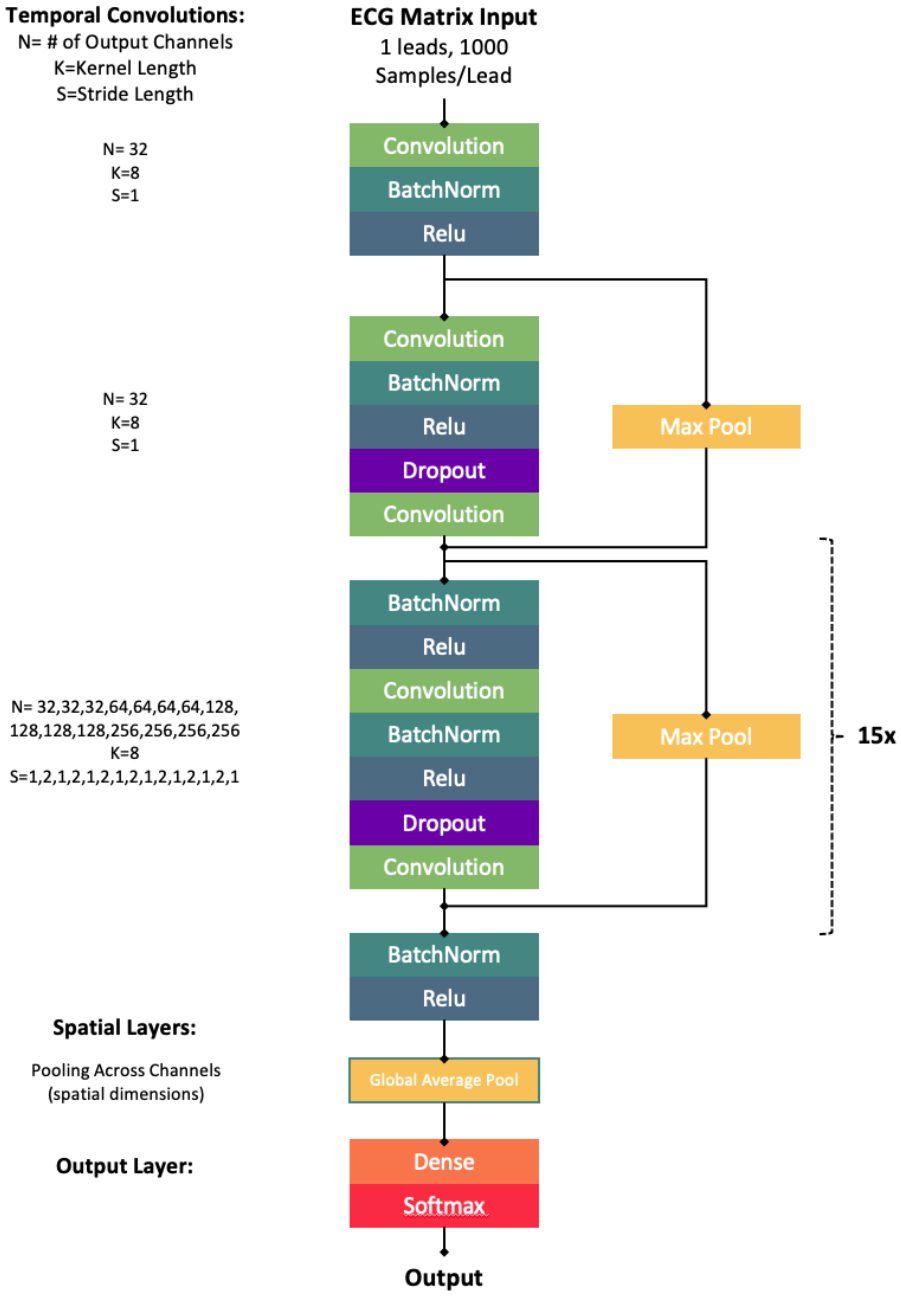


Figure 5. **Diagram of the ECG model architecture.**

A.6. Explanation Generation

The explanation validation set was used to tune each explanation method's hyperparameter, where the best performing explanation set for the associated metric was selected. The gradient-based methods were implemented using TensorFlow's native backprop functionality. Both IntGrad and SmoothGrad have a single hyperparameter that controls the number of samples along the path from the baseline input to the final input and the number of noisy inputs to sample, respectively. We tunned this hyperparamater for both explanation methods across $64, 128, 256, 512$, and 1024 samples. GradCAM does not have any hyperparameters; however, it can only be applied to CNNs models. Therefore, we could not generate GradCAM explanations for our BERT text classification model.

Feature removal was performed at different granularities for the removal-based methods depending on the data type. We removed segments of the input instead of individual features — 32 by 32 super-pixels segments for images, 0.08 second segments, and tokens for text. We implement LIME and SHAP using their respective open-source packages, where feature removal is simulated by replacing the removed features with a baseline value for all three data types. We simply alter SHAP's value function to implement SHAP-S and SHAP-KL. Image and ECG data were replaced using the zero baseline, while text data was replaced using the `[MASK]` token. Both LIME, SHAP, SHAP-S, and SHAP-KL have a single hyperparameter that controls the number of feature subsets to sample. We tunned this hyperparamater for the explanation methods across $512, 1024, 2048, 4096$, and 8192 subset samples. LIME[2] is made available under the BSD 2-Clause "Simplified" License. SHAP[3] is made available under the MIT License.

