Explaining Predictions by Approximating the Local Decision Boundary

Georgios Vlassopoulos
Jheronimus Academy of Data Science
georgiosvlassopoulos@gmail.com

Tim van Erven
Leiden University
tim@timvanerven.nl

Henry Brighton
Tilburg University
brighton@uvt.nl

Vlado Menkovski
Eindhoven University of Technology
v.menkovski@tue.nl

June 16, 2020

Abstract

Constructing accurate model-agnostic explanations for opaque machine learning models remains a challenging task. Classification models for high-dimensional data, like images, are often complex and highly parameterized. To reduce this complexity, various authors attempt to explain individual predictions locally, either in terms of a simpler local surrogate model or by communicating how the predictions contrast with those of another class. However, existing approaches still fall short in the following ways: a) they measure locality using a (Euclidean) metric that is not meaningful for non-linear high-dimensional data; or b) they do not attempt to explain the decision boundary, which is the most relevant characteristic of classifiers that are optimized for classification accuracy; or c) they do not give the user any freedom in specifying attributes that are meaningful to them. We address these issues in a new procedure for local decision boundary approximation (DBA). To construct a meaningful metric, we train a variational autoencoder to learn a Euclidean latent space of encoded data representations. We impose interpretability by exploiting attribute annotations to map the latent space to attributes that are meaningful to the user. A difficulty in evaluating explainability approaches is the lack of a ground truth. We address this by introducing a new benchmark data set with artificially generated Iris images, and showing that we can recover the latent attributes that locally determine the class. We further evaluate our approach on the CelebA image data set.

1 Introduction

Over the last few years, explaining opaque machine learning (ML) models has become a topic of increasing attention [2, 25, 8, 23, 37, 26]. This attention

arises from multiple needs of ML users, such as ensuring model *trustworthiness*, detecting and removing unwanted biases (*fairness*) and understanding *causal relationships* [2]. For many of these needs, it is crucial that explanations are able to identify and communicate which properties of the input are the most important for the model's predictions. These properties can be identified either by modifying the underlying mechanism of the model in a trade-off with predictive accuracy, or by designing algorithms that explain the model's behavior post-hoc after training. For the latter, one may take into account the model's internal mechanisms and develop frameworks for a specific category of models (e.g. for neural networks [3, 14, 40] or random forests [39]), or take a *model-agnostic* approach that can explain many different types of models [30, 21, 13, 36, 4, 5].

Predictive models are often complex and controlled by many parameters that cannot all be communicated to a user. This complexity can be reduced by explaining only the model's behavior in a *local* region of the feature space near a given input $\boldsymbol{x}_0$. Local model-agnostic explanations may be *example-based*, by simulating a data point $\boldsymbol{x}_\mathrm{c}$ from a different class that is close to $\boldsymbol{x}_0$ [16, 5]. This has the drawback that it is left to the user to identify the relevant differences between $\boldsymbol{x}_\mathrm{c}$ and $\boldsymbol{x}_0$, which may be difficult for high-dimensional data like images, especially when $\boldsymbol{x}_\mathrm{c}$ and $\boldsymbol{x}_0$ are close. An alternative is to approximate the model locally by an interpretable *surrogate model*, which is usually a linear model, a set of decision rules or a decision tree [30, 31].

We propose three criteria that local model-agnostic explanations should satisfy. First, locality needs to be measured by a *metric* that is *meaningful* for the data at hand. For unstructured low-dimensional data without strong correlations between the features, it may be very reasonable to use Euclidean distance, but for highly structured data we typically need a more refined measure. For instance, the difference in pixel values between two images will be affected greatly by the slight movement of an object, by a variation in lighting or by adding minor noise, without users perceiving any relevant difference in content. Lack of an appropriate metric prevents some explainability methods from scaling up to image data [17] and is not even fully avoided by some methods that do apply to image data [22]. Second, we posit that explanations of classifiers should depend solely on the *decision boundary* between classes, and not on any class probability estimates the classifiers may output in addition. Since classifiers are typically optimized for prediction accuracy and not for probability estimation, they often output probabilities that are uncalibrated [9, 27] and therefore lack interpretability. Moreover, users are directly affected by the decision boundary and not by probabilities: someone who is refused a mortgage may not care whether a better credit score could have reduced their estimated probability of defaulting on payment from 0.62 to 0.61, but they likely would care what would have changed the bank's decision. Third, a consensus appears to be forming in the literature that explanations need to be tailored to the target audience and context [2, 25]: an ML developer who is trying to fix classification errors should not be given the same output as an authority assessing compliance with fairness regulations. Consequently, there should be a way for the user to *specify attributes* of the data that are interpretable and relevant to them.

New Method We introduce the first model-agnostic method to explain binary classifiers that meets all three of these criteria. The procedure performs local decision boundary approximation (DBA) by searching for the closest decision boundary point $\boldsymbol{x}_\mathrm{b}$ to $\boldsymbol{x}_0$, and generating a sample of artificial data points around it which are labeled by the model that we want to explain. A linear surrogate model is then fit to this sample. We interpret the coefficients of this linear model as the direction of minimal change to switch the class of $\boldsymbol{x}_0$. Instead of measuring distance in the input space, we train a variational autoencoder (VAE) [15] to learn a Euclidean latent space of encoded data representations, and measure distance in this latent space. We also use attribute annotations to learn mappings from the latent space to user-specified attributes, and fit the linear surrogate in terms of the attributes. We restrict the search for $\boldsymbol{x}_\mathrm{b}$ to the data manifold by performing bisecting line searches between $\boldsymbol{x}_0$ and a selection of nearby training data points from the opposite class. This has the additional benefit that the run-time of the search procedure (almost) does not depend on the dimension. Both the line searches and the sampling step take place in the latent space learned by the VAE. Our main contributions lie in the new search and sampling procedures and the way they are combined with a VAE and user-specified attributes. We further adjust the training procedure of the VAE to favor preservation of class probabilities, which can have a significant effect.

New Evaluation Data A difficulty in evaluating explainability approaches is the lack of a ground truth. To remedy this, we introduce the Artificial Iris (AIris) data set, with simulated images of flowers that can be used as a benchmark for explainability methods. We label the classes based on two hyperplanes defined by latent parameters of the flowers like petal length and sepal width. We then show that our method is able to recover the true coefficients of the hyperplanes with high accuracy from the predictions of a convolutional neural network (CNN) that is trained to classify near-perfectly.

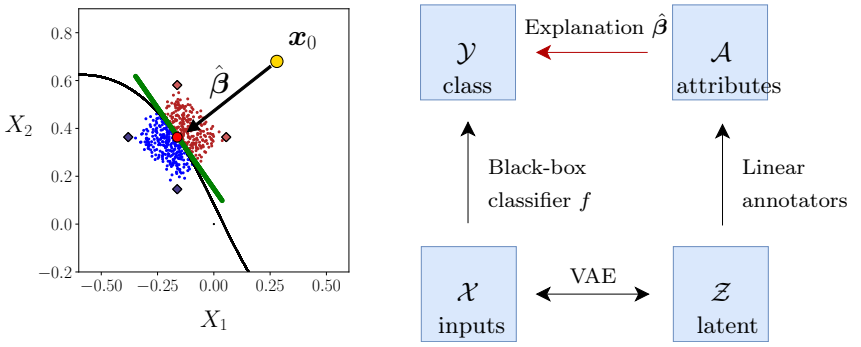
Related Work Perhaps the most established approach to local model-agnostic explanation is the LIME algorithm [30], with versions available for tabular data, text data, and images. LIME does not try to explain the decision boundary (so it does not satisfy our second criterion), but explains class probabilities by generating a sample of new points similar to $\boldsymbol{x}_0$ and fitting these points with a linear surrogate model that downweights sample points exponentially by their distance from $\boldsymbol{x}_0$. The original method uses Euclidean distance in the input space (which does not satisfy our first criterion). A variant called ALIME [34] is available that measures distance in the latent space of a denoising autoencoder [38], which is not quite Euclidean but makes progress towards our first criterion. For both LIME and ALIME the sampling procedure and the surrogate model are specified in terms of abstract attributes of $\boldsymbol{x}_0$, for which the authors propose specific choices (e.g. superpixels for images), but which can in principle be specified by the user (thus satisfying our third criterion). As shown by Laugel et al. [17], an important limitation of LIME occurs when $\boldsymbol{x}_0$ is

in between multiple regions of the decision boundary, in which case LIME will output the mean of the decision boundary parts within its sampling radius, which is not a faithful representation of the original classification model. Laugel et al. [17, 16] therefore introduce the LocalSurrogate procedure, which is similar to our DBA approach in that it searches for the closest decision boundary point $\boldsymbol{x}_\mathrm{b}$ and samples around it to fit the local decision boundary (thus meeting our second criterion). Unfortunately, the search and sampling procedures for LocalSurrogate are ineffective beyond very small dimensions. The procedure is also based on Euclidean distance in the input space (thus failing our first criterion), and does not allow the user to specify interpretable attributes (failing our third criterion). An alternative to approximating the decision boundary or class probabilities is to provide a so-called contrastive explanation, as implemented in CEM [5] and its extension CEM-MAF [22]. A contrastive explanation $\boldsymbol{x}_\mathrm{c}$ is a generated example from the opposite class that is close to $\boldsymbol{x}_0$, but usually not on the decision boundary (going against our second criterion). CEM-MAF measures the distance to $\boldsymbol{x}_0$ both in the input space (which goes against our first criterion) and in the latent space of a VAE (which fits with our first criterion) or generative adversarial network (which does not necessarily produce a Euclidean space, so it does not satisfy our first criterion). It further encourages the difference between $\boldsymbol{x}_\mathrm{c}$ and $\boldsymbol{x}_0$ to be sparse in terms of user-specified attributes (satisfying our third criterion). Finally, we mention the Anchors method, which produces local decision rules that are consistent with the decision boundary [31], and TCAV, which uses attribute annotations to learn a mapping from the internal state of a neural network to user-defined attributes [14].

Outline In the next section we describe our new DBA method. Section 3.1 then introduces the Artificial Iris data set and presents experiments comparing DBA to a variant of LIME and to CEM-MAF. Section 3.2 extends the evaluation to the CelebA data set [18] with real-world celebrity images. We show there that blurriness of images and open mouths improperly affect classifications. Finally, we conclude with a discussion of the limitations of our approach. Details and additional experiments are postponed to the appendix.

2 Method: Local Decision Boundary Approximation

Let $f : \mathcal{X} \to \mathcal{Y} = \{-1, +1\}$ be the binary classifier, whose decision for input $\boldsymbol{x}_0 \in \mathcal{X} \subset \mathbb{R}^d$ is to be explained. Suppose also that the training data $D = \big((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\big)$ on which f has been trained are still available, with $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. We first present a simplified version of our method: DBA-Tab, which is suitable for tabular data. This simplified version applies if the inputs $\boldsymbol{x}$ consist of d features that are already interpretable to the user, so we do not need a VAE or separate interpretable attributes. It further requires that the features have been suitably standardized and decorrelated to make Euclidean



(a) DBA-Tab on a toy example (b) DBA-Att spaces and their relations

Figure 1: Illustration of the DBA-Tab and DBA-Att procedures

distance an appropriate metric for $\mathcal{X}$. DBA-Tab consists of the following steps, which are illustrated in Figure 1a:

1. *Detection:* Search along the manifold of the training data D to find the point $\boldsymbol{x}_\mathrm{b}$ on the decision boundary that is closest to $\boldsymbol{x}_0$.

This is implemented by selecting the k closest points to $\boldsymbol{x}_0$ from D that are from the opposite class. For each selected point $(\boldsymbol{x}_j, y_j)$, we then perform a line search using the bisection method to find a point $\boldsymbol{x}_{\mathrm{b},j}$ on the line segment between $\boldsymbol{x}_0$ and $\boldsymbol{x}_j$ that is on the decision boundary of f. We call $\boldsymbol{x}_j$ the *corresponding bisected point* for $\boldsymbol{x}_{\mathrm{b},j}$. Finally, among these decision boundary points $\boldsymbol{x}_{\mathrm{b},1}, \ldots, \boldsymbol{x}_{\mathrm{b},k}$, we choose $\boldsymbol{x}_\mathrm{b}$ to be the point that is closest to $\boldsymbol{x}_0$. A notable advantage of this approach is computational efficiency: instead of searching in d dimensions for a boundary of unknown shape, it only needs to perform k one-dimensional line searches.

2. *Simulation:* Randomly generate m points near $\boldsymbol{x}_\mathrm{b}$ on both sides of the decision boundary of f and label them with f to obtain a sample $S = \big((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\big)$.

A natural first idea would be to sample densely from a sphere around $\boldsymbol{x}_\mathrm{b}$, but then the curse of dimensionality [10] would imply that the sample size m would need to grow exponentially with d, so this is hopeless except for very small d. Instead, we generate a sample that contains variation in all the feature directions: for each feature $j = 1, \ldots, d$, we first create two vertices $\boldsymbol{v}_{j,-1}$ and $\boldsymbol{v}_{j,+1}$ that are equal to $\boldsymbol{x}_\mathrm{b}$ except that they respectively decrease and increase the value of feature j by an amount α. We set $\alpha = r\|\boldsymbol{x}_\mathrm{b} - \boldsymbol{x}_0\|$ proportional to the distance between $\boldsymbol{x}_\mathrm{b}$ and $\boldsymbol{x}_0$ with a proportionality parameter $r > 0$ that is specified below. Each point $(\boldsymbol{x}, y) \in S$ is then sampled independently from the convex hull of the vertices $\boldsymbol{v}_{1,\pm1}, \ldots, \boldsymbol{v}_{d,\pm1}$ by drawing random weights $w_{j,\pm1}$ uniformly at random from the $2d$-dimensional probability simplex, and setting $\boldsymbol{x} = \sum_{j=1}^{d} w_{j,-1}\boldsymbol{v}_{j,-1} + \sum_{j=1}^{d} w_{j,+1}\boldsymbol{v}_{j,+1}$. We note that this approach

does not sample uniformly from the convex hull of the vertices, but instead induces approximate sparsity in the samples, because the influences of $\boldsymbol{v}_{j,-1}$ and $\boldsymbol{v}_{j,+1}$ approximately cancel each other out when their weights $w_{j,-1}$ and $w_{j,+1}$ are similar. Our use of (approximate) sparsity bears a resemblance to LIME [30], which samples sparse perturbations of $\boldsymbol{x}_0$.

3. *Explanation:* Fit a linear surrogate model $g(\boldsymbol{x}) = \boldsymbol{x}^\top \hat{\boldsymbol{\beta}} + \hat{\beta}_0$ on the sample S using (unpenalized) logistic regression.

The parameters of the method are k, m and r. We tune r automatically from a grid $\mathcal{R}$ of possible values by choosing the value for which the resulting direction $\hat{\boldsymbol{\beta}}$ minimizes the distance from $\boldsymbol{x}_0$ to the decision boundary.[1] The output of the algorithm is the linear surrogate model g. Its most important component is the coefficient vector $\hat{\boldsymbol{\beta}}$, which is the normal vector to the decision boundary of g and can be interpreted as the direction of minimal change to switch the class of $\boldsymbol{x}_0$, as illustrated in Figure 1a. See Appendix A for experiments illustrating the behavior of DBA-Tab.

2.1 Extension to Structured High-dimensional Data

DBA-Tab is not suitable for structured high-dimensional data like images, because individual input pixels lack interpretability and Euclidean distance between images is not meaningful. We therefore provide an extension called DBA-Att, which stands for DBA with attributes. DBA-Att measures distance in a latent representation space $\mathcal{Z}$, which is learned by a VAE from the training data D. The user is further required to describe attributes that are meaningful to them by providing additional data annotations. We use these annotations to predict which attributes are present for any given latent representation $\boldsymbol{z} \in \mathcal{Z}$. See Figure 1b for an overview of all the spaces and mappings between them.

Learning the Latent Representation Space Variational autoencoders provide an unsupervised procedure to learn (non-linear) mappings back and forth between inputs in $\mathcal{X}$ and latent representations in $\mathcal{Z} \subset \mathbb{R}^l$. The dimensionality l of the latent space is taken to be much smaller than the input dimension d, which forces a dimensionality reduction. VAEs have the important property that the marginal distribution over the latent space is approximately standard Gaussian, which means that Euclidean distance is an appropriate metric in $\mathcal{Z}$. We do not commit to any single choice of VAE, but allow the VAE to be customized for the data under consideration. We further adjust the training procedure of the VAE to favor preservation of class probabilities, which can significantly improve the *label stability*, i.e. $f(\boldsymbol{x}') = f(\boldsymbol{x})$, where $\boldsymbol{x}'$ is the reconstruction of $\boldsymbol{x}$ that is obtained by mapping $\boldsymbol{x}$ to $\mathcal{Z}$ and back using the VAE. See Appendix A for

[1] The distance to the decision boundary is again determined by the bisection method; this time on the line segment between $\boldsymbol{x}_0$ and $\boldsymbol{x}' = \boldsymbol{x}_0 - f(x_0)\gamma\hat{\boldsymbol{\beta}}/\|\hat{\boldsymbol{\beta}}\|$, where we take γ large enough that $\boldsymbol{x}'$ is always on the other side of the decision boundary. Specifically, we use $\gamma = \|\boldsymbol{x}_0 - \boldsymbol{x}_\text{b}\| + 0.1$ in all our experiments.

details. Label stability for $\boldsymbol{x}_0$ is crucial, because otherwise it hopeless to use the VAE in any procedure that tries to explain the decision boundary.

Predicting User-specified Attributes We assume the user provides annotations for p attributes, where each annotation $A_j = \big((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{n_j}, y_{n_j})\big)$ consists of pairs of inputs $\boldsymbol{x}_i$ and binary labels $y_i \in \{-1, +1\}$ that indicate whether attribute j is true or false. For example, an attribute might indicate whether the person in an image is smiling or not. The user may annotate (part of) the training data D, or provide separate annotated data. For each annotation A_j, we first map the inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_j}$ to their latent representations $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_j}$ using the VAE and then use L_2-penalized logistic regression to train what we call an *annotator* $a_j : \mathcal{Z} \to [0, 1]$, which predicts the probability that attribute j is true from the latent representation of an input. We denote the coefficients of the logistic regression model for a_j by $\hat{\boldsymbol{\theta}}_j \in \mathbb{R}^l$ and its intercept by $\hat{\theta}_{0,j}$. Thus, we can go from any input $\boldsymbol{x} \in \mathcal{X}$ to a latent representation $\boldsymbol{z} \in \mathcal{Z}$ to a vector $\boldsymbol{a} = \big(a_1(\boldsymbol{z}), \ldots, a_p(\boldsymbol{z})\big)$ of attribute probabilities. We call the space that $\boldsymbol{a}$ lives in the *attribute space* $\mathcal{A} \subset \mathbb{R}^p$.

The DBA-Att Method Our extended method DBA-Att differs from DBA-Tab as follows. First, we map the point to be explained, $\boldsymbol{x}_0$, to its latent representation $\boldsymbol{z}_0$. The detection step then runs in the latent space $\mathcal{Z}$ instead of the input space $\mathcal{X}$ and detects a point $\boldsymbol{z}_\text{b}$ on the decision boundary of f. Labels for any point $\boldsymbol{z} \in \mathcal{Z}$ are obtained by mapping $\boldsymbol{z} \mapsto \boldsymbol{x} \mapsto f(\boldsymbol{x})$. Second, the simulation step also runs in $\mathcal{Z}$. Since the coordinates of $\mathcal{Z}$ need not correspond directly to interpretable features, we define the vertices in terms of the user-specified attributes: $\boldsymbol{v}_{j,\pm 1} = \boldsymbol{z}_\text{b} \pm \alpha \hat{\boldsymbol{\theta}}_j$ for $j = 1, \ldots, p$. The interpretation is that $\hat{\boldsymbol{\theta}}_j$ represents the direction that increases the probability of attribute j being true. Finally, in the explanation step we fit the linear surrogate model in the attribute space, by mapping each sample point $(\boldsymbol{z}_i, y_i)$ to $(\boldsymbol{a}_i, y_i)$ using the annotators. To make the coefficients of the surrogate model comparable between attributes, we standardize all attributes $a_{i,j}$ based on their mean and standard deviation in the sample S.

3 Experiments

We design two controlled experiments in which interpretable attributes are available and we know how the class labels are assigned based on these attributes. In the first experiment we generate our own artificial data set of flower images (AIris). In the second experiment we use the CelebA data set [18], for which attribute annotations are available.

3.1 AIris: Artificial Iris Data

Inspired by Anderson's classical Iris flower data set [1], which was made famous by Fisher [7], we have created an image generation program that generates

128×128 RGB images of flowers based on 5 continuous parameters that all have ranges inside $[0, 1]$. See Appendix C for examples and further details. The first four parameters control the shape of the flower. They are petal length (PL), petal width (PW), sepal length (SL) and sepal width (SW). The last parameter, color (C), is a mixing parameter that interpolates between red and magenta. We have used this program to generate a training set D of 4000 flowers, as well as a test set with 2000 images. Each image was generated independently by sampling the five parameters from the uniform distribution over their range. We assign a flower to class A if

$$0.33\text{PL}+0.33\text{PW}+0.33\text{C} < 0.5 \quad \text{and} \quad 0.33\text{PL}+0.33\text{PW}+0.33\text{SL} > 0.4, \quad (1)$$

and to class B otherwise. This assignment defines a non-linear ground truth consisting of two hyperplanes that are defined in terms of latent parameters which are not available to the classifier f. The hyperplanes were chosen to achieve approximately balanced classes. We further annotate the training data with binarized versions of the parameters: for each parameter $\text{PL}, \text{PW}, \ldots, \text{C}$ we define an attribute that measures whether the parameter value is large or not. We set this attribute to $+1$ if the parameter exceeds the midpoint of its range, and to -1 otherwise. Thus the parameters are also not directly available to the explanation methods. The appeal of the AIris data is that it is sufficiently simple for a standard convolutional VAE to learn good latent parameters, but sufficiently difficult to illustrate the differences between existing explanation methods.

3.1.1 Experiments and Results

We train a 5-layer CNN $c : \mathcal{X} \to [0, 1]$ on the training data to learn the probability of class A. The corresponding binary classifications are: $f(\boldsymbol{x}) = +1$ if $c(\boldsymbol{x}) > 0.5$ and $f(\boldsymbol{x}) = -1$ otherwise. See Appendix C for details. The CNN achieves 99.33% accuracy on the training data and 98.75% accuracy on the test set, which is sufficiently high that, on images from the same source, its decision boundary must be very similar to the ground truth specified by the two hyperplanes. As described in Appendix C, we further train a convolutional VAE on the training data. The VAE achieves 90% label stability on the test set (up from 74% without our adjustment to favor preservation of class probabilities). We compare DBA-Att to the CEM-MAF pertinent negative method [22]. We also want to compare to LIME, but since LIME [30] and ALIME [34] do not have the option to learn user-specified attributes from annotations, we instead compare to a hybrid method LIME-Att, which is a variant of DBA-Att in which we have replaced the detection and simulation steps by LIME. We also include a non-local baseline that we call the Global method. Global is given an advantage because it gets as features the original parameters $(\text{PL}, \text{PW}, \ldots, \text{C})$ used to generate the images, but it is also at a disadvantage because it fits a global linear model in terms of these parameters, so it cannot exploit locality. We run DBA-Att with $k = 1000, m = 500$ and r tuned automatically from grid $\mathcal{R} = \{0.1, 0.2, \ldots, 0.9, 1, 1.5, 2, \ldots, 9.5, 10\}$. For details about the other methods,

see Appendix C. DBA-Att, LIME-Att and Global all output a coefficient vector $\hat{\boldsymbol{\beta}} \in \mathbb{R}^p$ that expresses the relative importance of the attributes. This corresponds to a direction $\boldsymbol{\delta} = \sum_{j=1}^{p} \hat{\beta}_j \hat{\boldsymbol{\theta}}_j$ in the latent space $\mathcal{Z}$, where $\hat{\boldsymbol{\theta}}_j \in \mathbb{R}^l$ are the coefficients of the annotator for attribute j (see Section 2.1). CEM-MAF outputs a contrastive example $\boldsymbol{x}_\mathrm{c}$, but, as described in Appendix C, it may also be used to obtain a direction $\boldsymbol{\delta}$ in the latent space and a vector of coefficients $\hat{\boldsymbol{\beta}}$ for the attributes.

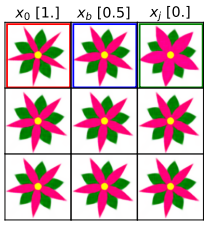


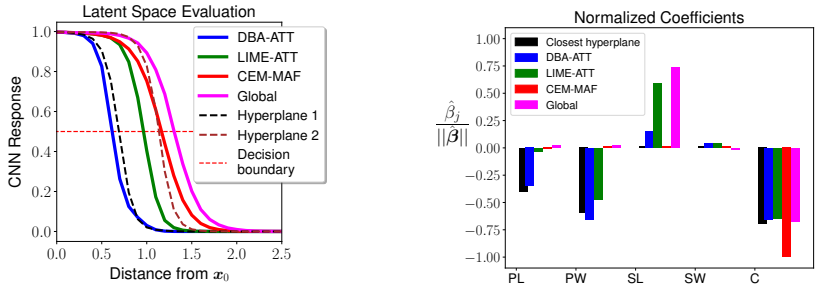
Figure 2: Illustration of DBA-Att detection and sampling steps

A First Impression Before reporting aggregate statistics for multiple explanations, we first illustrate our results on a single representative input image $\boldsymbol{x}_0$. Two other inputs illustrating best and worst performance for DBA-Att are shown in Appendix C. Figure 2 shows the input $\boldsymbol{x}_0$ in a red frame. It further shows the closest decision boundary point $\boldsymbol{x}_\mathrm{b}$ in a blue frame, and in a green frame there is the corresponding bisected point $\boldsymbol{x}_j$ from the detection step of the algorithm. The other images are 6 random samples from S that were generated in the simulation step. Appendix C shows the principal component analysis (PCA) projection of S onto two dimensions. It can be seen there that the two classes occur in roughly equal proportions, and can be separated quite well with a linear decision boundary. The corresponding linear local surrogate model g is indeed highly faithful to f: its fidelity is 99.7% on S.

Figure 3 compares the quality of explanations on a single test image. We see in Figure 3a that DBA-Att has found a direction that gets to the decision boundary much faster than the other methods. Both figures in Figure 3 also show that the DBA-Att direction is similar to the direction of the closest hyperplane[2], whereas the directions for the other methods are different. We proceed to show that the behaviors observed in these examples in fact hold generally.

Aggregate Evaluation All methods are evaluated on the same set of 50 randomly selected images $\boldsymbol{x}_0$ from the test data on which the VAE achieves label stability. (See Section 2.1 for a discussion on label stability.) In Table 1 we report the means of the following statistics. First, fidelity measures how

[2]For proper comparison, we standardize each coefficient of the hyperplane by multiplying it by the standard deviation of its corresponding parameter. See Appendix C.



(a) CNN probabilities moving in direction $\boldsymbol{\delta}/\|\boldsymbol{\delta}\|$ from $\boldsymbol{x}_0$

(b) Explanation coefficients $\hat{\boldsymbol{\beta}}$

Figure 3: Evaluation of explanation directions on the AIris test image from Figure 2

Table 1: AIris evaluation statistics, averaged over 50 test images

	DBA Fidelity	LIME R^2-Fidelity	Global Fidelity	Class Balance	Latent Distance	Cosine Similary-	Cosine Similarity+
DBA-Att	97.0%	-	-	**50.2%**	**0.6**	0.858	**0.925**
LIME-Att	-	21.7%	-	47.7%	1.0	0.570	0.719
CEM-MAF	-	-	-	-	1.0	0.458	0.640
Global	-	-	73.6%	47.4%	1.8	0.403	0.454

well the surrogate models approximate the original classifier f. For DBA-Att we measure "DBA Fidelity" by the classification accuracy of the surrogate in predicting the labels of f on the local sample S generated by DBA-Att; for "LIME Fidelity" we use the local Weighted Regression R^2 measure proposed by the LIME authors on the LIME-Att sample [34]; and for Global we measure "Global Fidelity" by its classification accuracy on all the training data. Since CEM-MAF does not directly produce a surrogate model, we do not measure its fidelity. Second, we measure the class balance of the samples produced by DBA-Att and LIME-Att by the percentage of sample points labeled as $+1$ by f. For Global we report the class balance of all the training data. Third, we measure the "Latent Distance" from $\boldsymbol{z}_0$ to the decision boundary in the latent space along the direction $\boldsymbol{\delta}$. Finally, we include two measures that capture the cosine similarity between $\hat{\boldsymbol{\beta}}$ and the coefficients of the true hyperplanes in (1).[2] The difference is that "Cosine Similarity-" measures similarity with the closest of the two hyperplanes and "Cosine Similarity+" measures the maximum of the similarities with the two hyperplanes.

Aggregate Results For DBA-Att fidelity is high, while fidelity is low for LIME-Att: its local surrogate model is only able to explain 21.7% of the variance in the class probabilities of the CNN c. We further see that class balance is good (close to 50%) for all methods. From Latent Distance we see that

DBA-Att is much better at identifying the nearest decision boundary direction than LIME-Att and CEM-MAF, which perform equally well. Global is worse still, showing that local explanations are much more informative than a single global explanation. Finally, we see from Cosine Similarity+ that DBA-Att recovers the coefficients of one of the hyperplanes extremely well: the similarity is close to its maximum value 1. The other methods are significantly worse. The gap between the two Cosine Similarities for all methods indicates that the methods do not necessarily focus on the closest hyperplane. This can be explained by the fact that, for a subset of data points from class B, the closest hyperplane does not switch the class.

3.2 CelebA: Annotated Celebrity Image Data

CelebA is a large image data set of 202 599 celebrity faces, annotated with 40 binary attributes [18]. We restrict attention to 10 attributes: MALE, SMILE, BANGS, PALE, OPEN MOUTH, YOUNG, BLOND, MAKE-UP and BLURRY, and define two classes: A) SMILE and not MALE $\approx$ smiling females vs B) other. Class A was chosen because it is one of the few combinations of two attributes that gives approximately balanced classes. We split the data into $n = 162\,079$ training images (80%) and 40 520 test images (20%). A CNN is trained with 93.2% training accuracy and 91.5% testing accuracy. We further train a VAE based on Hou et al. [11], which gives 91% label stability. For the explanation methods, we only use the annotations from 9 000 randomly selected images from the training data, and we explain 30 random images from the test set on which the VAE achieves label stability using DBA-Att, LIME-Att and CEM-MAF. See Appendix D for further details.

Table 2: CelebA evaluation statistics, averaged over 30 test images

	DBA Fidelity	LIME R^2-Fidelity	Class Balance	Latent Distance
DBA-Att	97.3%	-	**50.1%**	**2.0**
LIME-Att	-	32.7%	47.2%	2.1
CEM-MAF	-	-	-	2.7

3.2.1 Experiments and Results

In Table 2 we report the same aggregate statistics as for the AIris data, except for the ones that require knowledge of the true data generating mechanism. We see that DBA-Att and LIME-Att are approximately equally good in terms of distance to the decision boundary, which suggests that for this data the local CNN probabilities increase more or less orthogonally from the decision boundary. In this case, CEM-MAF is significantly worse at indicating the direction of the nearest decision boundary. A possible explanation why the low R^2-fidelity of LIME-Att does not prevent it from achieving small Latent Distance is that its

surrogate model might approximate the decision boundary reasonably well even if it cannot approximate the CNN probabilities very precisely.

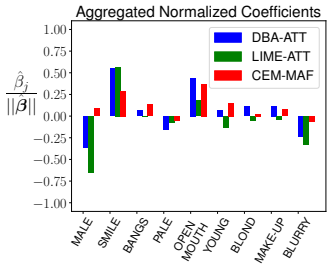
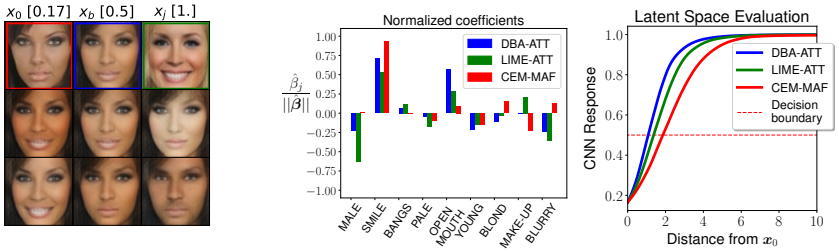


Figure 4: Mean explanation coefficients for CelebA

In Figure 4 we further show the mean coefficient values for the three methods. Both DBA-Att and LIME-Att detect the importance of MALE and SMILE, and further show that the CNN is improperly sensitive to OPEN MOUTH and BLURRY. The effect of blurriness is particularly interesting because it cannot be expressed by highlighting the most important pixels, which is a common approach in explainability [33, 30]. CEM-MAF disqualifies itself because it gives the wrong dependence on the MALE attribute.

Zooming in on a single image, Figure 5 shows a very similar pattern to the aggregate explanations, with perhaps increased importance of SMILE and OPEN MOUTH.



(a) Illustration of detection and sampling steps for DBA-Att

(b) Explanation coefficients

(c) CNN probabilities as we move in direction $\boldsymbol{\delta}/\|\boldsymbol{\delta}\|$ from $\boldsymbol{x}_0$

Figure 5: Explanations of the single image shown in the top-left corner of Figure 5a

4 Discussion

We provide explanations in terms of abstract user-specified attributes. This ambition comes with clear limitations in applicability. The most important

limitations are the dependence on successfully training a VAE and on the expressiveness of the attributes: a linear model in terms of the attributes, which corresponds to a linear model in a subspace of the latent space of the VAE, must be able to locally approximate the decision boundary of the classifier f with high fidelity. A less obvious limitation is that user annotations are not always univocal: if a user annotates a data set in which all men have short hair and all women have long hair, then the corresponding attribute MALE will not just correspond to gender but also to hair length. This is the problem of *entangled* latent representations, which recent work on VAEs is starting to address [24, 19, 20].

References

[1] E. Anderson. The irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935.

[2] A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, 2019.

[3] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. Muller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS ONE*, 10, 2015.

[4] P. Cortez and M. J. Embrechts. Opening black box data mining models using sensitivity analysis. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 2011.

[5] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *32nd Conference on Neural Information Processing Systems (NIPS 2018)*, pages 592–603, 2018.

[6] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[7] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(II):179–188, 1936.

[8] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89, 2018.

[9] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330, 2017.

[10] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[11] X. Hou, L. