Journal of Machine Learning Research 26 (2025) 1-35 Submitted 10/22; Revised 7/25; Published 7/25

EMaP: Explainable AI with Manifold-based Perturbations

Minh Nhat Vu MVU@LANL.GOV
Theoretical Division, Los Alamos National Laboratory,
Los Alamos, NM, 87545, USA

Huy Quang Mai HUYQMAI1@GMAIL.COM
Hanoi, 10000, Vietnam

My T. Thai MYTHAI@CISE.UFL.EDU
Department of Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611, USA

Editor: Florence d'Alche-Buc

Abstract

In the last few years, many explanation methods based on the perturbations of input data have been introduced to shed light on the predictions generated by black-box models. The goal of this work is to introduce a novel perturbation scheme so that more faithful and robust explanations can be obtained. Our study focuses on the impact of perturbing directions on the data topology. We show that perturbing along the orthogonal directions of the input manifold better preserves the data topology, both in the worst-case analysis of the discrete Gromov-Hausdorff distance and in the average-case analysis via persistent homology. From those results, we introduce EMaP algorithm, realizing the orthogonal perturbation scheme. Our experiments show that EMaP not only improves the explainers' performance but also helps them overcome a recently developed attack against perturbation-based explanation methods.

Keywords: black-box explanations, explainability, topological data analysis, adversarial robustness

1. Introduction

In recent years, many attempts to explain predictions of deep learning models have been conducted, which resulted in various explanation methods called *explainers* (Lipton, 2018; Murdoch et al., 2019). A common technique used by many explainers (Štrumbelj and Kononenko, 2013; Ribeiro et al., 2016; Lundberg and Lee, 2017) is first to generate some perturbations in the input space, then forward them through the model, and later provide an explanation based on the captured outputs. For that reason, these methods are also known as *perturbation-based explainers*.

Even though the perturbation-generating step has a strong influence on the performance of explainers (Ribeiro et al., 2016; Lundberg and Lee, 2017), very few works closely examined this step. Current perturbation schemes often ignore the data topology and distort it significantly as a result. These distortions may considerably degrade explainers' performance since models are not trained to operate on the deformed topology. Additionally, the difference between the perturbations and the original data creates opportunities for mali-

cious intent. For example, the work (Slack et al., 2020) demonstrates that a discriminator trained to recognize the explainer's perturbations can be exploited to fool the explainer.

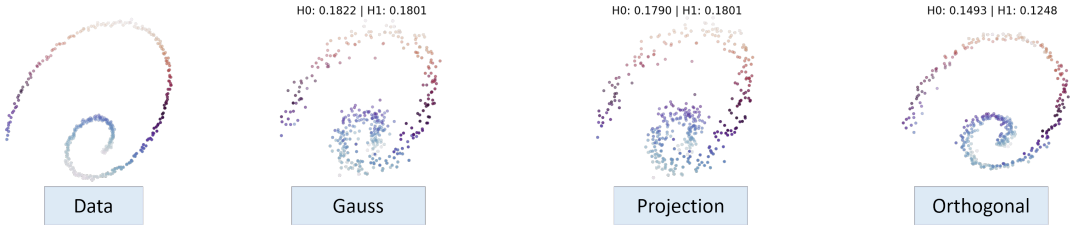


Figure 1: Visualization of perturbations with the same magnitude generated from a point cloud of a 2-dimensional spiral. Perturbations along the orthogonal directions of the data subspace (far-right) result in lower topological distortion, i.e. smaller Bottleneck distances H_0 and H_1 (the Bottleneck distance is discussed in Section 5).

Motivated by that lack of study, our work aims to redesign the perturbation step in the explanation process. We begin with the observation that there exists important topological information of the data having strong influences on the model trained on that data. For example, training a neural network to distinguish points from two separated point clouds is generally easier than on two overlapping/connected point clouds. Since perturbing the data can significantly change the topological information, perturbation methods can significantly change the model's behavior and influence the subsequent explaining tasks. As such, we aim to generate perturbations so that the topological structure of the original data is better preserved. Our key result is that, assuming the input data is embedded in an affine subspace whose dimension is significantly smaller than that of the data dimension, eliminating the perturbations' components along that affine subspace would better preserve the topological integrity of the original manifold. An illustration of that result is provided in Fig. 1, which shows that perturbing along the orthogonal directions (i.e., no subspace's directions) results in smaller distortion in the topological structure of the original data. This phenomenon is also reflected in the smaller Bottleneck distances in dimensions 0 and 1, denoted by H_0 and H_1.

Based on that result, we further propose a novel manifold-based perturbation method aiming to preserve the topological structure of the original data, called EMaP. The high-level operations of EMaP are shown in Fig. 2. Given some sampled data, EMaP first learns a function mapping the samples to their low-dimensional representations in the data subspace. Then, that function is used to approximate a local affine subspace, shortened to local-subspace, containing the data in the neighborhood of the input to be explained. Finally, the EMaP perturbations are generated by adding the noise vectors that are orthogonal to that local-subspace to the data.

Contributions. **(a)** We provide theoretical results showing that the worst-case discrete Gromov-Hausdorff distance between the data and the perturbations along the manifold's directions is larger than that along the orthogonal directions. **(b)** The worst-case analysis suggests that eliminating perturbation's components along the manifold's directions can

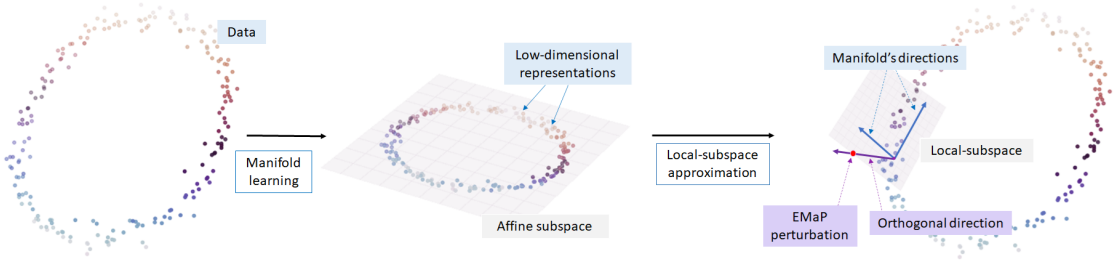


Figure 2: EMaP's perturbation: Assume the data is embedded in a low-dimensional affine subspace (middle figure), EMaP approximates that subspace locally at some given data points and performs perturbation along orthogonal directions of that subspace (right figure).

generally better maintain the topological integrity of the original manifold, i.e. the average case. We then provide synthetic and real-world experiments based on persistent homology and Bottleneck distance to support that hypothesis. **(c)** We propose EMaP, an algorithm generating perturbations along the manifold's orthogonal directions for explainers. EMaP first approximates the input's manifold locally at some given data points, called pivots, and the explained data point. The perturbations are then generated along the orthogonal directions of these local subspaces. EMaP also computes the low-dimensional distances from the perturbations to the explained data point so that the explainers can better explain the model. **(d)** Finally, we provide experiments on four text data sets, two tabular data sets, and two image data sets, showing that EMaP can improve the explainer's performance and protect explainers from adversarial discriminators.

Organization. The remainder of the paper is structured as follows. Sections 2 and 3 briefly discuss related work and preliminaries. Section 4 presents our analysis of the discrete Gromov-Hausdorff distances of different perturbation directions, which suggests orthogonal directions are preferable. We strengthen that result with a persistent homology analysis in Section 5. Sections 6 and 7 describes our proposed EMaP algorithm and its experimental results. Section 8 concludes the paper.

2. Related work

This work intersects several emerging research fields, including explainers and their attack/defense techniques. Our approach also uses recent results in topological data analysis. We provide an overview of the related work below.

Perturbation-based explanation methods. Perturbation-based explainers are becoming more popular among explanation methods for black-box models since they hardly require any knowledge of the explained model. Notable ones are LIME (Ribeiro et al., 2016), SHAP (Lundberg and Lee, 2017), and some others (Štrumbelj and Kononenko, 2013; Zeiler and Fergus, 2014; Sundararajan et al., 2017; Chang et al., 2018; Schwab and Karlen, 2019; Lundberg et al., 2020; Ying et al., 2019; Vu and Thai, 2020). While they share the same goal to explain the model's predictions, they are not only different in the objectives but also

in their perturbation schemes: some zero out features (Zeiler and Fergus, 2014; Schwab and Karlen, 2019) or replace features with neutral values (Ribeiro et al., 2016; Sundararajan et al., 2017), others marginalize over some distributions on the data (Ribeiro et al., 2016; Lundberg and Lee, 2017; Lundberg et al., 2020). There also exist methods relying on separate models to generate perturbations (Štrumbelj and Kononenko, 2013; Chang et al., 2018). The work (Covert et al., 2021) provides a comprehensive survey on those perturbation-based explanation methods and how they perturb the data.

Adversarial attack on explainers. We focus on the attack framework (Slack et al., 2020), in which the adversary intentionally hides a biased model from the explainer by training a discriminator to recognize its query. The framework will be discussed in detail in Section 3. There are other emerging attacks on explainers focusing on modifying the model's weights and tampering with the input data (Ghorbani et al., 2019; Dombrowski et al., 2019; Heo et al., 2019; Dimanov et al., 2020).

Defense techniques for perturbation-based explainers. Since most attacks on perturbation-based explainers were only developed recently, defense techniques against them are quite limited. Existing defenses generate perturbations either from carefully sampling the training data (Chakraborty et al., 2020) or from learning some generative models (Saito et al., 2020; Vres and Robnik-Sikonja, 2021). The advantage of EMaP is that it does not require any generative model, which not only reduces the attack surface but also allows theoretical study of the perturbations.

Topological Data Analysis. Topological data analysis (TDA) is an emerging field in mathematics, applying the techniques of topology (which was traditionally very theoretical) to real-world problems. Notable applications are data science, robotics, and neuroscience. TDA uses deep and powerful mathematical tools in algebraic topology to explore topological structures in data and to provide insights that normal metric-based methods fail to discern. The most common tool in the TDA arsenal is persistent homology, developed in the early 2000s by Gunnar Carlsson and his collaborators. We refer readers to Ghrist (2014); Edelsbrunner and Harer (2010) for an overview of both persistent homology and TDA as a whole.

3. Preliminaries

This section discusses the notations and some preliminaries for this work. In particular, we use the standard setting of the learning tasks where the set of input X is sampled from a distribution on $\mathbb{R}^N$. X is also assumed to be in a manifold embedded in an affine subspace $\mathbb{R}^V$, where V is much smaller than N. We also consider a black-box classifier f mapping each input $x \in X$ to a prediction y, a local explainer g, a (adversarial) discriminator $\mathcal{D}$, and a masking-model f'.

In the following, we describe how explainers and the explanation process are formulated. After that, we provide two examples demonstrating how perturbations along different directions can have significant impacts on the explaining process. We end this section with an attacking framework on perturbation-based explainers and describe how better perturbations can help defend against that class of attacks.

Explainers. An explanation of the prediction $y = f(x)$ can be obtained by running an explainer g on x and f. We denote such an explanation by $g(x)$. In additive feature

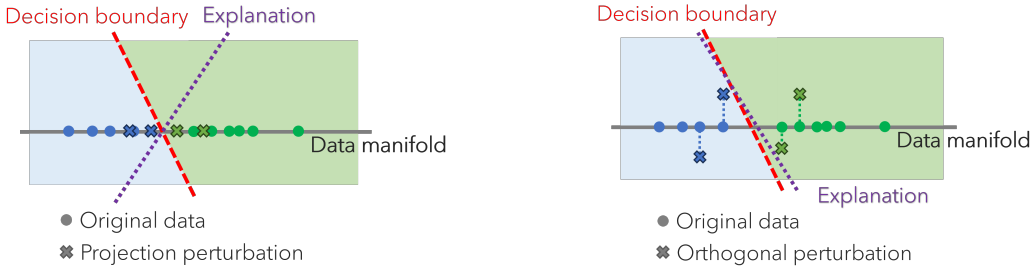
attribution methods, the range of $g(x)$ is a set of features' importance scores. We focus our analysis on the class of perturbation-based explanation, i.e., the importance scores are computed based on the model's predictions of some perturbations of the input data. Typically, the perturbations are generated by perturbing x or/and some samples in X. We denote the perturbations by X_r, where $r \geq 0$ specifies the amount of perturbation. A more rigorous definition for this notation will be provided in Section 4.

