Explanation-based Training with Differentiable Insertion/Deletion Metric-aware Regularizers

Yuya Yoshikawa
STAIR Lab, Chiba Institute of Technology

Tomoharu Iwata
NTT Corporation

Abstract

The quality of explanations for the predictions made by complex machine learning predictors is often measured using insertion and deletion metrics, which assess the faithfulness of the explanations, i.e., how accurately the explanations reflect the predictor's behavior. To improve the faithfulness, we propose insertion/deletion metric-aware explanation-based optimization (ID-ExpO), which optimizes differentiable predictors to improve both the insertion and deletion scores of the explanations while maintaining their predictive accuracy. Because the original insertion and deletion metrics are non-differentiable with respect to the explanations and directly unavailable for gradient-based optimization, we extend the metrics so that they are differentiable and use them to formalize insertion and deletion metric-based regularizers. Our experimental results on image and tabular datasets show that the deep neural network-based predictors that are fine-tuned using ID-ExpO enable popular post-hoc explainers to produce more faithful and easier-to-interpret explanations while maintaining high predictive accuracy. The code is available at `https://github.com/yuyay/idexpo`.

1 Introduction

Complex machine learning predictors, such as deep neural networks (DNNs), have become indispensable components of many modern AI systems because of their remarkable predictive accuracy. In addition to having high predictive accuracy, it has been crucial in AI systems for medical diagnostics (Holzinger et al., 2017), autonomous driving (Omeiza et al., 2022), cybersecurity (Capuano et al., 2022), e-commerce (Y. Zhang et al., 2020), etc., to explain the rationale for the predictor's behavior so that users can trust the AI system. Such explanations also help researchers and developers to identify flaws caused by biases in training datasets (Stein et al., 2023) and the errors in implementation and modeling (Lertvittayakumjorn et al., 2021).

To understand the behavior of a predictor, it is crucial to know what features are essential to the individual predictions produced by the predictor and to what extent they are essential. To obtain the explanations, researchers and practitioners rely on post-hoc explainers or use inherently interpretable predictors instead of opaque predictors. Popular post-hoc explainers include local interpretable model-agnostic explanations (LIME) (Ribeiro et al., 2016), kernel Shapley additive explanations (KernelSHAP) (Lundberg et al., 2017), and gradient-weighted class activation mapping (Grad-CAM) (Selvaraju et al., 2020). One advantage of using such post-hoc explainers is that they can focus on developing predictors that achieve the highest accuracy because they place few constraints on the architecture of the predictors. Inherently interpretable predictors make predictions and offer explanations for those predictions in a single model. These include classical models, such as generalized additive models (Hastie et al., 1986) and recent DNN-based models (Alvarez Melis et al., 2018).

Explanations for predictions can be obtained using the above approaches. Are these explanations appropriate? There are several evaluation metrics that assess, from different perspectives, the quality of explanations. How correctly explanations reflect the predictor's behavior is measured with *faithfulness* of the explanations. To evaluate the faithfulness, *insertion* and *deletion metrics* have been widely used in the literature (Petsiuk et al., 2018; Gevaert et al., 2022). Intuitively, these metrics are calculated on the assumption that if the features, e.g., the pixels in an image, which are deemed important

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

to the explanation are truly important to the predictor, the presence or absence of the features should strongly affect the output of the predictor. If the insertion and deletion scores are both excellent, then we assume the explanation is faithful to the predictor.

The present study enables explainers to generate more faithful explanations with better insertion and deletion scores. To this end, we propose insertion/deletion metric-aware explanation-based optimization (ID-ExpO), which is a framework for optimizing predictors to improve both the insertion and deletion scores of the explanations produced by the explainers while maintaining the predictive accuracy of the predictors. Because the original insertion and deletion metrics are non-differentiable with respect to the explanations and are directly unavailable for gradient-based optimization, we extend the metrics so that they are differentiable, and we use them to formalize the insertion and deletion metric-based regularizers. By optimizing the predictors based on the standard prediction loss together with our regularizers simultaneously, ID-ExpO equips the predictors with capabilities that both produce accurate predictions and enable the explainers to produce more faithful explanations. ID-ExpO can be applied to both post-hoc explainers and inherently interpretable models because it does not require any change in the architecture of the predictors. In general, the post-hoc explainers are modeled differently than predictors. For example, the LIME explainer approximates the predictor's behaviors using linear models around individual input data points, and the Grad-CAM explainer produces the explanation (saliency map) by aggregating the feature maps of the predictor's internal layer differently from the inference process of the predictor. Owing to these differences, the explanations by the post-hoc explainers are likely not to reflect the actual feature contributions in the predictor. Therefore, in this study, we focus on employing ID-ExpO to improve the post-hoc explainers and present its implementations for perturbation-based explainers, such as LIME and KernelSHAP, and gradient-based explainers, such as Grad-CAM.

In experiments on image and tabular datasets, we demonstrate the effectiveness of fine-tuning DNN predictors based on ID-ExpO compared with that of the existing stability-aware and fidelity-aware explanation-based optimization (Plumb et al., 2020) and the standard fine-tuning approach. The experimental results show that ID-ExpO significantly improves the insertion and deletion scores on all the datasets while maintaining high predictive accuracy. In a qualitative evaluation, we show that, by calculating our regularizers with only the top 30% or 50% of important features in explanations, the explanations for the predictor trained using ID-ExpO highlight the appropriate parts of features well.

2 Related Work

Existing Explainers. To interpret feature contributions in the individual predictions of complex machine learning models, various types of post-hoc explainers have been proposed, such as gradient-based explainers (including some of the CAM-based ones) (Selvaraju et al., 2020; Chattopadhay et al., 2018; Fu et al., 2020; Jiang et al., 2021), perturbation-based explainers (Ribeiro et al., 2016; Lundberg et al., 2017; Zhao et al., 2021), and occlusion-based explainers (Petsiuk et al., 2018; Wang et al., 2020). One advantage of using the proposed method is that it enables an improvement in the explanations by the existing post-hoc explainers without changing their formulation, as long as the explainers are differentiable. Some studies on the post-hoc explainers have proposed approaches that optimize or select explanations so that the features that are deemed important to the explanations contribute to better prediction (Petsiuk et al., 2018; Fong et al., 2019; H. Zhang et al., 2023). However, H. Zhang et al. (2023) reported that their proposed method, one of those explainers, did not improve the insertion and deletion scores. Another approach for this purpose is to use inherently interpretable predictors (Molnar, 2022), including generalized additive models (Hastie et al., 1986) and DNN-based models (Alvarez Melis et al., 2018; Agarwal et al., 2021; Yoshikawa et al., 2022), which can achieve both high predictive accuracy and transparency. Several studies have proposed inherently interpretable DNN-based predictors whose attention maps and feature weights, which produce explanations, affect predictions and optimize them to improve the predictive accuracy (Schwab et al., 2018; Fukui et al., 2019; Iida et al., 2022). Because explanations that lead to higher accuracy do not always result in better insertion and deletion scores, it is expected that the proposed method will also be helpful for the inherently interpretable predictors.

Evaluation Metrics for Explanation. The ground truths of explanations are rarely observed because they are inside the complex predictor we would like to understand. Therefore, many studies have assessed explanations quantitatively using various proxy evaluation metrics (Zhou et al., 2021). In computer vision literature, insertion and deletion metrics are widely used to evaluate the faithfulness of the explanations (Petsiuk et al., 2018; Gevaert et al., 2022). Several evaluation metrics that are related to insertion and deletion metrics have been proposed, such as sensitivity-n (Ancona et al., 2018), increase and drop rates (Chattopadhay et al., 2018; Ramaswamy et al., 2020), and the iterative

removal of features (IROF) (Rieger et al., 2020). For tabular data, stability (Alvarez Melis et al., 2018), sensitivity (Ghorbani et al., 2019), and faithfulness (Bhatt et al., 2021), which is another formulation for the insertion and deletion metrics, have been measured. Although insertion and deletion metrics have not been used frequently for tabular data, employing these metrics on tabular data is beneficial for evaluating the combinatorial effects of features on prediction.

Explanation-Based Optimization. Our study was inspired by the work of Plumb et al. (2020). They proposed an explanation-based optimization to improve the fidelity (Ribeiro et al., 2016) and stability (Alvarez Melis et al., 2018) of the explanations produced by perturbation-based post-hoc explainers for tabular data. The main differences between the proposed method and their method are threefold: The proposed method 1) aims to improve the faithfulness of the explanations by optimizing the insertion and deletion metrics, which are different from the fidelity and stability metrics; 2) is applicable to several data types, including images and tabular data; and 3) is effective for both perturbation-based and gradient-based explainers. Ismail et al. (2021) proposed saliency-guided training, which optimized predictors such that the predictions between an original input and an input in which some of the pixels were masked according to the gradient-based explanations were similar. Unlike our method, their method did not optimize the explanations to improve the insertion and deletion metrics.

3 Proposed Method

For the sake of concreteness, we consider a multiclass image classification task because the insertion and deletion metrics for this kind of task are widely used in the image domain. Note that the proposed method can be applied to other data, such as text and tabular data, with slight changes.

Problem Formulation. We are given an image $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{C \times H \times W}$ of the number of channels C, height H, and width W, which can be classified to a class within the set of classes $\mathcal{Y} = \{1, 2, \cdots, L\}$. In addition, we have a pretrained trained predictor $f_\theta : \mathcal{X} \to [0, 1]^L$, e.g., a deep neural network, which outputs the probabilities of the classes, where θ is the set of model parameters. We assume that the outputs of f_θ are normalized by the softmax function. Next, we want to explain the prediction $\hat{y} = \mathrm{argmax}_l f_\theta(\boldsymbol{x})_l$ produced by f_θ using the given post-hoc explainer e, e.g., LIME, KernelSHAP, and Grad-CAM. Here, the post-hoc explainer e outputs pixel-level contributions (an explanation) $\boldsymbol{\phi}^{\hat{y}} \in \mathbb{R}^{H \times W}$ for image $\boldsymbol{x}$ and predicted label $\hat{y}$, i.e., $\boldsymbol{\phi}^{\hat{y}} = e(\boldsymbol{x}, f_\theta; \hat{y})$, where a larger positive value within $\boldsymbol{\phi}^{\hat{y}}$ means that its corresponding pixel is of greater importance to label $\hat{y}$. In addition, in LIME and KernelSHAP, a large negative value in $\boldsymbol{\phi}^{\hat{y}}$ indicates that its corresponding pixel is not associated with label $\hat{y}$. When we do not need to specify the label, we denote the explanation as $\boldsymbol{\phi} \in \mathbb{R}^{H \times W}$.

With the above setup, our goal is to optimize (i.e., fine-tune) predictor f_θ on labeled data to force post-hoc explainer e to produce faithful explanations with the best insertion and deletion scores while maintaining the predictor's inherent predictive capability. To achieve our goal, we assume that predictor f_θ and post-hoc explainer e are differentiable. This assumption is commonly used because the DNN predictors trained through backpropagation are differentiable, and most post-hoc explainers, including perturbation-based, gradient-based, CAM-based, and occlusion-based explainers, are also differentiable.

Preliminaries: Insertion and Deletion Metrics. The insertion and deletion metrics are widely used to assess the faithfulness of an explanation. In particular, the insertion metric evaluates the increase in the predicted probability for a target label when pixels that are deemed important to the explanation are gradually added to a blank image. Conversely, the deletion metric evaluates the decrease in the predicted probability for the target label when such important pixels are gradually deleted from the input image. Therefore, if the insertion score is high and the deletion score is low, we can say the explanation is faithful to the predictor. More formally, for image $\boldsymbol{x} \in \mathcal{X}$ and label $y \in \mathcal{Y}$, the insertion and deletion metrics are defined as follows:

$$\texttt{Ins}_S(\boldsymbol{x}, y, \boldsymbol{\phi}^y, \boldsymbol{b}; f_\theta) = \frac{1}{S} \sum_{s=1}^{S} f_\theta(\alpha(\boldsymbol{x}; \boldsymbol{\phi}^y, \boldsymbol{b}, s))_y, \quad (1)$$

$$\alpha(\boldsymbol{x}; \boldsymbol{\phi}^y, \boldsymbol{b}, s)_{ijk} = \begin{cases} x_{ijk}, & (j,k) \in \text{arg top-}s(\boldsymbol{\phi}^y) \\ b_{ijk}, & \text{otherwise} \end{cases}, \quad (2)$$

$$\texttt{Del}_S(\boldsymbol{x}, y, \boldsymbol{\phi}^y, \boldsymbol{b}; f_\theta) = \frac{1}{S} \sum_{s=1}^{S} f_\theta(\beta(\boldsymbol{x}; \boldsymbol{\phi}^y, \boldsymbol{b}, s))_y, \quad (3)$$

$$\beta(\boldsymbol{x}; \boldsymbol{\phi}^y, \boldsymbol{b}, s)_{ijk} = \begin{cases} b_{ijk}, & (j,k) \in \text{arg top-}s(\boldsymbol{\phi}^y) \\ x_{ijk}, & \text{otherwise} \end{cases}, \quad (4)$$

where $S \in \{1, 2, \cdots, HW\}$ is the number of pixels that are used to evaluate these metrics. Although S is typically set to HW, in some studies, S is a smaller value than HW, e.g., 3.6%, 30%, and 50% of HW (Q. Zhang et al., 2021; Huber et al., 2023; Zeng et al., 2022), because some of the pixels in the image are often critical to correct classification. Here, s can be incremented by a positive integer that is larger than one to reduce

the number of times that f_θ is applied. arg top-$s(\boldsymbol{\phi}^y)$ outputs a set of s pairs of coordinate indices with the top-s values in $\boldsymbol{\phi}^y$. $\boldsymbol{b} \in \mathcal{X}$ is background values, e.g., channel-wise mean values calculated over all the training images. $\alpha : \mathcal{X} \to \mathcal{X}$ and $\beta : \mathcal{X} \to \mathcal{X}$ are mask functions that mask a part of the pixels according to $\boldsymbol{\phi}^y$ and replace them with background values $\boldsymbol{b}$. The difference between these metrics lies in the choice of the mask functions. The insertion metric masks the pixels other than arg top-$s(\boldsymbol{\phi}^y)$ using α, whereas the deletion metric masks the pixels in arg top-$s(\boldsymbol{\phi}^y)$ using β. The insertion and deletion scores range between 0 and 1, where higher insertion scores and lower deletion scores, respectively, are better.