Explanation generation with REAL-X, FastSHAP, and FastSHAP-KL involves a three-step process: 1) training a surrogate model to simulate feature removal, 2) training an explanation model, and 3) computing explanations with a single forward pass through the explanation model. We trained the surrogate and explanations models and tuned them using the same training and validation sets we used to train the original model. The surrogate model was trained with random feature removals, where the removed features were replaced with their aforementioned baseline. The surrogate model simulates marginalizing out features from the original model with their conditional distribution. The surrogate model's training procedure and model architecture directly mirrored that of the corresponding original model. We adapted the explanation model architectures from their corresponding classification models being explained by truncating the architectures (see Appendix A.7 for details). For FastSHAP and FastSHAP-KL, we tunned the hyperparameter controlling the number of feature subsets to sample for each input in a mini-batch explanation model across $1, 2, 4, 8, 16$. For REAL-X, we tunned the regularization hyperparameter that enforces sparse subset selections of the input across $10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}$, and 10^{-1}. All other training hyperparameters used to train the original models were conserved when training and tunning the explanation models. FastSHAP[4] and REAL-X[5] are both made available under the MIT License.

[2] https://github.com/marcotcr/lime
[3] https://github.com/slundberg/shap
[4] https://github.com/neiljethani/fastshap
[5] https://github.com/rajesh-lab/realx

A.7. Explanation Model Architectures

A.7.1. ECG Explanation Model

We modified the ECG model architecture (see Figure 5) to return a tensor of size 125×1 for REAL-X/FastSHAP-KL and 125×2 (one for each class) for FastSHAP. For the 10s ECG's input size of 1000×1, this process provides 0.08 second segment explanations. First, the layers after the 6^{th} residual connection were removed; the output of this block was 125×64. We then appended a 1D convolutional layer with filters of size 1 × 1, one filter for REAL-X/FastSHAP-KL and 2 filters for FastSHAP, such that the output was 125×1 or 125×2 respectively. For FastSHAP, the y^{th} 125 dimensional array slice corresponded to the segment-level Shapley values for the class $y \in \{0, 1\}$.

A.8. Retinal Fundus Image Explanation Model

We modified the DenseNet121 architecture to return a tensor of size $17 \times 17 \times 1$ for REAL-X/FastSHAP-KL and $17 \times 17 \times 5$ (one for each class) for FastSHAP. For a input image size of 544×544 this process provides 32×32 super-pixel explanations. First, the classification layers (global average pooling and fully-connected layers) were removed; the output of this block was $17 \times 17 \times 1024$. We then appended a 2D convolutional layer with filters of size 1 × 1, one filter for REAL-X/FastSHAP-KL and 5 filters for FastSHAP, such that the output is $17 \times 17 \times 1$ or $17 \times 17 \times 5$ respectively. For FastSHAP, the y^{th} 17×17 slice corresponded to the superpixel-level Shapley values for the class $y \in \{0, 1, 2, 3, 4\}$.

A.8.1. Discharge Summary Explanation Model

We simply modified the BERT architecture by appending two fully connected layers to the output for the last encoder layer. The output of the final encoder layer was a 128×768 tensor, such that there was a 768 dimensional array outputted for each input token. We first appended a fully connected layer with 768 units, GeLU activation, and layer norm. Then, we appended another fully connected layer with either 1 unit for REAL-X/FastSHAP-KL or 2 units for FastSHAP, yielding a 128×1 or 128×2 output. This output provided attributions for each token in the input text segment. For FastSHAP, the y^{th} 128 dimensional slice corresponded to the token-level Shapley values for the class $y \in \{0, 1\}$.

A.9. Evaluation Model vs. Original Prediction Model Performance

Table 2. **AUROC of the original prediction model compared to the evaluation model.** The prediction model is trained using the full feature set while the surrogate evaluation model is trained using random subsets of the input. The performance of each model on the full feature set is compared using the AUROC (micro-averaged for Eye-PACs). Model performance is negligibly affected by randomly removing subets of the input during training.

	$F_{\text{model}}(\mathbf{y} \mid \mathbf{x}; \theta)$	$F_{\text{eval}}(\mathbf{y} \mid \mathbf{x}; \alpha)$
PTB-XL	0.997	0.997
Eye-PACs	0.947	0.951
MIMIC-IV	0.775	0.774

A.10. Explanation Efficiency

Table 3. **Training and explanation run-times for the explanation sets (in minutes).**

		PTB-XL	EyePACs	MIMIC-III
	# of Samples	2163	2500	3063
Explain	GradCAM	56.39	31.11	—
	IntGrad	15.82	3902.53	717.18
	SmoothGrad	63.17	2398.13	415.74
	SHAP	143.02	6389.12	1343.03
	SHAP-S	150.49	6909.33	1354.13
	SHAP-KL	144.47	6348.32	1356.01
	LIME	138.08	6649.19	1340.14
	FastSHAP	0.01	0.66	0.30
	FastSHAP-KL	0.02	0.51	0.27
	REAL-X	0.01	0.51	—
Train	FastSHAP	54.67	3597.14	2038.53
	FastSHAP-KL	76.51	2918.45	3320.96
	REAL-X	89.27	3611.14	—
	SHAP-S	16.73	1320.97	38.63
	SHAP-KL	16.73	1320.97	38.63