Given the perturbations, the explainer typically finds the explanation g such that its behavior on the perturbations is similar to that of the explained function f. That is the reason why they are also known as *surrogate methods*. For example, the LIME explanation is a linear model whose coefficients are the importance score of the features:

$$\arg\min_{g \in \mathcal{G}} \mathcal{L}(f, g, \pi_x) + \Omega(g), \tag{1}$$

where $\mathcal{L}$ is a loss between the explained function f and the explanation function g, $\mathcal{G}$ is the class of linear models and π_x is an exponential kernel defined on some distance function and $\Omega(g)$ is a function measuring the complexity of g. For the SHAP explainer, the main differences are the choices of the kernel π_x and the complexity measure $\Omega(g)$ so that the explanations satisfy certain desired properties of Game Theory. For details of the two methods, we refer readers to the original papers (Ribeiro et al., 2016; Lundberg and Lee, 2017).

Topological-awareness perturbations for explaining tasks. We now provide two examples demonstrating how leveraging topological information of the data can be beneficial for the explanation tasks. The examples aim to demonstrate two key observations. The first is that the usage of only projecting perturbations, i.e., perturbations in the manifold, might lead to high explaining errors. The second example heuristically shows orthogonal perturbations resulting in a more desirable model response for the explaining task.

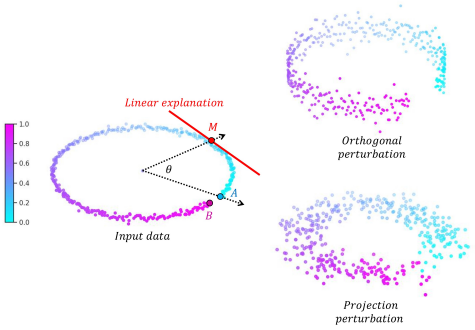


(a) Explaining using projecting perturbation. (b) Explaining using orthogonal perturbation.

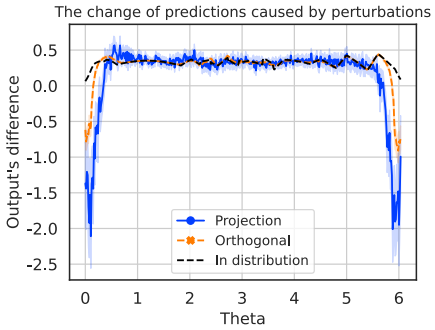
Figure 3: A 1-D example showing the impact of the perturbation's direction.

In the first example, we consider 2-dimensional data lying in a 1-dimensional manifold (Fig. 3). The task of the model is to differentiate the blue data from the green data. The decision boundary of the model is a line intersecting the manifold (the red dashed line). The goal of perturbation-based explainers is to determine that decision boundary. Fig. 3a demonstrates the scenario when only projecting perturbations are used. Since the perturbations and the data lie in a 1-dimensional manifold, they cannot help the explainer differentiate the decision boundary from any other lines going through the intersection.

Thus, the resulting explanation (purple dot line) might have high errors. On the other hand, as depicted in Fig. 3b, orthogonal perturbations can differentiate lines in the 2-dimensional space and help the explainer learn solutions with lower errors.



(a) A C-shaped data in a regression task (left) and its perturbations (right). The projection perturbation of the data is more likely to destroy the *gap* at the two ends of the original C-shaped data.



(b) The figure reports the normalized change $\frac{|f(x+\delta)-f(x)|}{\|\delta\|_{L_2}}$ for x along the C-shaped data (located by the angle θ). The perturbation $x+\delta$ belongs to the projection point cloud, the orthogonal point cloud, and the original data as indicated in the legend.

Figure 4: An example showing how leveraging topological information can be beneficial for the explanation task.

Our second example is the experiment shown in Fig. 4. The experiment consists of a C-shaped data set and a regression model f mapping the data set uniformly to the range $(0, 1)$, i.e., $f(x) = \theta_x/(2\pi-0.5)$ for $\theta_x \in [0, 2\pi-0.5]$, where θ_x is the angular coordinate of the point x on the shape. That angle θ_x, the mapping f, and the data set are illustrated in Fig. 4a. The first observation is that the optimal linear local explanation (1) of this regression model is the tangent line at the point of prediction. To obtain that explanation, one necessary condition of the perturbations is that the following quantity $f(x+\delta) - f(x) = \frac{\theta_{x+\delta}-\theta_x}{2\pi-0.5}$ should depend only on δ but not x. In other words, the change $f(x+\delta) - f(x)$ should be almost constant when x moves along the shape. Particularly, Fig. 4b demonstrates the low variance of that quantity (black line) when the perturbation $x+\delta$ is sampled from the original C-shaped data, i.e., in distribution[1].

For projection and orthogonal perturbations, we can observe that $f(x+\delta)-f(x)$ matches that of in-distribution perturbation, except for data at the two ends of the C-shape. The reason is that the two ends have important topological information on the data, and that information also has a strong impact on the model: if the data is slightly perturbed at those ends, the data might become connected (a circle appears,) and the two extreme values of the regression model will also be connected. Thus, the evaluations of the model on the perturbations connecting the two ends, i.e,. on those that destroy the associated

1. To sample the perturbation $x+\delta$, we first fix a perturbation magnitude Δ, then randomly select the point x' in the specified point cloud (in-data, projection or orthogonal perturbations) such that the L_2 distance between x' and x equal to Δ, i.e., $\|\delta\|_{L_2} = \Delta$. The value of Δ for this experiment is 0.25.

topological feature, will result in unstable responses. Consequently, explaining the model based on those perturbations is not desirable. In fact, the distortion $f(x+\delta) - f(x)$ caused by projection perturbations has significantly higher fluctuation than that of the orthogonal. The reason is that projection perturbations near the two ends of the C-shaped have a higher chance of connecting the ends and destroying the shape.

Attack framework. We study the discriminator-based attack framework introduced by Slack et al. (2020), which is illustrated in Fig. 5. The framework focuses on an adversary with an incentive to deploy a biased-model f. This adversary can bypass the detection of the explainer by forwarding the explainer's perturbations X_r to a masking model f'. The decision whether to forward an input to the masking model is made by a discriminator $\mathcal{D}$. Thus, the success of the attack is dictated by the capability of the discriminator $\mathcal{D}$ to distinguish the perturbations X_r from the actual data X. Intuitively, if the explainer can craft an X_r similar to X, it not only improves the explainer's performance but also prevents the adversary from hiding its bias.

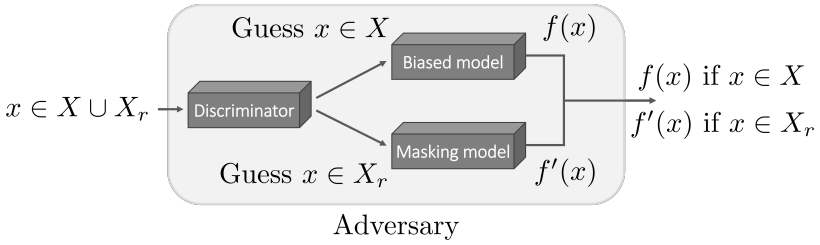


Figure 5: The discriminator-based attack framework: By recognizing and forwarding the perturbations X_r generated by an explainer to the masking model f', the biased model f can be deployed without detection.

4. Analysis of Discrete Gromov-Hausdorff distances of perturbations

We consider the following perturbation problem: Given a manifold embedded in $\mathbb{R}^N$, how do we perturb it so that we preserve as much topological information as possible? More concretely, given a finite set of points sampled from such a manifold, is there a consistent method to perturb the original data set while preserving some notion of topology?

To begin talking about differences between (metric) spaces, we need to introduce a notion of distance between them. One such commonly used distance is the **Gromov-Hausdorff distance**, which is derived from the **Hausdorff distance**. Intuitively, a small Gromov-Hausdorff distance means that the two spaces are very similar as metric spaces. Thus, we can focus our study on the Gromov-Hausdorff distances between the data and different perturbation schemes. However, as it is infeasible to compute the distance in practice, we instead study an approximation of it, which is the ***discrete* Gromov-Hausdorff distance**. Specifically, we show that, when the perturbation is significantly small, the worst-case discrete Gromov-Hausdorff distance resulting from orthogonal perturbation is smaller than that of projection perturbation, i.e., perturbation along the manifold (Theorem 3). The proof of that claim relies on Lemma 4, which states that, with a small perturbation, the

discrete Gromov-Hausdorff distance between the original point cloud and the perturbation point cloud equals the largest change in the distances of any pair of points in the original point cloud. With the Lemma, the problem of comparing point clouds is further reduced to the problem of comparing the change in distances.

The structure of this section is as follows: first, we recall the definition of the Hausdorff distance and the Gromov-Hausdorff distance. We then introduce the discrete Gromov-Hausdorff distance, which will be the main focus for computational purposes. Finally, using such a discrete distance, we prove Lemma 4 and Theorem 3.

We now state the formal definitions. Let (M, d) be a metric space. For a subset $S \subseteq M$ and a point $y \in M$, the distance between S and y is given by $d(S, y) := \inf_{x \in S} d(x, y)$.

Definition 1 (Hausdorff distance) *(Gromov, 1981) Let S and S' be two non-empty subsets of a metric space (M, d). The Hausdorff distance between S and S', denoted by $d_H(S, S')$ is:*

$$d_H(S, S') := \max\big(\sup_{x \in S} d(S', x), \sup_{y \in S'} d(S, y)\big).$$

Definition 2 (Gromov-Hausdorff distance) *(Gromov, 1981) Let X, Y be two compact metric spaces. The Gromov-Hausdorff distance between X and Y is given by:*

$$d_{GH}(X, Y) = \inf_{f,g} d_H(f(X), g(Y)),$$

where the infimum is taken over all metric spaces M and all isometric embeddings $f : X \to M$, $g : Y \to M$.

Even though the Gromov-Hausdorff distance is mathematically desirable, it is practically non-computable since the above infimum is taken over all possible metric spaces. In particular, this includes the computation of the Gromov-Hausdorff distance between any two point clouds. In 2004, Mémoli and Sapiro (2004) addressed this problem by using a discrete approximation of Gromov-Hausdorff, which looks at the distortion of pairwise distances over all possible matchings between the two point clouds. Formally, given two finite sets of points $X = \{x_1, ..., x_n\}$ and $Y = \{y_1, ..., y_n\}$ in a metric space (M, d), the discrete Gromov-Hausdorff distance between X and Y is given by

$$d_J(X, Y) = \min_{\pi \in S_n} \max_{i,j} \frac{1}{2}\big|d(x_i, x_j) - d(y_{\pi(i)}, y_{\pi(j)})\big|, \tag{2}$$

where S_n is the set of all n-permutations.

Let $X = \{x_1, ..., x_k\} \in \mathbb{R}^V \subseteq \mathbb{R}^N$ be a point cloud contained in some affine subspace $\mathbb{R}^V$ of $\mathbb{R}^N$. We say X is generic if the pairwise distances between the points in X are not all equal, i.e. there exist some points $x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4} \in X$ such that $d(x_{i_1}, x_{i_2}) \neq d(x_{i_3}, x_{i_4})$.

Let X_r be a finite set of points in $\mathbb{R}^N$ s.t. for every $x_i \in X$, there exists a unique $\tilde{x}_i \in X_r$ such that $d(x_i, \tilde{x}_i) = r$. X_r realizes a perturbation of X with the radius of perturbation being equal to r. We also denote $X_r^{\perp}$ (resp. X_r^{Proj}) as a finite set of points in $\mathbb{R}^N$ such that for every $x_i \in X$, there exists a unique $\tilde{x}_i \in X_r^{\perp}$ such that $d(x_i, \tilde{x}_i) = r$ and $\overline{x_i\tilde{x}_i} \perp \mathbb{R}^V$ (resp. $\overline{x_i\tilde{x}_i} \subset \mathbb{R}^V$), where $\overline{x_i\tilde{x}_i}$ denotes the line connecting the points x_i and $\tilde{x}_i$. We are now ready to state the following key theorem:

Theorem 3 *Given a generic point-cloud $X = \{x_1, ..., x_k\} \in \mathbb{R}^V \subset \mathbb{R}^N$, there exists an $r_0 > 0$ such that for any $r < r_0$ and for any instances of $X_r^{\perp}$, there exists an X_r^{Proj} such that:*

$$d_J(X, X_r^{\perp}) \leq d_J(X, X_r^{\text{Proj}}).$$

We prove Theorem 3 is based on the following lemma:

Lemma 4 *There exists an $\epsilon > 0$ such that for any $r < \epsilon$, we have*

$$d_J(X, X_r) = \frac{1}{2} \max_{i,j} |d(x_i, x_j) - d(\tilde{x}_i, \tilde{x}_j)|.$$

for any X_r.

To prove Lemma 4, we show that, for a small enough ϵ, the optimal permutation in Eq. (2) is the identity $\pi(i) = i$. Thus, the minimization in the computation of d_J can be eliminated. The detail is shown in the following.

Proof of Lemma 4. Given a permutation $\pi \in S_n$ and two point clouds X, Y of the same cardinality, denote:

$$D_\pi(X, X_r) = \frac{1}{2} \max_{i,j} |d(x_i, x_j) - d(\tilde{x}_{\pi(i)}, \tilde{x}_{\pi(j)})|.$$

Let N_0 be the set of permutations $\pi \in S_n$ such that $D_\pi(X, X) = 0$. Let $N_1 = S_n \backslash N_0$. Since X is generic, N_0 does not include all of S_n and $N_1 \neq \emptyset$.

Let $\delta = \min_{\pi \in N_1} D_\pi(X, X) > 0$. We claim that choosing $\epsilon = \frac{\delta}{4}$ proves the lemma. To be more precise, the radius of perturbation r is chosen such that $r < \epsilon = \frac{\delta}{4}$, i.e. $4r < \delta$.

Given an X_r, for any $\pi \in S_n$, we consider two cases:

1. If $\pi \in N_0$, then $d(x_i, x_j) = d(x_{\pi(i)}, x_{\pi(j)}) \ \forall \ (i, j)$. Note that the identity permutation belongs to N_0 as:

$$\begin{aligned} D_\pi(X, X_r) &= \frac{1}{2} \max_{i,j} |d(x_i, x_j) - d(\tilde{x}_{\pi(i)}, \tilde{x}_{\pi(j)})| \\ =& \frac{1}{2} \max_{i,j} |d(x_{\pi(i)}, x_{\pi(j)}) - d(\tilde{x}_{\pi(i)}, \tilde{x}_{\pi(j)})| = \frac{1}{2} \max_{i,j} |d(x_i, x_j) - d(\tilde{x}_i, \tilde{x}_j)|. \end{aligned} \tag{3}$$

   Since $d(x_i, \tilde{x}_i) = d(x_j, \tilde{x}_j) = r$, from Triangle inequality, we have:

$$\begin{aligned} d(\tilde{x}_i, \tilde{x}_j) &\leq d(x_i, x_j) + d(x_i, \tilde{x}_i) + d(x_j, \tilde{x}_j) = d(x_i, x_j) + 2r, \\ d(\tilde{x}_i, \tilde{x}_j) &\geq d(x_i, x_j) - d(x_i, \tilde{x}_i) - d(x_j, \tilde{x}_j) = d(x_i, x_j) - 2r. \end{aligned}$$

   Therefore, for all i, j, we have:

$$|d(\tilde{x}_i, \tilde{x}_j) - d(x_i, x_j)| \leq 2r. \tag{4}$$

   This implies $D_\pi(X, X_r) \leq r$ for $\pi \in N_0$.

2. If $\pi \in N_1$, without loss of generality, we assume the pair (x_1, x_2) maximizes $|d(x_i, x_j) - d(x_{\pi(i)}, x_{\pi(j)})|$. For convenience, we denote $\pi(1) = 3$ and $\pi(2) = 4$. From the fact that $\pi \in N_1$, we have $|d(x_1, x_2) - d(x_3, x_4)| \geq \delta$. On the other hand, from (4), $|d(\tilde{x}_3, \tilde{x}_4) - d(x_3, x_4)| \leq 2r$. Thus, from the Triangle inequality, we obtain:

$$\begin{aligned} |d(\tilde{x}_3, \tilde{x}_4) - d(x_1, x_2)| \geq & |d(x_1, x_2) - d(x_3, x_4)| - |d(\tilde{x}_3, \tilde{x}_4) - d(x_3, x_4)| \\ \geq & \delta - 2r \geq 2r, \end{aligned}$$

Since $D_\pi(X, X_r) \geq \frac{1}{2}|d(x_1, x_2) - d(\tilde{x}_3, \tilde{x}_4)|$, we establish $D_\pi(X, X_r) \geq r$ for $\pi \in N_1$.

From the above analysis, we can conclude $D_\pi(X, X_r) \leq D_\tau(X, X_r)$ for all $\pi \in N_0$ and $\tau \in N_1$. Combining this with (3), we have that the identity permutation is the solution of (2), which proves the Lemma. ■

With the Lemma, Theorem 3 can be proved by choosing a specific projection perturbation such that its discrete Gromov-Hausdorff distance is always bigger than the upper bound for such distance of any orthogonal perturbations. The proof of the Theorem is shown below:

Proof of Theorem 3. Applying Lemma 4 to the orthogonal perturbation $Y = X_r^{\perp}$ and projection perturbation $Z = X_r^{\text{Proj}}$, and for any r less than or equal to the minimum of the ϵ corresponding to each perturbation specified in Lemma 4, we obtain:

$$d_J(X, Y) = \frac{1}{2} \max_{i,j} |d(x_i, x_j) - d(y_i, y_j)| \quad \text{and} \quad d_J(X, Z) = \frac{1}{2} \max_{i,j} |d(x_i, x_j) - d(z_i, z_j)|.$$

From the triangle inequality (similar to how we show Eq. (4) in the proof of Lemma 4), we have:

$$\frac{1}{2} \max_{i,j} |d(x_i, x_j) - d(y_i, y_j)| < r \quad \text{and} \quad \frac{1}{2} \max_{i,j} |d(x_i, x_j) - d(z_i, z_j)| \leq r,$$

where the first inequality is strict due to the orthogonality of the perturbation.

Given such r, consider the following perturbation $Z = X_r^{\text{Proj}}$ where x_1, x_2, z_1, z_2 are collinear and $|d(x_1, x_2) - d(z_1, z_2)| = 2r$, and $z_i = x_i$ for $i = 3, ..., n$. The d_J distance between such Z and X is greater than or equal to r, which proves our claim. ■

This theoretical result suggests the following perturbation scheme: Given a manifold embedded in an affine subspace of $\mathbb{R}^N$ and a fixed amplitude r of perturbation, perturbing the manifold in the orthogonal directions with respect to the affine subspace is preferable to random perturbation, as it minimizes the topological difference between the perturbed manifold and the original.

5. Persistent homology analysis with the Bottleneck distance

The results from the previous section show that on the worst-case basis, the orthogonal perturbation is preferable to the projection perturbation. However, when we apply them to

actual data sets, how do they compare on average? Since the discrete Gromov-Hausdorff distance is still computationally intractable for a Monte-Carlo analysis, we choose a different approach: **persistent homology**.

For the last 30 years, there have been new developments in the field of algebraic topology, which was classically very abstract and theoretical, toward real-world applications. The newly discovered field, commonly referred to as *applied topology* or *topological data analysis*, is centered around a concept called *persistent homology*. Interested readers can refer to (Ghrist, 2014; Edelsbrunner and Harer, 2010; Ghrist, 2008) for an overview of the subject.

For any topological space, homology is a topological invariant that counts the number of *holes* or *voids* in the space. Intuitively, given any space, the 0^{th}-homology counts the number of connected components, the 1^{st}-homology counts the number of loops, the 2^{nd}-homology counts the number of 2-dimensional voids, and so on. The homology groups of dimension i are denoted by H_i.

Given a point cloud sampled from a manifold, we want to recapture the homological features of the original manifold from these discrete points. The idea is to construct a sequence of topological spaces along some *timeline* and track the evolution of the topological features across time. The longer the features *persist* (and hence the name *persistent homology*), the more likely they are the actual features of the original manifold. Given a point cloud X and a dimension i, the persistence diagram $D_i(X)$ is the set of points $(b, d) \in \mathbb{R}^2$ corresponding to the birth and death time of these features in the aforementioned timeline.

For two point clouds, and also for their two persistence diagrams, there are several notions of distances between them, representing how *similar* they are as topological spaces. The most commonly used distance in practice is the **Bottleneck distance**.

Definition 5 (Bottleneck distance) *Let X and Y be two persistence diagrams. The Bottleneck distance $W_\infty(X, Y)$ is given by*

$$W_\infty(X, Y) = \inf_{\varphi: X \to Y} \sup_{x \in X} \|x - \varphi(x)\|_\infty,$$

where the infimum is taken over all matchings $\varphi : X \to Y$ (which allows matchings to points with equal birth and death time).

For simplicity, we shorthand the Bottleneck distance between persistence diagrams of X and Y in dimension i to $H_i(X, Y)$ instead of $W_\infty(D_i(X), D_i(Y))$. As the notation takes in 2 parameters in X and Y, this is not to be confused with the homology group of the specified spaces. Note that two point clouds with small Bottleneck distances can be considered topologically similar.

Notably, the bottleneck distance is highly correlated to the Gromov-Hausdorff distance (Section 4), as the bottleneck distance of two persistence diagrams of the same dimension is bounded above by their Gromov-Hausdorff distance (Chazal et al., 2009):

$$W_\infty(D_i(X), D_i(Y)) \leq d_{GH}(X, Y),$$

for every dimension i.

Monte-Carlo simulations. The Bottleneck distance is much more calculable than the Gromov-Hausdorff distance, and there are available software packages depending on the use

cases. As such, we run Monte-Carlo simulations to compute the Bottleneck distances on 5 synthetic data sets, 3 real-world tabular data sets, and 2 real-world image data sets to confirm our hypothesis that the orthogonal perturbation preserves the topology better than the projection perturbation *on average*. The synthetic data sets are some noisy point clouds of certain 2-dimensional shapes in 3-dimensional space. The tabular data sets are the COMPAS (Jeff Larson and Angwin, 2016), German Credit (Hofmann, 1994), and Communities and Crime (Redmond, 2011). The image data sets are MNIST (LeCun and Cortes, 2010) and Fashion-MNIST (Xiao et al., 2017). Table 1 and 2 provide more details about those data sets. We use the Ripser Python library (Tralie et al., 2018) to compute the Bottleneck distances in our experiments. All reported Bottleneck distances are normalized with the noise added to the point clouds for more intuitive visualization.

Table 3 reports the means H_0 and H_1 Bottleneck distances of the perturbations on the synthetic data sets. The number of data points and the noise level are chosen mainly for nice visualizations (shown in Appendix A). The results show that orthogonal perturbation consistently results in lower H_0 distances for line-shaped data and lower H_1 distances for cycle-shaped data. Note that in general, H_1 is the better topological indicator for cycle-shaped data sets compared to H_0, since cycles or holes (detected by H_1) are harder to replicate than connected components (detected by H_0).

For the real-world data set, we conduct the experiments with perturbations of different noise levels and report results in Fig. 6 and 7. It can be observed that both H_0 and H_1 Bottleneck distances of the persistence diagrams of the orthogonal perturbation are significantly smaller than those of the projection perturbation on all experiments.

6. EMaP Algorithm

We now discuss our perturbation-generating algorithm, called EMaP, leveraging the topological information of the input data. To generate better perturbations for the explaining task, EMaP not only exploits the perturbation direction but also uses low-dimensional distances to better capture the notion of locality.

In particular, predictions on the perturbations do not necessarily hold local information about the explained inputs, which is one main reason for the usage of some distance or kernel functions measuring how similar the perturbations are to the explained input. Note that those distances and kernels are normally functions of other distances, such as L_1 and L_2 in the input space $\mathbb{R}^N$, which might not correctly capture the notion of similarity. By operating in the low-dimensional manifold, EMaP can overcome those issues. First, if topological similarity implies similarity in the model's predictions, maintaining the topological structure of the original data should improve the relevance of the model's predictions on the perturbations. Therefore, explaining the model with orthogonal perturbations, which helps preserve the topology better, should be more beneficial. Furthermore, the manifold provides a natural way to improve the similarity measurement among data points. Fig. 8 shows the issue of similarity measurement based on Euclidean distance in the input space $\mathbb{R}^N$. As the distance ignores the layout of the data, further points on the manifold might result in the same similarity measure. On the other hand, the low-dimensional distances computed on the manifold take into account the layout and can overcome that issue.

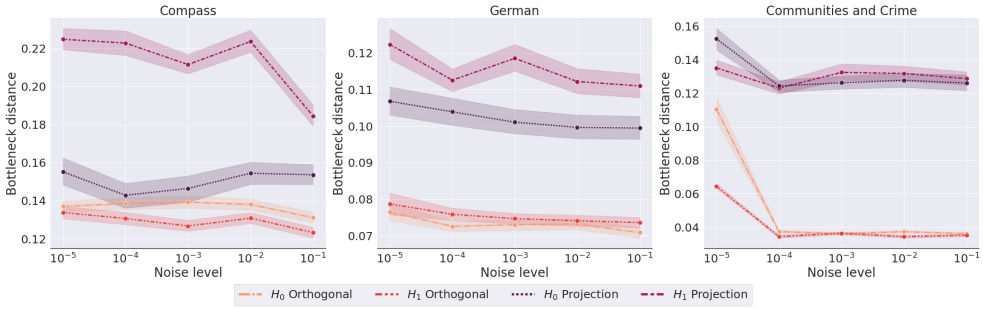


Figure 6: The normalized H_0 and H_1 Bottleneck distances for orthogonal and projection perturbations on 3 real-world data sets at different noise levels. The x-axis shows the average perturbations' radius applied to each data point (log-scale).

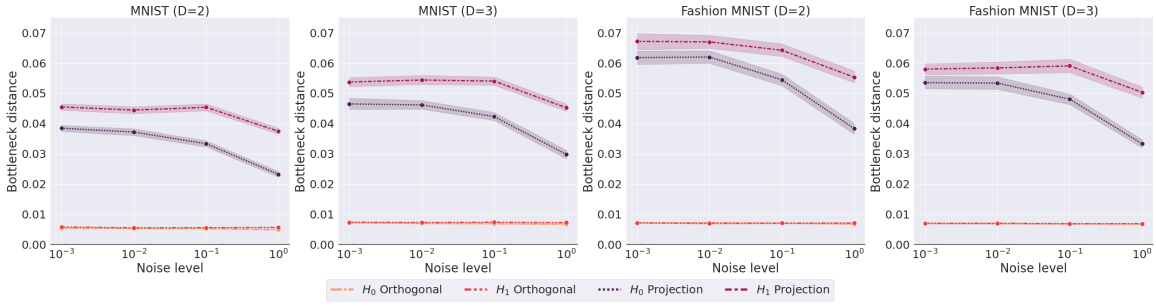


Figure 7: The normalized H_0 and H_1 Bottleneck distances for orthogonal and projection perturbations on 2 image data sets at different noise levels. The x-axis shows the average perturbations' radius applied on each data point (log-scale).

data set	Parameter	No. points	Data's noise	Perturbation	No. runs	EMaP's dim
Line	Length: 10	100	0.1	≈ 0.15	100	1
Circle	Radius: 1	400	0.1	≈ 0.1	100	2
2 intersecting circles	Radius: 1	400	0.01	≈ 0.1	100	2
2 concentric circles	Radius: 1	400	0.01	≈ 0.1	100	2
Spiral	Radius: [0,2]	1000	0.02	≈ 0.05	100	2

Table 1: The parameters of the Bottleneck distance's experiments on the synthetic data. The perturbation column shows the average magnitude of the perturbation on the data.

Experiment	No. data points	No. feats	Feature's values	No. runs	EMaP's dim
COMPAS	7214	100	{0,1}	100	2
German Credit	1000	28	{0,1}	100	2
CC	2215	100	{0,1}	100	2
MNIST	60000	28×28	[0,1]	100	2 and 3
Fashion-MNIST	60000	28×28	[0,1]	100	2 and 3

Table 2: The parameters of the Bottleneck distance's experiments on the real-world data.

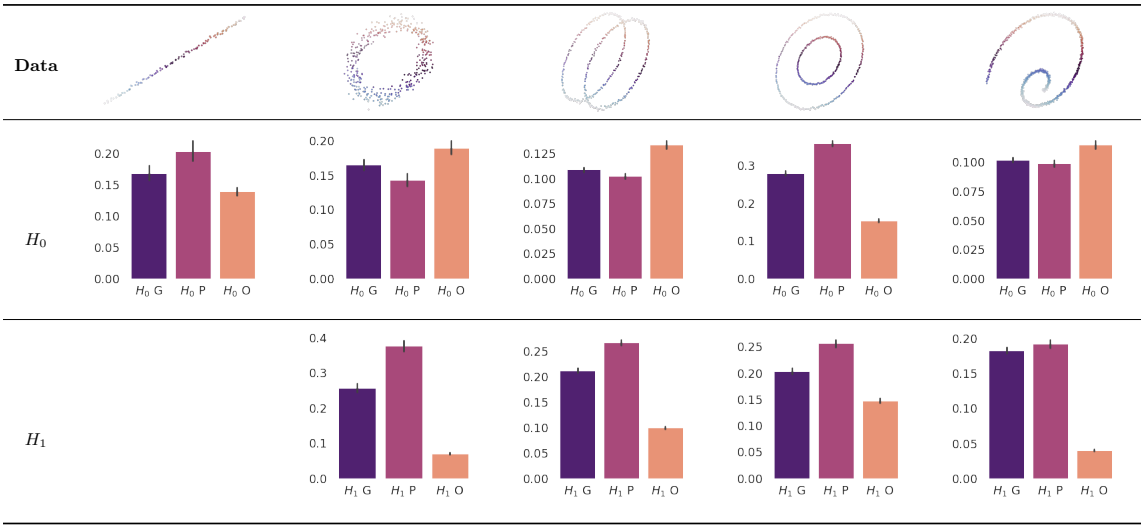


Table 3: The normalized H_0 and H_1 Bottleneck distances for Gaussian (G), projection (P), and orthogonal (O) perturbations on synthetic data. Visualizations of the actual perturbations are provided in Appendix A.

Algorithm overview. Given an input to be explained, the outputs of EMaP are the perturbations along the manifold's orthogonal directions X_r and their low-dimensional distances D_r to that input. The pseudo-code of EMaP is shown in Alg. 1. The first step is to learn an embedding function, i.e., a *mapper*, transforming the data to a low dimension (line 2). Then, p samples from each label are selected and combined with the

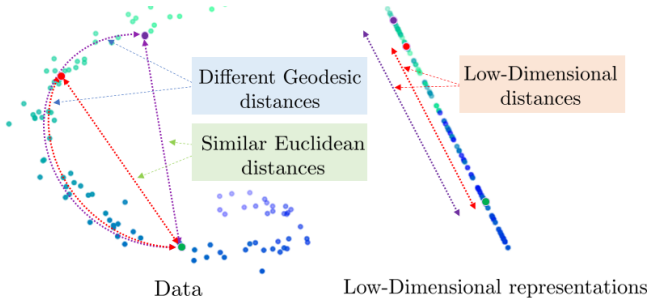


Figure 8: Euclidean distances computing in the input space might not capture the actual distances between the data points (left). Distances in low-dimensional space can help with the issue (right).

explained input x_0 into a set, called the *pivots* (lines 3 to 7). After that, EMaP generates perturbations along the orthogonal directions of the manifold from each pivot (line 10). Pivots are used to provide the explainer with a broader range of perturbations, leading to improved performance. Specifically, since the manifold learned by UMAP is optimized for capturing global structure, orthogonal directions computed directly on this manifold can be highly inaccurate. By introducing pivots, EMaP can learn multiple local subspaces, enabling the generation of more accurate orthogonal perturbations. In the next paragraphs, we will describe those key steps in more detail.

Algorithm 1 EMaP

Input: Data to explain x_0, a subset of training data (X, y), number of pivots per labels p, number of perturbations per pivot k, lower dimension V and noise level r.

Output: X_r and D_r. X_r contains $k(pl + 1)$ orthogonal perturbations locally around x_0 and points in X. D_r contains the low-dimensional distances of points in X_r to x_0 (l is the number of unique labels in y).

1: Initialized an EMaP sampler object $\mathcal{M}$.
2: $\mathcal{M}$.mapper $\leftarrow$ Mapper to the manifold of dimension V of X
3: $\mathcal{M}$.pivots $\leftarrow \emptyset$
4: Include x_0 to $\mathcal{M}$.pivots
5: **for** each class l in y **do**
6: Include p samples of class l to $\mathcal{M}$.pivots
7: **end for**
8: $X_r \leftarrow \emptyset$, $D_r \leftarrow \emptyset$
9: **for** each data point x in $\mathcal{M}$.pivots **do**
10: $\tilde{X}, \tilde{D} \leftarrow \mathcal{M}$.gen_perturbation$(x, k, r)$
11: Include $\tilde{X}$ to X_r and include $\tilde{D}$ to D_r
12: **end for**
13: **return** X_r and D_r.

The mapper. In our implementation of EMaP, the mapper is learned from a manifold approximated by UMAP (McInnes et al., 2018). Since the manifold learned by UMAP is optimized for global information, the orthogonal directions computed on top of that manifold at local data points are prone to high error. This can degrade the correctness of orthogonal perturbations significantly. To overcome this issue, EMaP learns a local subspace for each pivot and generates the orthogonal perturbations on top of that subspace. Intuitively, the local subspace is a local affine approximation of the manifold. Therefore, the resulting orthogonal directions are more finely tuned for the local points. We denote G_x as the $N \times V$ matrix characterizing the local subspace at x. Since G_x is a linear approximation of data points near x, by denoting $\omega : \mathbb{R}^N \to \mathbb{R}^V$ as the function embedding input data to the manifold, we have $z^{\text{low}} := \omega(z) \approx G_x^\top z$ and $z \approx G_x\ z^{\text{low}}$, where $z \in \mathbb{R}^N$ are points near x and z^{low} are their embedding in $\mathbb{R}^V$. In our current implementations, the set of pivots contains the explained data point and p data points sampled from each class label l (see Algorithm 1).

EMaP orthogonal perturbations. The key step of EMaP is the generation of orthogonal perturbations $\tilde{x}$ from x (line 10, Alg. 1), which can be expressed as:

$$\tilde{x} = x + \text{noise} - \text{Proj}_{G_x}(\text{noise}), \tag{5}$$

where the noise is sampled from a multivariate normal distribution and $\text{Proj}_{G_x}(\text{noise})$ is the projection of the noise on the local subspace characterized by G_x. Upon obtaining the orthogonal perturbations, we can compute their low-dimensional embedding using the mapper transform function ω. The pseudo-code for this computation is in Algorithm 2.

Algorithm 2 gen_perturbation

Input: Input x, number of perturbation k and noise level r.
Output: k orthogonal perturbations of x and their low-dimension distances to x.

1: $G_x \leftarrow$ self.get_local_subspace(x)
2: $\tilde{X} \leftarrow \emptyset$
3: **for** $1 \le i \le k$: **do**
4: noise $\leftarrow \mathcal{N}(0, \Sigma_r)$
5: $\tilde{x} \leftarrow x + \text{noise} - \text{Proj}_{G_x}(\text{noise})$
6: Include $\tilde{x}$ into $\tilde{X}$
7: **end for**
8: $\tilde{X}^{\text{low}} \leftarrow$ self.mapper.transform($\tilde{X}$)
9: $x^{\text{low}} \leftarrow$ self.mapper.transform(x)
10: $\tilde{D} \leftarrow$ distances of each member in $\tilde{X}^{\text{low}}$ to x^{low}
11: **return** $\tilde{X}$ and $\tilde{D}$

Local-subspace approximation. The correctness of the orthogonal perturbations, i.e. whether the perturbations are actually lying in the orthogonal subspace, is heavily dependent on the correctness of the computation of G_x. We now discuss how EMaP learns the local-subspace matrix G_x.

Ideally, given a set of data Z near x in the manifold, we can solve the following optimization for G_x :

$$G_x = \arg\min_G \sum_{z \in Z} \|G\omega(z) - z\|, \tag{6}$$

where ω is the transform function embedding input data to the manifold learned in the previous step. An intuition is that, for all z in the manifold and near x, we are searching for a matrix G such that the inverse mapping $G\omega(z) \approx Gz^{\text{low}}$ approximately equals its original value z. Note that if the embedding ω is exactly on Z and the manifold is affine, the optimization (6) can achieve its optimal value 0 for some G. Since it is not trivial to obtain the set Z belonging to the manifold, EMaP perturbs around x with some random noise and solves the following optimization instead:

$$\hat{G}_x = \arg\min_G \sum_{r \in B} \|G\omega(x + r) - (x + r)\|, \tag{7}$$

where B is a ball centering at 0 with noise radius r. EMaP solves (7) for an approximation of the local subspace. Further details of this step are provided in Algorithm 3.

Algorithm 3 get_local_subspace

Input: Data point z.
Hyper-parameters: Number of training samples k_T and noise level for training r_T.
Output: Matrix G characterize the local-subspace at z

$\tilde{Z} \leftarrow \emptyset$
for $1 \leq i \leq k_T$: **do**
  noise $\leftarrow \mathcal{N}(0, \Sigma_{r_T})$
  $\tilde{z} \leftarrow z +$ noise
  Include $\tilde{z}$ into $\tilde{Z}$
end for
$\tilde{Z}^{\text{low}} \leftarrow$ self.mapper.transform($\tilde{Z}$)
$G \leftarrow \arg\min_W \|\tilde{Z} - W\tilde{Z}^{\text{low}}\|_2$
return G

We now discuss the gap between the ideal solution G_x and the approximation $\hat{G}_x$ used by EMaP, i.e., the solution of (6) and (7). This can be characterized by bounding the error between $\{G_x\omega(z)\}$ and $\{\hat{G}_x\omega(z)\}$, i.e. the reconstructed signals in $\mathbb{R}^N$ by using G_x and $\hat{G}_x$, respectively. Lemma 6 provides a bound on that reconstruction error where the set Z in (6) is the projection of a ball B on the data manifold. The bound holds under a mild assumption that the optimal objective of (6) is not larger than that of (7). We find this assumption reasonable and intuitive: as the set Z is in a subspace of dimension V and the set of $x + r$ is a ball in $\mathbb{R}^N$, finding a subspace of dimension V approximating the subspace containing Z should give a much lower error.

Lemma 6 *Assume that all data points x belong to the same affine space $\mathbb{R}^V$. Let Proj be the projection onto $\mathbb{R}^V$, then under the above assumption on the optimization (6) and (7),*

the reconstruction error on perturbed data points is upper bounded by:

$$\|\hat{G}_x\omega(x+r) - G_x\omega(\text{Proj}(x+r))\| \leq F_B(\omega) + \|r^\perp\|,$$

where $r^\perp$ *is the orthogonal components of* r *and* $F_B(\omega) := \min_G \sum_{r\in B} \|G\omega(x+r)-(x+r)\|$.

Proof For simplicity, we rewrite:

$$G_1 = \arg\min_G \sum_{r\in B} \|G\omega(x+r) - (x+r)\|,$$
$$G_2 = \arg\min_G \sum_{r\in B} \|G\omega(\text{Proj}(x+r)) - \text{Proj}(x+r)\|.$$

The assumption regarding the objectives (6) and (7) mentioned in the Lemma can be rewritten as:

$$\sum_{r\in B} \|G_2\omega(\text{Proj}(x+r)) - \text{Proj}(x+r)\| \leq \sum_{r\in B} \|G_1\omega(x+r) - (x+r)\|.$$

We find this assumption reasonable since its left-hand side is equal to 0 in the ideal scenario, i.e., $x \in \mathbb{R}^V$ for all x in our data set. With that, we have:

$$\begin{aligned}
&2\|G_1\omega(x+r) - (x+r)\| \\
\geq&\|G_1\omega(x+r) - (x+r)\| + \|G_2\omega(\text{Proj}(x+r)) - \text{Proj}(x+r)\| \\
\geq&\|G_1\omega(x+r) - G_2\omega(\text{Proj}(x+r)) - (x+r-\text{Proj}(x+r))\| \\
\geq&\|G_1\omega(x+r) - G_2\omega(\text{Proj}(x+r))|| - ||(x+r) - \text{Proj}(x+r))\| \\
=&\|G_1\omega(x+r) - G_2\omega(\text{Proj}(x+r))|| - \|r - \text{Proj}(r)\|,
\end{aligned}$$

where the last two inequalities are from the Triangle Inequality. The last equality is due to the fact that $(x+r) - \text{Proj}(x+r)$ is the orthogonal components of $x+r$ and that x has no orthogonal components. ■

Note that $F_B(\omega)$ is small if the manifold is affine in the neighborhood B of x and ω is good, i.e. $\hat{G}_x$ is a good estimator for G_x under the above assumption.

7. Experiments

We evaluate EMaP on two main objectives: the explainer's performance and the perturbations' robustness. Our experiments are conducted on 3 tabular data sets, 2 image data sets, and 4 text data sets of reviews in multiple domains. They are COMPAS (Jeff Larson and Angwin, 2016), Communities and Crime (Redmond, 2011), German Credit (Hofmann, 1994), MNIST (LeCun and Cortes, 2010), Fashion-MNIST (Xiao et al., 2017), and 4 reviews data sets in the Multi-Domain Sentiment (Blitzer et al., 2007).

7.1 Experimental settings

We first describe how the data sets and the testing models are processed in our experiments. Then, we provide some details of the implementation of EMaP along with the baselines.

Data sets and Models' hyperparameters. All reported results for real-world data sets include at least 2000 data points and 100 runs, except for those of German Credit where the data only consists of 1000 samples. The text data sets are tokenized into vectors of 1000 input features while the MNIST and Fashion-MNIST data sets are of dimension $1 \times 28 \times 28$ and $3 \times 32 \times 32$, respectively. The model's inputs of all experiments are normalized between 0 and 1. The experimental model for the text data set is the $L1$ logistic regression as used by LIME (Ribeiro et al., 2016). The models of testing for the two image data sets are 2-layer convolutional networks implemented in Pytorch (Paszke et al., 2019) with test set accuracy of 98% for MNIST and 93% for Fashion-MNIST.

Technical implementation of EMaP and baselines. Generally, EMaP perturbations can be used to leverage any model-agnostic perturbation-based methods; however, the modification may require significant changes to the existing implementations of the methods. In our experiments, we use EMaP to leverage LIME (Ribeiro et al., 2016) as a proof of work, i.e., we show that EMaP can improve LIME in terms of performance. We chose LIME to demonstrate the usage of EMaP since it requires a few modifications to integrate EMaP's perturbations into LIME. This helps demonstrate fairly the gain of applying EMaP on explanation methods. We use the notation EMaP-LIME (or EMaP for short) to indicate *LIME with EMaP's perturbations* in our following experimental results.

To apply EMaP into LIME, we simply choose the loss function (defined in Eq.(1)) as:

$$\mathcal{L}(f, g, \pi_x) = \sum_{\tilde{x} \in X_r} \pi_x(\tilde{x})(f(\tilde{x}) - g(\tilde{x})),$$

where $\pi_x(\tilde{x}) = \exp(-\tilde{D}(x, \tilde{x})^2/\sigma^2)$ with the distance $\tilde{D}$ computed as in Algorithm 2 and g is the linear function of the features of the perturbation.

Besides LIME, we also heuristically compare EMaP-LIME to some other black-box and white-box explanation methods. The following are brief descriptions of those methods:

- LIME zero: LIME with perturbations whose perturbed features are set to zero. This method is used by LIME in explaining text data.
- LIME+: LIME with perturbations whose perturbed features are added with Gaussian noise. This method is used by LIME in explaining image data.
- LIME*: LIME with perturbed whose perturbed features that are multiplied with uniform noise between 0 and 1. This can be considered a smoother version of LIME zero. We include this variant of LIME because it provides nice explanations in some specific experiments (see Appendix A for some examples).
- KernelSHAP: a black-box method based on Shapley value (Lundberg and Lee, 2017), whose perturbed features are set to the average of some background data.
- GradientSHAP: a white-box method based on Shapley value (Lundberg and Lee, 2017), which relies on the gradient of the model.

- DeepLIFT: a white-box method based on back-propagating the model (Shrikumar et al., 2017).
- Parzen (Baehrens et al., 2010): an explanation method that approximates the model globally and uses the gradients of that approximating model to explain the predictions.
- Greedy: the features contributing the most to the predicted class are removed and selected until the prediction changes. The implementation is taken from (Ribeiro et al., 2016).

The noise vector used to generate perturbations for all applicable explainers has a radius 10^{-3} for text data and 10^{-4} for image data, which are in the range analyzed in the previous Bottleneck distance experiments in Fig. 6 and 7. The noise radius r_T used to approximate the local subspaces of EMaP (Algorithm 3) is chosen equal to the noise radius r for perturbation. For UMAP's hyper-parameters, we use n_components $\in \{2, 3\}$ and min_dist $= 0.1$ (default value). Our source code is attached in the supplementary material of this submission. Finally, for a fair comparison, the number of perturbations used to generate the explanation of any reported methods is 1000.

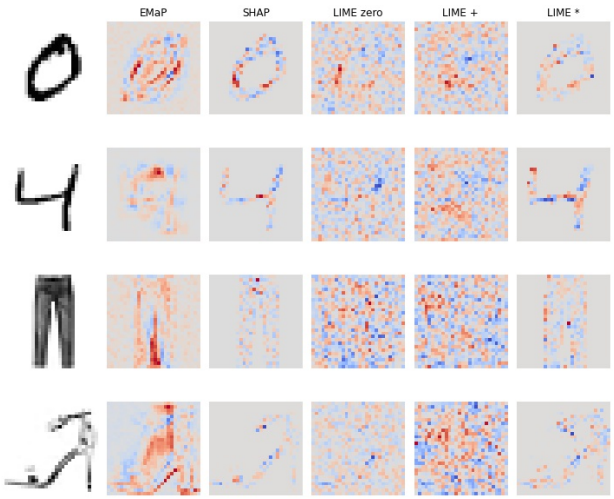


Figure 9: Examples of explanations for different methods on MNIST and Fashion-MNIST: altering white pixels to dark or vice versa in the most red (or blue) regions would weaken (or reinforce) the original prediction.

7.2 Explaining performance of explanation methods

For illustrative purposes, Fig. 9 shows the resulting explanations on MNIST and Fashion-MNIST generated by EMaP and some other black-box methods (More examples are in Appendix A). All methods use 1000 perturbations to generate the corresponding explanations. We can see that, while EMaP and SHAP capture relevant features, LIME includes a lot of noisy features in the background. Note that the reason LIME* does not include

the background is that the background of the original image has a value of 0 and, through multiplication, the features are not perturbed. This makes their weights 0 in the explaining model g. Regarding the low quality of LIME zero and LIME+, the main reason is that the method would require a much larger number of perturbations to generate good results at the pixel level. We want to point out that the only difference between EMaP and LIME is in the perturbations and how to compute the distances between the perturbations and the original data points. This clearly demonstrates that optimizing the perturbations can greatly improve the explanation's quality. Another interesting observation is the red areas on top of the digit 4 and at the bottom of the *Trouser* (the second and third rows of Fig. 9) only identified by EMaP. In fact, those areas are essential to the predictions since they differentiate the samples from other classes, i.e., digit 9 and the *Dress*, respectively. Those areas, on the other hand, are neglected by all other examples.

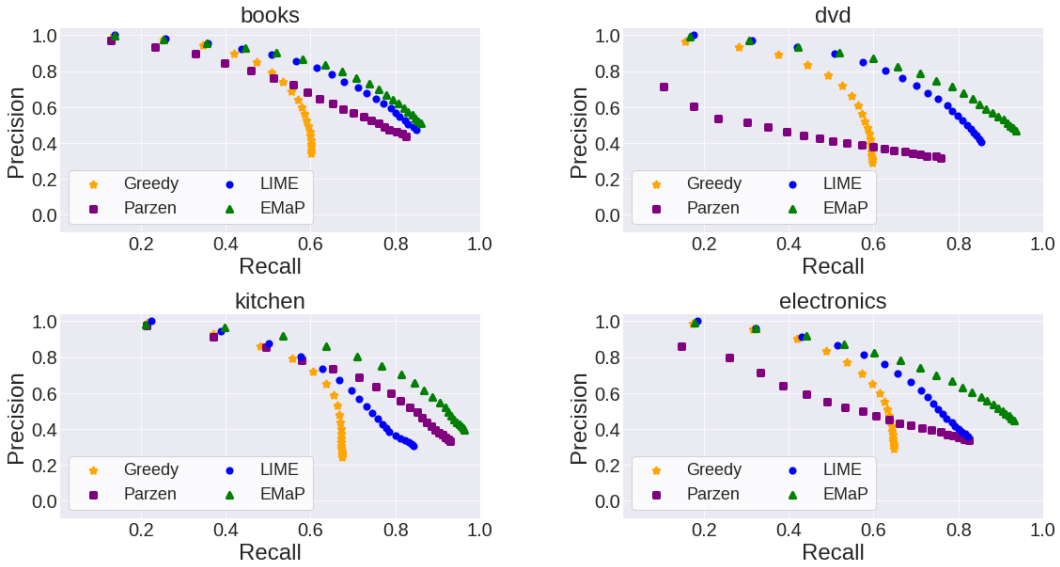


Figure 10: The precision and recall of explanations returned by Greedy, LIME, Parzen, and LIME-EMaP (the higher the better). The dots are in the increasing order of the number of features in explanations (left to right).

Text data sets. We report our experimental results for the sentiment classification task in the Multi-Domain Sentiment data set (Blitzer et al., 2007). We follow the experimental setup in (Ribeiro et al., 2016), in which the ground-truth explanatory features of the $L1$ logistic regression model are known. In particular, a regression model is trained for each data set, and the ground-truth explanatory features are non-zero coefficients. For each explanatory budget, i.e., the number of features included in the explanations, the fraction of these ground-truth features recovered by the explanations is computed. The performance of an explainer is evaluated by the precision and the recall rate of the features in the explanations. Intuitively, the more features included in the explanation, the higher the recall and the lower the precision.

Fig. 10 shows the scatter plot of precision vs. recall of LIME and LIME with EMaP on 4 review data sets of *books*, *dvds*, *kitchen*, and *electronics*. Similar to how LIME is evaluated in these data sets in its original paper, we compare LIME with EMAP to LIME, Greedy, and Parzen explanation methods (Baehrens et al., 2010). In Greedy, the features contributing the most to the predicted class are removed until the prediction changes. On the other hand, Parzen approximates the model globally with Parzen windows, and the explanation is the gradient of the prediction. The results clearly show that EMaP consistently improves the faithfulness of LIME.

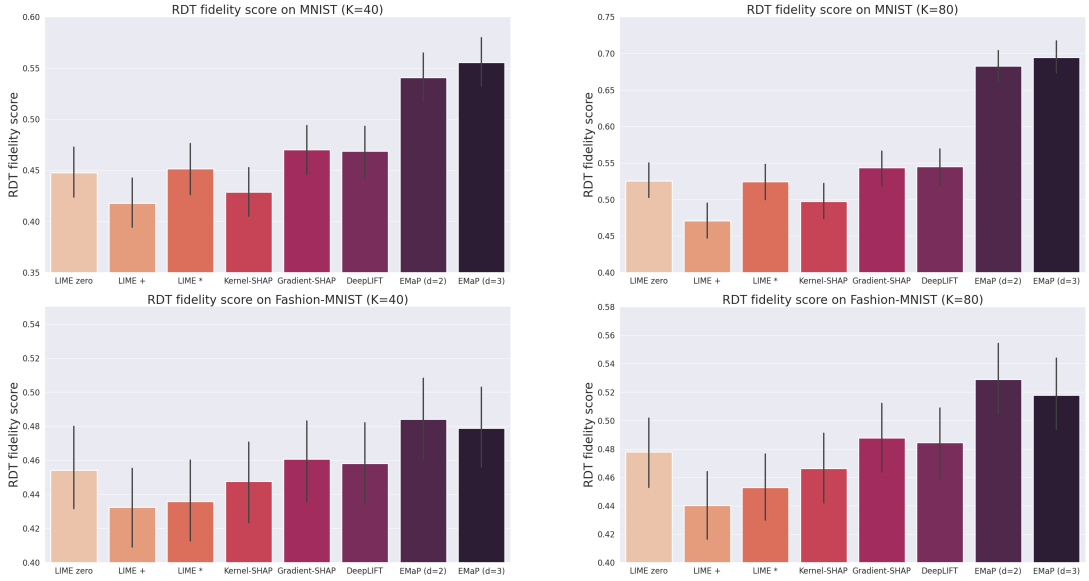


Figure 11: RDT fidelity scores of different perturbation-based methods on MNIST and Fashion-MNIST (the higher the better).

Image data sets. Since there are no ground-truth explanations for the MNIST and Fashion-MNIST image data sets, we evaluate explanations using the RDT-fidelity Funke et al. (2023), the infidelity scores (Yeh et al., 2019), and the log-odds scores (Shrikumar et al., 2017). The RDT-fidelity of an explanation S with respect to a model f is the probability that the model's prediction does not change if the features in the explanation remain unchanged:

$$RDT(S) = \mathbb{E}\left[\mathbf{1}\left(f(x) = f(x_S)\right)\right],$$

where x_S is x with noise added on features not in S, and the expectation is taken over the noise. On the other hand, the infidelity score measures the expected error between the explanation multiplied by a meaningful perturbation and the differences between the predictions at its input and at the perturbation. The metric can be considered as a generalized notion of Sensitivity-n (Ancona et al., 2018). Intuitively, explanations with lower infidelity are more desirable. Finally, given the importance weights on features (i.e., the explanation), the log-odds score measures the difference between the image and the modified image whose

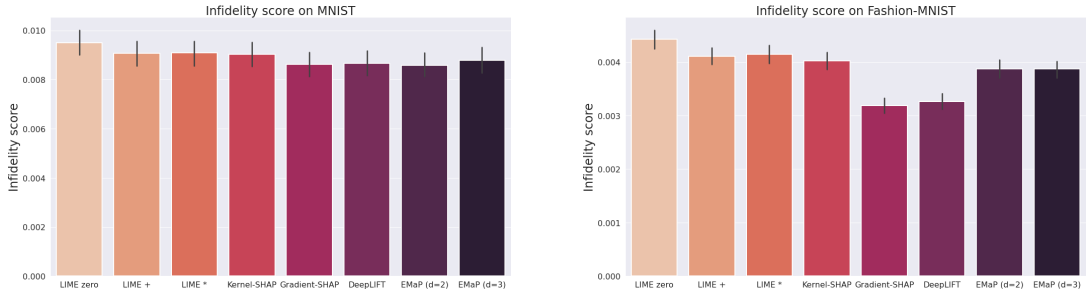


Figure 12: Infidelity scores of different perturbation-based methods on MNIST and Fashion-MNIST (the lower the better).

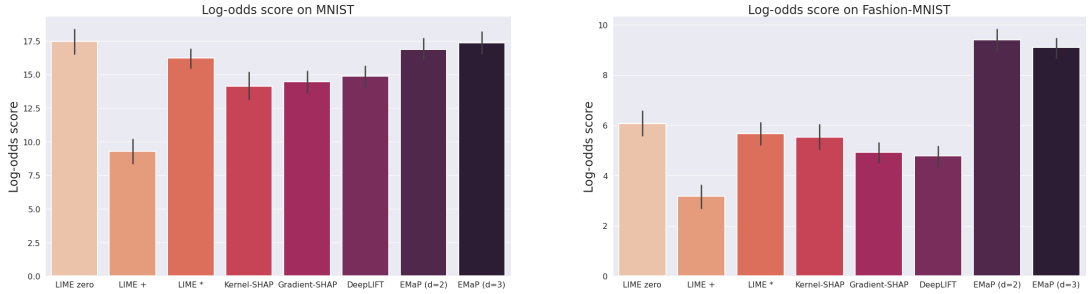


Figure 13: Log-odds scores of different perturbation-based methods on MNIST and Fashion-MNIST (the higher the better).

pixels are erased based on their importance weights. Intuitively, the higher the log-odds score, the better the explanation.

The evaluations using the RDT-fidelity are computed on explanations of $K = 40$ and 80 features of the explained image (Fig. 11). The results clearly demonstrate the high fidelity of our EMaP in both image data sets. Fig. 12 shows the infidelity scores. It is clear that EMaP has the lowest infidelity score among all black-box methods. Even though the white box methods, KernelSHAP and DeepLIFT, have more information on the explained models than EMaP, they can only outperform EMaP in Fashion-MNIST. Additionally, Fig. 13 shows the log-odds scores of EMaP with low-dimension $d = 2$ and $d = 3$, along with other explanation methods and other perturbation schemes. In these experiments, the scores are computed on 20% erased pixels. In MNIST, we can see that EMaP does not degrade the explainer performance compared to LIME in terms of log-odds (note that the default setting for LIME for image data is LIME+). For Fashion-MNIST, EMaP improves the log-odds significantly. More examples of EMaP and other explanation methods are provided in Appendix A.

7.3 Perturbation's robustness

We evaluate the robustness of the perturbation scheme based on the discriminator's performance in differentiating perturbations from the original data. Following the setup in (Slack

et al., 2020), the discriminator is trained with full knowledge of the explainer's parameters. This discriminator has been shown to be able to recognize perturbations of LIME and SHAP explainers. Note that, as DeepLIFT and GradientSHAP are white-box methods and do not use perturbations, we only evaluate the robustness of LIME's perturbations and KernelSHAP's perturbations. Since there is no ambiguity, we use SHAP to denote KernelSHAP in the following results.

Our experimental results show that EMaP's perturbation is more robust to the discriminator. Specifically, Figs. 14, 15, 16, and 17 show the True-Positive (TP) and True-Negative (TN) rates of discriminators on perturbations of 2 image data sets and 2 tabular data sets. For image data sets, the discriminators can easily recognize perturbations generated by LIME and SHAP. On the other hand, EMaP perturbations significantly lower the success rates of discriminators in recognizing the perturbations. While the explainers' perturbation schemes prove to be slightly more robust in the tabular data sets compared to the image data sets, EMaP still improves the perturbations' robustness remarkably.

7.4 Computational resource and run time

Our experiments are conducted on a single GPU-assisted compute node that is installed with a Linux 64-bit operating system. The allocated resources include 32 CPU cores (AMD EPYC 7742 model) with 2 threads per core and 100GB of RAM. The node is also equipped with 8 GPUs (NVIDIA DGX A100 SuperPod model), with 80GB of memory per GPU.

	EMaP Initialization	LIME	EMaP (d=2)	EMaP (d=3)
MNIST	240-260	0.763	1.311	1.493
Fashion-MNIST	240-260	0.726	1.502	1.467

Table 4: Run time (in seconds) of EMaP and LIME in generating explanations using 1000 perturbations per explanation. The reported numbers are *seconds* (for initialization column) and *seconds per explanation* (for other columns).

The run time of EMaP is mostly dictated by the learning of the embedding function (line 2 of Algorithm 1). That initialization step in the tabular data set for about 2000 data points takes less than 2 minutes. It takes between 240 and 260 seconds for all images of 60000 MNIST/Fashion-MNIST images. Note that the manifold and local subspaces can be computed before deployment since it does not depend on the explained inputs. Thus, this overhead of EMaP can be mitigated at deployment. Table 4 reports the actual run time of EMaP and LIME in the image data sets.

8. Conclusion, limitations and future research

From our theoretical and experimental results, we exploit the data manifold to preserve the topology information of its perturbation. We implement the EMaP to realize the idea and demonstrate its benefits in the explanation task. We recognize the main limitation of EMaP is in its requirement for low-dimensional representations of the data and the local affine subspaces. For more complex data, computing it correctly can be very challenging.

There are several interesting open questions of EMaP that we leave for our future work. For instance, it is important to study the impact of the underlying manifold-learning algorithm, i.e., the UMAP, on the perturbations and the explanations. It is also interesting to examine the behavior of EMaP in a wider range of explainers and applications.

Acknowledgement

This work is partially supported by the National Science Foundation under Grant No. FAI-1939725 and SCH-2123809.

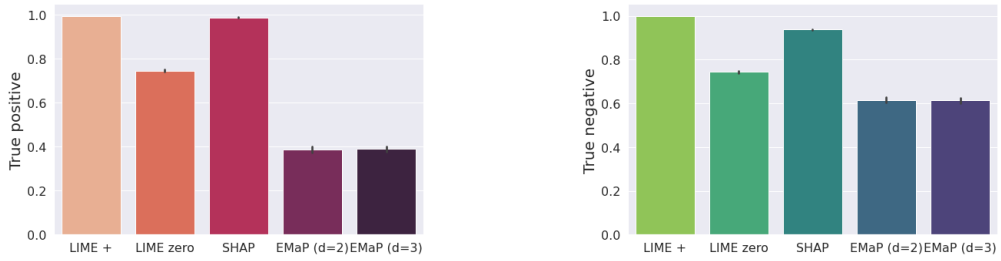


Figure 14: True-Positive and True-Negative rates of the discriminators on perturbations of different methods on the MNIST data set (the lower the better).

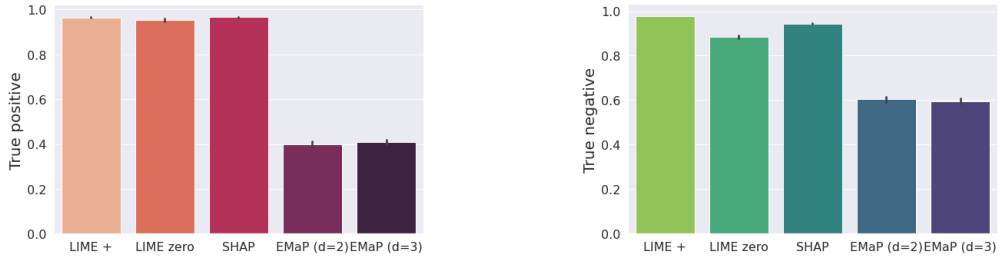


Figure 15: True-Positive and True-Negative rates of the discriminators on perturbations of different methods on the Fashion-MNIST data set (the lower the better).

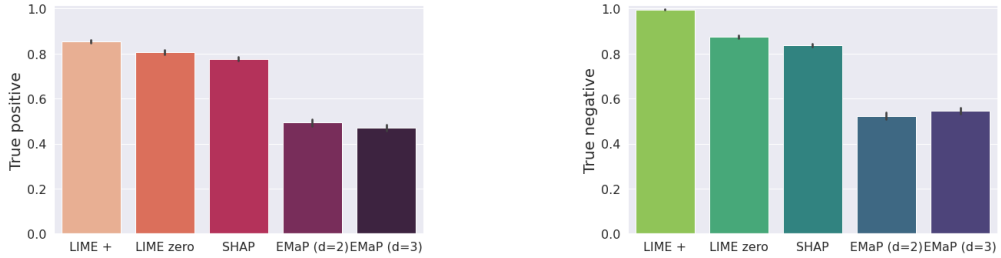


Figure 16: True-Positive and True-Negative rates of the discriminators on perturbations of different methods on Communities and Crime data set (the lower the better).

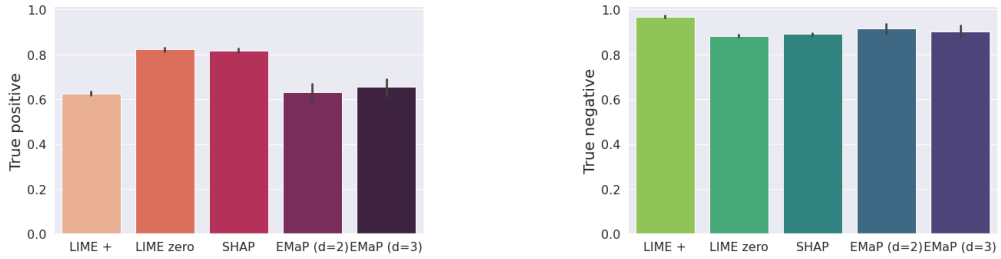


Figure 17: True-Positive and True-Negative rates of the discriminators on perturbations of different methods on the German Credit data set (the lower the better).

Appendix A. Visualizations of synthetic data and explanations generated with EMaP

This Appendix provides some visualizations of our synthetic data (in experiments of Table 3) and explanations generated with or without EMaP. The explanations of EMaP shown in this Appendix are those that appeared in the experiments reported in Section 7 of the main manuscript.

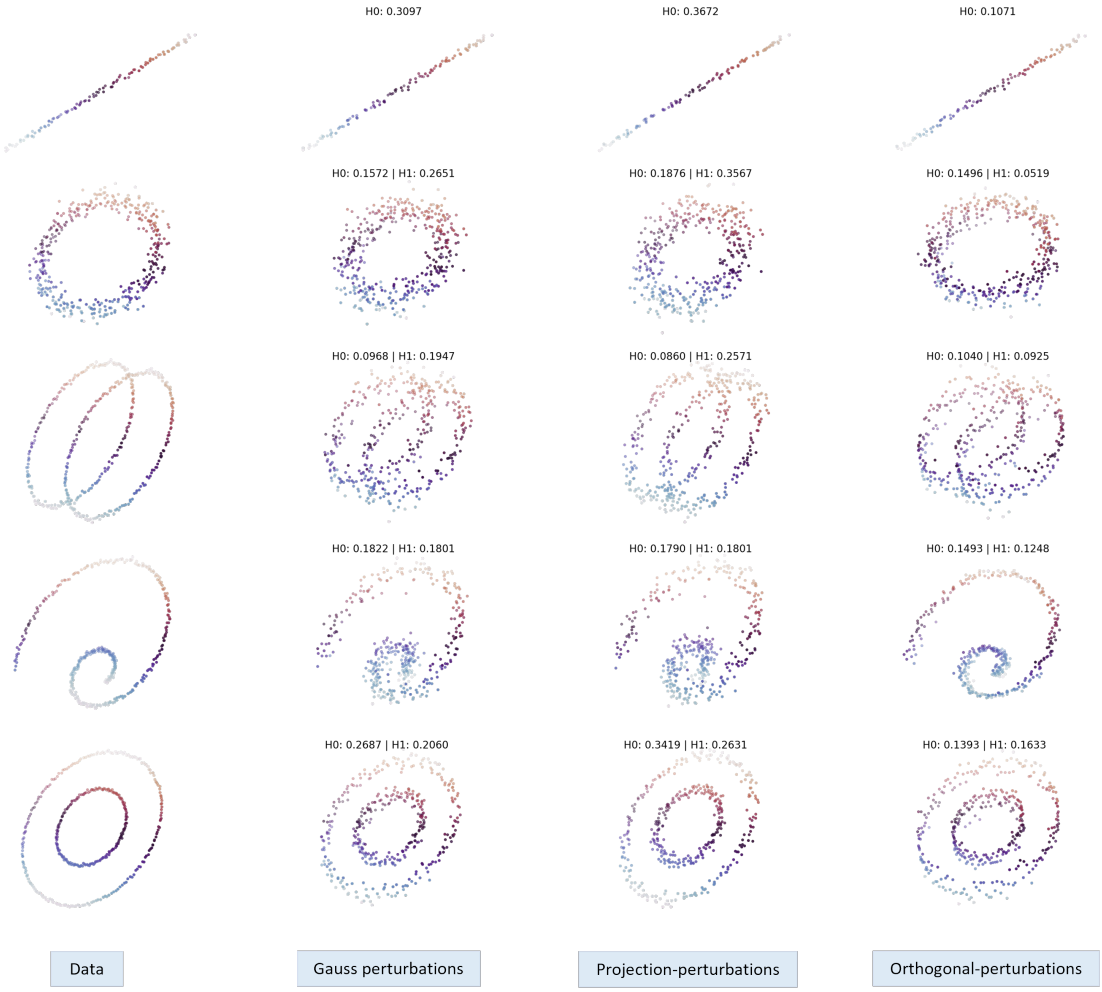


Figure 18: The visualization of some synthetic data in experiments of Table 3.

In Fig. 18, we visualize the synthetic data of different shapes and their perturbations in three dimensions. We also report H_0 and H_1 Bottleneck distances between the perturbations and the original data in that figure.

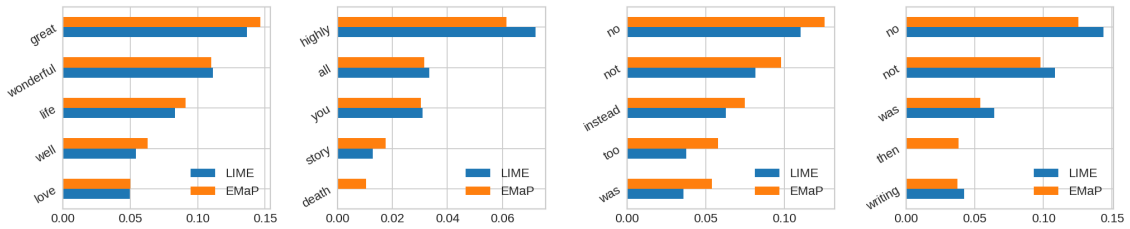


Figure 19: Visualization of EMaP-LIME explanations of the Multi-polarity-books review data sets (Blitzer et al., 2007).

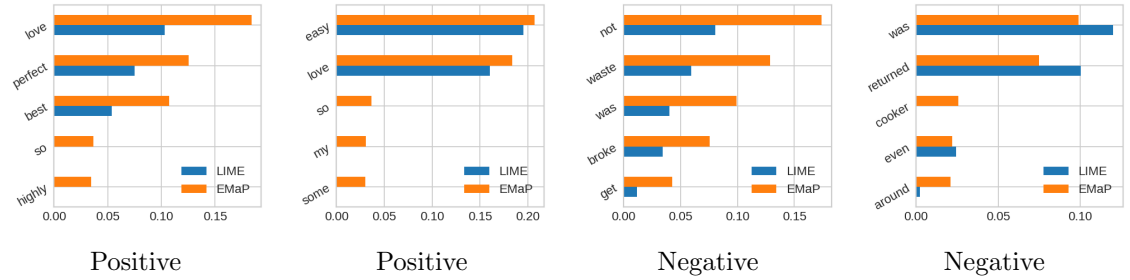


Figure 20: Visualization of EMaP-LIME explanations of the Multi-polarity-kitchen review data sets (Blitzer et al., 2007).

Fig. 19 and 20 compare the actual explanations returned by LIME and EMaP-LIME (EMaP) in the books' reviews in the Multi-Domain Sentiment data sets. We can see that the weights of features included in the explanations are quite similar between the two methods.

Fig 21 shows the explanations of LIME with EMaP for all classes in the MNIST data set. The red (blue) areas mean, if the features in those areas are unchanged (changed), the prediction for that class will be stronger (weaker).

Finally, Fig. 22 and 23 provide some explanations of EMaP along with explanations of other methods in MNIST and Fashion-MNIST, respectively. The color code also has a similar meaning to that of Fig 21.

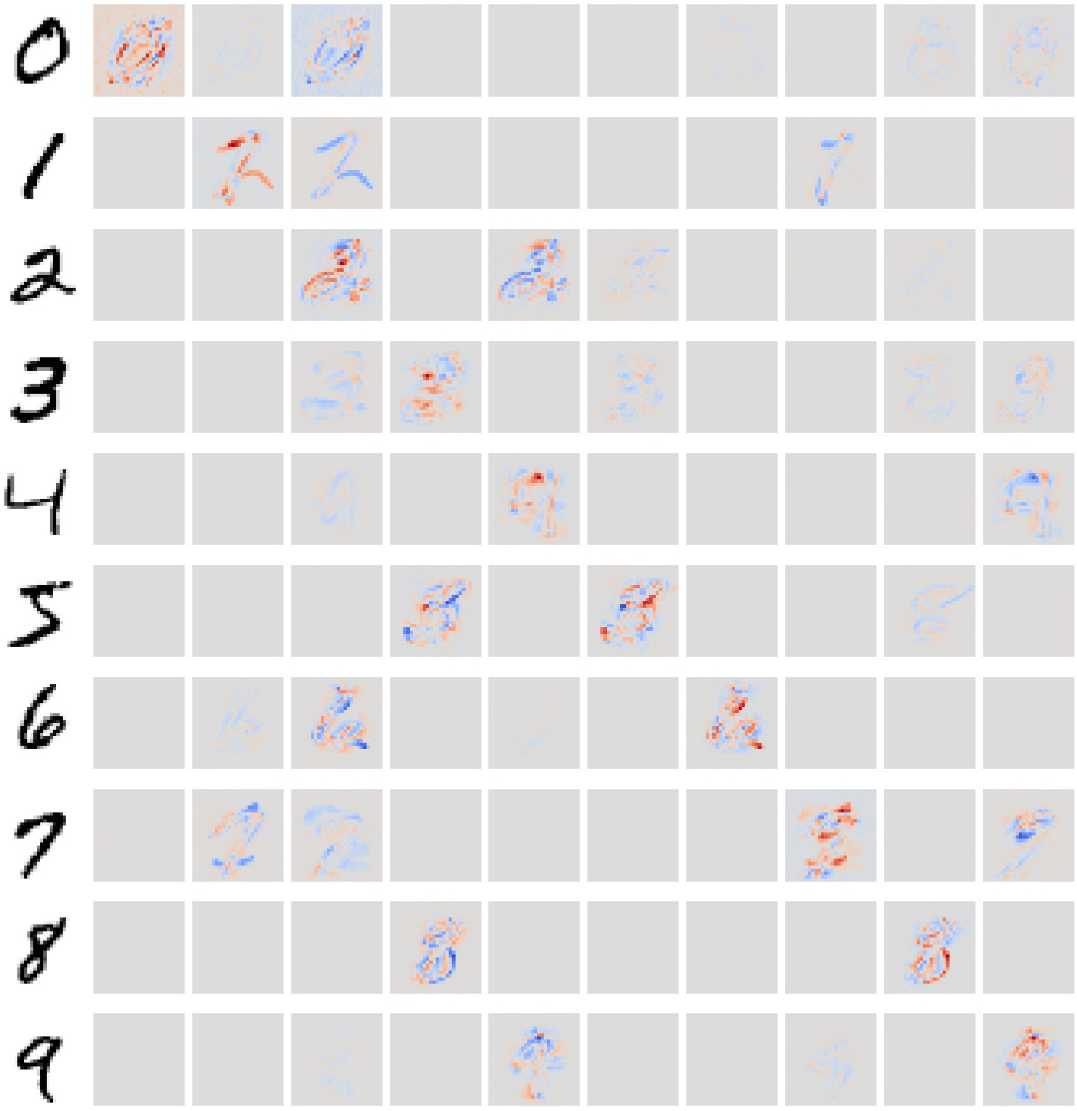


Figure 21: Visualization of EMaP-LIME explanations of the MNIST data set. The column indicates the class label to be explained.

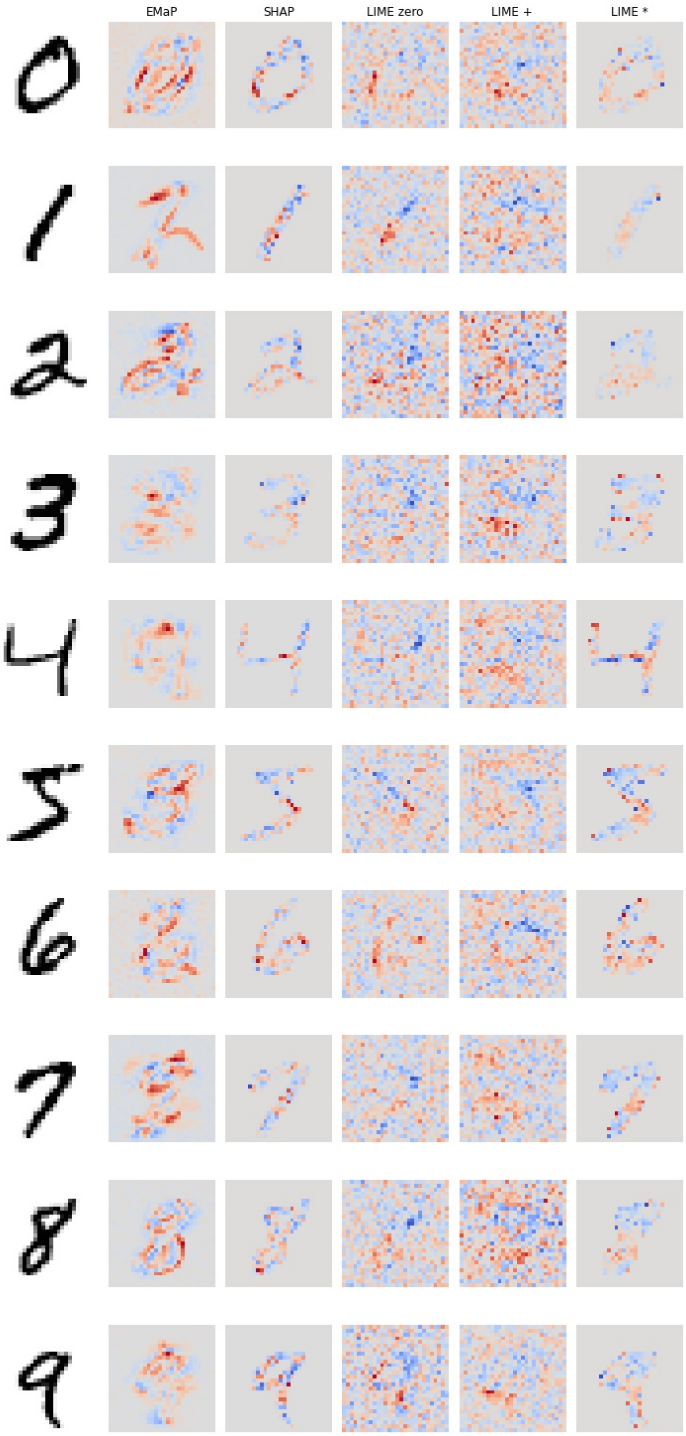


Figure 22: Examples of explanations in MNIST. Modifying the red-est (blue-est) area would negate (strengthen) the original prediction.

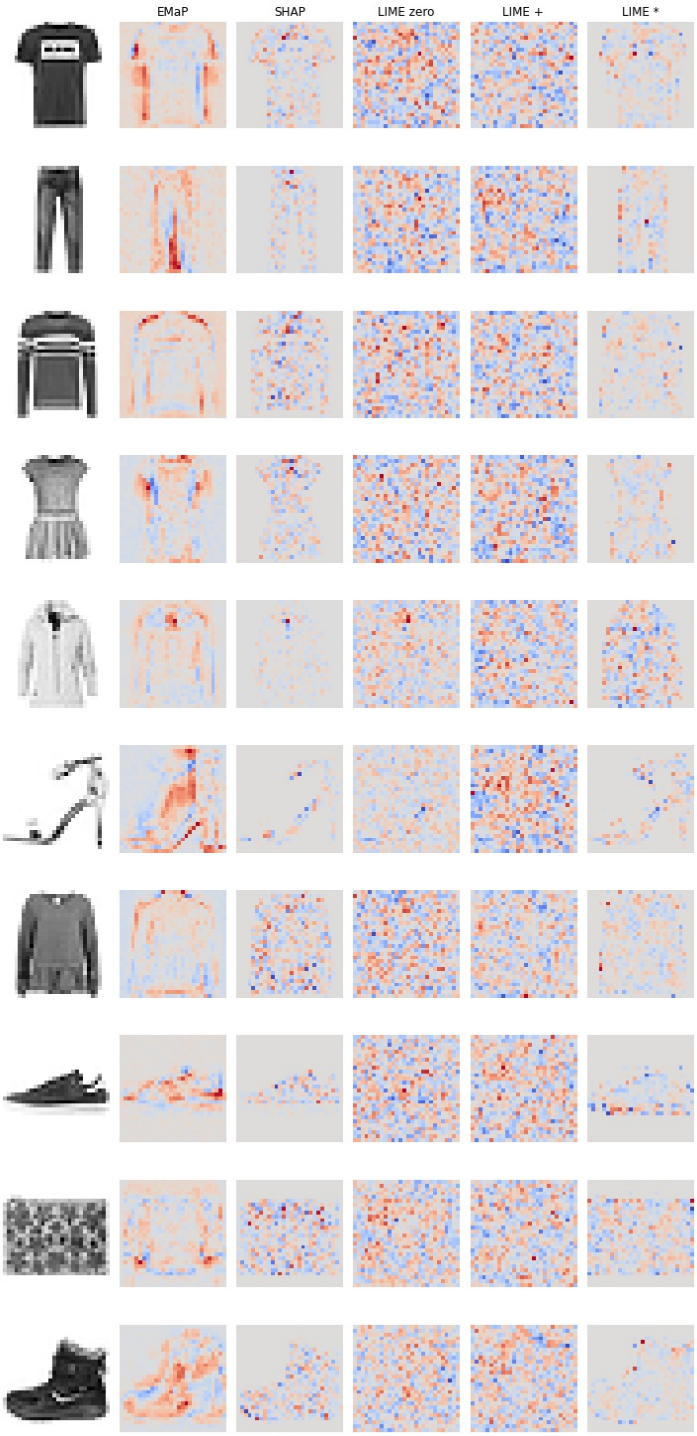


Figure 23: Examples of explanations in Fashion-MNIST. Modifying the red-est (blue-est) area would negate (strengthen) the original prediction.

References

M. Ancona, E. Ceolini, C. Öztireli, and M. Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sy21R9JAW.

D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11 (61):1803–1831, 2010. URL http://jmlr.org/papers/v11/baehrens10a.html.

J. Blitzer, M. Dredze, and F. Pereira. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/P07-1056.

J. Chakraborty, K. Peng, and T. Menzies. Making fair ML software using trustworthy explanation. *CoRR*, abs/2007.02893, 2020. URL https://arxiv.org/abs/2007.02893.

C.-H. Chang, E. Creager, A. Goldenberg, , and D. Duvenaud. Explaining image classifiers by adaptive dropout and generative in-filling. *arXiv preprint arXiv:1807.08024*, 2018.

F. Chazal, D. Cohen-Steiner, L. Guibas, F. Mémoli, and S. Oudot. Gromov-hausdorff stable signatures for shapes using persistence. *Comput. Graph. Forum*, 28:1393–1403, 07 2009. doi: 10.1111/j.1467-8659.2009.01516.x.

I. C. Covert, S. Lundberg, and S.-I. Lee. Explaining by removing: A unified framework for model explanation. *J. Mach. Learn. Res.*, 22(1), jan 2021. ISSN 1532-4435.

B. Dimanov, U. Bhatt, M. Jamnik, and A. Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In *SafeAI@AAAI*, 2020.

A.-K. Dombrowski, M. Alber, C. J. Anders, M. Ackermann, K.-R. Müller, and P. Kessel. Explanations can be manipulated and geometry is to blame. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. B. Fox, and R. Garnett, editors, *NeurIPS*, pages 13567–13578, 2019. URL http://dblp.uni-trier.de/db/conf/nips/nips2019.html#DombrowskiAAAMK19.

H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010. ISBN 9780821849255. URL https://books.google.com.vn/books?id=MDXa6gFRZuIC.

T. Funke, M. Khosla, M. Rathee, and A. Anand. Zorro: Valid, sparse, and stable explanations in graph neural networks. *IEEE Transactions on Knowledge & Data Engineering*, 35(08):8687–8698, aug 2023. ISSN 1558-2191. doi: 10.1109/TKDE.2022.3201170.

A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3681–3688, Jul. 2019. doi: 10.1609/aaai.v33i01.33013681. URL https://ojs.aaai.org/index.php/AAAI/article/view/4252.

R. Ghrist. Barcodes: The persistent topology of data. *BULLETIN (New Series) OF THE AMERICAN MATHEMATICAL SOCIETY*, 45, 02 2008. doi: 10.1090/S0273-0979-07-01191-3.

R. Ghrist. *Elementary Applied Topology*. CreateSpace Independent Publishing Platform, 2014. ISBN 9781502880857. URL https://books.google.com.vn/books?id=Z5ATogEACAAJ.

M. L. Gromov. Groups of polynomial growth and expanding maps. *Publ. Math., Inst. Hautes Étud. Sci*, pages 53–73, 1981.

J. Heo, S. Joo, and T. Moon. Fooling neural network interpretations via adversarial model manipulation. In *NeurIPS*, 2019.

H. Hofmann. UCI machine learning repository, 1994. URL http://archive.ics.uci.edu/ml.

L. K. Jeff Larson, Surya Mattu and J. Angwin. How we analyzed the compas recidivism algorithm, 2016.

Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Z. C. Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, June 2018. ISSN 1542-7730. doi: 10.1145/3236386.3241340. URL https://doi.org/10.1145/3236386.3241340.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774. 2017.

S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):56–67, Jan 2020. ISSN 2522-5839. doi: 10.1038/s42256-019-0138-9. URL https://doi.org/10.1038/s42256-019-0138-9.

L. McInnes, J. Healy, N. Saul, and L. Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1900654116. URL https://www.pnas.org/content/116/44/22071.

F. Mémoli and G. Sapiro. Comparing point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, volume 71, pages 33–42, 01 2004. doi: 10.1145/1057432.1057436.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8026–8037. 2019.

M. Redmond. UCI machine learning repository, 2011. URL `http://archive.ics.uci.edu/ml`.

M. T. Ribeiro, S. Singh, and C. 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*, page 1135–1144, 2016. ISBN 9781450342322. doi: 10.1145/2939672.2939778.

S. Saito, E. Chua, N. Capel, and R. Hu. Improving lime robustness with smarter locality sampling. *ArXiv*, abs/2006.12302, 2020.

P. Schwab and W. Karlen. CXPlain: Causal Explanations for Model Interpretation under Uncertainty. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3145–3153, 06–11 Aug 2017.

D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society (AIES)*, 2020. URL `https://arxiv.org/pdf/1911.02508.pdf`.

E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41:647–665, 2013.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017. URL `http://proceedings.mlr.press/v70/sundararajan17a.html`.

C. Tralie, N. Saul, and R. Bar-On. Ripser.py: A lean persistent homology library for python. *The Journal of Open Source Software*, 3(29):925, Sep 2018. doi: 10.21105/joss.00925. URL `https://doi.org/10.21105/joss.00925`.

D. Vres and M. Robnik-Sikonja. Better sampling in explanation methods can prevent dieselgate-like deception. *CoRR*, abs/2101.11702, 2021. URL `https://arxiv.org/abs/2101.11702`.

M. Vu and M. T. Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages

12225–12235. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/8fb134f258b1f7865a6ab2d935a897c9-Paper.pdf.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*, abs/1708.07747, 2017.

C.-K. Yeh, C.-Y. Hsieh, A. S. Suggala, D. I. Inouye, and P. Ravikumar. On the (in)fidelity and sensitivity of explanations. In *NeurIPS*, 2019.

Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. GNNexplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems 32*, pages 9244–9255. 2019.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10590-1.