3.1 Insertion/Deletion Metric-Aware Explanation-Based Optimization (ID-ExpO)

Although the insertion and deletion metrics are used to assess explainers in the literature, we consider using them to optimize predictor f_θ such that the explanations by explainer e have higher insertion scores and lower deletion scores. We expect that such an optimization will furnish two benefits: 1) the explainer can produce feature contributions that are more faithful to the predictor's behaviors; 2) the explainer can clearly separate the important pixels from the less important ones in the explanation. However, as these metrics are non-differentiable with respect to $\boldsymbol{\phi}$ due to the arg top-s operation in the mask functions, we cannot use them directly to optimize f_θ with gradient-based optimization, such as stochastic gradient descent (SGD).

To solve this problem, we present differentiable insertion and deletion metrics with *soft* mask functions. We rewrite the mask functions in (2) and (4) as follows:

$$\alpha(\boldsymbol{x}; \boldsymbol{\phi}, \boldsymbol{b}, s)_{ijk} = \begin{cases} x_{ijk}, & \phi_{jk} \geq s\text{th-val}(\boldsymbol{\phi}) \\ b_{ijk}, & \text{otherwise} \end{cases}, \quad (5)$$

$$\beta(\boldsymbol{x}; \boldsymbol{\phi}, \boldsymbol{b}, s)_{ijk} = \begin{cases} b_{ijk}, & \phi_{jk} \geq s\text{th-val}(\boldsymbol{\phi}) \\ x_{ijk}, & \text{otherwise} \end{cases}, \quad (6)$$

where sth-val$(\boldsymbol{\phi})$ indicates the sth largest value in $\boldsymbol{\phi}$. Here, the reformulation is equivalent to (2) and (4), except in the case where the same value as sth-val$(\boldsymbol{\phi})$ exists in $\boldsymbol{\phi}$. The mask functions (5) and (6) are step functions that distinctly switch between x_{ijk} and b_{ijk} using the value of sth-val$(\boldsymbol{\phi})$ as a boundary, which are non-differentiable at the boundary and they have zero derivatives elsewhere. To make them smooth, we approximate (5) and (6) with soft step functions as follows:

$$\alpha_{\text{soft}}(\boldsymbol{x}, \boldsymbol{\phi}; \boldsymbol{b}, s)_{ijk} = r(\phi_{jk}; s)x_{ijk} + (1 - r(\phi_{jk}; s))b_{ijk}, \quad (7)$$

$$\beta_{\text{soft}}(\boldsymbol{x}, \boldsymbol{\phi}; \boldsymbol{b}, s)_{ijk} = r(\phi_{jk}; s)b_{ijk} + (1 - r(\phi_{jk}; s))x_{ijk}, \quad (8)$$

where $r(\phi_{jk}; s) \in [0, 1]$ is defined as $r(\phi_{jk}; s) = \sigma\left(T \cdot (\phi_{jk} - t_s)\right)$. σ is a sigmoid function, $T > 0$ is a temperature parameter, and t_s is the boundary value for the s-th largest value in $\boldsymbol{\phi}$ and is calculated as $t_s = \left(s\text{th-val}(\boldsymbol{\phi}) + (s{+}1)\text{th-val}(\boldsymbol{\phi})\right)/2$. Here, if $T = \infty$, then (7) and (8) are equivalent to (5) and (6), respectively. Using (7) and (8) as the mask functions in (1) and (3), respectively, we obtain the insertion and deletion metrics that are differentiable with respect to $\boldsymbol{\phi}$.

On the basis of the differentiable insertion and deletion metrics, we define insertion and deletion metric-based regularizers that regularize the predictor to maximize the insertion scores and minimize the deletion scores, as follows:

$$\Omega_{\texttt{Ins}}(\boldsymbol{\phi}^y, f_\theta; \boldsymbol{x}, y, \boldsymbol{b}) = -\frac{1}{S}\sum_{s=1}^{S} \log f_\theta(\alpha_{\text{soft}}(\boldsymbol{x}, \boldsymbol{\phi}^y; \boldsymbol{b}, s))_y, \quad (9)$$

$$\Omega_{\texttt{Del}}(\boldsymbol{\phi}^y, f_\theta; \boldsymbol{x}, y, \boldsymbol{b}) = -\frac{1}{S}\sum_{s=1}^{S} \log \frac{f_\theta(\boldsymbol{x})_y}{f_\theta(\beta_{\text{soft}}(\boldsymbol{x}, \boldsymbol{\phi}^y; \boldsymbol{b}, s))_y}, \quad (10)$$

where we use $\log f_\theta$ instead of using f_θ directly for numerical stability. We attempted three types of formulations using the deletion metric-based regularizer. Consequently, (10) achieved the high performance, as shown in Appendix A.

During the training, we used an f_θ that had been pretrained in a supervised learning manner, and we fine-tuned it using the regularizers together with the standard prediction loss. In particular, given N training samples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ where $\boldsymbol{x}_n \in \mathcal{X}$ and $y_n \in \mathcal{Y}$, we solve the following minimization problem using SGD:

$$\begin{aligned} \operatorname*{argmin}_\theta \sum_{n=1}^{N} & \ell_{\text{CE}}(f_\theta(\boldsymbol{x}_n), y_n) + \lambda_1 \Omega_{\texttt{Ins}}(\boldsymbol{\phi}_n^{y_n}, f_\theta; \boldsymbol{x}_n, y_n, \boldsymbol{b}) \\ & + \lambda_2 \Omega_{\texttt{Del}}(\boldsymbol{\phi}_n^{y_n}, f_\theta; \boldsymbol{x}_n, y_n, \boldsymbol{b}) + \lambda_3 \|\boldsymbol{\phi}_n^{y_n}\|_2^2, \end{aligned} \quad (11)$$

where $\ell_{\text{CE}}(f_\theta(\boldsymbol{x}_n), y_n)$ is the cross-entropy loss between the prediction $f_\theta(\boldsymbol{x}_n)$ and the label y_n; λ_1, λ_2, and λ_3 are hyperparameters; $\lambda_1, \lambda_2 \geq 0$ are the weights for the regularizers, respectively. $\|\boldsymbol{\phi}_n^{y_n}\|_2^2$ is an L2 regularizer to prevent the divergence of $\boldsymbol{\phi}_n^{y_n}$, and $\lambda_3 \geq 0$ is its

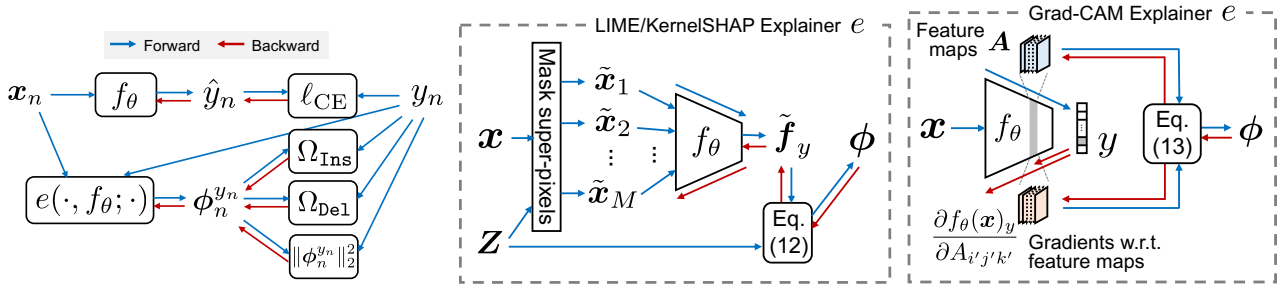


Figure 1: Overview of the forward and backward computations during training using ID-ExpO. **(Left)** the entire computational flow for each training sample $(\boldsymbol{x}_n, y_n)$. The computation flows inside the **(center)** LIME/KernelSHAP and **(right)** Grad-CAM explainers. Here, the red double line in Grad-CAM indicates that it computes second-order derivatives when it updates predictor f_θ, as it uses the gradients w.r.t. feature maps to obtain $\boldsymbol{\phi}$.

weight[1]. The left side of Figure 1 illustrates an overview of forward and backward computations involved in executing (11).

Because $\boldsymbol{\phi}_n^{y_n}$ is obtained using the post-hoc explainer $e(\boldsymbol{x}_n, f_\theta; y_n)$, the implementation of (11) depends on which post-hoc explainer we use. Below, we describe the implementations of perturbation-based and gradient-based explainers.

ID-ExpO for Perturbation-Based Explainers. The perturbation-based explainers for an image learn interpretable functions that capture the relationship between the predictor's inputs and outputs using perturbations around the image. The representative methods for the perturbation-based explanation are LIME and KernelSHAP, which calculate pixel-level contributions using the coefficients of linear regression models that have been learned on the perturbations around the input image. We illustrate the computational flow of these explainers at the center of Figure 1. First, we partition the image into D superpixels. Then, we generate M binary random vectors where the mth vector is denoted by $\boldsymbol{z}_m \in \{0,1\}^D$, and its lth element of that vector indicates whether its corresponding superpixel is masked ($z_{ml} = 0$) or not ($z_{ml} = 1$). According to $\boldsymbol{z}_m$, we obtain the masked perturbed image $\tilde{\boldsymbol{x}}_m$ from the original image $\boldsymbol{x}$. Where $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_M]^\top \in \{0,1\}^{M \times D}$ and $\tilde{\boldsymbol{f}}_y = [f_\theta(\tilde{\boldsymbol{x}}_m)_y]_{m=1}^M$, the LIME and KernelSHAP explainers calculate the pixel-level contributions by first solving the weighted least-squares problem and then expanding the obtained coefficients into the pixels of the image, as follows:

$$e(\boldsymbol{x}, f_\theta; y) = \texttt{expand}_{D \to H \times W}\Big((\boldsymbol{Z}^\top \boldsymbol{K} \boldsymbol{Z} + \epsilon \boldsymbol{I}_D)^{-1} \boldsymbol{Z} \boldsymbol{K} \tilde{\boldsymbol{f}}_y\Big), \tag{12}$$

where $\texttt{expand}_{D \to H \times W}$ is a function that expands the contribution assigned to each of D superpixels to the pixels associated with the superpixel. $\boldsymbol{K}$ is a diagonal matrix of size $M \times M$ whose (m,m)-element is the kernel value between a D-dimensional all-ones vector and $\boldsymbol{z}_m$. $\boldsymbol{I}_D$ is an identity matrix of size D, and $\epsilon \geq 0$ is the hyperparameter of the L2 regularizer. The key difference between LIME and KernelSHAP is the kernel function used to compute $\boldsymbol{K}$: LIME uses an L2 kernel (for images), whereas KernelSHAP uses a Shapley kernel.

