
Label-Free Explainability for Unsupervised Models

Jonathan Crabbé¹ Mihaela van der Schaar^{1 2 3}

Abstract

Unsupervised black-box models are challenging to interpret. Indeed, most existing explainability methods require labels to select which component(s) of the black-box’s output to interpret. In the absence of labels, black-box outputs often are representation vectors whose components do not correspond to any meaningful quantity. Hence, choosing which component(s) to interpret in a label-free unsupervised/self-supervised setting is an important, yet unsolved problem. To bridge this gap in the literature, we introduce two crucial extensions of post-hoc explanation techniques: (1) label-free feature importance and (2) label-free example importance that respectively highlight influential features and training examples for a black-box to construct representations at inference time. We demonstrate that our extensions can be successfully implemented as simple wrappers around many existing feature and example importance methods. We illustrate the utility of our label-free explainability paradigm through a qualitative and quantitative comparison of representation spaces learned by various autoencoders trained on distinct unsupervised tasks.

1. Introduction

Are machine learning models ready to be deployed in high-stakes applications? Recent years have witnessed a success of deep models on nontrivial tasks such as computer vision (Voulodimos et al., 2018), natural language processing (Young et al., 2018) and scientific discovery (Jumper et al., 2021; Davies et al., 2021). The success of these models comes at the cost of their complexity. Deep models typically involve millions to billions operations in order to turn their input data into a prediction. Since it is not pos-

¹University of Cambridge ²The Alan Turing Institute
³University of California Los Angeles. Correspondence to: Jonathan Crabbé <jc2133@cam.ac.uk>, Mihaela van der Schaar <mv472@cam.ac.uk>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

sible for a human user to analyze each of these operations, the models appear as *black-boxes*. When the deployment of these models impact critical areas, such as healthcare, finance or justice, their opacity appears as a major obstruction (Lipton, 2016; Ching et al., 2018; Tjoa & Guan, 2020).

Post-Hoc Explainability. As a response to this transparency problem, the field of *explainable artificial intelligence* (XAI) received an increasing interest, see (Adadi & Berrada, 2018; Barredo Arrieta et al., 2020; Das & Rad, 2020) for reviews. In order to retain the approximation power of deep models, many *post-hoc explainability* methods were developed. These methods complement the predictions of black-box models with various explanations. In this way, models can be understood through the lens of explanations regardless their complexity. We focus on two types of such methods. **(1) Feature Importance** explanations highlight crucial features for the black-box to issue a prediction. Examples include Saliency (Simonyan et al., 2013), Lime (Ribeiro et al., 2016), Integrated Gradients (Sundararajan et al., 2017), Shap (Lundberg & Lee, 2017), DeepLift (Shrikumar et al., 2017) and Perturbation Masks (Fong & Vedaldi, 2017; Crabbé & Van Der Schaar, 2021). **(2) Example Importance** explanations highlight crucial training examples for the black-box to issue a prediction. Examples include Influence Function (Cook & Weisenberg, 1982; Koh & Liang, 2017), Deep K-Nearest Neighbours (Papernot & McDaniel, 2018), TraceIn (Pruthi et al., 2020) and SimplEx (Crabbé et al., 2021).

Supervised Setting. The previous works almost exclusively focus on explaining models obtained in a *supervised setting*. In this setting, the black box $f : \mathcal{X} \rightarrow \mathcal{Y}$ connects two meaningful spaces: the feature space \mathcal{X} and the label space \mathcal{Y} . By meaningful, we mean that each axis of these spaces corresponds to a quantity known by the model’s user. This is illustrated at the top of Figure 1 with an idealized prostate cancer risk predictor. Each axis of \mathcal{Y} corresponds to a clear label: the probability of mortality with and without treatment. We can explain the predictions of this model with the importance of features/examples to make a prediction on each individual axis y_1, y_2 or for a combination of those axes (e.g. in this case $y_1 - y_2$ is associated to the treatment effect). The key point is that, in the supervised setting, the user knows the meaning of the black-box output they try to interpret. This is not always the case in machine learning.

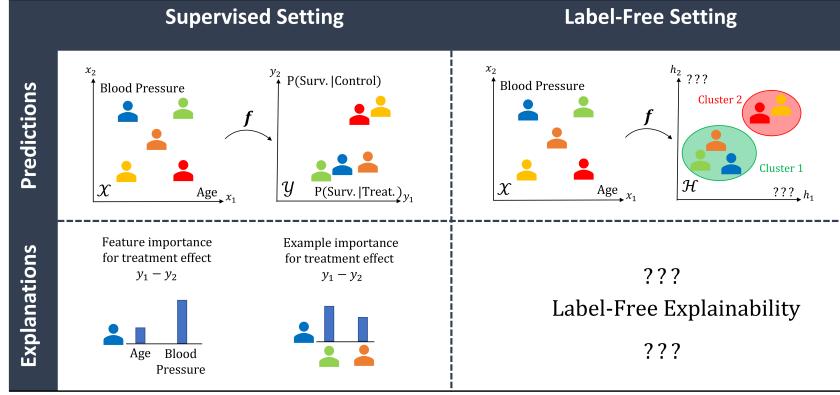


Figure 1. Supervised and Label-Free Settings.

Label-Free Setting. In the label-free setting, we are interested in black-boxes $f : \mathcal{X} \rightarrow \mathcal{H}$ that connect a meaningful feature space \mathcal{X} to a latent (or representation) space \mathcal{H} . Unlike the feature space \mathcal{X} and the label space \mathcal{Y} , the axes of the latent space \mathcal{H} do not correspond to labelled quantities known by the user. This is illustrated at the bottom of Figure 1, where the examples are mapped in a representation space for clustering purposes. Unlike in the supervised setting, there is no obvious way for the user to choose an axis among h_1, h_2 to interpret. This distinction goes well beyond the philosophical consideration. As we show in Sections 2 and 3, the aforementioned feature and example importance methods require labels to select an axis to interpret or evaluate a loss. The usage of these methods in the label-free setting hence requires a non-trivial extension.

Motivation. To illustrate the significance of the label-free extension of explainability, we outline 2 widespread setups where it is required. Note that these 2 setups are not mutually exclusive. **(1) Unsupervised Learning:** Whenever we solve an unsupervised task such as clustering or density estimation, it is common to learn a function like f that projects the data onto a low dimensional representation space. Due to the unsupervised nature of the problem, any explanation of the latent space has to be done without label. **(2) Self-Supervised Learning:** Even in a supervised learning setting, we have often more unlabelled than labelled data available. When this is the case, self-supervised learning preconises to leverage unlabelled data to solve an unsupervised pretext tasks such as predicting the missing part of a masked image (Jing & Tian, 2021). This yields a representation f of the data that we can use as part of a model to solve the downstream task. If we want to interpret the model’s representations of unlabelled examples, label-free explainability is a must. Let us now review related work.

Related Work. The majority of the relevant literature focuses on increasing the interpretability of representations spaces \mathcal{H} . Disentangled-VAEs constitute the best example (Higgins et al., 2017; Burgess et al., 2018; Chen et al.,

2018; Sarhan et al., 2019), we discuss them in more details in Section 4.3. When it comes to post-hoc approaches for explainability of latent spaces, *concept-based explanations* (Kim et al., 2017; Brocki & Chung, 2019) are central. These methods identify a dictionary between human concepts (like the presence of stripes on an image) and latent vectors. These methods are only partially relevant here since concepts are typically learned with labelled examples, although early works challenge this assumption (Ghorbani et al., 2019). When it comes to label-free feature importance, we note that *Layer Relevance Propagation* (LRP) permit to assign feature importance scores in the absence of labels (Bach et al., 2015; Eberle et al., 2022; Holzinger et al., 2022). That said, these works have been restricted to specific settings (e.g. clustering and similarity models) and come with the natural limitations of LRP-based methods (e.g. no implementation invariance). Similarly, works discussing label-free example importance are also restricted to specific settings, like Kong & Chaudhuri (2021) that adapt TraceIn to VAEs.

Contribution. **(1) Label-Free Feature Importance:** We introduce a general framework to extend *linear* feature importance methods to the label-free setting (Section 2). Our extension is done by defining an auxiliary scalar function as a wrapper around the label-free black-box to interpret. This permits to compute feature importance in the label-free setting by retaining useful properties of the original methods without increasing their complexity. **(2) Label-Free Example Importance:** We extend example importance methods to the label-free setting (Section 3). In this work, we treat two types of example importance methods that we call *loss-based* and *representation-based*. For the former, the extension requires to specify a label-free loss and a set of relevant model parameters to differentiate the loss with. For the latter, the extension is straightforward. Our feature and example importance extensions are validated experimentally (Section 4.1) and their practical utility is demonstrated with a use case motivated by self-supervised

learning (Section 4.2). **(3) Challenging Interpretability of Disentangled Representations:** In testing the limits of our feature importance hypotheses with disentangled VAEs, we noticed that the interpretability of saliency maps associated to individual latent units seems unrelated to the strength of disentanglement between the units (Section 4.3). We analyze this phenomenon both qualitatively and quantitatively. This insight could be the seed of future developments in interpretable representation learning.

2. Feature Importance

In this section, we present our approach to extend linear feature importance methods to the label-free setting. We start by reviewing the typical setup with label to grasp some useful insights for our extension. With these insights, the extension to the label-free regime is immediate.

2.1. Feature Importance with Labels

We consider an input (or feature) space $\mathcal{X} \subset \mathbb{R}^{d_X}$ and an output (or label) space $\mathcal{Y} \subset \mathbb{R}^{d_Y}$, where $d_X \in \mathbb{N}^*$ and $d_Y \in \mathbb{N}^*$ are respectively the dimension of the input and output spaces. We are given a black-box model $f : \mathcal{X} \rightarrow \mathcal{Y}$ from a hypothesis set¹ $\mathcal{A}(\mathcal{Y}^\mathcal{X})$ mapping each input $\mathbf{x} \in \mathcal{X}$ to an output $\mathbf{y} = f(\mathbf{x}) \in \mathcal{Y}$. Note that we use bold symbols to emphasize that those elements are typically vectors with more than one component ($d_X, d_Y > 1$). Feature importance methods explain the black-box prediction $f(\mathbf{x})$ by attributing an importance score $a_i(f, \mathbf{x})$ to each feature x_i of \mathbf{x} for² $i \in [d_X]$. Note that feature importance methods require to select one component $f_j(\mathbf{x}) \in \mathbb{R}$ for some $j \in [d_Y]$ of the output in order to compute these scores: $a_i(f, \mathbf{x}) \equiv a_i(f_j, \mathbf{x})$. In a classification setting, j typically corresponds to the ground-truth label (when it is known) $j = \arg \max_{k \in [d_Y]} [y_k]$ or to the label with maximal predicted probability $j = \arg \max_{k \in [d_Y]} [f_k(\mathbf{x})]$.

We now suggest an alternative approach: combining the importance scores for each component of f by weighing them with the associated class probability: $b_i(f, \mathbf{x}) \equiv \sum_{j=1}^{d_Y} f_j(\mathbf{x}) \cdot a_i(f_j, \mathbf{x})$. We note that, when a class probability $f_j(\mathbf{x}) \approx 1$ dominates, this reduces to the previous definition. However, when the class probabilities are balanced, this accounts for the contribution of each class. In the image classification context, this might be more appropriate than cherry-picking the saliency map of the appropriate class while disregarding the others (Rudin, 2019). To the best of our knowledge, this approach has not been used in the literature. A likely reason for this is that this definition requires to compute $d_Y \cdot d_X$ importance scores per sample, which quickly becomes expensive as the number of

classes grows. This limitation can easily be avoided when the importance scores are linear with respect to the black-box³. In this case, the weighted importance score can be rewritten as $b_i(f, \mathbf{x}) = a_i(\sum_{j=1}^{d_Y} f_j(\mathbf{x}) \cdot f_j, \mathbf{x})$. With this trick, we can compute the weighted importance score by only calling the auxiliary function $g_\mathbf{x}$ defined for all $\tilde{\mathbf{x}} \in \mathcal{X}$ as $g_\mathbf{x}(\tilde{\mathbf{x}}) = \sum_{j=1}^{d_Y} f_j(\mathbf{x}) \cdot f_j(\tilde{\mathbf{x}})$. We will now use a similar reasoning in the label-free setting.

2.2. Label-Free Feature Importance

We now turn our setting of interest. We consider a latent (or representation) space $\mathcal{H} \subset \mathbb{R}^{d_H}$, where $d_H \in \mathbb{N}^*$ is the dimension of the latent space. We are given a black-box model $f : \mathcal{X} \rightarrow \mathcal{H}$ from a hypothesis set $\mathcal{A}(\mathcal{H}^\mathcal{X})$ mapping each input $\mathbf{x} \in \mathcal{X}$ to a representation $\mathbf{h} = f(\mathbf{x}) \in \mathcal{H}$. As aforementioned, the latent space dimensions $h_j, j \in [d_H]$ are not related to labels with clear interpretations. Again, we would like to attribute an importance score $b_i(f, \mathbf{x})$ to each feature x_i of \mathbf{x} for $i \in [d_X]$. Ideally, this score should reflect the importance of feature x_i in assigning the representation $\mathbf{h} = f(\mathbf{x})$. Unlike in the previous setting, we do not have a principled way of choosing a component f_j for some $j \in [d_H]$. How can we compute importance scores?

We can simply mimic the approach described in the previous section. For a feature importance method, we use the weighted importance score $b_i(f, \mathbf{x}) = a_i(\sum_{j=1}^{d_H} f_j(\mathbf{x}) \cdot f_j, \mathbf{x})$. We stress that the individual components $f_j(\mathbf{x})$ do not correspond to probabilities in this case. Does it still make sense to compute a sum weighted by these components? In most cases, it does. The components will typically correspond to a neuron’s activation function (Glorot et al., 2011). With typical activation functions such as ReLU or Sigmoid, inactive neurons will correspond to a vanishing component $f_j(\mathbf{x}) = 0$. From the above formula, this implies that these neurons will not contribute in the computation of $b_i(f, \mathbf{x})$. Neurons that are more activated, on the other hand, will contribute more to the weighted sum. By linearity of the feature importance method, this reasoning extends to linear combinations of neurons. We note that the weighted sum is a latent space inner product. This leads to the following definition.

Definition 2.1 (Label-Free Feature Importance). Let $f : \mathcal{X} \rightarrow \mathcal{H}$ be a black-box latent map and for all $i \in [d_X]$ a $a_i(\cdot, \cdot) : \mathcal{A}(\mathbb{R}^\mathcal{X}) \times \mathcal{X} \rightarrow \mathbb{R}$ be a feature importance score linear w.r.t. its first argument. We define the *label-free* feature importance as a score $b_i(\cdot, \cdot) : \mathcal{A}(\mathcal{H}^\mathcal{X}) \times \mathcal{X} \rightarrow \mathbb{R}$:

$$b_i(f, \mathbf{x}) \equiv a_i(g_\mathbf{x}, \mathbf{x}) \quad (1)$$

$g_\mathbf{x} : \mathcal{X} \rightarrow \mathbb{R}$ such that for all $\tilde{\mathbf{x}} \in \mathcal{X}$:

$$g_\mathbf{x}(\tilde{\mathbf{x}}) = \langle f(\mathbf{x}), f(\tilde{\mathbf{x}}) \rangle_{\mathcal{H}}, \quad (2)$$

¹Typically neural networks.

²We denote by $[N]$ the positive integers from 1 to $N \in \mathbb{N}^*$.

³Which is the case for most methods, including Shap.

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes an inner product for the space \mathcal{H} .

Remark 2.2. This definition gives a simple recipe to extend any linear feature importance method a_i to the label-free setting. In practice, this is implemented by defining the auxiliary scalar function $g_{\mathbf{x}}$ as a simple wrapper around the black-box function f . We then feed $g_{\mathbf{x}}$ to any feature importance method a_i .

Arguably one of the most important property shared by many feature importance methods is *completeness*. Feature importance methods endowed with this property produce importance scores whose sum equals the black-box prediction up to a constant baseline $a_0 \in \mathbb{R}$: $\sum_{i=1}^{d_X} a_i(g, \mathbf{x}) = g(\mathbf{x}) - a_0$. This provides a meaningful connection between importance scores and black-box predictions. Typical examples of baselines are $a_0 = 0$ for Lime; the expected prediction $a_0 = \mathbb{E}_{\mathbf{X}}[g(\mathbf{X})]$ for Shap and a baseline prediction $a_0 = g(\bar{\mathbf{x}})$ for Integrated Gradients. We show that our label-free feature importance scores are endowed with an analogous property in higher dimension.

Proposition 2.3 (Label-Free Completeness). *If the feature importance scores $a_i(\cdot, \cdot) : \mathcal{A}(\mathbb{R}^{\mathcal{X}}) \times \mathcal{X} \rightarrow \mathbb{R}$ are linear and satisfy completeness, then the label-free importance scores $b_i(f, \mathbf{x}), i \in [d_X]$ sum to the black-box representation's norm $\|f(\mathbf{x})\|_{\mathcal{H}}^2 = \langle f(\mathbf{x}), f(\mathbf{x}) \rangle_{\mathcal{H}}$ up to a constant baseline $b_0 \in \mathbb{R}$ for all $\mathbf{x} \in \mathcal{X}$:*

$$\sum_{i=1}^{d_X} b_i(f, \mathbf{x}) = \|f(\mathbf{x})\|_{\mathcal{H}}^2 - b_0. \quad (3)$$

Proof. The proof is provided in Appendix A. \square

This property is more general than its LRP counterpart (Proposition 1 in (Eberle et al., 2022)) that holds only for neural network with biases set to zero. Furthermore, in Appendix A.2, we demonstrate that our label-free extension of feature importance verifies crucial invariance properties with respect to latent space symmetries.

3. Example Importance

In this section, we present our approach to extend example importance methods to the label-free setting. Since example importance methods are harder to unify, the structure of this section differs from the previous one. We split the example importance methods in two families: the loss-based and representation-based methods. The extension to the label-free setting works differently for these two families. Hence, we treat them in two distinct subsections. In both cases, we work with an input space \mathcal{X} and a latent space \mathcal{H} . We are given a training set of $N \in \mathbb{N}^*$ examples $\mathcal{D}_{\text{train}} = \{\mathbf{x}^n \mid n \in [N]\}$. This training set is used to fit a black-box latent map $f : \mathcal{X} \rightarrow \mathcal{H}$. We want to assign an

example importance score $c^n(f, \mathbf{x})$ to each training example $\mathbf{x}^n \in \mathcal{D}_{\text{train}}$ for the black-box f to build a representation $f(\mathbf{x}) \in \mathcal{H}$ of a test example $\mathbf{x} \in \mathcal{X}$. Note that we use upper indices for examples in contrast with the lower indices for the features. Hence, x_i^n denotes feature i of training example n . Similarly, c^n denotes an example importance in contrast with b_i that is used for feature importance.

3.1. Loss-Based Example Importance

Supervised Setting. Loss-based example importance methods assign a score to each training example \mathbf{x}^n by simulating the effect of their removal on the loss for a test example. To make this more explicit, we denote by \mathbf{z} the data of an example that is required to evaluate the loss. In a supervised setting, this typically correspond to a couple $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ with an input $\mathbf{x} \in \mathcal{X}$ and a label $\mathbf{y} \in \mathcal{Y}$. Similarly, the training set is of the form $\mathcal{D}_{\text{train}} = \{\mathbf{z}^n \mid n \in [N]\}$. This training set is used to fit a black-box model f_{θ} parametrized by $\theta \in \Theta \subset \mathbb{R}^P$, where $P \in \mathbb{N}^*$ is the number of model parameters. Training is done by solving $\theta_* = \arg \min_{\theta \in \Theta} \sum_{n=1}^N L(\mathbf{z}^n, \theta)$ with the loss $L : \mathcal{Z} \times \Theta \rightarrow \mathbb{R}$. This yields a model f_{θ_*} . If we remove an example \mathbf{z}^n from $\mathcal{D}_{\text{train}}$, the optimization problem turns into $\theta_*^{-n} = \arg \min_{\theta \in \Theta} \sum_{m=1, m \neq n}^N L(\mathbf{z}^m, \theta)$. This creates a parameter shift $\delta^n \theta \equiv \theta_*^{-n} - \theta_*$. This parameter shift, in turns, impacts the loss $L(\mathbf{z}, \theta)$ on a test example \mathbf{z} . This shift is reflected by the quantity $\delta^n \theta L(\mathbf{z}, \theta_*) \equiv L(\mathbf{z}, \theta_*^{-n}) - L(\mathbf{z}, \theta_*)$. This loss shift permits to draw a distinction between proponents (\mathbf{z}^n that decrease the loss: $\delta^n \theta L(\mathbf{z}, \theta_*) > 0$) and opponents (\mathbf{z}^n that increase the loss: $\delta^n \theta L(\mathbf{z}, \theta_*) < 0$). Hence, it provides a meaningful measure of example importance. In order to estimate the loss shift without retraining the model, Koh & Liang (2017) propose to evaluate the influence function:

$$\delta^n \theta L(\mathbf{z}, \theta_*) \approx \frac{1}{N} \langle \nabla_{\theta} L(\mathbf{z}, \theta_*), \mathbf{H}_{\theta_*}^{-1} \nabla_{\theta} L(\mathbf{z}^n, \theta_*) \rangle_{\Theta},$$

where $\langle \cdot, \cdot \rangle_{\Theta}$ is an inner product on the parameter space Θ and $\mathbf{H}_{\theta_*} \equiv \sum_{n=1}^N N^{-1} \nabla_{\theta}^2 L(\mathbf{z}^n, \theta_*) \in \mathbb{R}^{P \times P}$ is the training loss Hessian matrix. Similarly, Pruthi et al. (2020) propose to use checkpoints during the model training to evaluate the loss shift:

$$\delta^n \theta L(\mathbf{z}, \theta_*) \approx \sum_{t=1}^T \eta_t \langle \nabla_{\theta} L(\mathbf{z}, \theta_t), \nabla_{\theta} L(\mathbf{z}^n, \theta_t) \rangle_{\Theta},$$

where η_t and θ_t are respectively the learning rate and the model's parameters at checkpoint $t \in [T]$, $T \in \mathbb{N}^*$ is the total number of checkpoints. Similar approaches building on the theory of Shapley values exist (Ghorbani & Zou, 2019; Ghorbani et al., 2020).

Label-free Setting. We now turn to the label-free setting. In this case, we train our model with a label-free loss L :

$\mathcal{X} \times \Theta \rightarrow \mathbb{R}$. Is it enough to drop the label and fix $\mathbf{z} = \mathbf{x}$ in all the above expressions? Most of the time, no. It is important to notice that the latent map $\mathbf{f} : \mathcal{X} \rightarrow \mathcal{H}$ that we wish to interpret is not necessarily equal to the model that we use to evaluate the loss $L(\mathbf{z}, \theta)$. To understand, it is instructive to consider a concrete example. Let us assume that we are in a self-supervised setting and that we train an autoencoder $\mathbf{f}_d \circ \mathbf{f}_e : \mathcal{X} \rightarrow \mathcal{X}$ that consists in an encoder $\mathbf{f}_e : \mathcal{X} \rightarrow \mathcal{H}$ and a decoder $\mathbf{f}_d : \mathcal{H} \rightarrow \mathcal{X}$ on a pretext task such as denoising. At the end of the pretraining, we typically keep the encoder \mathbf{f}_e and drop the decoder \mathbf{f}_d . If we want to compute the loss-based example importance, we are facing a difficulty. On the one hand, we would like to draw interpretations that rely solely on the encoder \mathbf{f}_e that is going to be used in a downstream task. On the other hand, loss-based example importance scores are computed with the autoencoder $\mathbf{f}_d \circ \mathbf{f}_e$ that also involves the irrelevant decoder \mathbf{f}_d . Whenever the loss evaluation involves more than the black-box latent map \mathbf{f} we wish to interpret, replacing \mathbf{z} by \mathbf{x} in the above expressions is therefore insufficient.

To provide a more satisfactory solution, we split the parameter space $\Theta = \Theta_r \times \Theta_{irr}$ into a relevant component $\Theta_r \subseteq \mathbb{R}^{P_r}$ and an irrelevant component $\Theta_{irr} \subseteq \mathbb{R}^{P_{irr}}$. The black-box \mathbf{f} to interpret is parametrized only by the relevant parameters θ_r and can be denoted \mathbf{f}_{θ_r} . It typically corresponds to an encoder, as in the previous example. Concretely, we are interested in isolating the part of loss shift $\delta_{\theta}^n L(\mathbf{z}, \theta_*)$ caused by the variation of these relevant parameters. We note that the above estimators for this quantity involve gradients with respect to the parameters θ . We decompose the overall parameters gradient in terms of relevant and irrelevant parameters gradients: $\nabla_{\theta} = \nabla_{\theta_r} + \nabla_{\theta_{irr}}$. Ignoring the variation of irrelevant parameters is equivalent to setting $\nabla_{\theta_{irr}} = 0$ in the above expressions. This is trivially equivalent to the replacement of δ_{θ}^n by $\delta_{\theta_r}^n$. This motivates the following definition.

Definition 3.1 (Label-Free Loss-Based Importance). Let $\mathbf{f}_{\theta_r} : \mathcal{X} \rightarrow \mathcal{H}$ be a black-box latent map trained to minimize a loss $L : \mathcal{X} \times \Theta \rightarrow \mathbb{R}$ on a training set $\mathcal{D}_{\text{train}} = \{\mathbf{x}^n \mid n \in [N]\}$. To measure the impact of removing example \mathbf{x}^n from $\mathcal{D}_{\text{train}}$ with $n \in [N]$, we define the *Label-Free Loss-Based Example Importance* as a score $c^n(\cdot, \cdot) : \mathcal{A}(\mathcal{H}^{\mathcal{X}}) \times \mathcal{X} \rightarrow \mathbb{R}$ such that

$$c^n(\mathbf{f}_{\theta_r}, \mathbf{x}) = \delta_{\theta_r}^n L(\mathbf{x}, \theta_*) \quad (4)$$

Remark 3.2. This definition gives a simple recipe to extend any loss-based example importance method to the label-free setting. In practice, this is implemented by using the unsupervised loss L trained to fit the model and differentiating L with respect to parameters of the encoder we wish to interpret.

We stress that the loss depends only on a *single* input ex-

ample \mathbf{x} . This is obviously not the case for *contrastive losses* that involve *pairs* of input examples (Chen et al., 2020). There is not obvious extension of loss-based example importance in this setting. We will now present another type of example importance method that better extends to contrastive learning.

3.2. Representation-Based Example Importance

Supervised Setting. Although representation-based example importance methods are introduced in a supervised context, their extension to the label-free setting is straightforward. These methods assign a score to each training example \mathbf{x}^n by analyzing the latent representations of these examples. To make this more concrete, we start with a typical supervised setting. Consider a model $\mathbf{f}_l \circ \mathbf{f}_e : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathbf{f}_e : \mathcal{X} \rightarrow \mathcal{H}$ maps inputs to latent representations and $\mathbf{f}_l : \mathcal{H} \rightarrow \mathcal{Y}$ maps representations to labels. In the case of deep neural networks, the representation space typically corresponds to the output of an intermediate layer. We would like to see how the representation map relates a test example $\mathbf{x} \in \mathcal{X}$ to the training set examples. This can be done by mapping the training set inputs into the representation space $\mathbf{f}_e(\mathcal{D}_{\text{train}}) = \{\mathbf{f}_e(\mathbf{x}^n) \mid n \in [N]\}$. To quantify the affinity between \mathbf{x} and the training set examples, we attempt a reconstruction of $\mathbf{f}_e(\mathbf{x})$ with training representations from $\mathbf{f}_e(\mathcal{D}_{\text{train}})$: $\mathbf{f}_e(\mathbf{x}) \approx \sum_{n=1}^N w^n(\mathbf{x}) \cdot \mathbf{f}_e(\mathbf{x}^n)$. Following (Papernot & McDaniel, 2018), the most obvious approach to define weights $w^n(\mathbf{x})$ is to identify the indices $\text{KNN}(\mathbf{x}) \subset [N]$ of the K nearest neighbours (DKNN) of $\mathbf{f}_e(\mathbf{x})$ in $\mathbf{f}_e(\mathcal{D}_{\text{train}})$ and weigh them according to a Kernel function $\kappa : \mathcal{H}^2 \rightarrow \mathbb{R}^+$:

$$w^n(\mathbf{x}) = \mathbf{1}[n \in \text{KNN}(\mathbf{x})] \cdot \kappa[\mathbf{f}_e(\mathbf{x}^n), \mathbf{f}_e(\mathbf{x})], \quad (5)$$

where $\mathbf{1}$ denotes the indicator function. Similarly, Crabbé et al. (2021) propose to learn the weights by solving

$$\mathbf{w}(\mathbf{x}) = \arg \min_{\lambda \in [0,1]^N} \left\| \mathbf{f}_e(\mathbf{x}) - \sum_{n=1}^N \lambda^n \mathbf{f}_e(\mathbf{x}^n) \right\|_{\mathcal{H}} \quad (6)$$

such that $\sum_{n=1}^N \lambda^n = 1$. Similar approaches building on the representer theorem exist (Yeh et al., 2018).

Label-free Setting. We now turn to the label-free setting. The above discussion remains valid if we replace the supervised representation map \mathbf{f}_e by an unsupervised representation map \mathbf{f} . In short, we can take $c^n = w^n$ without any additional work for representation-based methods. A major advantage of representation-based methods over loss-based methods is that they only require latent representations. Therefore, they naturally extend to representation spaces learned in contrastive learning. Moreover, we argue in Appendix A that only representation-based methods are invariant to latent space symmetries.

4. Experiments

In this section, we conduct quantitative evaluations of the label-free extensions of various explanation methods. We start with simple consistency checks to ensure that these methods provide sensible explanations for unsupervised models. Then, we demonstrate how our label-free explanation paradigm makes it possible to compare representations learned from different pretext tasks. Finally, we challenge Definition 2.1 by studying saliency maps of VAEs. A more detailed description of each experiment can be found in Appendix C. The implementation is available online^{4 5}.

4.1. Consistency Checks

We would like to assess whether the approaches described in Sections 2 and 3 provide a sensible way to extend feature and example importance scores to the unsupervised setting.

Setup. We fit 3 models on 3 datasets: a denoising autoencoder CNN on the MNIST image dataset (LeCun et al., 1998), a LSTM reconstruction autoencoder on the ECG5000 time series dataset (Goldberger et al., 2000) and a SimCLR (Chen et al., 2020) neural network with a ResNet-18 (He et al., 2015) backbone on the CIFAR-10 image dataset (Krizhevsky, 2009). We extract an encoder f_e to interpret from each model. ▶ **Feature Importance:** We compute the label-free feature importance $b_i(f_e, \mathbf{x})$ of each feature (pixel/time step) x_i for building the latent representation of the test example $\mathbf{x} \in \mathcal{D}_{\text{test}}$. To verify that high-scoring features are salient, we use an approach analogous to pixel-flipping (Montavon et al., 2018): we mask the M most important features with a mask $\mathbf{m} \in \{0, 1\}^{d_x}$. We measure the latent shift $\|f_e(\mathbf{x}) - f_e(\mathbf{m} \odot \mathbf{x} + (1 - \mathbf{m}) \odot \bar{\mathbf{x}})\|_{\mathcal{H}}$ induced by replacing the most important features with a baseline $\bar{\mathbf{x}}$, where \odot denotes the Hadamard product. We expect this shift to increase with the importance of masked features. We report the average shift over the testing set for several values of M and feature importance methods in Figure 2. ▶ **Example Importance:** We sample $N = 1000$ training examples $\mathbf{x}^n \in \mathcal{D}_{\text{train}}, n \in [N]$ without replacement. We compute the importance score $c^n(f_e, \mathbf{x})$ of each training example \mathbf{x}^n for predicting the latent representation of the test images $\mathbf{x} \in \mathcal{D}_{\text{test}}$. To verify that high scoring examples are salient, we use an approach analogous to Kong & Chaudhuri (2021). We select the M most important training examples $\mathbf{x}^{n_1}, \dots, \mathbf{x}^{n_M}$. We compare their ground truth label y^{n_1}, \dots, y^{n_M} to the label y of \mathbf{x} . We compute the similarity rates $\sum_{m=1}^M \delta_{y, y^{n_m}} / M$, where δ denotes the Kronecker delta. We reproduce the above steps for the M least important examples. If the encoder meaningfully represents

the data, we expect the similarity rate of the most important examples to be higher than for the least important examples. We report the distribution of similarity rates across 1,000 test examples for various values of M and example importance methods in Figure 3.

Results. ▶ **Feature Importance:** Label-free feature importance methods exhibit a similar behaviour: the latent shift increases sharply as we perturb the few most important pixels. This increase decelerates when we start perturbing pixels that are less relevant. Furthermore, selecting the perturbed pixels according to the various importance scores $b_i(f_e, \mathbf{x})$ yields latent shifts that are significantly larger than the shift induced by perturbing random pixels. Label-free Integrated Gradients outperform other methods for each model. These observations confirm that the label-free importance scores $b_i(f_e, \mathbf{x})$ allow us to identify the features that are the most relevant for the encoder f_e to build a latent representation for the example \mathbf{x} . ▶ **Example Importance:** For all example importance method, we observe that the similarity rate is substantially higher among the most similar examples than among the least similar examples. This observation confirms that the label-free importance scores $c^n(f_e, \mathbf{x})$ allow us to identify training examples that are related to the test example we wish to explain. Representation-based methods usually outperform loss-based methods. In this case, the verification also validates our models since no label was used during training.

4.2. Use Case: Comparing the Representations Learned with Different Pretext Tasks

In a self-supervised learning setting, many unsupervised pretext task can be chosen to learn a meaningful representation of the data. How do the representations from different pretext tasks compare to each other? We show that label-free explainability permits to answer this question through qualitative and quantitative comparisons of representations.

Setup. We work with the MNIST denoising autoencoder from Section 4.1. Besides denoising, we consider 2 additional pretext tasks along with their autoencoders: reconstruction and inpainting (Pandit et al., 2019). Finally, we use the labelled training set to fit a MNIST classifier and extract the representations from the penultimate layer. We are interested in comparing the representation spaces learned by the encoder f_e for the various tasks. ▶ **Feature Importance:** For each encoder f_e , we use our label-free Gradient Shap to produce saliency maps $b_i(f_e, \mathbf{x})$ for the test images $\mathbf{x} \in \mathcal{D}_{\text{test}}$. To compare the saliency maps obtained by different models, a common approach is to compute their Pearson correlation coefficient (Le Meur & Baccino, 2013). We report the average Pearson coefficients between the encoders across 5 runs in Table 1. ▶ **Example Importance:** For each encoder f_e , we use our label-free Deep-KNN to produce

⁴<https://github.com/JonathanCrabbe/Label-Free-XAI>

⁵<https://github.com/vanderschaarlab/Label-Free-XAI>

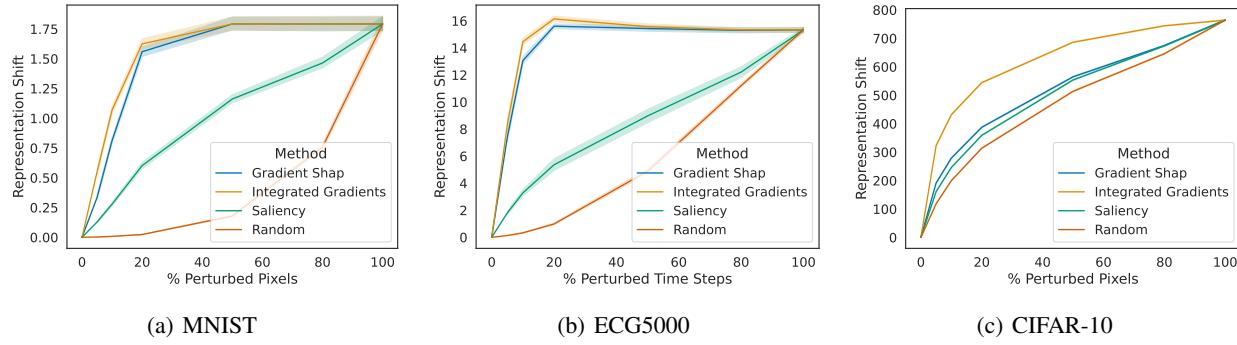


Figure 2. Consistency check for label-free feature importance (average and 95% confidence interval).

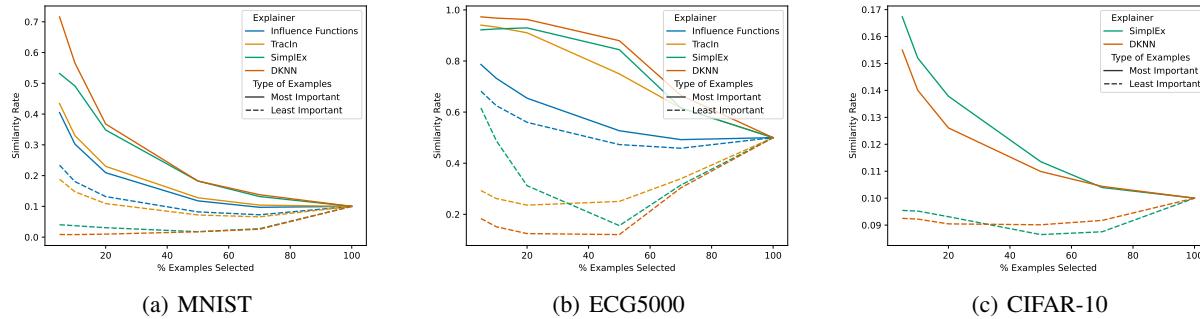


Figure 3. Consistency check for label-free example importance (only representation-based methods apply to SimCLR).

example importance $c^n(\mathbf{f}_e, \mathbf{x})$ of 1,000 training examples $\mathbf{x}^n \in \mathcal{D}_{\text{train}}$ for 1,000 test images $\mathbf{x} \in \mathcal{D}_{\text{test}}$. Again, we use the Pearson correlation coefficient to compare different encoders. We report the average Pearson coefficients between the encoders across 5 runs in Table 2.

Results. ▶ Not all representations are created equal. For saliency maps, the Pearson correlation coefficients range from .31 to .44. This corresponds to moderate positive correlations. A good baseline to interpret these results is provided by Ouerhani et al. (2003): the correlation between the fixation of two human subjects (human saliency maps) are typically in the same range. Hence, two encoders trained on distinct pretext tasks pay attention to different parts of the image like two separate human subjects typically do. For example importance scores, the Pearson correlation coefficients range from .06 to .12, which correspond to weak correlations. For both explanation types, these quantitative results strongly suggest that distinct pretext tasks do not yield interchangeable representations. ▶ **What makes classification special?** For saliency maps, the autoencoder-classifier correlations are comparable to those of the autoencoder-autoencoder. This shows that using labels creates a shift in the saliency maps comparable to changing the unsupervised pretext task. Hence, classification does not appear as a privileged task in terms of feature importance. Things are different for example importance: the autoencoder-classifier correlations are substantially lower than those of the autoencoder-autoencoder. One likely reason is that the

classifier groups examples together according to an external label that is unknown to the autoencoders.

Table 1. Pearson correlation for saliency maps (avg +/- std).

PEARSON	RECON.	DENOIS.	INPAINT.	CLASSIF.
RECON.				
DENOIS.	.39 ± .01			
INPAINT.	.31 ± .02	.32 ± .01		
CLASSIF.	.44 ± .02	.40 ± .00	.32 ± .02	

Table 2. Pearson correlation for example importance (avg +/- std).

PEARSON	RECON.	DENOIS.	INPAINT.	CLASSIF.
RECON.				
DENOIS.	.10 ± .04			
INPAINT.	.11 ± .03	.12 ± .03		
CLASSIF.	.07 ± .03	.06 ± .02	.07 ± .01	

Qualitative Analysis. Beyond quantitative analysis, label-free explainability allows us to appreciate qualitative differences between different encoders. To illustrate, we plot the most important DKNN and the saliency maps for the various encoders in Figure 4. ▶ **Feature Importance:** In accordance with our quantitative analysis, the saliency maps between different tasks look different. For instance, the denoising encoder seems to focus on small contiguous parts of the images. In contrast, the classifier seems to focus on a few isolated pixels. ▶ **Example Importance:** The top examples are rarely similar across various pretext tasks, as suggested

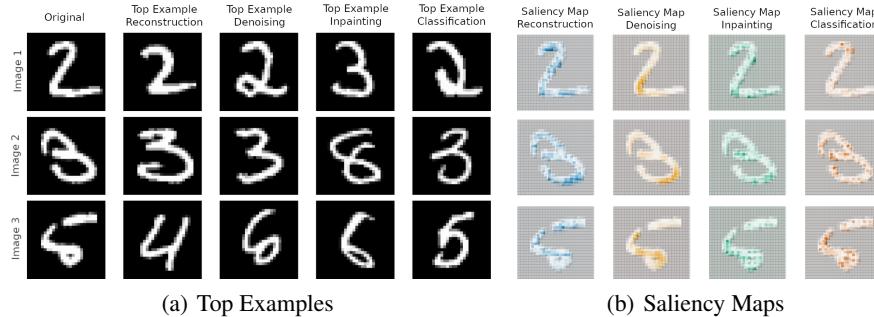


Figure 4. Explanations for different pretext tasks.

by the quantitative analysis. The classifier is the only one that associates an example of the same class given an ambiguous image like Image 3. ► **Synergies:** Sometimes, saliency maps permit to better understand example importance. For instance, let us consider Image 3. In comparison to the other encoders, the reconstruction encoder pays less attention to the crucial loop at the image bottom. Hence, it is not surprising that the corresponding top example is less relevant than those selected by the other encoders.

4.3. Challenging our Assumptions with Disentangled VAEs

In Definition 2.1, the inner product appearing in the label-free importance expression corresponds to a sum over the latent space dimensions. In practice, this has the effect of mixing the feature importance for each latent unit (neuron) to compute an overall feature importance. While this is reasonable when no particular meaning is attached to each latent unit, it might be inappropriate when the units are designed to be interpretable. Disentangled VAEs, for instance, involve latent units that are sensitive to change in single data generative factors, while invariants to other factors. This selective sensitivity permits to assign a meaning to each unit. An important question ensues: can we identify the generative factor associated to each latent unit by using their saliency maps? To answer, we propose a qualitative and quantitative analysis of the saliency maps from disentangled latent units. We show that, even for disentangled β -VAEs, the saliency maps of individual latent units are hard to interpret on their own.

Setup. We study two popular disentangled VAEs : the β -VAE (Higgins et al., 2017) and the TC-VAE (Chen et al., 2018). Those two VAEs involve a variational encoder computing the expected representation $\mu : \mathcal{X} \rightarrow \mathcal{H}$ as well as its standard deviation $\sigma : \mathcal{X} \rightarrow \mathcal{H}$ and a decoder $f_d : \mathcal{H} \rightarrow \mathcal{X}$. Latent samples are obtained with the reparametrization trick (Kingma & Welling, 2013): $h = \mu(x) + \sigma(x) \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$. These VAEs are trained on the MNIST and dSprites datasets (Matthey et al., 2017) (90% – 10% train-test split) to minimize their objective. We use $d_H = 3$ latent

units for MNIST and $d_H = 6$ for dSprites. We train 20 disentangled VAEs of each type for $\beta \in \{1, 5, 10\}$.

Qualitative Analysis. We use Gradient Shap to evaluate the importance⁶ $a_i(\mu_j, \mathbf{x})$ of each pixel x_i from the image \mathbf{x} to predict each latent unit $j \in [d_H]$. Again, we use the Pearson correlation to compare the saliency maps for each pair of latent unit⁷. In this case, a low Pearson correlation corresponds to latent units paying attention to distinct parts of the images. Clearly, this is desirable if we want to easily identify the specialization of each latent unit. Therefore, use this criterion to select a VAE to analyse among the 120 VAEs we trained on each dataset. This corresponds to a β -VAE with $\beta = 10$ for MNIST and a TC-VAE with $\beta = 1$ for dSprites. We show the various saliency maps for 4 test images on Figure 5. The saliency maps appear to follow patterns that make the interpretation difficult. Here are a couple of examples that we can observe: (1) A latent unit is sensitive to a given image while insensitive to a similar image (e.g. Latent Unit 1 of the MNIST VAE is sensitive to Image 1 but not to Image 2). (2) The focus of a latent unit changes completely between two similar images (e.g. Latent Unit 4 of the dSprites VAE focuses on the interior of the square from Image 1 but only on the edges of the square from Image 2). (3) Several latent units focus on the same part of the image (e.g. Image 4 of MNIST and Image 3 of dSprites). Additional examples can be found in Appendix C.

Quantitative Analysis. If we increase the disentanglement of a VAE, does it imply that distinct latent units are going to focus on more distinct features? For the disantangled VAEs we consider, the strength of disentanglement increases with β . Hence, the above question can be formulated more precisely : does the correlation between the latent units saliency map decrease with β ? To answer this question, we make box-plots of the Pearson correlation coefficients for both VAE types (Beta and TC) and various values of β . Results can be observed in Figure 6. For MNIST, the Pearson correlation slightly decreases with β (Spearman $\rho = -0.15$). For dSprites, the Pearson correlation moderately increases with

⁶In this case, we *don't* use label-free feature importance b_i .

⁷We average the correlation over pairs of latent units.

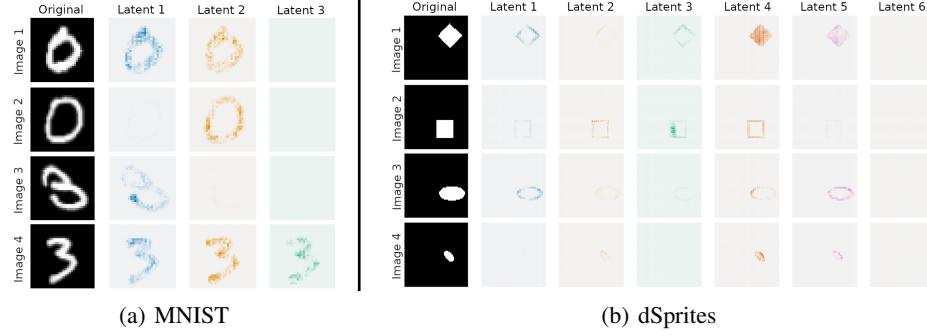


Figure 5. Saliency maps for each unit of the disentangled VAEs. The scale is constant for each image.

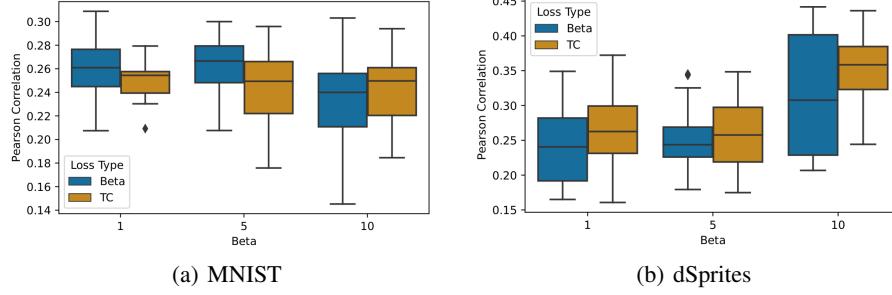


Figure 6. Pearson correlation between saliency maps for different values of β .

β (Spearman $\rho = .43$). This analysis shows that increasing β does not imply that latent units are going to pay attention to distinct part of the images. In fact, the opposite is true for dSprites. If this results is surprising at first, it can be understood by thinking about disentanglement. As aforementioned, increasing disentanglement encourages distinct latent units to pay attention to distinct generative factors. There is no guarantee that distinct generative factors are associated to distinct features. To illustrate, let us consider two generative factors of the dSprites dataset: the position of a shape and its scale. Clearly, these two generative factors can be identified by paying attention to the edges of the shape appearing on the image, as various latent units do in Figure 5(b). Whenever generative factors are not unambiguously associated to a subset of features, latent units can identify distinct generative factors by looking at similar features. In this case, increasing disentanglement of latent units does not necessarily make their saliency maps more decorrelated. We conclude two things: (1) If we want to identify the role of each latent unit with their saliency maps, disentanglement might not be the right approach. Perhaps it is possible to introduce priors on the saliency map to control the features the model pays attention to, like it was done by Erion et al. (2021) in a supervised setting. We leave this for future works. (2) Taking a weighted sum of these saliency maps (as done by our label-free wrappers) does not sacrifice any interpretable information specific to each unit.

5. Discussion

We introduced label-free explainability, a new framework to extend linear feature importance and example importance methods to the unsupervised setting. We showed that our framework guarantees crucial properties, such as completeness and invariance with respect to latent symmetries. We validated the framework on several datasets and verified that it permits to compare different representation spaces, both qualitatively and quantitatively. Finally, we challenged some common beliefs about the interpretability of β -VAEs.

Label-free explainability opens up many interesting avenues for future work. A first one is the extension of loss-based example importance methods to contrastive losses, hence completing Section 3. Another one is to compare the representation learned by different state of the art encoders with the approach from Section 4.2. A third one, as suggested in Section 4.3, is to regularize latent units to make their individual saliency maps more interpretable. Finally, a more radical way to interpret representation spaces is to use symbolic regression (Crabbe et al., 2020) to express latent units as closed form expressions of the input features.

Acknowledgements

The authors are grateful to Zhaozhi Qian, Alicia Curth and the 3 anonymous ICML reviewers for their useful comments on an earlier version of the manuscript. Jonathan Crabbe is funded by Aviva and Mihaela van der Schaar by the Office of Naval Research (ONR), NSF 172251.

References

- Adadi, A. and Berrada, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018. ISSN 21693536. doi: 10.1109/ACCESS.2018.2870052.
- Agarwal, N., Bullins, B., and Hazan, E. Second-Order Stochastic Optimization for Machine Learning in Linear Time. *Journal of Machine Learning Research*, 18:1–40, 2016. ISSN 15337928.
- Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., and Samek, W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7):e0130140, 2015. ISSN 1932-6203. doi: 10.1371/JOURNAL.PONE.0130140.
- Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Benetot, A., Tabik, S., Barbado, A., García, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020. ISSN 15662535. doi: 10.1016/j.inffus.2019.12.012.
- Brocki, L. and Chung, N. C. Concept saliency maps to visualize relevant features in deep generative models. *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*, pp. 1771–1778, 2019. doi: 10.1109/ICMLA.2019.000287.
- Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., Lerchner, A., and London, D. Understanding disentangling in β -VAE. *arXiv*, 2018.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. *37th International Conference on Machine Learning, ICML 2020*, pp. 1575–1585, 2020. doi: 10.48550/arxiv.2002.05709.
- Chen, T. Q., Li, X., Grosse, R., and Duvenaud, D. Isolating Sources of Disentanglement in Variational Autoencoders. *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 2018.
- Ching, T., Himmelstein, D. S., Beaulieu-Jones, B. K., Kalinin, A. A., Do, B. T., Way, G. P., Ferrero, E., Agapow, P.-M., Zietz, M., Hoffman, M. M., Xie, W., Rosen, G. L., Lengerich, B. J., Israeli, J., Lanchantin, J., Woloszynek, S., Carpenter, A. E., Shrikumar, A., Xu, J., Cofer, E. M., Lavender, C. A., Turaga, S. C., Alexandari, A. M., Lu, Z., Harris, D. J., DeCaprio, D., Qi, Y., Kundaje, A., Peng, Y., Wiley, L. K., Segler, M. H. S., Boca, S. M., Swamidass, S. J., Huang, A., Gitter, A., and Greene, C. S. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018. ISSN 1742-5689. doi: 10.1098/rsif.2017.0387.
- Cook, R. D. and Weisberg, S. *Residuals and influence in regression*. Chapman and Hall, New York, 1982.
- Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. Wiley, 2005. ISBN 9780471241959. doi: 10.1002/047174882X.
- Crabbé, J. and Van Der Schaar, M. Explaining time series predictions with dynamic masks. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2166–2177. PMLR, 2021.
- Crabbe, J., Zhang, Y., Zame, W. R., and van der Schaar, M. Learning outside the black-box: The pursuit of interpretable models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20*, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- Crabbé, J., Qian, Z., Imrie, F., and van der Schaar, M. Explaining Latent Representations with a Corpus of Examples. In *Advances in Neural Information Processing Systems*, 2021.
- Das, A. and Rad, P. Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey. *arXiv*, 2020.
- Davies, A., Veličković, P., Buesing, L., Blackwell, S., Zheng, D., Tomašev, N., Tanburn, R., Battaglia, P., Blundell, C., Juhász, A., Lackenby, M., Williamson, G., Hassabis, D., and Kohli, P. Advancing mathematics by guiding human intuition with AI. *Nature* 2021 600:7887, 600(7887):70–74, 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-04086-x.
- Eberle, O., Buttner, J., Krautli, F., Muller, K. R., Valleriani, M., and Montavon, G. Building and Interpreting Deep Similarity Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1149–1161, 2022. ISSN 19393539. doi: 10.1109/TPAMI.2020.3020738.
- Erion, G., Janizek, J. D., Sturmels, P., Lundberg, S. M., and Lee, S.-I. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature Machine Intelligence* 2021 3:7, 3(7):620–631, 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00343-w.

- Fong, R. C. and Vedaldi, A. Interpretable Explanations of Black Boxes by Meaningful Perturbation. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3449–3457, 2017. ISSN 15505499. doi: 10.1109/ICCV.2017.371.
- Ghorbani, A. and Zou, J. Data Shapley: Equitable Valuation of Data for Machine Learning. *36th International Conference on Machine Learning, ICML 2019*, 2019-June: 4053–4065, 2019.
- Ghorbani, A., Wexler, J., Zou, J., and Kim, B. Towards Automatic Concept-based Explanations. *Advances in Neural Information Processing Systems*, 32, 2019. ISSN 10495258.
- Ghorbani, A., Kim, M. P., and Zou, J. A Distributional Framework for Data Valuation. *arXiv*, 2020.
- Glorot, X., Bordes, A., and Bengio, Y. Deep Sparse Rectifier Neural Networks, 2011. ISSN 1938-7228.
- Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C. K., and Stanley, H. E. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23), 2000. doi: 10.1161/01.CIR.101.23.E215.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, 2015. ISSN 10636919. doi: 10.48550/arxiv.1512.03385.
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A., and Deepmind, G. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*, 2017.
- Holzinger, A., Goebel, R., Fong, R., Moon, T., Müller, K.-R., and Samek, W. *xxAI - Beyond Explainable Artificial Intelligence*, pp. 3–10. Springer International Publishing, 2022. ISBN 978-3-031-04083-2. doi: 10.1007/978-3-031-04083-2_1.
- Hooker, S., Erhan, D., Kindermans, P. J., and Kim, B. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Neural information processing systems foundation, 2019.
- Jing, L. and Tian, Y. Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4037–4058, 2021. doi: 10.1109/TPAMI.2020.2992393.
- Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with AlphaFold. *Nature* 2021 596:7873, 596 (7873):583–589, 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2.
- Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised Contrastive Learning. *Advances in Neural Information Processing Systems*, 2020-December, 2020. ISSN 10495258.
- Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., and Sayres, R. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). *35th International Conference on Machine Learning*, 6:4186–4195, 2017.
- Kingma, D. P. and Ba, J. L. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2014.
- Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2013.
- Koh, P. W. and Liang, P. Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning*, volume 34, pp. 2976–2987. PMLR, 2017.
- Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Al-sallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., and Reblitz-Richardson, O. Captum: A unified and generic model interpretability library for PyTorch. *arXiv*, 2020.
- Kong, Z. and Chaudhuri, K. Understanding Instance-based Interpretability of Variational Auto-Encoders. In *Advances in Neural Information Processing Systems*, 2021.
- Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.
- Le Meur, O. and Baccino, T. Methods for comparing scanpaths and saliency maps: Strengths and weaknesses. *Behavior Research Methods*, 45(1):251–266, 2013. ISSN 1554351X. doi: 10.3758/S13428-012-0226-9/TABLES/2.

- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998. ISSN 00189219. doi: 10.1109/5.726791.
- Lipton, Z. C. The Mythos of Model Interpretability. *Communications of the ACM*, 61(10):35–43, 2016.
- Lundberg, S. and Lee, S.-I. A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, pp. 4766–4775, 2017.
- Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. dSprites: Disentanglement testing Sprites dataset. <https://github.com/deepmind/dsprites-dataset/>, 2017.
- Montavon, G., Samek, W., and Müller, K. R. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018. ISSN 1051-2004. doi: 10.1016/J.DSP.2017.10.011.
- Ouerhani, N., Hügli, H., Müri, R., and Von Wartburg, R. Empirical Validation of the Saliency-based Model of Visual Attention. *Electronic Letters on Computer Vision and Image Analysis*, 3(1):13–23, 2003.
- Pandit, P., Sahraee-Ardakan, M., Rangan, S., Schniter, P., Fletcher, A. K., Pandit, P., Sahraee-Ardakan, M., and Fletcher, A. K. Inference with Deep Generative Priors in High Dimensions. *IEEE Journal on Selected Areas in Information Theory*, 1(1):336–347, 2019. doi: 10.1109/jsait.2020.2986321.
- Papernot, N. and McDaniel, P. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. *arXiv*, 2018.
- Pearlmutter, B. A. Fast Exact Multiplication by the Hessian. *Neural Computation*, 6:160, 1994.
- Pruthi, G., Liu, F., Sundararajan, M., and Kale, S. Estimating Training Data Influence by Tracing Gradient Descent. In *Advances in Neural Information Processing Systems*, pp. 19920–19930. Curran Associates, Inc., 2020.
- Rai, A. Explainable AI: from black box to glass box. *Journal of the Academy of Marketing Science 2019 48:1*, 48(1):137–141, 2019. ISSN 1552-7824. doi: 10.1007/S11747-019-00710-5.
- Rajkomar, A., Dean, J., and Kohane, I. Machine Learning in Medicine. *New England Journal of Medicine*, 380(14):1347–1358, 2019. ISSN 0028-4793. doi: 10.1056/NEJMRA1814259/SUPPL_FILE/NEJMRA1814259_DISCLOSURES.PDF.
- Rasmussen, C. E. Gaussian Processes in Machine Learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3176:63–71, 2003. ISSN 16113349. doi: 10.1007/978-3-540-28650-9_4.
- Ribeiro, M. T., Singh, S., and Guestrin, C. "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17, pp. 1135–1144. Association for Computing Machinery, 2016. ISBN 9781450342322. doi: 10.1145/2939672.2939778.
- Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence 2019 1:5*, 1(5):206–215, 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0048-x.
- Sarhan, M. H., Eslami, A., Navab, N., and Albarqouni, S. Learning Interpretable Disentangled Representations Using Adversarial VAEs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11795 LNCS:37–44, 2019. doi: 10.1007/978-3-030-33391-1_5.
- Shannon, C. E. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(4):623–656, 1948. ISSN 00058580. doi: 10.1002/j.1538-7305.1948.tb00917.x.
- Shrikumar, A., Greenside, P., and Kundaje, A. Learning Important Features Through Propagating Activation Differences. *34th International Conference on Machine Learning, ICML 2017*, 7:4844–4866, 2017.
- Simonyan, K., Vedaldi, A., and Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *2nd International Conference on Learning Representations, ICLR 2014 - Workshop Track Proceedings*, 2013.
- Sundararajan, M., Taly, A., and Yan, Q. Axiomatic Attribution for Deep Networks. *34th International Conference on Machine Learning, ICML 2017*, 7:5109–5118, 2017.
- Tjoa, E. and Guan, C. A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020. ISSN 2162-237X. doi: 10.1109/tnnls.2020.3027314.
- Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, 2018, 2018. ISSN 16875273. doi: 10.1155/2018/7068349.

Wachter, S., Mittelstadt, B., and Russell, C. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electronic Journal*, 2017. ISSN 1556-5068. doi: 10.2139/ssrn.3063289.

Yeh, C.-K., Kim, J. S., Yen, I. E. H., and Ravikumar, P. Representer Point Selection for Explaining Deep Neural Networks. *Advances in Neural Information Processing Systems*, 31:9291–9301, 2018.

Young, T., Hazarika, D., Poria, S., and Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018. ISSN 15566048. doi: 10.1109/MCI.2018.2840738.

A. Properties of the Label-Free Extensions

In this appendix, we prove the completeness property. Next, we motivate and prove the orthogonal invariance of our label-free extensions.

A.1. Completeness

Let us prove Proposition 2.3.

Proof. The proof is an immediate consequence of Definition 2.1 and the completeness property of the feature importance score a_i :

$$\begin{aligned} \sum_{i=1}^{d_X} b_i(\mathbf{f}, \mathbf{x}) &\stackrel{(1)}{=} \sum_{i=1}^{d_X} a_i(g_{\mathbf{x}}, \mathbf{x}) \\ &= g_{\mathbf{x}}(\mathbf{x}) - a_0, \end{aligned}$$

where we used the completeness property to obtain the second equality and $a_0 \in \mathbb{R}$ is the baseline for the importance score a_i . By noting that $g_{\mathbf{x}}(\mathbf{x}) = \langle \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}) \rangle_{\mathcal{H}} = \|\mathbf{f}(\mathbf{x})\|_{\mathcal{H}}^2$, we obtain the desired equality (3) with the identification $b_0 = a_0$. \square

A.2. Invariance with respect to latent symmetries

In Section 1, we described the ambiguity associated to the axes of the latent space \mathcal{H} . This line of reasoning can be made more formal with symmetries. Due to the fact that each axis of the latent space is not associated with a fixed and predetermined label, there exists many latent spaces that are equivalent to each other. For instance, if we swap two axes of the latent space by relabelling $h_1 \mapsto h_2$ and $h_2 \mapsto h_1$, we do not change the latent space structure. More generally, given an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, the set of transformations that leave the geometry of the latent space \mathcal{H} invariant is the set of orthogonal transformations.

Definition A.1 (Orthogonal Transformations). Let \mathcal{H} be a real vector space equipped with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}} : \mathcal{H}^2 \rightarrow \mathbb{R}$. An orthogonal transformation is a linear map $\mathbf{T} : \mathcal{H} \rightarrow \mathcal{H}$ such that for all $\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}$, we have:

$$\langle \mathbf{T}(\mathbf{h}_1), \mathbf{T}(\mathbf{h}_2) \rangle_{\mathcal{H}} = \langle \mathbf{h}_1, \mathbf{h}_2 \rangle_{\mathcal{H}}.$$

Remark A.2. In the case where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is the standard euclidean inner product $\langle \mathbf{h}_1, \mathbf{h}_2 \rangle_{\mathcal{H}} = \mathbf{h}_1^\top \mathbf{h}_2$, the orthogonal transformations are represented by orthogonal matrices in $O(d_H) = \{\mathbf{M} \in \mathbb{R}^{d_H \times d_H} \mid \mathbf{M}^\top \mathbf{M} = \mathbf{I}\}$. These transformations include rotations, axes permutations and mirror symmetries.

Of course, since these transformations leave the geometry of the latent space invariant, we would expect the same for the explanations. We verify that this is indeed the case for our label-free extension of feature importance.

Proposition A.3 (Label-Free Feature Importance Invariance). *The label-free importance scores $b_i(\cdot, \cdot)$, $i \in [d_X]$ are invariant with respect to orthogonal transformations in the latent space \mathcal{H} , $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. More formally, for all $\mathbf{f} \in \mathcal{A}(\mathcal{H}^{\mathcal{X}})$, $\mathbf{x} \in \mathcal{X}$ and $i \in [d_X]$:*

$$b_i(\mathbf{T} \circ \mathbf{f}, \mathbf{x}) = b_i(\mathbf{f}, \mathbf{x}),$$

where \mathbf{T} is any orthogonal transformation of the latent space \mathcal{H} , $\langle \cdot, \cdot \rangle_{\mathcal{H}}$.

Remark A.4. This property is a further motivation for the usage of an inner product in Definition 2.1.

Proof. This proposition is a trivial consequence of the inner product appearing in Definition 2.1. Let $g_{\mathbf{x}}^T$ be the auxiliary function associated to $\mathbf{T} \circ \mathbf{f}$ for some $\mathbf{x} \in \mathcal{X}$. We note that for all $\tilde{\mathbf{x}} \in \mathcal{X}$:

$$\begin{aligned} g_{\mathbf{x}}^T(\tilde{\mathbf{x}}) &\stackrel{(2)}{=} \langle \mathbf{T} \circ \mathbf{f}(\mathbf{x}), \mathbf{T} \circ \mathbf{f}(\tilde{\mathbf{x}}) \rangle_{\mathcal{H}} \\ &= \langle \mathbf{f}(\mathbf{x}), \mathbf{f}(\tilde{\mathbf{x}}) \rangle_{\mathcal{H}} \\ &\stackrel{(2)}{=} g_{\mathbf{x}}(\tilde{\mathbf{x}}), \end{aligned}$$

where we used the fact that \mathbf{T} is orthogonal in the second equality and $g_{\mathbf{x}}(\tilde{\mathbf{x}})$ is the auxiliary function associated to \mathbf{f} . Since this holds for any $\tilde{\mathbf{x}} \in \mathcal{X}$, we have that $g_{\mathbf{x}}^T = g_{\mathbf{x}}$ for all $\mathbf{x} \in \mathcal{X}$. This allows us to write:

$$\begin{aligned} b_i(\mathbf{T} \circ \mathbf{f}, \mathbf{x}) &\stackrel{(1)}{=} a_i(g_{\mathbf{x}}^T, \mathbf{x}) \\ &= a_i(g_{\mathbf{x}}, \mathbf{x}) \\ &\stackrel{(1)}{=} b_i(\mathbf{f}, \mathbf{x}) \end{aligned}$$

for all $\mathbf{x} \in \mathcal{X}$ and $i \in [d_X]$. This is the desired identity. \square

When it comes to example importance methods, the same guarantee holds for representation-based methods:

Proposition A.5 (Representation-Based Example Importance Invariance). *Let $\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}}$ be a latent space. The label-free importance scores $c^n(\cdot, \cdot), n \in [N]$ outputted by DKNN (5) are invariant with respect to orthogonal transformations of \mathcal{H} if they are defined with a kernel $\kappa : \mathcal{H}^2 \rightarrow \mathbb{R}^+$ that is invariant with respect to orthogonal transformations:*

$$\kappa(\mathbf{T}(\mathbf{h}_1), \mathbf{T}(\mathbf{h}_2)) = \kappa(\mathbf{h}_1, \mathbf{h}_2)$$

for all orthogonal transformation \mathbf{T} and $\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}$. Similarly, the importance scores outputted by SimplEx (6) are invariant with respect to orthogonal transformations of \mathcal{H} . In both cases, the invariance property can be written more formally: for all $\mathbf{f} \in \mathcal{A}(\mathcal{H}^{\mathcal{X}}), \mathbf{x} \in \mathcal{X}$ and $n \in [N]$:

$$c^n(\mathbf{T} \circ \mathbf{f}, \mathbf{x}) = c^n(\mathbf{f}, \mathbf{x})$$

Remark A.6. The invariance property for the kernel function is verified for kernels that involve the inner-product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ in their definition. This includes RBF, Matern and Polynomial Kernels (Rasmussen, 2003). Note that constant kernels trivially verify this property. Finally, replacing the kernel function by the inverse-distance in latent space (as it is done in our implementation) $\kappa(\mathbf{h}_1, \mathbf{h}_2) = \|\mathbf{h}_1 - \mathbf{h}_2\|_{\mathcal{H}}^{-1}$ also preserves the invariance property.

Proof. We start by noting that the latent space distance is invariant under orthogonal transformations. for all $\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}$:

$$\begin{aligned} \|\mathbf{T}(\mathbf{h}_1) - \mathbf{T}(\mathbf{h}_2)\|_{\mathcal{H}}^2 &= \|\mathbf{T}(\mathbf{h}_1 - \mathbf{h}_2)\|_{\mathcal{H}}^2 \\ &= \langle \mathbf{T}(\mathbf{h}_1 - \mathbf{h}_2), \mathbf{T}(\mathbf{h}_1 - \mathbf{h}_2) \rangle_{\mathcal{H}} \\ &= \langle \mathbf{h}_1 - \mathbf{h}_2, \mathbf{h}_1 - \mathbf{h}_2 \rangle_{\mathcal{H}} \\ &= \|\mathbf{h}_1 - \mathbf{h}_2\|_{\mathcal{H}}^2, \end{aligned}$$

where we successively used the linearity and orthogonality of \mathbf{T} . Note that this equation is equivalent to $\|\mathbf{T}(\mathbf{h}_1) - \mathbf{T}(\mathbf{h}_2)\|_{\mathcal{H}} = \|\mathbf{h}_1 - \mathbf{h}_2\|_{\mathcal{H}}$ as both norms are positive. Since the latent KNNs in (5) are computed with this latent space distance, we deduce their invariance under orthogonal transformations. By combining this to the invariance of the kernel, we obtain the desired invariance for the DKNN importance scores (5). For all $\mathbf{x} \in \mathcal{X}, \mathbf{f} \in \mathcal{A}(\mathcal{H}^{\mathcal{X}})$ and orthogonal transformation \mathbf{T} :

$$\begin{aligned} c_{\text{DKNN}}^n(\mathbf{T} \circ \mathbf{f}, \mathbf{x}) &\stackrel{(5)}{=} \mathbf{1}[n \in \text{KNN}(\mathbf{x})] \cdot \kappa[\mathbf{T} \circ \mathbf{f}(\mathbf{x}^n), \mathbf{T} \circ \mathbf{f}(\mathbf{x})] \\ &= \mathbf{1}[n \in \text{KNN}(\mathbf{x})] \cdot \kappa[\mathbf{f}(\mathbf{x}^n), \mathbf{f}(\mathbf{x})] \\ &\stackrel{(5)}{=} c_{\text{DKNN}}^n(\mathbf{f}, \mathbf{x}), \end{aligned}$$

where we have used the invariance property to obtain the second equality. We can proceed similarly for SimplEx (6):

$$\begin{aligned} c_{\text{SimplEx}}^n(\mathbf{T} \circ \mathbf{f}, \mathbf{x}) &\stackrel{(6)}{=} \arg \min_{\boldsymbol{\lambda} \in [0, 1]^N} \left\| \mathbf{T} \circ \mathbf{f}(\mathbf{x}) - \sum_{n=1}^N \lambda^n \mathbf{T} \circ \mathbf{f}(\mathbf{x}^n) \right\|_{\mathcal{H}} \\ &= \arg \min_{\boldsymbol{\lambda} \in [0, 1]^N} \left\| \mathbf{T} \left[\mathbf{f}(\mathbf{x}) - \sum_{n=1}^N \lambda^n \mathbf{f}(\mathbf{x}^n) \right] \right\|_{\mathcal{H}} \\ &= \arg \min_{\boldsymbol{\lambda} \in [0, 1]^N} \left\| \mathbf{f}(\mathbf{x}) - \sum_{n=1}^N \lambda^n \mathbf{f}(\mathbf{x}^n) \right\|_{\mathcal{H}} \\ &\stackrel{(6)}{=} c_{\text{SimplEx}}^n(\mathbf{f}, \mathbf{x}), \end{aligned}$$

where we have successively used the linearity and orthogonality of \mathbf{T} . Those are the desired identities. \square

The only label-free extension that we have not yet discussed are the loss-based example importance methods from Section 3.1. Unfortunately, due to the fact that the black-box \mathbf{f} is only a component required in the evaluation of the loss L , it is not possible to provide a general guarantee like in the previous examples. If we take the example of the autoencoder $\mathbf{f}_d \circ \mathbf{f}_e$, we note that applying an orthogonal transformation $\mathbf{f}_e \mapsto \mathbf{T} \circ \mathbf{f}_e$ to the encoder leaves the autoencoder invariant only if this transformation is undone by the decoder $\mathbf{f}_d \mapsto \mathbf{f}_d \circ \mathbf{T}^{-1}$. Unlike the other methods, the invariance of loss-based example importance scores therefore requires restrictive assumptions. If invariance of the explanations under orthogonal transformations is required, this might be an argument in favour of representation-based methods.

B. Implementation Details

In this appendix, we detail the implementation of our label-free extensions.

B.1. Label-Free Feature Importance

The label-free feature importance methods used in our experiments are described in Table 3:

Table 3. Feature Importance Methods.

Method	Ref.	Linearity	Completeness	Label-Free Expression
Saliency	(Simonyan et al., 2013)	✓	✗	$b_i(\mathbf{f}, \mathbf{x}) = \frac{\partial g_{\mathbf{x}}}{\partial x_i}(\mathbf{x})$
Integrated Gradients	(Sundararajan et al., 2017)	✓	✓	$b_i(\mathbf{f}, \mathbf{x}) = (x_i - \bar{x}_i) \int_0^1 \frac{\partial g_{\mathbf{x}}}{\partial x_i}(\mathbf{x} + \tau(\mathbf{x} - \bar{\mathbf{x}})) d\tau$
Gradient Shap	(Lundberg & Lee, 2017)	✓	✓	$b_i(\mathbf{f}, \mathbf{x}) = (x_i - \bar{x}_i) \mathbb{E}_{\epsilon, U} \left[\frac{\partial g_{\mathbf{x}}}{\partial x_i}(\mathbf{x} + \epsilon + U(\mathbf{x} - \bar{\mathbf{x}})) \right]$
DeepLift	(Shrikumar et al., 2017)	✓	✓	$b_i(\mathbf{f}, \mathbf{x}) = C_{\Delta x_i \Delta g_{\mathbf{x}}}$

where $\bar{\mathbf{x}} \in \mathcal{X}$ is a baseline input, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $U \sim \text{Uniform}(0, 1)$ and $C_{\Delta x_i \Delta g_{\mathbf{x}}}$ is used by propagating the Deeplift rules along the computational graph of $g_{\mathbf{x}}$. Note that, in each case, partial derivatives are computed with respect to the argument of $g_{\mathbf{x}}$ only (hence we do not consider derivatives of the form $g_{\frac{\partial g_{\mathbf{x}}}{\partial x_i}}$). We use the Captum (Kokhlikyan et al., 2020) implementation of each method.

To extend this implementation to the label-free setting, it is necessary to define an auxiliary function $g_{\mathbf{x}}$ associated to the vectorial black-box function \mathbf{f} for each testing example $\mathbf{x} \in \mathcal{D}_{\text{test}}$. With libraries such as Pytorch, it is possible to define an auxiliary function as a wrapper around the module that represents \mathbf{f} . This allows us to compute the importance scores with a single batch call of the original feature importance method, as described in Algorithm 1.

Algorithm 1 Label-Free Feature Importance

Input: Batch $\mathbf{X} \in \mathcal{X}^B$ of size B , Black-box $\mathbf{f} : \mathcal{X} \rightarrow \mathcal{H}$, Feature importance method $a_i(\cdot, \cdot) : \mathcal{A}(\mathcal{H}^{\mathcal{X}}) \times \mathcal{X}^B \rightarrow \mathbb{R}^B$

Output: Batch label-free feature importance $b_i(\mathbf{f}, \mathbf{X})$.

Define batch auxiliary function $g_{\mathbf{X}}$ as a wrapper around \mathbf{f} according to (2).

Compute the label-free importance scores for the batch $b_i(\mathbf{f}, \mathbf{X}) \leftarrow a_i(g_{\mathbf{X}}, \mathbf{X})$.

B.2. Label-Free Example Importance

We detail the label-free example importance methods used in our experiments in Table 4:

Table 4. Example Importance Methods.

Method Type	Method	Ref.	Label-Free Expression
Loss-Based	Influence Functions	(Koh & Liang, 2017)	$c^n(\mathbf{f}, \mathbf{x}) = \frac{1}{N} \langle \nabla_{\theta_r} L(\mathbf{x}, \theta_*), \mathbf{H}_{\theta_r}^{-1} \nabla_{\theta_r} L(\mathbf{x}^n, \theta_*) \rangle_{\Theta_r}$
	TracIn	(Pruthi et al., 2020)	$c^n(\mathbf{f}, \mathbf{x}) = \sum_{t=1}^T \eta_t \langle \nabla_{\theta_r} L(\mathbf{x}, \theta_t), \nabla_{\theta_r} L(\mathbf{x}^n, \theta_t) \rangle_{\Theta_r}$
Representation-Based	Deep K-Nearest Neighbours	(Papernot & McDaniel, 2018)	$c^n(\mathbf{f}, \mathbf{x}) = \mathbf{1}[n \in \text{KNN}(\mathbf{x})] \cdot \kappa[\mathbf{f}(\mathbf{x}^n), \mathbf{f}(\mathbf{x})]$
	SimplEx	(Crabbé et al., 2021)	$c^n(\mathbf{f}, \mathbf{x}) = \arg \min_{\lambda \in [0,1]^N} \left\ \mathbf{f}(\mathbf{x}) - \sum_{n=1}^N \lambda^n \mathbf{f}(\mathbf{x}^n) \right\ _{\mathcal{H}}$ s.t. $\sum_{n=1}^N \lambda^n = 1$

where θ_r are the parameters of the black-box \mathbf{f} . Our implementation closely follows the above references with some subtle differences. For completeness, we detail the algorithm used for each method. We start with Influence Functions in Algorithm 2.

Algorithm 2 Label-Free Influence Functions

Input: Test input $\mathbf{x} \in \mathcal{X}$, Black-box $\mathbf{f} : \mathcal{X} \rightarrow \mathcal{H}$, Optimized parameters $\theta_* \in \Theta$, Relevant black-box parameters $\theta_r \in \Theta_r$, Loss function $L : \mathcal{X} \times \Theta \rightarrow \mathbb{R}$ used to train the black-box, Training set $\mathcal{D}_{\text{train}} = \{\mathbf{x}^n \mid n \in [N]\}$, Number of samples $S \in \mathbb{N}^*$, Number of recursions $R \in \mathbb{N}^*$.

Output: Label-free influence functions $c^n(\mathbf{f}, \mathbf{x})$.

Initialize $\mathbf{v}_0 \leftarrow \nabla_{\theta_r} L(\mathbf{x}, \theta_*)$.

Initialize $\mathbf{v} \leftarrow \mathbf{v}_0$.

for recursion in $[R]$ **do**

 Sample S training points $\mathbf{x}^{n_1}, \dots, \mathbf{x}^{n_S}$ from the training set $\mathcal{D}_{\text{train}}$.

 Make a Monte-Carlo estimation of the training loss Hessian $\mathbf{H} \leftarrow \frac{1}{S} \nabla_{\theta_r}^2 \sum_{s=1}^S L(\mathbf{x}^{n_s}, \theta_*)$.

 Update the estimate for the inverse Hessian-vector product $\mathbf{v} \leftarrow \mathbf{v}_0 + (\mathbf{I} - \mathbf{H})\mathbf{v}$.

end for

Compute the influence function $c^n(\mathbf{f}, \mathbf{x}) \leftarrow \mathbf{v}^\top \nabla_{\theta_r} L(\mathbf{x}^n, \theta_*)$

This implementation follows the original implementation by Koh & Liang (2017) that leverages the literature on second-order approximation techniques (Pearlmutter, 1994; Agarwal et al., 2016). Note that Monte-Carlo estimations of the Hessian quickly become expensive when the number of model parameters grows. Due to our limited computing infrastructure, we limit the number of recursions to $R = 100$. Furthermore, we only compute influence functions for smaller subsets of the training and testing set. The label-free version of TracIn is described in Algorithm 3. In our implementation, we create a checkpoint after each interval of 10 epochs during training.

When it comes to DKNN, the formula (5) can be computed explicitly without following a particular procedure. In our implementation, we replaced the kernel function by an inverse distance $\kappa(\mathbf{h}_1, \mathbf{h}_2) = \|\mathbf{h}_1 - \mathbf{h}_2\|_{\mathcal{H}}^{-1}$. Further, to make it more fair with other baselines that assign a score to each examples (and not only to $K \in \mathbb{N}^*$ examples), we removed the indicator in (5): $\mathbf{1}[n \in \text{KNN}(\mathbf{x})] \mapsto 1$. In this way, the K most important examples always correspond to the K nearest neighbours. Finally, the label-free version of SimplEx is described in Algorithm 4.

Note that our implementation of SimplEx is identical to the original one. It relies on an Adam (Kingma & Ba, 2014) with the default Pytorch parameters (learning rate = .001, $\beta_1 = .9$, $\beta_2 = .999$, $\epsilon = 10^{-8}$, weight decay = 0).

Algorithm 3 Label-Free TracIn

Input: Test input $\mathbf{x} \in \mathcal{X}$, Black-box $\mathbf{f} : \mathcal{X} \rightarrow \mathcal{H}$, Checkpoint parameters $\{\boldsymbol{\theta}_t \in \Theta \mid t \in [T]\}$, Relevant black-box parameters $\boldsymbol{\theta}_r \in \Theta_r$, Loss function $L : \mathcal{X} \times \Theta \rightarrow \mathbb{R}$ used to train the black-box, Training set $\mathcal{D}_{\text{train}} = \{\mathbf{x}^n \mid n \in [N]\}$, Checkpoint learning rates $\{\eta_t \in \mathbb{R} \mid t \in [T]\}$.

Output: Label-free TraceIn scores $c^n(\mathbf{f}, \mathbf{x})$.

Initialize $c \leftarrow 0$.

for t in $[T]$ **do**

Update the estimate $c \leftarrow c + \eta_t \nabla_{\boldsymbol{\theta}_r}^\top L(\mathbf{x}^n, \boldsymbol{\theta}_t) \nabla_{\boldsymbol{\theta}_r} L(\mathbf{x}, \boldsymbol{\theta}_t)$.

end for

Return the score $c^n(\mathbf{f}, \mathbf{x}) \leftarrow c$

Algorithm 4 SimplEx

Input: Test input $\mathbf{x} \in \mathcal{X}$, Black-box $\mathbf{f} : \mathcal{X} \rightarrow \mathcal{H}$, Training set $\mathcal{D}_{\text{train}} = \{\mathbf{x}^n \mid n \in [N]\}$, Number of epochs $E \in \mathbb{N}^*$.

Output: SimplEx scores $c^n(\mathbf{f}, \mathbf{x})$.

Initialize weights $(w^n)_{n=1}^N \leftarrow \mathbf{0}$.

for epoch in $[E]$ **do**

Normalize weights $(\lambda^n)_{n=1}^N \leftarrow \text{Softmax}[(w^n)_{n=1}^N]$

Estimate Error $\mathcal{L} = \left\| \mathbf{f}(\mathbf{x}) - \sum_{n=1}^N \lambda^n \mathbf{f}(\mathbf{x}^n) \right\|_{\mathcal{H}}$.

Update weights with Adam $(w^n)_{n=1}^N \leftarrow \text{Adam Step}(\mathcal{L})$.

end for

Return the score $c^n(\mathbf{f}, \mathbf{x}) \leftarrow \lambda^n$

In this section, we have presented many label-free implementations of feature and example importance methods. For some types of explanations, like counterfactual explanations (Wachter et al., 2017), the label plays an essential role. Hence, it does not always make sense to extend an explanation to the label-free setting.

C. Experiments Details

In this appendix, we provide further details to support the experiments described in Section 4. All our experiments have been performed on a machine with Intel(R) Core(TM) i5-8600K CPU @ 3.60GHz [6 cores] and Nvidia GeForce RTX 2080 Ti GPU. Our implementation is done with Python 3.8 and Pytorch 1.10.0.

C.1. Consistency Checks

We provide some details for the experiments in Section 4.1.

ECG5000 dataset. The dataset \mathcal{D} contains 5000 univariate time series⁸ $(x_t)_{t=1}^T \in \mathcal{X}^T$ describing the heartbeat of a patient. Each time series describes a single heartbeat with a resolution of $T = 140$ time steps. For the sake of notation, we will represent univariate time series by vectors: $\mathbf{x} = (x_t)_{t=1}^T$. Each time series comes with a label $y \in \{0, 1\}$ indicating if the heartbeat is normal ($y = 0$) or abnormal ($y = 1$). Since it is laborious to manually annotate 5000 time series, those labels were generated automatically. Of course, those labels are not going to be used in training our model. We only use the labels to perform consistency checks once the model has been trained.

⁸Note that t is used to index the time series steps, as opposed to model checkpoints in Section 3.1.

MNIST autoencoder. We use a denoising autoencoder $\mathbf{f}_d \circ \mathbf{f}_e : \mathcal{X} \rightarrow \mathcal{X}$ that consists in an encoder $\mathbf{f}_e : \mathcal{X} \rightarrow \mathcal{H}$ and a decoder $\mathbf{f}_d : \mathcal{H} \rightarrow \mathcal{X}$ with $d_X = 28^2$, $d_H = 4$. The architecture for the autoencoder is described in Table 5. We corrupt each training image $\mathbf{x} \in \mathcal{D}_{\text{train}}$ with random noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}/3)$ where \mathbf{I} is the identity matrix on \mathcal{X} . The autoencoder is trained to minimize the denoising loss $L_{\text{den}}(\mathbf{x}) = \mathbb{E}_\epsilon [\mathbf{x} - \mathbf{f}_d \circ \mathbf{f}_e(\mathbf{x} + \epsilon)]^2$. The autoencoder is trained for 100 epochs with patience 10 by using Pytorch’s Adam with hyperparameters: learning rate = .001, $\beta_1 = .9$, $\beta_2 = .999$, $\epsilon = 10^{-8}$, weight decay = 10^{-5} . The testing set is sometimes used for early stopping. This is acceptable because assessing the generalization of the learned model is *not* the focus of our paper. Rather, we only use the test set to study the explanations of the learned model.

ECG5000 autoencoder. ▶ **Feature importance:** We train a reconstruction autoencoder $\mathbf{f}_d \circ \mathbf{f}_e : \mathcal{X}^T \rightarrow \mathcal{X}^T$ that consists in an encoder $\mathbf{f}_e : \mathcal{X}^T \rightarrow \mathcal{H}$ and a decoder $\mathbf{f}_d : \mathcal{H} \rightarrow \mathcal{X}^T$ with $d_H = 64$. This model is trained with a training set $\mathcal{D}_{\text{train}}$ of 2919 time series from \mathcal{D} that correspond to *normal* heartbeats: $y = 0$. In this way, the testing set $\mathcal{D}_{\text{test}} = \mathcal{D} \setminus \mathcal{D}_{\text{train}}$ contains only abnormal heartbeats: $y = 1$. The model is trained to minimize the reconstruction loss $L_{\text{rec}}(\mathbf{x}) = \sum_{t=1}^T |x_t - [\mathbf{f}_d \circ \mathbf{f}_e(\mathbf{x})]_t|$. The autoencoder is trained for 150 epochs with patience 10 by using Pytorch’s Adam with hyperparameters: learning rate = .001, $\beta_1 = .9$, $\beta_2 = .999$, $\epsilon = 10^{-8}$, weight decay = 0. Its detailed architecture is presented in Table 6. ▶ **Example importance:** We use the autoencoder described in Table 6 with $d_H = 32$. The whole training process is identical to the one for feature importance with one difference: $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ are now obtained with a random split of \mathcal{D} (80% – 20%). This means that both subsets contain normal and abnormal heartbeats.

CIFAR-10 SimCLR. We use a SimCLR network $\mathbf{f}_p \circ \mathbf{f}_e : \mathcal{X} \rightarrow \mathcal{P}$ that consists in a Resnet-18 encoder $\mathbf{f}_e : \mathcal{X} \rightarrow \mathcal{H}$ and a multilayer perceptron projection head $\mathbf{f}_p : \mathcal{H} \rightarrow \mathcal{P}$ with $d_X = 3 \cdot 32^2$, $d_H = 512$. The architecture for the SimCLR network is described in Table 7. We use SimCLR’s contrastive loss to train the model (Chen et al., 2020). The model is trained for 100 epochs by using Pytorch’s SGD with hyperparameters: learning rate = .6, momentum = .9, weight decay = 10^{-6} .

Table 5. MNIST Autoencoder Architecture.

Component	Layer Type	Hyperparameters	Activation Function
Encoder	Conv2d	Input Channels:1 ; Output Channels:8 ; Kernel Size:3 ; Stride:2 ; Padding:1	ReLU
	Conv2d	Input Channels:8 ; Output Channels:16 ; Kernel Size:3 ; Stride:2 ; Padding:1	ReLU
	BatchNorm	Input Channels:16	ReLU
	Conv2d	Input Channels:16 ; Output Channels:32 ; Kernel Size:3 ; Stride:2 ; Padding:0	ReLU
	Flatten	Start Dimension:1	
	Linear	Input Dimension: 288 ; Output Dimension: 128	ReLU
	Linear	Input Dimension: 128 ; Output Dimension: 4	
Decoder	Linear	Input Dimension: 4 ; Output Dimension: 128	ReLU
	Linear	Input Dimension: 128 ; Output Dimension: 288	ReLU
	Unflatten	Dimension:1 ; Unflatten Size:(32, 3, 3)	
	ConvTranspose2d	Input Channels:32 ; Output Channels:16 ; Kernel Size:3 ; Stride:2 ; Output Padding:0	
	BatchNorm	Input Channels:16	ReLU
	ConvTranspose2d	Input Channels:16 ; Output Channels:8 ; Kernel Size:3 ; Stride:2 ; Output Padding:1	
	BatchNorm	Input Channels:8	ReLU
	ConvTranspose2d	Input Channels:8 ; Output Channels:1 ; Kernel Size:3 ; Stride:2 ; Output Padding:1	Sigmoid

Feature Importance. As a baseline for MNIST feature importance, we use a black image $\bar{\mathbf{x}} = \mathbf{0}$. For ECG5000, we use the average normal heartbeat as a baseline: $\bar{\mathbf{x}} = \sum_{\mathbf{x} \in \mathcal{D}_{\text{train}}} \mathbf{x} / |\mathcal{D}_{\text{train}}|$. For CIFAR-10, we use a blurred version of the image \mathbf{x} we wish to explain as a baseline: $\bar{\mathbf{x}} = \mathbf{G}_\sigma \otimes \mathbf{x}$, where \mathbf{G}_σ is a Gaussian blur with kernel of size 21 with width $\sigma = 5$ and \otimes denotes the convolution operator.

Table 6. ECG5000 Autoencoder Architecture.

Component	Layer Type	Hyperparameters	Activation Function
Encoder	LSTM	Input Size:1 ; Hidden Size: $2 \cdot d_H$	
	LSTM	Input Size: $2 \cdot d_H$; Hidden Size: d_H	
Representation	The latent representation \mathbf{h} is given by the output of the second LSTM at the last time step. This representation is copied at each time step to be a valid input for the first decoder LSTM.		
Decoder	LSTM	Input Size: d_H ; Hidden Size: d_H	
	LSTM	Input Size: d_H ; Hidden Size: $2 \cdot d_H$	
	Linear	Input Dimension: $2 \cdot d_H$; Output Dimension: 1	

Table 7. CIFAR-10 ResNet-18 Architecture.

Component	Layer Type	Hyperparameters	Activation Function
Encoder	ResNet-18	Similar to Appendix B.9 in (Chen et al., 2020).	
Projection Head	Linear	Input Dimension: 512 ; Output Dimension: 2048	ReLU
	Linear	Input Dimension: 2048 ; Output Dimension: 128	

ROAR Test. We perform the ROAR test (Hooker et al., 2019) for our label-free feature importance methods. The setup is similar to Section 4.1 except that the most important pixels are removed and a new autoencoder is fitted on the ablated dataset. We report the results in Figure 7. Again, the label-free feature importance methods discover pixels that increase the test loss more substantially than random pixels when removed, which supports the results from the main paper.

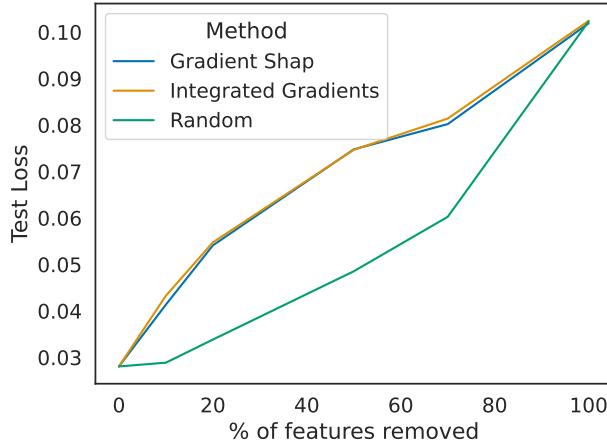


Figure 7. ROAR test for MNIST denoising autoencoder.

C.2. Pretext Task Sensitivity

We provide some details for the experiments in Section 4.2.

Models. All the autoencoders have the architecture described in Table 5. The classifier has all the layers from the encoder in Table 5 with an extra linear layer (Input Dimension:4 ; Output Dimension:10 ; Activation:Softmax) that converts the latent representations to class probabilities. Hence, it can be written as $f_l \circ f_e$, where $f_l : \mathcal{X} \rightarrow \mathcal{Y}$ is an extra linear layer followed by a softmax activation. The reconstruction autoencoder is trained to minimize the reconstruction loss $L_{\text{rec}}(\mathbf{x}) = [\mathbf{x} - f_d \circ f_e(\mathbf{x})]^2$. The inpainting autoencoder is trained to minimize the inptaiting loss

$L_{\text{in}}(\mathbf{x}) = \mathbb{E}_{\mathbf{M}}[\mathbf{x} - \mathbf{f}_d \circ \mathbf{f}_e(\mathbf{M} \odot \mathbf{x})]^2$, where \mathbf{M} is a random mask with $M_i \sim \text{Bernoulli}(0.8)$ for all $i \in [d_X]$. The classifier is trained to minimize the cross-entropy loss $L_{CE}(\mathbf{x}, \mathbf{y}) = -\mathbf{y} \odot \log [\mathbf{f}_l \circ \mathbf{f}_e(\mathbf{x})]$, where \mathbf{y} is the one-hot encoded label associated to the training example \mathbf{x} . All the models are trained to minimize their objective for 100 epochs with patience 10 by using Pytorch’s Adam with hyperparameters: learning rate = .001, $\beta_1 = .9$, $\beta_2 = .999$, $\epsilon = 10^{-8}$, weight decay = 10^{-5} .

Feature Importance. As a baseline for the feature importance methods, we use a black image $\bar{\mathbf{x}} = \mathbf{0}$.

Metrics. We use the Pearson coefficient to measure the correlation between two importance scores given a random test example and a random feature/training example. In our experiment, we compute the Pearson correlation between the label-free feature importance scores b_i outputted by two different encoder $\mathbf{f}_{e1}, \mathbf{f}_{e2} : \mathcal{X} \rightarrow \mathcal{H}$:

$$r_{\text{feat.}}(\mathbf{f}_{e1}, \mathbf{f}_{e2}) = \frac{\text{cov}_{\mathbf{X}, I}[b_I(\mathbf{f}_{e1}, \mathbf{X}), b_I(\mathbf{f}_{e2}, \mathbf{X})]}{\sigma_{\mathbf{X}, I}[b_I(\mathbf{f}_{e1}, \mathbf{X})] \sigma_{\mathbf{X}, I}[b_I(\mathbf{f}_{e2}, \mathbf{X})]}$$

$$\mathbf{X} \sim \text{Empirical Distribution}(\mathcal{D}_{\text{test}}), I \sim \text{Uniform}([d_X]),$$

where cov denotes the covariance between two random variables and σ denotes the standard deviation of a random variable. Similarly, for label-free example importance scores c^n :

$$r_{\text{ex.}}(\mathbf{f}_{e1}, \mathbf{f}_{e2}) = \frac{\text{cov}_{\mathbf{X}, I}[c^I(\mathbf{f}_{e1}, \mathbf{X}), c^I(\mathbf{f}_{e2}, \mathbf{X})]}{\sigma_{\mathbf{X}, I}[c^I(\mathbf{f}_{e1}, \mathbf{X})] \sigma_{\mathbf{X}, I}[c^I(\mathbf{f}_{e2}, \mathbf{X})]}$$

$$\mathbf{X} \sim \text{Empirical Distribution}(\mathcal{D}_{\text{test}}), I \sim \text{Uniform}(\mathcal{J}),$$

where $\mathcal{J} \subset [N]$ is the indices of the sampled training examples for which the example importance is computed. Those two Pearson correlation coefficients are the one that we report in Tables 1 and 2.

Supplementary Examples. To check that the qualitative analysis from Section 4.2 extends beyond the examples showed in the main paper, the reader can refer to Figures 9 and 10.

C.3. Challenging our assumptions with disentangled VAEs

We provide some details for the experiments in Section 4.3.

Model. The architecture of the MNIST VAE is described in Table 8 and those of the dSprites VAE is described in Table 9. Both of these architectures are reproductions of the VAEs from Burgess et al. (2018). The β -VAE is trained to minimize the objective $L_\beta(\mathbf{x}, \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{h} | \mathbf{x})}[\log p_\theta(\mathbf{x} | \mathbf{h})] - \beta D_{\text{KL}}[q_\phi(\mathbf{h} | \mathbf{x}) || p(\mathbf{h})]$, where $q_\phi(\mathbf{h} | \mathbf{x})$ is the distribution underlying the reparametrized encoder output, $p_\theta(\mathbf{x} | \mathbf{h})$ is the distribution underlying the decoder output, $p(\mathbf{h})$ is the density associated to isotropic unit Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$ underlying ϵ and D_{KL} is the KL-divergence. The objective of the TC-VAE is the same as in (Chen et al., 2018). We refer the reader to the original paper for the details. All the VAEs are trained to minimize their objective for 100 epochs with patience 10 by using Pytorch’s Adam with hyperparameters: learning rate = .001, $\beta_1 = .9$, $\beta_2 = .999$, $\epsilon = 10^{-8}$, weight decay = 10^{-5} .

Feature Importance. As a baseline for the feature importance methods, we use a black image $\bar{\mathbf{x}} = \mathbf{0}$.

Metrics. We use the Pearson coefficient to measure the correlation between two importance scores given a random test example and a random feature. In this case, one correlation coefficient can be computed for each couple $(i, j) \in [d_H]^2$ of latent units:

$$r_{ij} = \frac{\text{cov}_{\mathbf{X}, I}[a_I(\mu_i, \mathbf{X}), a_I(\mu_j, \mathbf{X})]}{\sigma_{\mathbf{X}, I}[a_I(\mu_i, \mathbf{X})] \sigma_{\mathbf{X}, I}[a_I(\mu_j, \mathbf{X})]}$$

$$\mathbf{X} \sim \text{Empirical Distribution}(\mathcal{D}_{\text{test}}), I \sim \text{Uniform}([d_X]),$$

where μ_i is the i -th component of the expected representation computed by the encoder for all $i \in [d_H]$. To have an overall measure of correlation between the VAE units, we sum over all pairs of distinct latent units:

$$r = \frac{1}{d_H(d_H - 1)} \sum_{i=1, i \neq j}^{d_H} \sum_{j=1}^{d_H} r_{ij}$$

Table 8. MNIST Variational Autoencoder Architecture.

Component	Layer Type	Hyperparameters	Activation Function
Encoder	Conv2d	Input Channels:1 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Padding:1	ReLU
	Conv2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Padding:1	ReLU
	Conv2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Padding:1	ReLU
	Flatten	Start Dimension:1	
	Linear	Input Dimension: 512 ; Output Dimension: 256	ReLU
	Linear	Input Dimension: 256 ; Output Dimension: 256	ReLU
	Linear	Input Dimension: 256 ; Output Dimension: 6	ReLU
Reparametrization Trick		The output of the encoder contains μ and $\log \sigma$. The latent representation is then generated via $\mathbf{h} = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$	
Decoder	Linear	Input Dimension: 3 ; Output Dimension: 256	ReLU
	Linear	Input Dimension: 256 ; Output Dimension: 256	ReLU
	Linear	Input Dimension: 256 ; Output Dimension: 512	ReLU
	Unflatten	Dimension:1 ; Unflatten Size:(32, 4, 4)	
	ConvTranspose2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Output Padding:1	ReLU
	ConvTranspose2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Output Padding:1	ReLU
	ConvTranspose2d	Input Channels:32 ; Output Channels:1 ; Kernel Size:4 ; Stride:2 ; Output Padding:1	Sigmoid

This averaged correlation coefficient is the one that we report in Figure 6. In our quantitative analysis, we also report the Spearman rank correlation between β and r . Concretely, this is done by performing the experiment $M \in \mathbb{N}^*$ times for different values β_1, \dots, β_M of β . We then measure the correlation coefficients r_1, \dots, r_M associated to each experiment. The Spearman rank correlation coefficient can be computed from this data:

$$\rho = \frac{\text{cov}_{M_1, M_2} [\text{rank}(r_{M_1}), \text{rank}(\beta_{M_2})]}{\sigma_{M_1}[\text{rank}(r_{M_1})] \sigma_{M_2}[\text{rank}(\beta_{M_2})]} \\ M_1, M_2 \sim \text{Uniform}([M]).$$

This coefficient ρ ranges from -1 to 1 , where $\rho = -1$ corresponds to a perfect monotonically decreasing relation, $\rho = 0$ corresponds to the absence of monotonic relation and $\rho = 1$ corresponds to a perfect monotonically increasing relation.

Entropy. To complete our quantitative analysis of the VAEs, we introduce a new metric called *entropy*. The purpose of this metric is to measure how the saliency for each feature is distributed across the different latent units. In particular, we would like to be able to distinguish the case where all latent units are sensitive to a feature and the case where only one latent unit is sensitive to a feature. As we have done previously, we can compute the importance score $a_i(\mu_j, \mathbf{x})$ of each feature x_i from $\mathbf{x} \in \mathcal{X}$ for a latent unit $j \in [d_H]$. For each latent unit $j \in [d_H]$, we define the proportion of attribution as

$$p_j(i, \mathbf{x}) = \frac{|a_i(\mu_j, \mathbf{x})|}{\sum_{k=1}^{d_H} |a_i(\mu_k, \mathbf{x})|}.$$

This corresponds to the fraction of the importance score attributed to j for feature i and example \mathbf{x} . Note that this quantity is well defined if at least one of the $a_i(\mu_k, \mathbf{x})$, $k \in [d_H]$ is non-vanishing. Hence, we only consider the features $i \in [d_X]$ that are salient for at least one latent unit. We can easily check that $\sum_{j=1}^{d_H} p_j(i, \mathbf{x}) = 1$ by construction. This means that the proportions of attributions can be interpreted as probabilities of saliency. This allows us to define an entropy that summarizes the distribution over the latent units:

$$S(i, \mathbf{x}) = - \sum_{j=1}^{d_H} p_j(i, \mathbf{x}) \ln p_j(i, \mathbf{x}).$$

Table 9. dSprites Variational Autoencoder Architecture.

Component	Layer Type	Hyperparameters	Activation Function
Encoder	Conv2d	Input Channels:1 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Padding:1	ReLU
	Conv2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Padding:1	ReLU
	Conv2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Padding:1	ReLU
	Conv2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Padding:1	ReLU
	Flatten	Start Dimension:1	
	Linear	Input Dimension: 512 ; Output Dimension: 256	ReLU
	Linear	Input Dimension: 256 ; Output Dimension: 256	ReLU
	Linear	Input Dimension: 256 ; Output Dimension: 12	ReLU
Reparametrization Trick		The output of the encoder contains μ and $\log \sigma$. The latent representation is then generated via $\mathbf{h} = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$	
Decoder	Linear	Input Dimension: 6 ; Output Dimension: 256	ReLU
	Linear	Input Dimension: 256 ; Output Dimension: 256	ReLU
	Linear	Input Dimension: 256 ; Output Dimension: 512	ReLU
	Unflatten	Dimension:1 ; Unflatten Size:(32, 4, 4)	
	Conv2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Padding:1	ReLU
	ConvTranspose2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Output Padding:1	ReLU
	ConvTranspose2d	Input Channels:32 ; Output Channels:32 ; Kernel Size:4 ; Stride:2 ; Output Padding:1	ReLU
	ConvTranspose2d	Input Channels:32 ; Output Channels:1 ; Kernel Size:4 ; Stride:2 ; Output Padding:1	Sigmoid

This entropy is analogous to Shannon’s entropy (Shannon, 1948). It can be checked easily that this entropy is minimal ($S^{\min} = 0$) whenever only one latent unit $j \in [d_H]$ is sensitive to feature i : $p_j(i, \mathbf{x}) = 1$. Conversely, it is well known (Cover & Thomas, 2005) that the entropy is maximal ($S^{\max} = \ln d_H$) whenever the distribution is uniform over the latent units: $p_j(i, \mathbf{x}) = 1/d_H$ for all $j \in [d_H]$. In short: the entropy is low when mostly one latent unit is sensitive to the feature of interest and high when several latent units are sensitive to the feature of interest. Clearly, the former situation is more desirable if we want to distinguish the different latent units. For each VAE, we evaluate the average entropy

$$S = \mathbb{E}_{\mathbf{X}, I} [S(I, \mathbf{X})] \\ \mathbf{X} \sim \text{Empirical Distribution}(\mathcal{D}_{\text{test}}), I \sim \text{Uniform}[\mathcal{I}(\mathbf{X})],$$

where $\mathcal{I}(\mathbf{x}) = \{i \in [d_X] \mid a_i(\mu_k, \mathbf{x}) \neq 0 \text{ for at least one } k \in [d_H]\}$ is the set of features that are salient for at least one latent unit. We measure the average entropy for each VAE and report the results as a function of β in Figure 8.

We clearly see that the entropy decreases as the disentanglement strength β increases for both MNIST (Spearman $\rho = -0.56$) and dSprites (Spearman $\rho = -0.76$). This means that disentangling has the effect of distributing the saliency over fewer units. This brings a nice complement to the quantitative analysis that we have conducted in Section 4.3: although increasing disentanglement does not make the latent units focus on different parts of the image (since the correlation does not decrease significantly), it does decrease the number of latent units that are simultaneously sensitive to a given part of the image (since the entropy decreases substantially). These two phenomena are not incompatible with each other. For instance, we see that the 6-th latent unit seems inactive in comparison with the other latent units in Figure 12. In fact, this latent unit might perfectly pay attention to the same parts of the image as the other units and, hence, be correlated. What distinguishes this unit from the others is that the feature importance scores on its saliency map are significantly smaller (we cannot appreciate it by plotting the saliency maps on the same scale) and, hence, reduces the entropy. Finally, we note that the entropies from Figure 8 remain fairly close to their maximal value ($S^{\max} = \ln 3 \approx 1.1$ for MNIST and $S^{\max} = \ln 6 \approx 1.8$ for dSprites). This means that the VAEs have several active units for each pixel.

Supplementary Examples. To check that the qualitative analysis from Section 4.3 extends beyond the examples showed in the main paper, the reader can refer to Figures 11 and 12. These saliency maps are produced with the main paper’s VAEs.

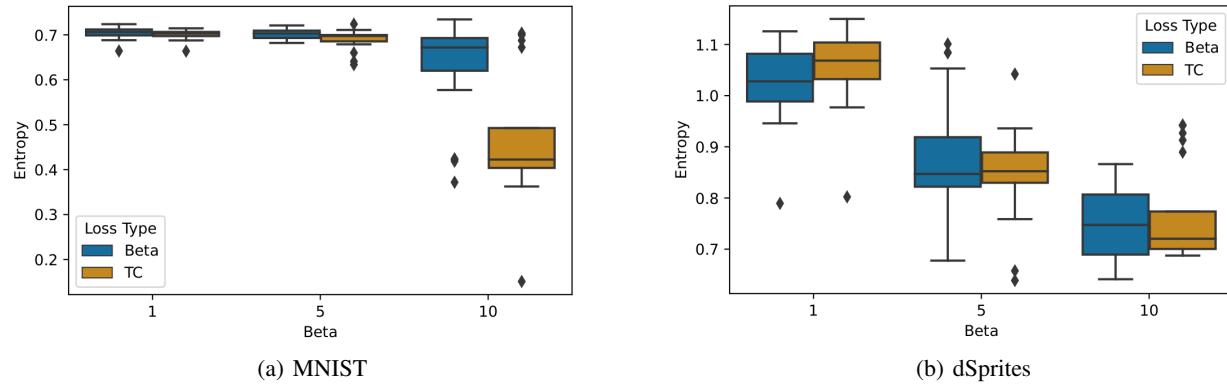


Figure 8. Entropy of saliency maps for different values of β .

We also plot saliency maps for vanilla ($\beta = 1$) VAEs in Figures 13 and 14. The issues mentioned in Section 4.3 are still present in this case.

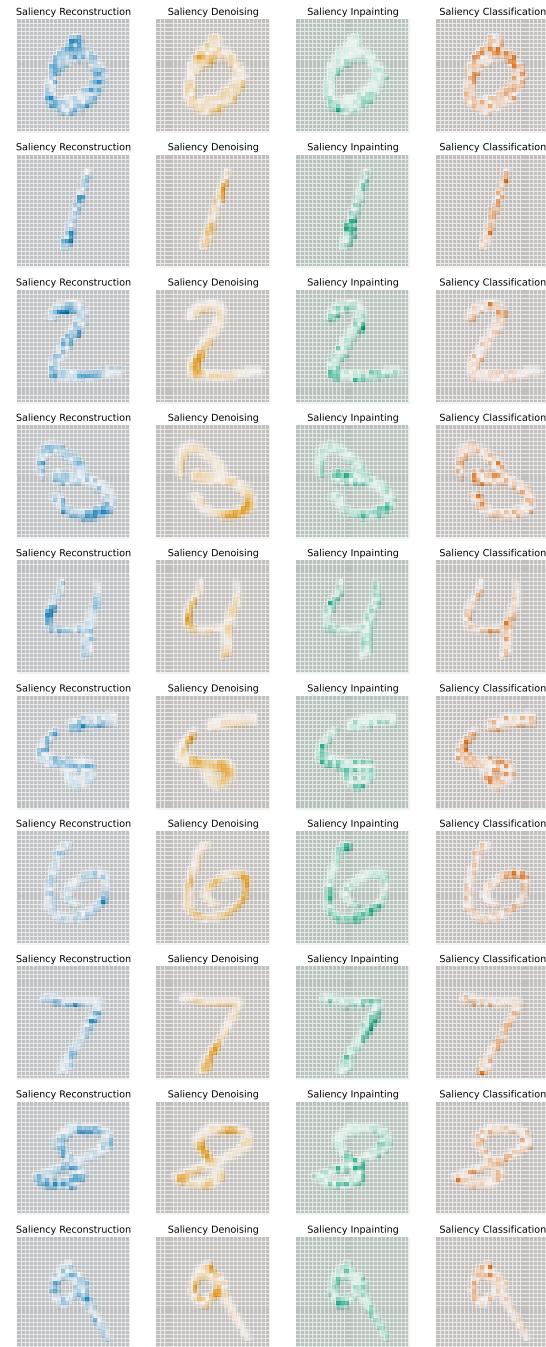


Figure 9. Label-free saliency for various pretext tasks.

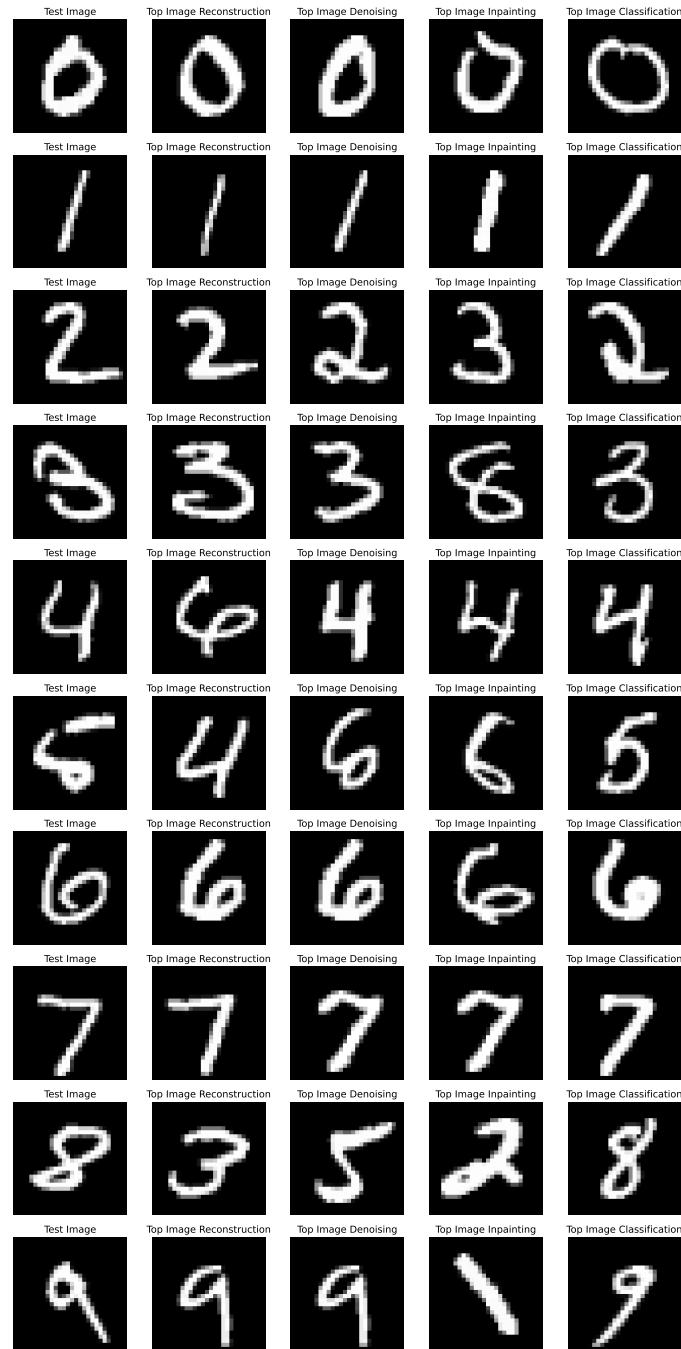


Figure 10. Label-free top example for various pretext tasks.

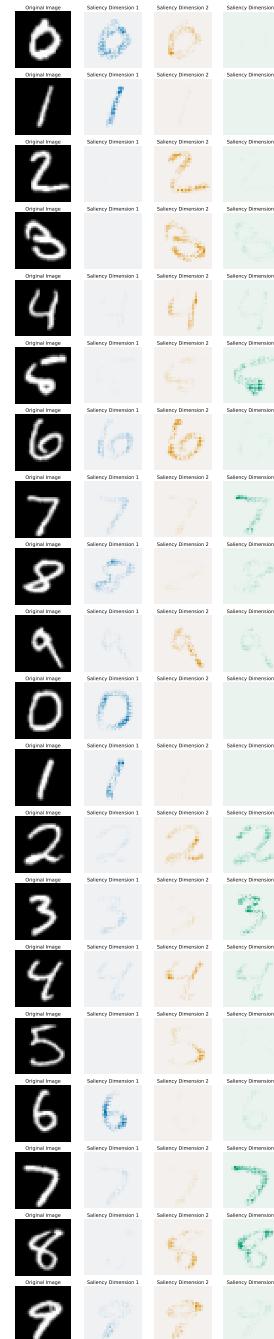


Figure 11. Saliency maps for the latent units of a MNIST VAE.

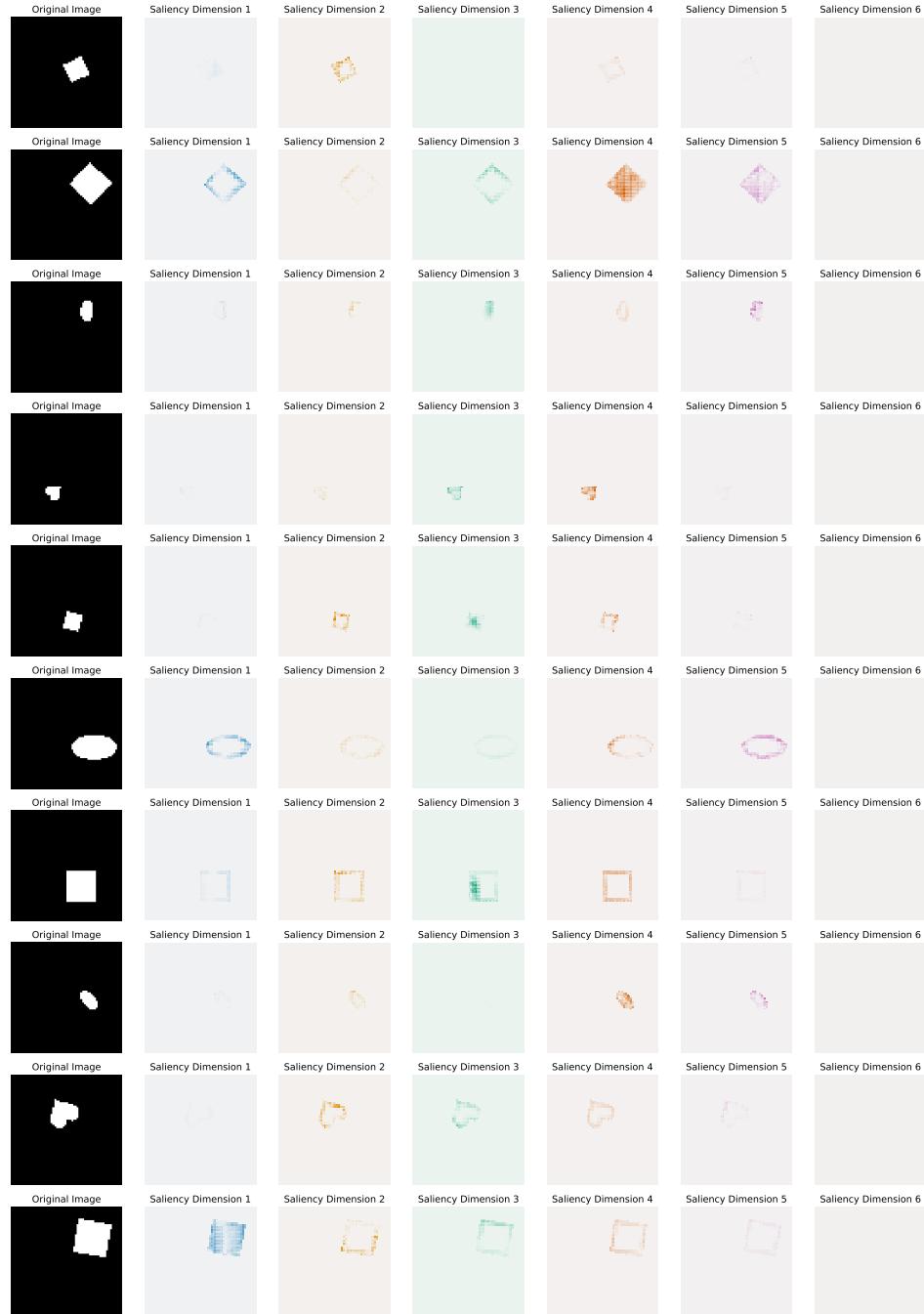


Figure 12. Saliency maps for the latent units of a dSprites VAE.

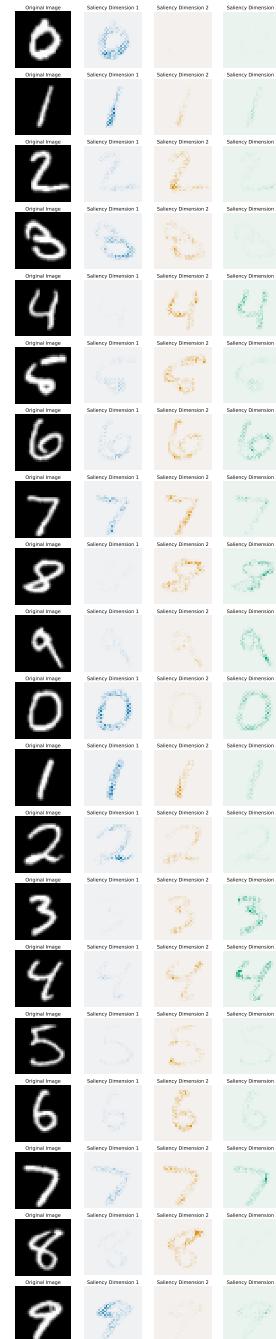


Figure 13. Saliency maps for the latent units of a vanilla ($\beta = 1$) MNIST VAE.

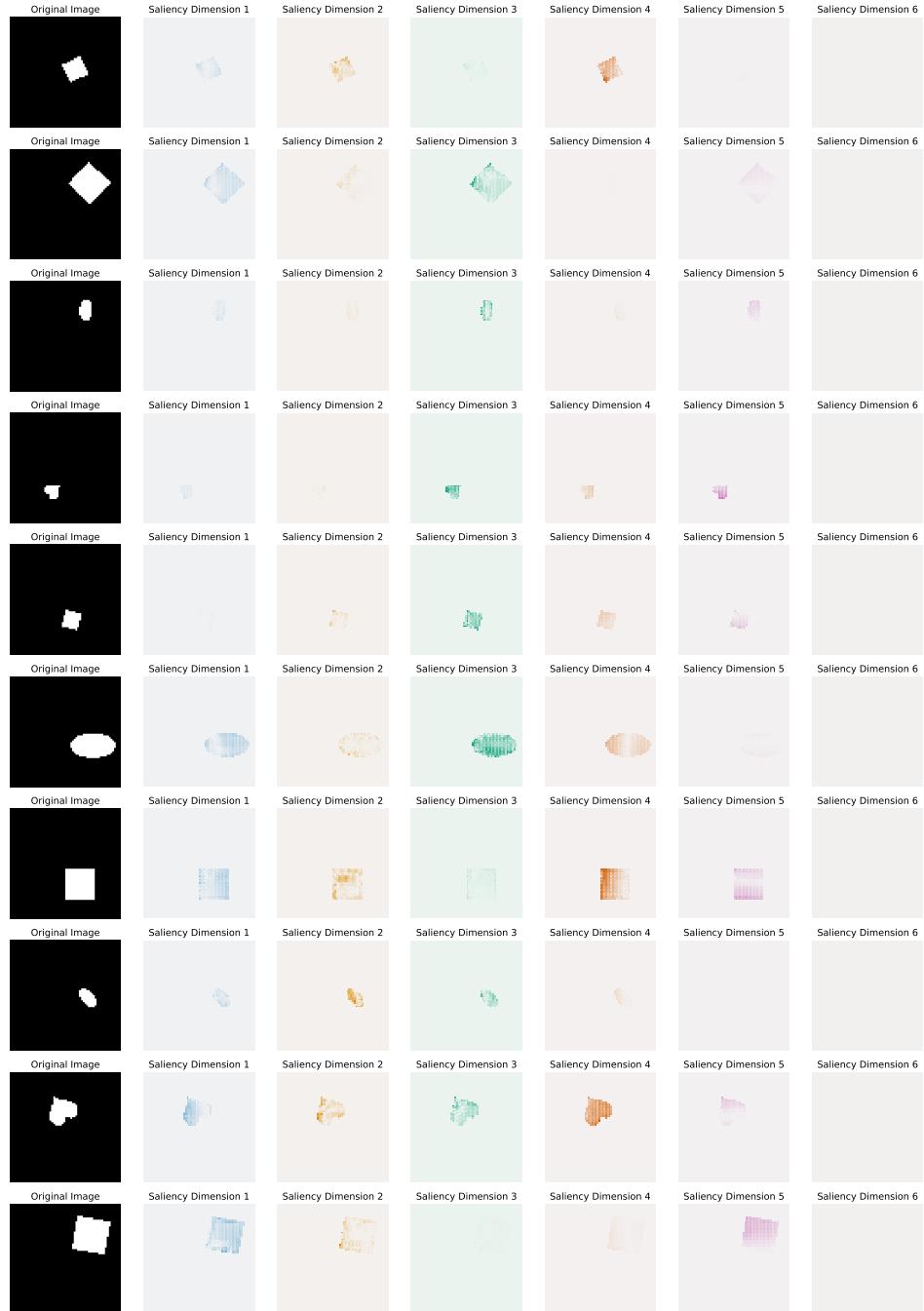


Figure 14. Saliency maps for the latent units of a vanilla ($\beta = 1$) dSprites VAE.