ID-ExpO for Gradient-Based Explainers. One of the most popular gradient-based explainers is Grad-CAM. We illustrate the computational flow of the Grad-CAM explainer on the right side of Figure 1. Grad-CAM obtains nonnegative pixel-level contributions by calculating the activation map for the intermediate layers of predictor f_θ. Typically, convolutional neural networks are used as the predictors. To compute the activation map, Grad-CAM uses the feature maps from a convolution layer of the CNN predictor, which are denoted by $\boldsymbol{A} \in \mathbb{R}^{C' \times H' \times W'}$ where C', H' and W' are the number of channels, height, and width of the feature maps, respectively. In addition, it calculates the gradient for the output of the CNN predictor with respect to each element in the feature maps, i.e., $\frac{\partial f_\theta(\boldsymbol{x})_y}{\partial A_{i'j'k'}}$. Using them, the Grad-CAM explainer calculates the activation map $\boldsymbol{M} \in \mathbb{R}_{\geq 0}^{H' \times W'}$ for label y, followed by the pixel-level importance from $\boldsymbol{M}$, as follows:

$$e(\boldsymbol{x}, f_\theta; y) = \texttt{expand}_{H' \times W' \to H \times W}\Bigg(\underbrace{\mathrm{ReLU}\Big(\sum_{i'=1}^{C'} \omega_{i'}^y \boldsymbol{A}_{i'}\Big)}_{=\boldsymbol{M}}\Bigg), \tag{13}$$

[1]Because our regularizers encourage separating important pixels from less important ones, the feature contributions for the important pixels may become larger, and those for the less important ones may become smaller (negatively larger). Therefore, we add $\lambda_3 \|\boldsymbol{\phi}_n^{y_n}\|_2^2$ to avoid the divergence of $\boldsymbol{\phi}_n^{y_n}$, which may adversely affect the predictor's parameter update.

where

$$\omega_{i'}^{y} = \frac{1}{H'W'} \sum_{j'=1}^{H'} \sum_{k'=1}^{W'} \frac{\partial f_\theta(\boldsymbol{x})_y}{\partial A_{i'j'k'}}. \tag{14}$$

Here, $\texttt{expand}_{H' \times W' \to H \times W}$, as with that in (12), expands the (j', k')-element of the activation map $\boldsymbol{M}$ to its corresponding pixels on the image; ReLU is the rectified linear unit.

For Data Other Than Images. ID-ExpO is also available for cases where the input data is a Q-dimensional vector $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^Q$, e.g., text classification and tabular classification, whose qth dimension represents the qth feature value. In these cases, we replace ijk and jk in (1)–(10) specifying indices in an image with $q \in \{1, 2, \cdots, Q\}$. In addition, the maximum value of S becomes Q instead of HW.

Missingness Bias. When we produce explanations by perturbation-based explainers and evaluate explanations using insertion/deletion metrics, we mask some of the pixels in the input image. This input masking causes a problem called *missingness bias*, which means that the masked images are left out of the training input distribution. Jain et al. (2022) reported that missingness bias causes the explanations by LIME not to be aligned with human intuition, and that training the model with random masking augmentation mitigates the problem. Our regularizers, (9) and (10), evaluate the predictive loss, $\log f(\cdot)$, when some of the input pixels are masked; this has effects similar to the training with random masking augmentation. Therefore, our regularizers naturally manage the missingness bias.

Computational Complexity. The computational time complexity of the forward computation for updating the predictor's parameters once is $O(B(E + S))$ in our framework, where B is the batch size, E is the computational time complexity for the generation of an explanation by the explainer, and S is the number of times that the predictor is applied in Eqs. (9) and (10). M depends on the explainer used. For LIME and KernelSHAP, E indicates the computational time complexity of Eq. (12), $O(M + D^3)$, where M is the number of predictions executed for perturbed images, and D is the number of superpixels. For Grad-CAM, E indicates the computational time complexity of Eq. (13), which is determined by the choice of predictor.

4 Experiments

We conducted experiments on two image datasets and six tabular datasets to evaluate the effectiveness of our ID-ExpO in the case of using LIME, KernelSHAP and Grad-CAM as post-hoc explainers. Due to length limitations, we report the results of the image classification task in this section and those of the tabular classification task in Appendix C. Our implementation is based on PyTorch v.1.13, and the experiments were performed on a computer consisting of an Intel Xeon Platinum 8360Y CPU, an NVIDIA A100 SMX4 GPU, and 512 GB of RAM.

Comparing Methods. We compared the proposed method (ID-ExpO) with the following methods: stability-aware explanation-based optimization (ExpO-S), fidelity-aware explanation-based optimization (ExpO-F), and fine-tuning without explanation regularizers (ℓ_{CE}-only). ExpO-S and ExpO-F were based on the study in (Plumb et al., 2020). They aimed to improve the stability and fidelity of explanations produced by perturbation-based explainers such as LIME. ExpO-S learns the predictor with a regularizer so that the outputs of the predictor do not change for neighborhoods of the input. ExpO-F learns the predictor with another regularizer so that the outputs of the predictor for the neighborhoods are fitted with a local linear function. Because the original ExpO-S and ExpO-F were not intended for use on image data, we modified the ExpO-S and ExpO-F to apply to both image and tabular data, as described in Appendix B.1. ℓ_{CE}-only learns the predictor without any regularizer of explanation, which is equivalent to optimizing (11) with $\lambda_1 = \lambda_2 = \lambda_3 = 0$.

Evaluation. We quantitatively assessed the proposed method and the comparing methods in terms of predictive accuracy, insertion score (1), and deletion score (3). Here, we set S to 30% or 50% of the number of features (pixels), which is the same value as that used by our regularizers (9) and (10). Instead of the deletion score, we use the one-minus-deletion score, which is calculated by subtracting the deletion score from one, for readability. Furthermore, to check if the proposed method improves other faithfulness metrics that are not optimized directly, we also assessed explanations in sensitivity-n (Ancona et al., 2018), which evaluates how much, when n features are randomly removed, the sum of the contributions of the removed features correlates with the decrease in the predicted confidence.

As the criterion used for model selection and for monitoring the progress of the training, we used the validation score function defined as follows:

$$\mathrm{valscore}(f_\theta; \eta) = \eta \cdot \texttt{Acc}(f_\theta) + \texttt{Ins}(f_\theta) + 1 - \texttt{Del}(f_\theta), \tag{15}$$

where $\texttt{Acc}(f_\theta)$, $\texttt{Ins}(f_\theta)$, and $\texttt{Del}(f_\theta)$ indicate predictive accuracy, average insertion score, and average deletion score for the validation set for the predictor with current parameters f_θ, respectively. $\eta \geq 0$ is an accuracy weight used to control the ratio of predictive and explanatory

capabilities. Unless otherwise noted, we set $\eta = 2$, which indicates that the predictive and explanatory capabilities should be equally evaluated.

4.1 Image Classification

Datasets. We used two standard benchmark image classification datasets, CIFAR-10 (Krizhevsky et al., 2009) and STL-10 (Coates et al., 2011). CIFAR-10 contains 60,000 color images having a resolution of 32x32, classified into ten distinct classes. There were originally 50,000 labeled images for training and 10,000 labeled images for testing. In our experiment, we left 10,000 images in the original training set for validation. STL-10 consists of color images having a resolution of 96x96, with ten classes and 1,300 images per class. It originally provided 5,000 labeled images for training and 8,000 labeled images for testing. In our experiment, we left 500 images in the original training set for validation.

Implementation. We used ResNet-18 (He et al., 2016) as a predictor. We trained it in advance on each training set using the standard supervised learning, with data augmentation via random horizontal flipping and random cropping. For the LIME explainer, we first partitioned the image into square-shaped superpixels of different sizes, depending on the dataset. We used the superpixels of size 4x4 for 32x32 pixel images in CIFAR-10, and we used the superpixels of size 12x12 for the 96x96 pixel images in STL-10. Therefore, the number of superpixels was set to $D = (32/4)^2 = 64$ for CIFAR-10 and $D = (96/12)^2 = 64$ for STL-10. We also used D as the constant to increment s because each pixel in a superpixel has the same contribution value. We generated $M = 200$ perturbations and we set $\epsilon = 0.01$ in (12). For the Grad-CAM explainer, we used the feature maps of the `conv3_x` and `conv4_x` building blocks in ResNet-18 for CIFAR-10 and STL-10, respectively. The sizes of the feature maps were 8x8 for CIFAR-10 and 6x6 for STL-10. Depending on these sizes, we decided s to increment by 16 for CIFAR-10 and by 256 for STL-10.

The hyperparameters in ID-ExpO were λ_1, λ_2 and λ_3 in (11), and the temperature parameter T. For the experiments in this study, we used the same value λ_{12} for λ_1 and λ_2. We determined the best hyperparameter values in the ranges of $\lambda_{12} \in \{0.1, 0.01, 0.001\}$ and $\lambda_3 \in \{0.001, 0\}$ based on (15). T in the soft step function r ensures that its value does not approach zero or one. Because the appropriate value of T is different for different samples, we determined it as $T = \left(\frac{1}{\#(\boldsymbol{\phi})-1}\left(\max(\boldsymbol{\phi}) - \min(\boldsymbol{\phi})\right)\right)^{-1}$ for each sample, where $\#(\boldsymbol{\phi})$ is the number of elements with nonduplicated values in $\boldsymbol{\phi}$. The hyperparameter in each

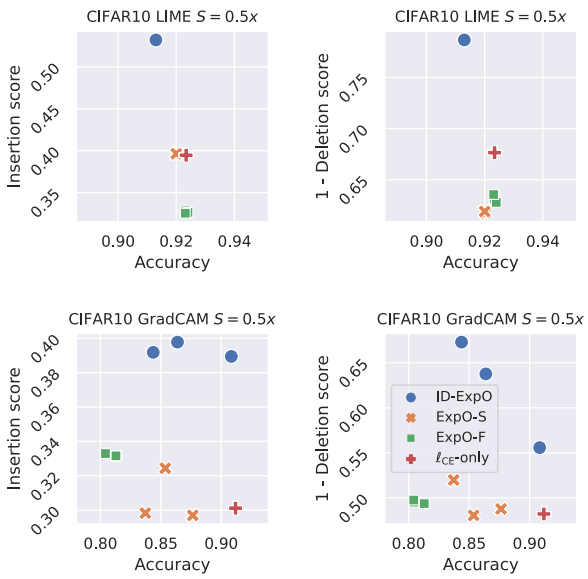


Figure 2: Mean insertion and mean one-minus-deletion scores against accuracy on CIFAR-10 in the case of $S = 0.5 \cdot HW$. The top row shows the results for LIME, and the bottom row shows the results for Grad-CAM. Each point indicates the result for the hyperparameters chosen on the basis of (15) with a different accuracy weight $\eta \in \{0.5, 1.0. \cdots, 3.0\}$ (different η values can be plotted in the same location). The higher the score, the better.

of ExpO-S and ExpO-F, which is the strength of its own regularizer, is determined as the best one in the range of $\{0.1, 0.01, 0.001\}$ based on (15). There are no hyperparameters specific to ℓ_{CE}-only. For the optimizer, we used an SGD optimizer with a minibatch size of 128, a momentum factor of 0.9, a weight decay of 0.0005, and Nesterov momentum. Its learning rate was determined on the basis of (15) within the range of $\{10^{-4}, 10^{-5}\}$. The training continued for 50 epochs or the value of valscore$(f_\theta; 2)$ did not increase for ten consecutive epochs.

4.1.1 Results

Figure 2 shows the insertion and one-minus-deletion scores against the predictive accuracy on CIFAR-10. Here, the scores are averaged over all the samples in the test set. ID-ExpO achieved the best insertion and deletion scores among the results. This fact indicates that the insertion and deletion metric-based regularizers used in ID-ExpO are effective, although the stability-aware regularizer in ExpO-S and the fidelity-aware regularizer in ExpO-F are not suitable for improving those scores. In terms of predictive accuracy, ID-ExpO was comparable with ℓ_{CE}-only when η was controlled in (15)

to achieve the highest accuracy, e.g., for $\eta = 3$, the insertion and deletion scores of ID-ExpO were better than those of ℓ_{CE}-only. Similar results were observed in the case of $S = 0.3 \cdot HW$ and on STL-10, as shown in Appendix B.2.

As described in Section 3.1, the regularizers in ID-ExpO naturally make predictors robust to the missingness bias. Missingness bias can unfairly make the insertion and deletion scores of the comparing methods better than those of ID-ExpO because the predictors that are sensitive to the bias may greatly change thier predictions, even if unimportant pixels are inserted in or masked out. Therefore, the fact that ID-ExpO achieved the best insertion and deletion scores indicates that the regularizers in ID-ExpO are effective in improving the faithfulness of explanations.

To investigate the impact of the proposed and the existing optimization methods on a faithfulness metric other than insertion/deletion metrics, we also evaluated the produced explanations in terms of sensitivity-n metric in Figure 3, which ID-ExpO does not directly optimize[2]. The figure shows that ID-ExpO consistently improved in terms of the sensitivity-n metric on CIFAR-10 and STL-10, whereas ExpO-S and ExpO-F did not. This result indicates that ID-ExpO does not overfit the insertion and deletion metrics, and can improve the faithfulness of explanations from various perspectives.

Figure 4 shows how much the insertion and deletion scores of individual samples changed before and after we fine-tuned the predictor based on each method. In ID-ExpO, ExpO-S, ExpO-F, and ℓ_{CE}-only, 57.8%, 41.9%, 38.7% and 34.6% of samples were located in the first quadrant, respectively. Because the first quadrant means that the changes in the insertion and one-minus-deletion scores are both positive, we found that ID-ExpO was the most effective in making explanations more faithful. Conversely, the ratios of the samples located in the third quadrant, which indicates that the samples became less faithful, were 6.9%, 6.7%, 7.7% and 11.1%, respectively. We also found that the possibility that ID-ExpO would worsen the faithfulness of the explanations was relatively low.

In Figure 5, we visualize the produced explanations for the predictors trained using ID-ExpO and ℓ_{CE}-only, and we show the distributions of the insertion and deletion metrics for the explanations. More results are shown in Appendix B.3. Here, we show only the examples in Figure 5, in which correct prediction was made. As shown in Figures 5(d) and 5(e), the distributions of the insertion and deletion metrics were quite different between ID-ExpO and ℓ_{CE}-only. This result indicates that our insertion and deletion metric-aware regularizers positively affected those distributions, as we expected. In addition, as shown in Figures 5(b) and 5(c), we found that the explanations differed from each other. In particular, in the heatmap

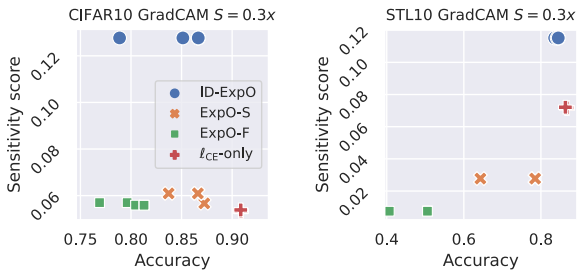


Figure 3: Mean sensitivity-n scores against accuracy on CIFAR-10 (left) and STL-10 (right) in the case of $S = 0.3 \cdot HW$. The higher the score, the better.

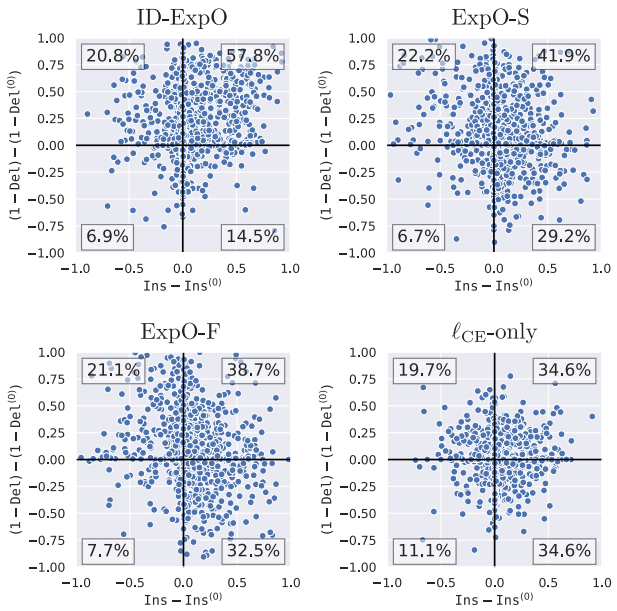
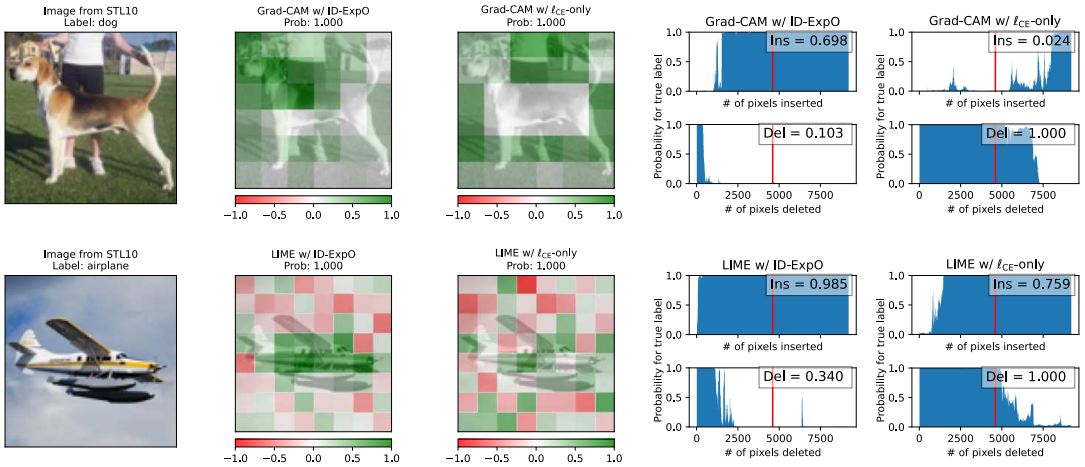


Figure 4: Differences in the insertion and one-minus-deletion scores between before and after predictors were fine-tuned, with Grad-CAM using each method for 1,000 randomly selected individual test samples on CIFAR-10. `Ins` and `Del` indicate the mean insertion and deletion scores over the test set when the predictors are used after fine-tuning, whereas $\texttt{Ins}^{(0)}$ and $\texttt{Del}^{(0)}$ indicate the same scores before the fine-tuning. The percentage in each quadrant is the ratio of the samples located in the quadrant.

[2]Although the original sensitivity-n evaluates feature contributions at a pixel level, it is not appropriate to assess the feature contributions produced by Grad-CAM and LIME, which are explained at a super-pixel level. Therefore, we modified the sensitivity-n so as to mask super-pixels randomly. Here, we were set to $n = 4$, i.e., we randomly masked four super-pixels instead of pixels in an image.



(a) (b) (c) (d) (e)

Figure 5: Examples of the produced explanations for STL-10. The first row shows the results obtained by Grad-CAM, while the other shows the results obtained by LIME. Each row illustrates (a) an input image, (b)–(c) the heatmaps of the explanations by the explainers with ID-ExpO and ℓ_{CE}-only, and (d)–(e) the insertion score (top) and the deletion score (bottom) for those explanations in the case of $S = 0.5 \cdot HW$, which means that the scores are the blue areas to the left of red vertical lines.

explanations for ID-ExpO (Figure 5(b)), large positive contributions were assigned to some of the superpixels that captured the object of the class label well. This is because the evaluation of the insertion and deletion scores in our regularizers is truncated at 30% or 50% of the number of pixels. These results indicate that ID-ExpO can change the explanations to preferentially assign larger positive contributions to the pixels that strongly affect the prediction to improve those metrics.

Comparison with Adversarially Robust Models. Shah et al. (2021) reported that adversarially robust models could make explanations based on input gradients more faithful to the model predictions than could models trained in the standard supervised learning manner. However, it is not clear whether the adversarially robust models were helpful in producing faithful explanations when LIME and Grad-CAM were used. To compare the models trained with ID-ExpO and with the adversarially robust models, we evaluated the adversarially robust ResNet-50 model (ADV for short) used in the work of Shah et al. (2021) in terms of insertion and one-minus-deletion metrics. Here, the weights for ADV that are trained on CIFAR-10 are publicly available, and the predictive accuracy of ADV was 0.854 in our setting, which was comparable to or lower than that of the ResNet-18 model that had been fine-tuned with ID-ExpO. The insertion and one-minus-deletion scores of ADV were 0.282 and 0.567 when Grad-CAM was used, and they were 0.456 and 0.778 when LIME was used. Compared with the results shown in Figure 2, we found that in terms of the insertion scores, the model trained with ID-ExpO significantly outperformed ADV, and in terms of the one-minus-deletion scores, the model trained with ID-ExpO was comparable to or better than ADV. This result indicates that for Grad-CAM and LIME, ID-ExpO was more effective than the adversarially robust model in improving the faithfulness of explanations.

5 Conclusion

We proposed an explanation-based optimization method that learns machine learning predictors, such as DNNs, with insertion and deletion metrics-aware regularizers. By fine-tuning the predictors based on the proposed method, we were able to confirm that several explainers, including perturbation-based and gradient-based explainers, could produce explanations that were faithful to the predictors' behaviors. In future work, we will further verify the effectiveness of our insertion and deletion metrics-aware regularizers in improving the faithfulness of the explanations made by inherently interpretable models (Alvarez Melis et al., 2018; Yoshikawa et al., 2021) and parameterized explainers (Situ et al., 2021).

Acknowledgments

We thank the anonymous reviewers for their valuable suggestions. This work was supported by JSPS KAKENHI Grant Number 22K17953.

References

Agarwal, Rishabh et al. (2021). "Neural additive models: Interpretable machine learning with neural nets". In: *Advances in Neural Information Processing Systems* 34, pp. 4699–4711.

Alvarez Melis, David and Tommi Jaakkola (2018). "Towards robust interpretability with self-explaining neural networks". In: *Advances in neural information processing systems* 31.

Ancona, Marco et al. (Feb. 2018). "Towards better understanding of gradient-based attribution methods for Deep Neural Networks". URL: https://openreview.net/pdf?id=Sy21R9JAW.

Balayan, Vladimir et al. (2020). *Teaching the Machine to Explain Itself using Domain Knowledge*.

Bhatt, Umang, Adrian Weller, and José MF Moura (2021). "Evaluating and aggregating feature-based model explanations". In: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 3016–3022.

Bischl, Bernd et al. (2021). "OpenML Benchmarking Suites." In: *Proceedings of the NeurIPS 2021 Datasets and Benchmarks Track*.

Capuano, Nicola et al. (2022). "Explainable Artificial Intelligence in CyberSecurity: A Survey". In: *IEEE Access* 10, pp. 93575–93600. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3204171. URL: http://dx.doi.org/10.1109/ACCESS.2022.3204171.

Chattopadhay, Aditya et al. (2018). "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks". In: *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE, pp. 839–847.

Coates, Adam, Andrew Ng, and Honglak Lee (Nov. 2011). "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, pp. 215–223. URL: https://proceedings.mlr.press/v15/coates11a.html.

Fong, Ruth, Mandela Patrick, and Andrea Vedaldi (2019). "Understanding deep networks via extremal perturbations and smooth masks". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2950–2958.

Fu, Ruigang et al. (2020). *Axiom-based grad-cam: Towards accurate visualization and explanation of cnns*.

Fukui, Hiroshi et al. (2019). "Attention branch network: Learning of attention mechanism for visual explanation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10705–10714.

Gao, Yuyang et al. (2022). "Aligning eyes between humans and deep neural network through interactive attention alignment". In: *Proceedings of the ACM on Human-Computer Interaction* 6.CSCW2, pp. 1–28.

Gevaert, Arne et al. (2022). *Evaluating Feature Attribution Methods in the Image Domain*. en. URL: https://www.semanticscholar.org/paper/65f8cf77942dd5067ba96debc90aa009467c990d.

Ghorbani, Amirata, Abubakar Abid, and James Zou (2019). "Interpretation of neural networks is fragile". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01, pp. 3681–3688.

Hastie, Trevor and Robert Tibshirani (1986). "Generalized Additive Models". In: *Statistical Science* 1.3, pp. 297–310. ISSN: 08834237. URL: http://www.jstor.org/stable/2245459 (visited on 05/16/2023).

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Holzinger, Andreas et al. (Dec. 2017). *What do we need to build explainable AI systems for the medical domain?* arXiv: 1712.09923 [cs.AI]. URL: http://arxiv.org/abs/1712.09923.

Huber, Marco et al. (2023). *Are Explainability Tools Gender Biased? A Case Study on Face Presentation Attack Detection*.

Iida, Tsumugi et al. (Dec. 2022). "Visual Explanation Generation Based on Lambda Attention Branch Networks". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pp. 3536–3551.

Ismail, Aya Abdelsalam, Hector Corrada Bravo, and Soheil Feizi (2021). "Improving deep learning interpretability by saliency guided training". In: *Advances in neural information processing systems* 34, pp. 26726–26739. ISSN: 1049-5258. URL: https://proceedings.neurips.cc/paper/2021/hash/e0cd3f16f9e883ca91c2a4c24f47b3d9-Abstract.html.

Jain, Saachi et al. (Apr. 2022). "Missingness Bias in Model Debugging". In: arXiv: 2204.08945 [cs.CV]. URL: http://arxiv.org/abs/2204.08945.

Jiang, Peng-Tao et al. (2021). "Layercam: Exploring hierarchical class activation maps for localization". In: *IEEE Transactions on Image Processing* 30, pp. 5875–5888.

Krizhevsky, Alex, Geoffrey Hinton, et al. (2009). "Learning multiple layers of features from tiny images". In.

Lage, Isaac et al. (Dec. 2018). "Human-in-the-Loop Interpretability Prior". en. In: *Advances in neural information processing systems* 31. ISSN: 1049-5258.

URL: https://www.ncbi.nlm.nih.gov/pubmed/33623354.

Lertvittayakumjorn, Piyawat and Francesca Toni (2021). "Explanation-Based Human Debugging of NLP Models: A Survey". In: *Transactions of the Association for Computational Linguistics* 9, pp. 1508–1528. DOI: 10.1162/tacl_a_00440. URL: https://aclanthology.org/2021.tacl-1.90.

Lundberg, Scott M and Su-In Lee (2017). "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I Guyon et al. Curran Associates, Inc., pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Molnar, Christoph (2022). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. URL: https://christophm.github.io/interpretable-ml-book.

Omeiza, Daniel et al. (Aug. 2022). "Explanations in Autonomous Driving: A Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.8, pp. 10142–10162. ISSN: 1558-0016. DOI: 10.1109/TITS.2021.3122865. URL: http://dx.doi.org/10.1109/TITS.2021.3122865.

Petsiuk, Vitali, Abir Das, and Kate Saenko (2018). *RISE: Randomized Input Sampling for Explanation of Black-box Models*. arXiv: 1806.07421 [cs.CV].

Plumb, Gregory et al. (2020). "Regularizing black-box models for improved interpretability". In: *Advances in Neural Information Processing Systems* 33, pp. 10526–10536.

Ramaswamy, Harish Guruprasad et al. (2020). "Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 983–991.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (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*, pp. 1135–1144.

Rieger, Laura and Lars Kai Hansen (2020). *Irof: a low resource evaluation metric for explanation methods*.

Ross, Andrew Slavin, Michael C Hughes, and Finale Doshi-Velez (2017). "Right for the right reasons: training differentiable models by constraining their explanations". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2662–2670.

Schwab, Patrick, Djordje Miladinovic, and Walter Karlen (2018). "Granger-Causal Attentive Mixtures of Experts: Learning Important Features with Neural Networks". In: *AAAI Conference on Artificial Intelligence*.

Selvaraju, Ramprasaath R et al. (Feb. 2020). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International journal of computer vision* 128.2, pp. 336–359. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: https://doi.org/10.1007/s11263-019-01228-7.

Shah, Harshay, Prateek Jain, and Praneeth Netrapalli (2021). "Do input gradients highlight discriminative features?" In: *Advances in neural information processing systems* 34, pp. 2046–2059. ISSN: 1049-5258. URL: https://arxiv.org/pdf/2102.12781.pdf.

Situ, Xuelin et al. (Aug. 2021). "Learning to Explain: Generating Stable Explanations Fast". 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)*. Online: Association for Computational Linguistics, pp. 5340–5355. DOI: 10.18653/v1/2021.acl-long.415. URL: https://aclanthology.org/2021.acl-long.415.

Stein, Bas van et al. (Apr. 2023). *Deep-BIAS: Detecting Structural Bias using Explainable AI*. arXiv: 2304.01869 [cs.NE]. URL: http://arxiv.org/abs/2304.01869.

Wang, Haofan et al. (2020). "Score-CAM: Score-weighted visual explanations for convolutional neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 24–25.

Yoshikawa, Yuya and Tomoharu Iwata (Dec. 2021). "Gaussian Process Regression With Interpretable Sample-Wise Feature Weights". en. In: *IEEE transactions on neural networks and learning systems* PP. ISSN: 2162-2388, 2162-237X. DOI: 10.1109/TNNLS.2021.3131234. URL: http://dx.doi.org/10.1109/TNNLS.2021.3131234.

– (May 2022). "Neural generators of sparse local linear models for achieving both accuracy and interpretability". In: *An international journal on information fusion* 81, pp. 116–128. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2021.11.009. URL: https://www.sciencedirect.com/science/article/pii/S1566253521002347.

Zeng, Chunyan et al. (2022). "Abs-CAM: a gradient optimization interpretable approach for explanation of convolutional neural networks". In: *Signal, Image and Video Processing*, pp. 1–8.

Zhang, Hanwei et al. (Jan. 2023). *Opti-CAM: Optimizing saliency maps for interpretability*. arXiv: 2301.07002 [cs.CV]. URL: http://arxiv.org/abs/2301.07002.

Zhang, Qinglong, Lu Rao, and Yubin Yang (2021). *Group-cam: Group score-weighted visual explanations for deep convolutional networks*.

Zhang, Yongfeng and Xu Chen (Mar. 2020). "Explainable Recommendation: A Survey and New Perspectives". In: *Found. Trends Inf. Retr.* 14.1, pp. 1–101. ISSN: 1554-0669. DOI: 10.1561/1500000066. URL: https://doi.org/10.1561/1500000066.

Zhao, Xingyu et al. (2021). "Baylime: Bayesian local interpretable model-agnostic explanations". In: *Uncertainty in artificial intelligence*. PMLR, pp. 887–896.

Zhou, Jianlong et al. (Mar. 2021). "Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics". en. In: *Electronics* 10.5, p. 593. ISSN: 2079-9292, 2079-9292. DOI: 10.3390/electronics10050593. URL: https://www.mdpi.com/2079-9292/10/5/593.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [**Yes**]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [**Yes**]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [**No**] However, we will release the source code after the paper is accepted.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [**Not Applicable**]
   (b) Complete proofs of all theoretical results. [**Not Applicable**]
   (c) Clear explanations of any assumptions. [**Not Applicable**]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [**Yes**]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [**Yes**]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [**Yes**]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [**Yes**]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [**Yes**]
   (b) The license information of the assets, if applicable. [**Not Applicable**]
   (c) New assets either in the supplemental material or as a URL, if applicable. [**Not Applicable**]
   (d) Information about consent from data providers/curators. [**Not Applicable**]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [**Not Applicable**]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [**Not Applicable**]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [**Not Applicable**]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [**Not Applicable**]

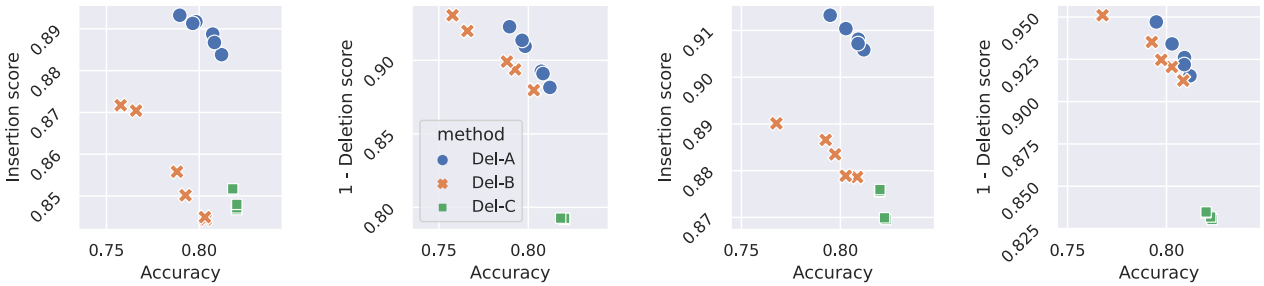


Figure A.1: Mean insertion and mean one-minus-deletion scores against mean accuracy averaged over six tabular datasets when LIME was used as the explainer among three types of deletion metric-aware regularizers. Each point has a different accuracy weight $\eta \in \{0.5, 1.0. \cdots, 3.0\}$. Figures of the first two columns are the results at $S = 0.3 \cdot HW$, while those of the other columns are the results at $S = 0.5 \cdot HW$. The higher the score, the better.

A Comparison Among Different Types of Deletion Metric-Based Regularizers

We formalized three types of deletion metric-based regularizers, as follows:

$$\textbf{Del-A}: \quad \Omega_{\texttt{Del}}(\boldsymbol{\phi}^y, f_\theta; \boldsymbol{x}, y, \boldsymbol{b}) = -\frac{1}{S}\sum_{s=1}^{S} \log \frac{f_\theta(\boldsymbol{x})_y}{f_\theta(\beta_{\text{soft}}(\boldsymbol{x}, \boldsymbol{\phi}^y; \boldsymbol{b}, s))_y}, \tag{A.1}$$

$$\textbf{Del-B}: \quad \Omega_{\texttt{Del}}(\boldsymbol{\phi}^y, f_\theta; \boldsymbol{x}, y, \boldsymbol{b}) = \frac{1}{S}\sum_{s=1}^{S} \log f_\theta(\beta_{\text{soft}}(\boldsymbol{x}, \boldsymbol{\phi}^y; \boldsymbol{b}, s))_y, \tag{A.2}$$

$$\textbf{Del-C}: \quad \Omega_{\texttt{Del}}(\boldsymbol{\phi}^y, f_\theta; \boldsymbol{x}, y, \boldsymbol{b}) = -\frac{1}{S}\sum_{s=1}^{S} \log \left(1 - f_\theta(\beta_{\text{soft}}(\boldsymbol{x}, \boldsymbol{\phi}^y; \boldsymbol{b}, s))_y\right), \tag{A.3}$$

where Del-A is what the proposed method employs. Del-B is similar to Del-A, but does not have the term improving the prediction for the original input. Del-C is similar to Del-B, but is formalized like the term for negative class in the binary cross entropy loss.

Figure A.1 shows the insertion and one-minus-deletion scores of the three formulations. We found that Del-A achieved the best balance of the accuracy and insertion/one-minus-deletion scores.

B On Experiments on Image Datasets

B.1 Modified Version of ExpO-S and ExpO-F

The original ExpO-Stability (ExpO-S for short) and ExpO-Fidelity (ExpO-F for short) aim at making predictions and their corresponding explanations robust to slight changes in the feature values of the input, respectively (Plumb et al., 2020). The authors stated that the ExpO-F did not evaluate for non-semantic features, such as images, as the fidelity metric is not appropriate for the non-semantic features (Plumb et al., 2020, Appendix A.8). Therefore, while keeping the idea of the original ExpO-F, we modified it to apply it to image data. In particular, we utilize the same approach to LIME for image data, which we describe in Section 3.1, as the fidelity regularizer mimics the derivation of the explanations produced by LIME. Below we use the same notation of the variables in Section 3.1. First, for a given training sample $(\boldsymbol{x}, y)$, we obtain a LIME explanation $\boldsymbol{\phi}^y$ by applying (12). Then, since $\boldsymbol{\phi}^y$ is the coefficients of a local linear model around input $\boldsymbol{x}$, we can calculate the fidelity-aware regularizer based on the original ExpO-F as follows:

$$\Omega_{\texttt{Fidelity}}(\boldsymbol{\phi}^y, f_\theta; \boldsymbol{x}, y, \{\boldsymbol{z}_m\}_{m=1}^M, \boldsymbol{K}) = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{K}_{mm}(f_\theta(\tilde{\boldsymbol{x}}_m)_y - \boldsymbol{\phi}^{y\top}\boldsymbol{z}_m)^2. \tag{A.4}$$

where $\boldsymbol{z}_m \in \{0,1\}^D$ is the mth binary random vector to mask D super-pixels, $f_\theta(\tilde{\boldsymbol{x}}_m)_y$ is the predicted probability of label y for perturbed mask image $\tilde{\boldsymbol{x}}_m$, $\boldsymbol{K}_{mm}$ is the value of a cosine kernel between a D-dimensional all-one

vector and $\boldsymbol{z}_m$.

The original ExpO-S can be applied to numerical features and non-semantic features as it calculates the differences between the prediction for the original input and that for the perturbed input, in which Gaussian perturbations are added to the features of the input. However, for the consistency of evaluation, we modified the original ExpO-S to perturb the input with binary random mask vectors $\boldsymbol{Z}$. In particular, we calculate the stability-aware regularizer as follows:

$$\Omega_{\texttt{Stability}}(\boldsymbol{\phi}^y, f_\theta; \boldsymbol{x}, y, \{\tilde{\boldsymbol{x}}_m\}_{m=1}^M, \boldsymbol{K}) = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{K}_{mm}(f_\theta(\tilde{\boldsymbol{x}}_m)_y - f_\theta(\boldsymbol{x}))^2. \tag{A.5}$$

B.2 All Quantitative Results on Image Datasets

Figure A.2 shows the insertion and the one-minus-deletion scores against the accuracy on each image dataset. On all the explainers and the datasets except Grad-CAM on STL-10 (Figure A.2(D)), we found that the insertion and one-minus-deletion scores of ID-ExpO are superior to those of the other methods. With the accuracy, by putting emphasis on the accuracy, i.e., by setting $\eta = 3$, we found that ID-ExpO in the case of $S = 0.5 \cdot HW$ could keep comparable or slightly low accuracy compared to the others, while the highest insertion and one-minus-deletion scores. However, in the case of $S = 0.3 \cdot HW$, we found that the accuracy of ID-ExpO was lower than the other methods and the accuracy of ID-ExpO in the case of $S = 0.5 \cdot HW$. The setting of $S = 0.3 \cdot HW$ means that our insertion and deletion metric-aware regularizers force the predictor to use only 30% of the pixels in an image in prediction. The result indicates that the regularization was too strong to predict correctly for the image datasets.

B.3 Additional Examples of Produced Explanations on Image Datasets

Figures A.3 and A.4 show additional visualization examples of the produced explanations on image datasets when the insertion and deletion scores of the explanations were improved by using ID-ExpO. Overall, compared to the explanations of ℓ_{CE}-only, the explanations of ID-ExpO had the tendency that large positive contributions were assigned to a part of super-pixels that captures the object of the class label well.

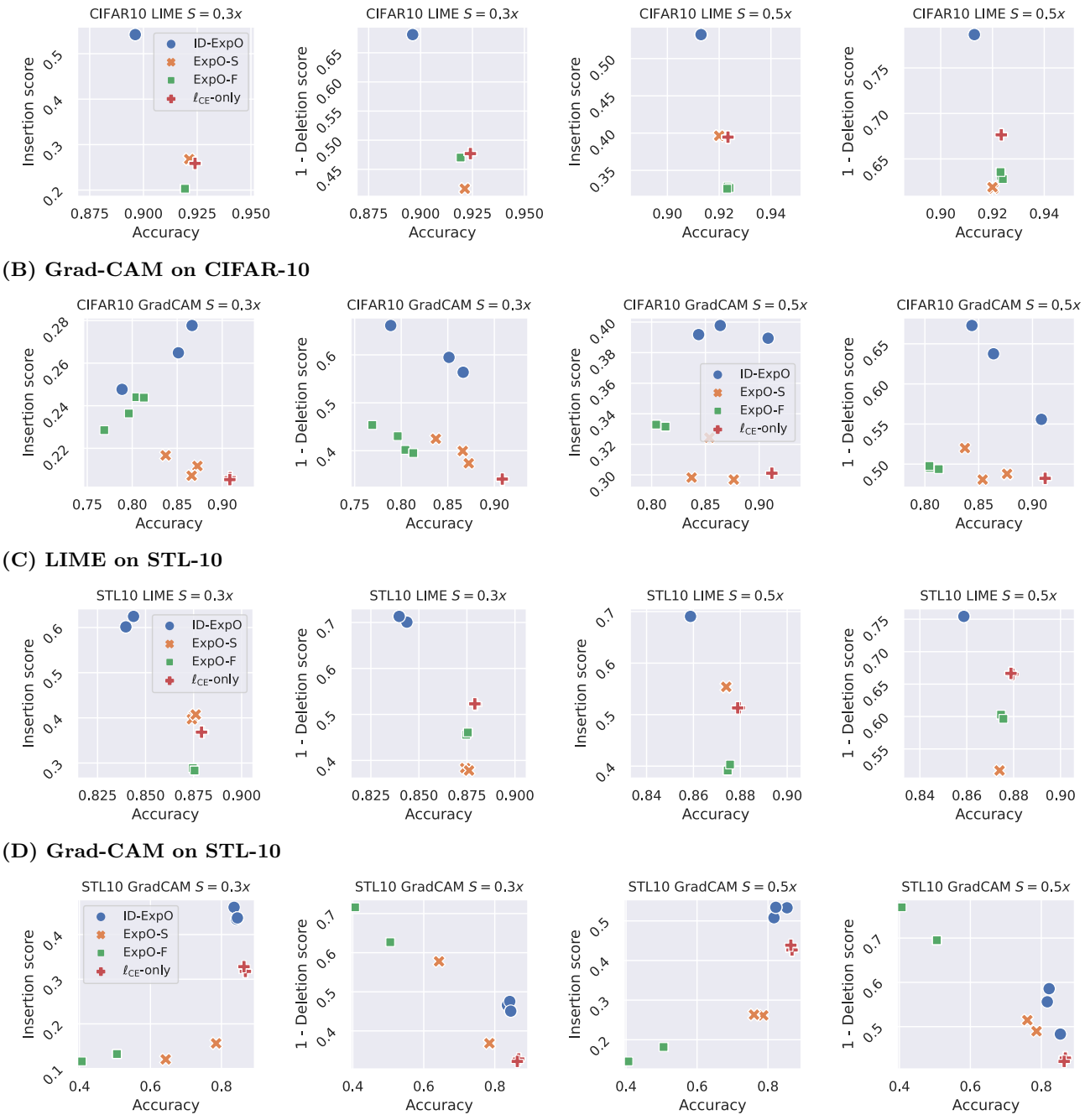
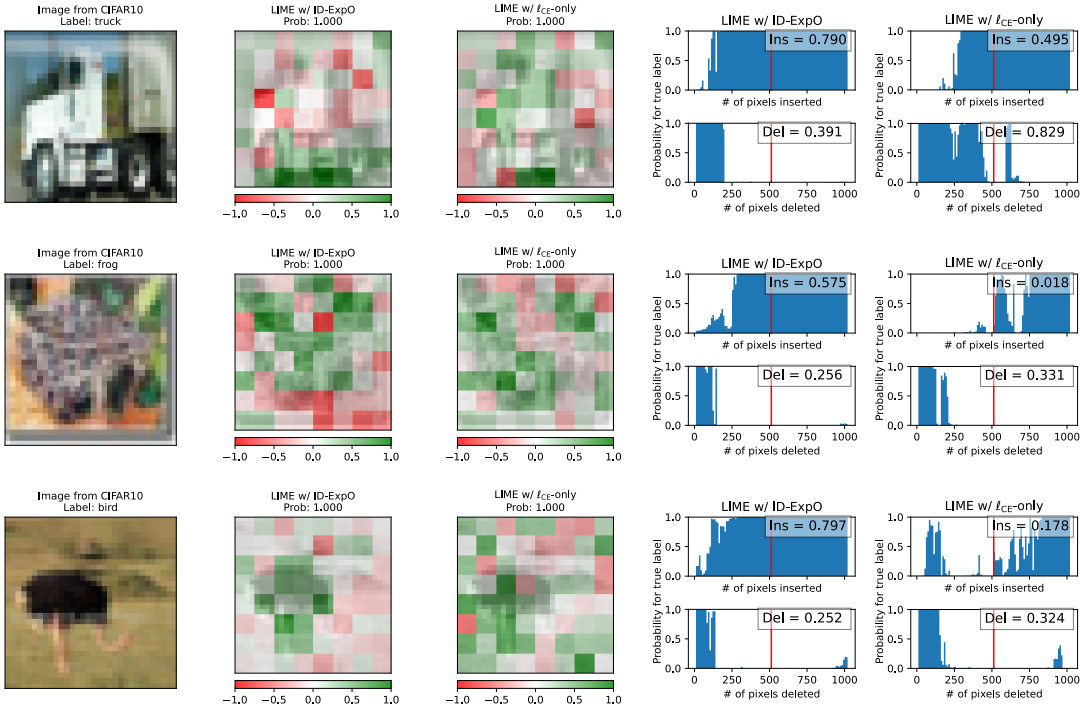
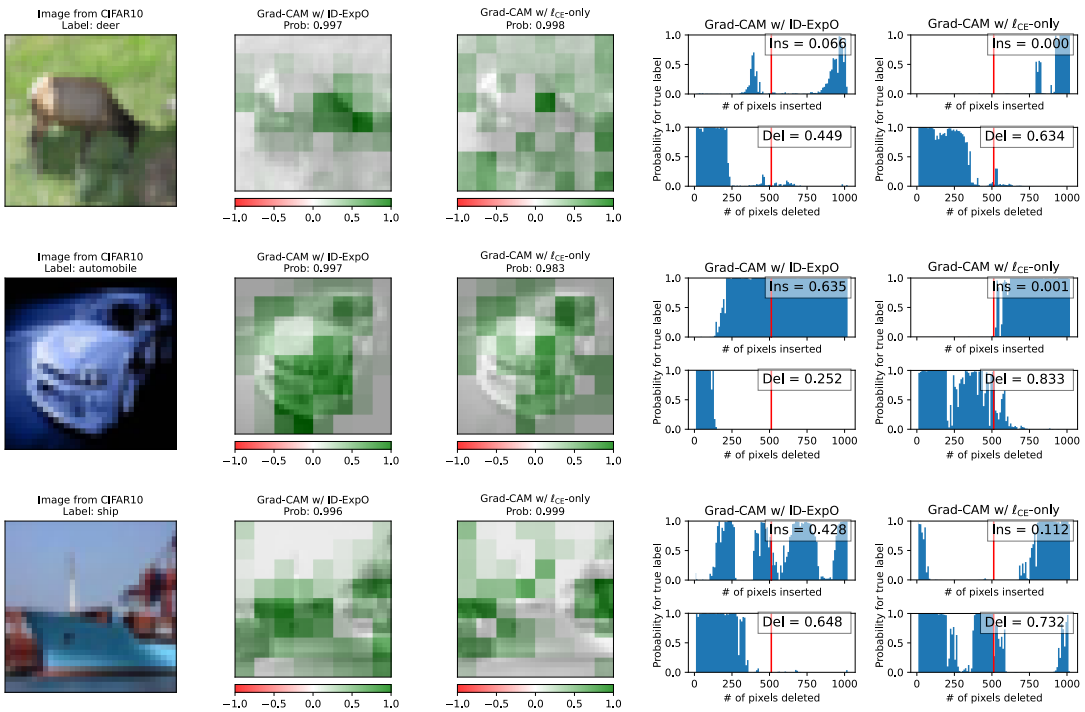


Figure A.2: Mean insertion and mean one-minus-deletion scores against accuracy on each image dataset. Each row indicates a different pair of an explainer and a dataset. The first two columns show the results when $S = 0.3 \cdot HW$, while the last two columns show the results when $S = 0.5 \cdot HW$. Each point indicates the result for the hyperparameters chosen based on (15) with a different accuracy weight $\eta \in \{0.5, 1.0. \cdots, 3.0\}$ (different η values can be plotted in the same location). The higher the score, the better.

(A) LIME on CIFAR-10 ($S = 0.5 \cdot HW$)



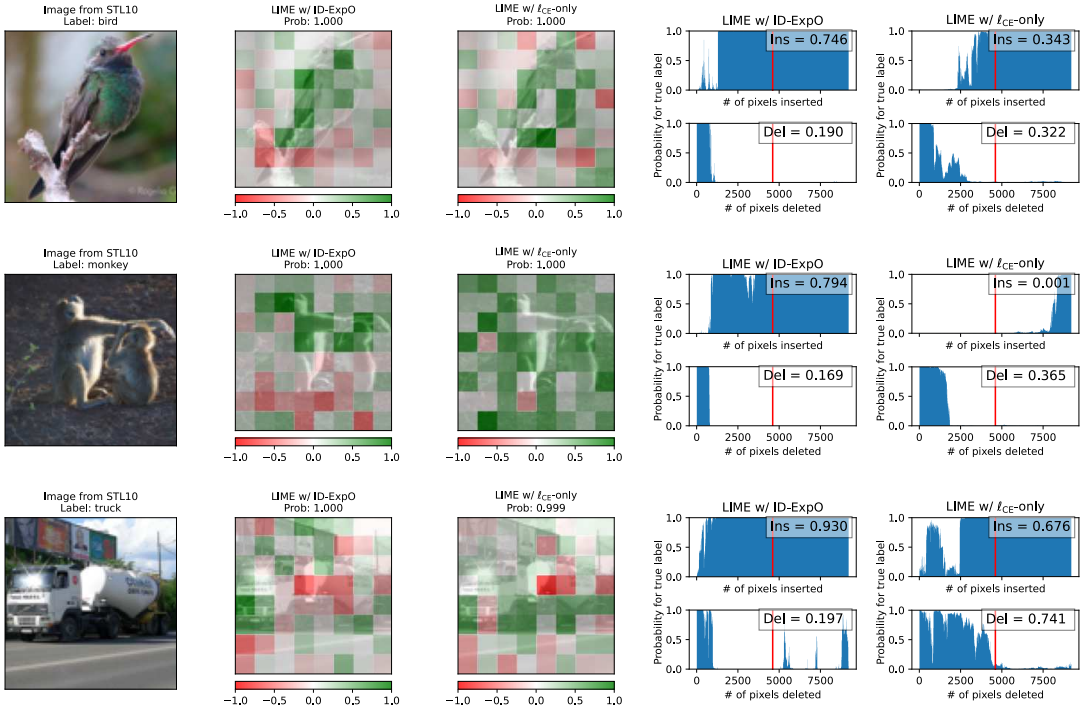
(B) Grad-CAM on CIFAR-10 ($S = 0.5 \cdot HW$)



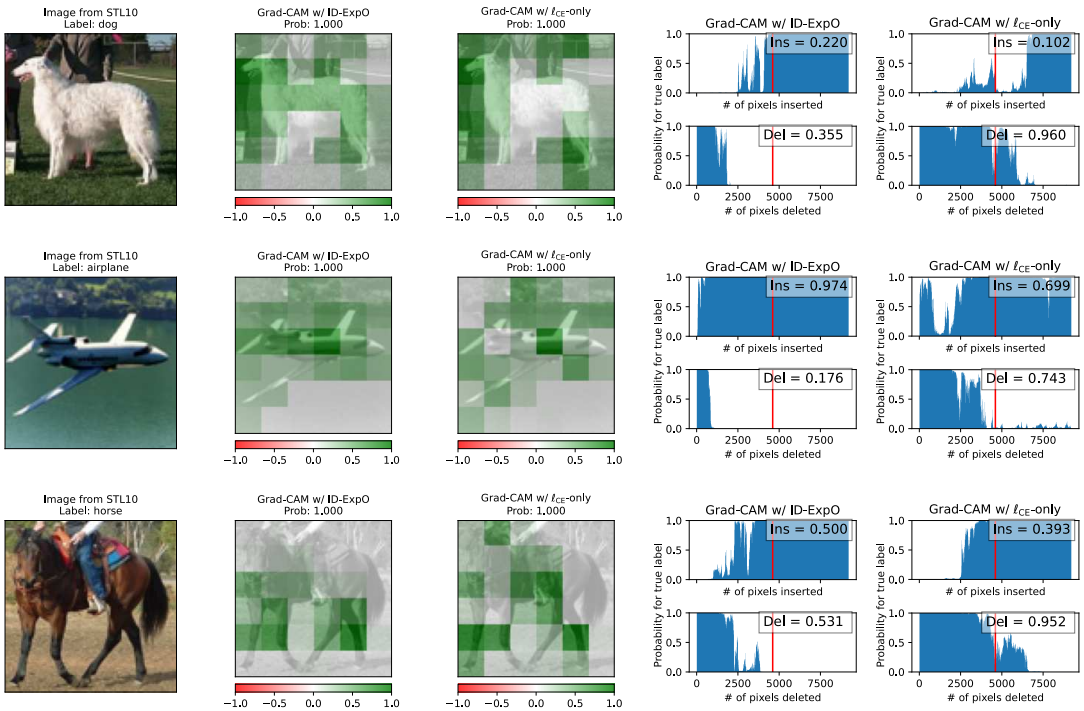
(a) (b) (c) (d) (e)

Figure A.3: Examples of the produced explanations on CIFAR-10. Each row illustrates (a) an input image, (b)–(c) the heatmaps of the explanations by the explainers with ID-ExpO and ℓ_{CE}-only, and (d)–(e) the insertion score (top) and the deletion score (bottom) for those explanations in the case of $S = 0.5 \cdot HW$, which means that the scores are the blue areas to the left of red vertical lines.

(A) LIME on STL-10 ($S = 0.5 \cdot HW$)



(B) Grad-CAM on STL-10 ($S = 0.5 \cdot HW$)



(a) (b) (c) (d) (e)

Figure A.4: Examples of the produced explanations on STL-10. How to read the figures is the same as Figure A.3.

Table A.1: Specification of tabular datasets.

Dataset	# samples	# features	# classes
collins	500	23	2
mfeat-fourier	2,000	77	10
one-hundred-plants-shape	1,600	65	100
qsar-biodeg	1,055	42	2
steel-plates-fault	1,941	28	7
wine-quality-red	1,599	12	6

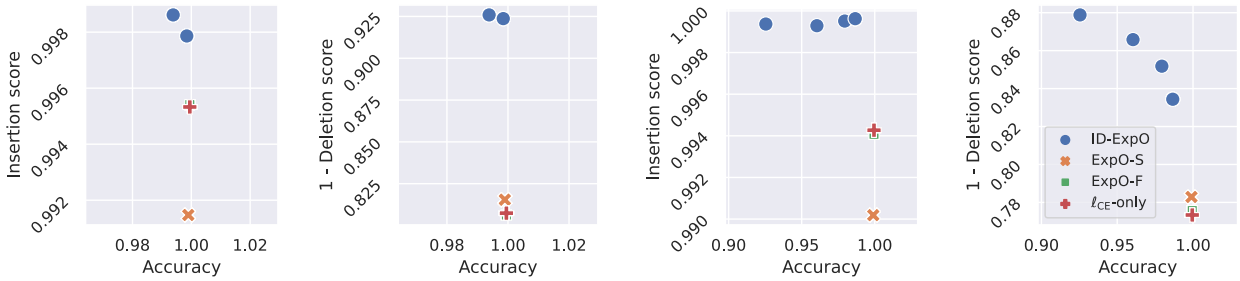


Figure A.5: Mean insertion and mean one-minus-deletion scores against accuracy on steel-plates-fault dataset in case of $S = 0.3 \cdot Q$. The scores are averaged over the five training/validation/test sets. The first two columns show the results on LIME, while the other shows the results on KernelSHAP.

C Experiments on Tabular Datasets

C.1 Tabular Datasets

We used six tabular classification datasets with numerical features from OpenML dataset repository (Bischl et al., 2021): collins, mfeat-fourier, one-hundred-plants-shape, qsar-biodeg, steel-plates-fault, and wine-quality-red. Table A.1 shows the numbers of samples, features and classes of each tabular dataset. For each dataset, we created five sets, each of which consists of training, validation and test sets, by randomly dividing the dataset in the ratio of 70%, 10% and 20%. We standardized each feature value using the training set.

C.2 Implementation Details for Tabular Datasets

We used a multilayer perceptron (MLP) with two hidden layers of 256 units and ReLU activation functions as a predictor, which was trained on the training set of each dataset in advance in the standard supervised learning manner. We used LIME and KernelSHAP with the same hyperparameter setting as the LIME for the image classification as explainers. Since there is no bunch of features like a super-pixel, we directly calculated the contributions of individual features in (12). We did not use Grad-CAM because it is impossible to associate the features with the activation maps of the intermediate layers of the MLP. The optimizer is the same as that for image classification, except we chose the learning rate in the range of $\{0.01, 0.001\}$. The training continued until 200 epochs were reached or the value of valscore$(f_\theta; 2)$ does not gain for 20 consecutive epochs.

C.3 Results of Tabular Datasets

Figure A.5 shows the insertion and one-minus-deletion scores against accuracy on steel-plates-fault dataset. To test the differences among the methods for each evaluation metric, we performed a paired t-test at 5% level for the results with $\eta = 3$. As a result, in the case of using LIME, ID-ExpO achieved the highest insertion and one-minus-deletion scores, and there was no statistical accuracy difference among the methods. In the case of using KernelSHAP, although the insertion and one-minus-deletion scores of ID-ExpO were the highest, its accuracy was superior to the comparing methods. LIME and KernelSHAP are similarly formalized as linear regression models. The main difference between them is the kernel functions in (12). Since the Shapley kernel

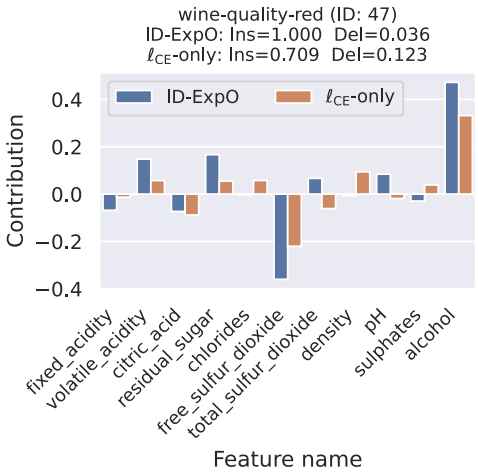


Figure A.6: Feature contributions on a sample in wine-quality-red dataset.

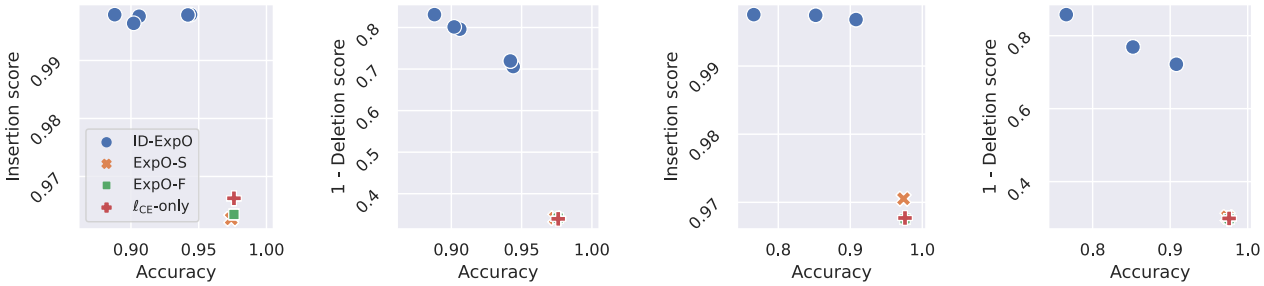
used in KernelSHAP often outputs extremely small values, resulting in unstable calculation in (12), it may have adversely affected the accuracy. As shown in Appendix C.4, similar results were observed on the other tabular datasets.

To analyze how the explanations changed by employing ID-ExpO, we visualize the feature contributions by LIME with ID-ExpO and ℓ_{CE}-only. We show a typical example in the wine-quality-red dataset in Figure A.6. In many samples, including this, we found that ID-ExpO tended to bring larger positive or negative feature contributions than ℓ_{CE}-only, although LIME for both has the same setting. This is because that ID-ExpO adjusts the predictor's behaviors so that important features are taken into account early in the calculation of the insertion and deletion metrics. Since the explanation with such feature contributions makes features that users should focus on more clear, ID-ExpO can be effective in producing explanations that are easy to understand for the users.

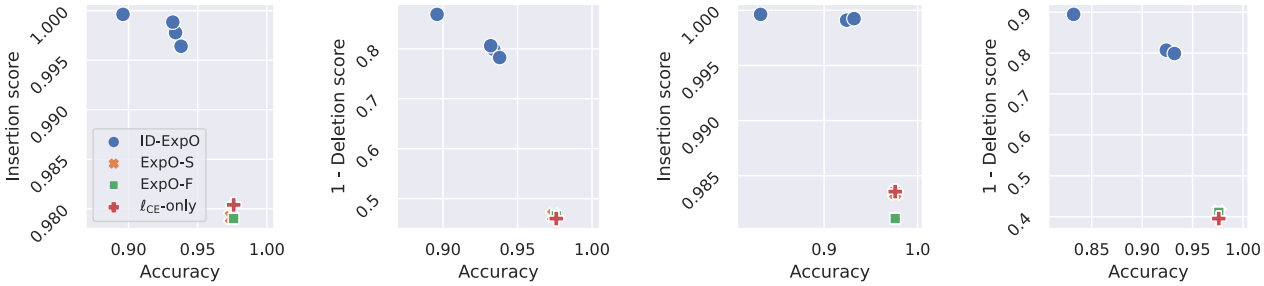
C.4 All Quantitative Results on Tabular Datasets

Figures A.7–A.9 show the insertion and the one-minus-deletion scores against the accuracy on the six tabular datasets. On all the datasets, ID-ExpO outperformed the others in terms of the insertion and the one-minus-deletion scores for any in the range of η. With the accuracy, by putting emphasis on the accuracy, i.e., by setting $\eta = 3$, we found that ID-ExpO could keep comparable or slightly low accuracy compared to the others, while the highest insertion and one-minus-deletion scores. However, in some cases of using KernelSHAP, e.g., Figures A.8(C) and A.9(C), we found that the accuracy of ID-ExpO degraded compared to the other methods.

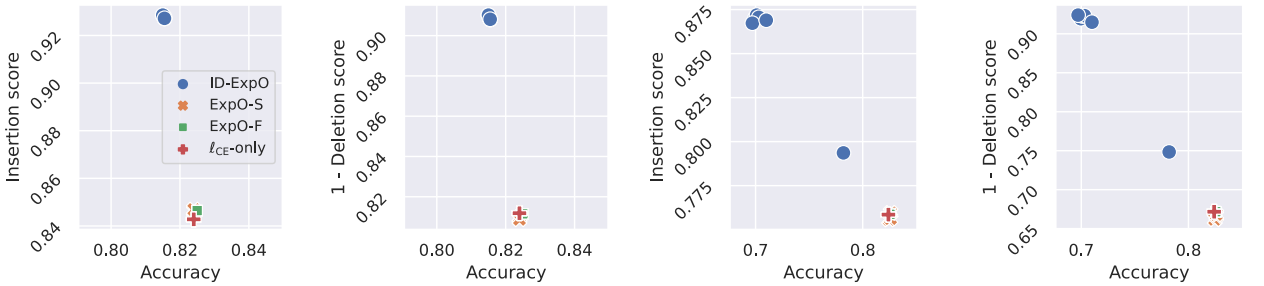
(A) LIME and KernelSHAP on collins ($S = 0.3 \cdot Q$)



(B) LIME and KernelSHAP on collins ($S = 0.5 \cdot Q$)



(C) LIME and KernelSHAP on mfeat-fourier ($S = 0.3 \cdot Q$)



(D) LIME and KernelSHAP on mfeat-fourier ($S = 0.5 \cdot Q$)

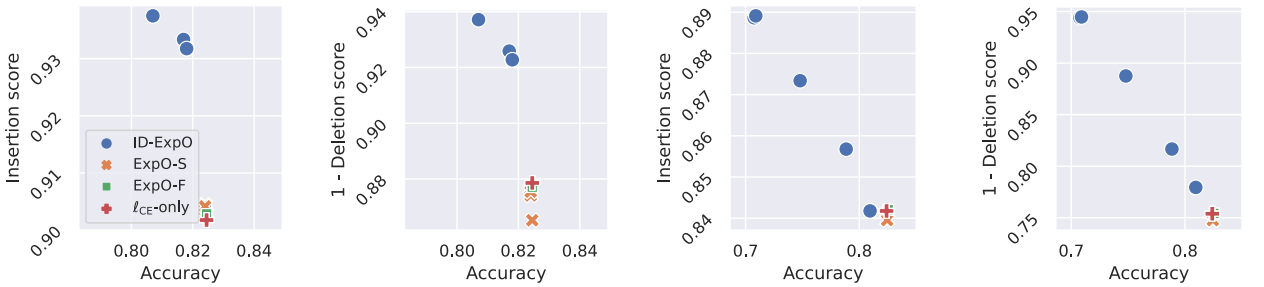
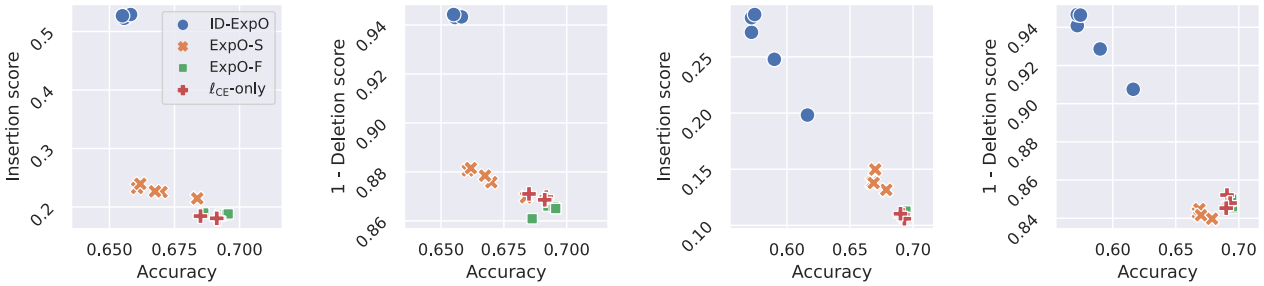
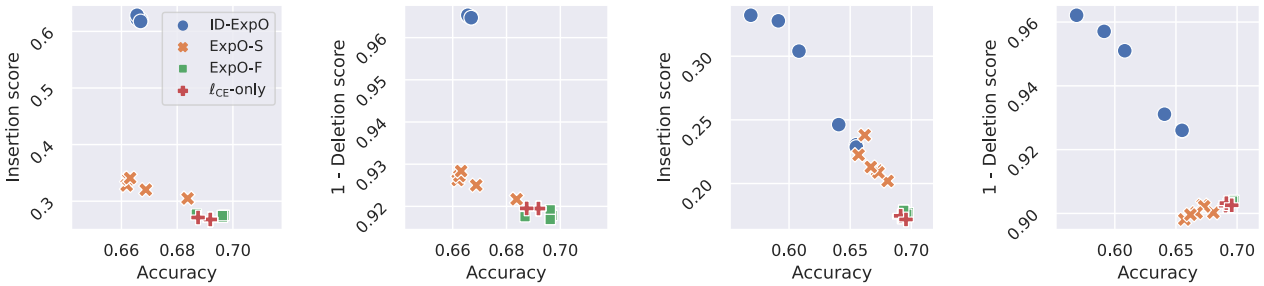


Figure A.7: Mean insertion and mean one-minus-deletion scores against accuracy on collins and mfeat-fourier datasets. The scores are averaged over the five training/validation/test sets. The first two columns show the results on LIME, while the others show the results on KernelSHAP. Each point has a different accuracy weight $\eta \in \{0.5, 1.0. \cdots, 3.0\}$. The higher the score, the better.

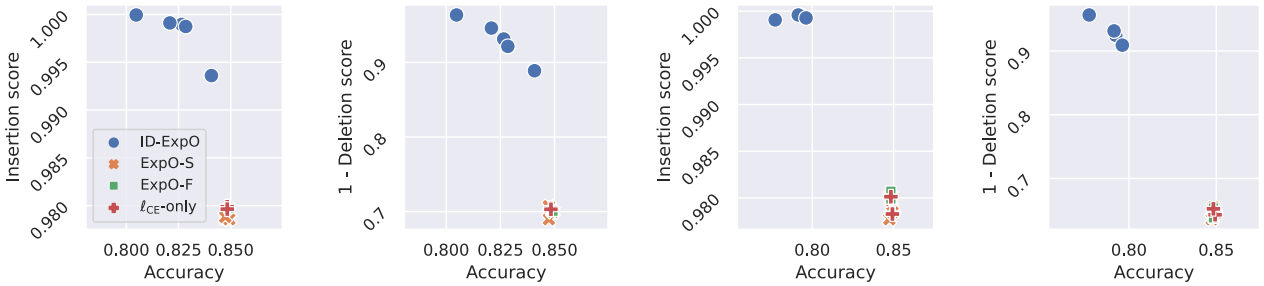
(A) LIME and KernelSHAP on one-hundred-plants-shape ($S = 0.3 \cdot Q$)



(B) LIME and KernelSHAP on one-hundred-plants-shape ($S = 0.5 \cdot Q$)



(C) LIME and KernelSHAP on qsar-biodeg ($S = 0.3 \cdot Q$)



(D) LIME and KernelSHAP on qsar-biodeg ($S = 0.5 \cdot Q$)

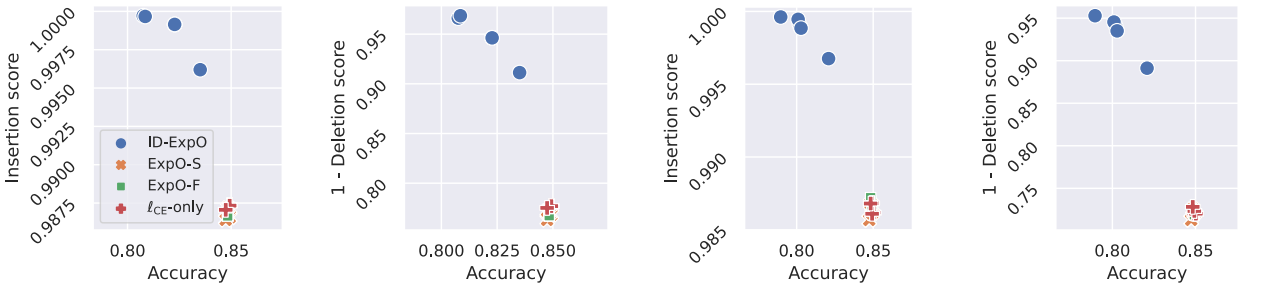
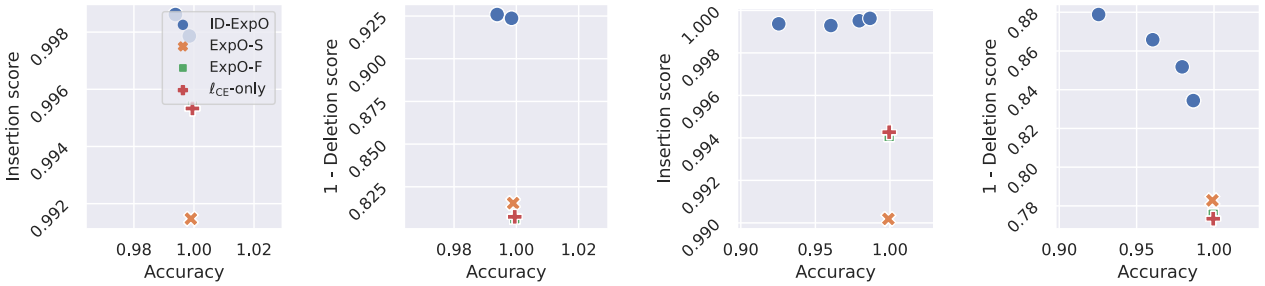
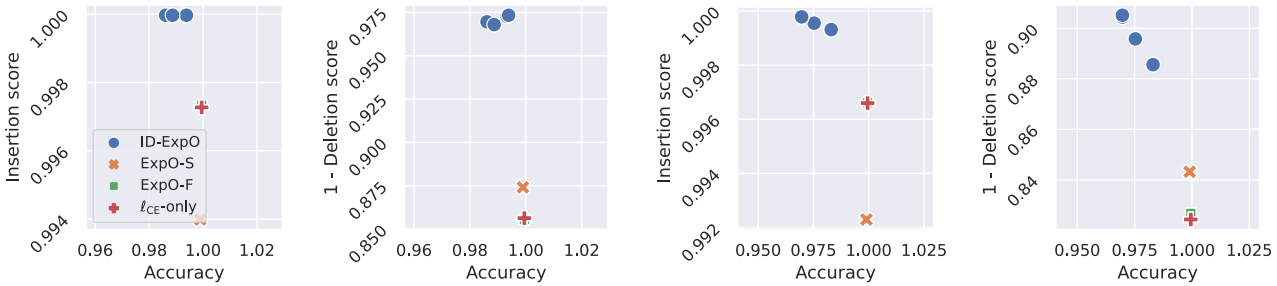


Figure A.8: Mean insertion and mean one-minus-deletion scores against accuracy on one-hundred-plants-shape and qsar-biodeg datasets. How to read these figures is the same as Figure A.7.

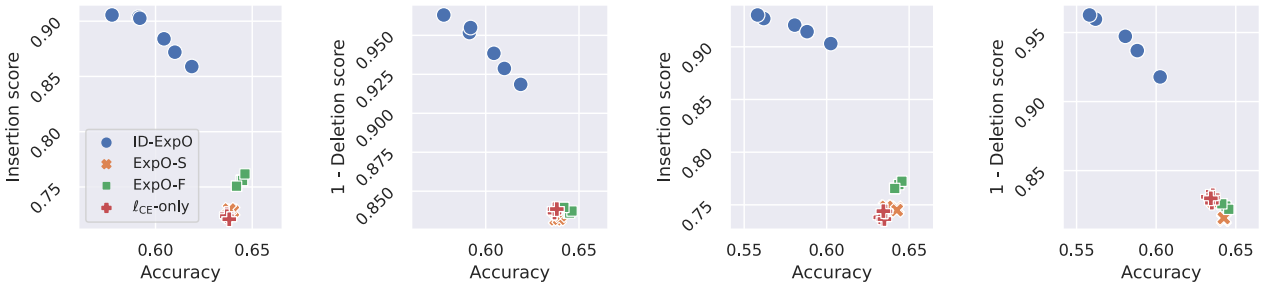
(A) LIME and KernelSHAP on steel-plates-fault ($S = 0.3 \cdot Q$)



(B) LIME and KernelSHAP on steel-plates-fault ($S = 0.5 \cdot Q$)



(C) LIME and KernelSHAP on wine-quality-red ($S = 0.3 \cdot Q$)



(D) LIME and KernelSHAP on wine-quality-red ($S = 0.5 \cdot Q$)

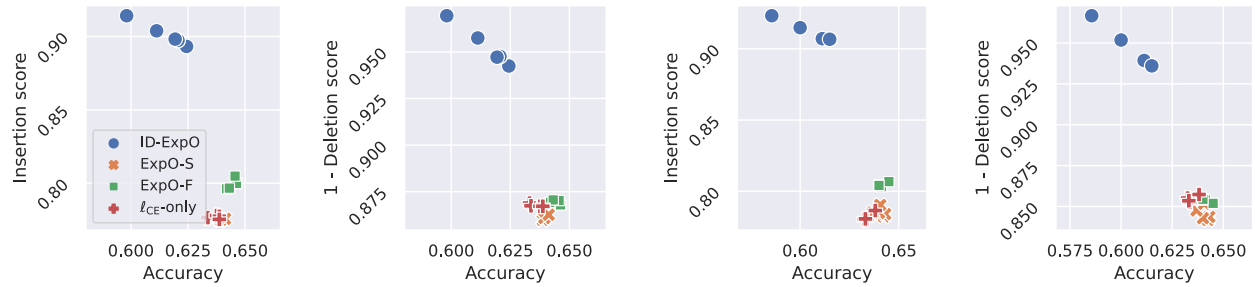


Figure A.9: Mean insertion and mean one-minus-deletion scores against accuracy on steel-plates-fault and wine-quality-red datasets. How to read these figures is the same as Figure A.7.

D Broader Impact

The proposed method can contribute to producing faithful explanations that capture the predictor's behaviors well. However, the fact does not guarantee that the explanations are easy-to-understand for humans. If the predictor and explainer that are trained using the proposed method produce explanations that are faithful but difficult to understand for humans, they might give users wrong interpretations of the prediction results. There are several studies that explore producing explanations that are easy to understand for humans, which include human-in-the-loop approaches (Lage et al., 2018; Gao et al., 2022) and using ground truths of explanations by human annotators (Ross et al., 2017; Balayan et al., 2020). By using the proposed method together with such approaches, we can alleviate the concern of such misinterpretation while keeping high faithfulness in the explanations.

E Limitations

The proposed method can contribute to producing faithful explanations that capture the predictor's behaviors well. However, the fact does not guarantee that the explanations are easy-to-understand for humans. There are several studies that explore producing explanations that are easy to understand for humans, which include human-in-the-loop approaches (Lage et al., 2018; Gao et al., 2022) and using the ground truth of explanations by human annotators (Ross et al., 2017; Balayan et al., 2020). When we require easy-to-understand explanations, combining the proposed method with such approaches would result in producing faithful and easy-to-understand explanations.

The proposed method is applicable to a wide range of predictors and explainers. In practice, the computational complexities of the predictor and explainer we use can be barriers to using the proposed method. The perturbation-based explainers, such as LIME and KernelSHAP, produce an explanation for an input sample by using M perturbed samples around the input sample. In our experiment, M was set to 200, and the mini-batch size was 128. This means that $200 \times 128 = 25,600$ samples were used to update the predictor's parameters θ once. For this issue, fast computation is possible by performing the data parallel training using multiple GPUs. On the other hand, using Grad-CAM, one of the gradient-based explainers, as an explainer in the proposed method is computationally more efficient than LIME and KernelSHAP because it does not need to increase the sample. Note that, although those computational complexities affect training efficiency using the proposed method, the computational complexities of the predictor and explainer in testing are invariant before and after applying the proposed method.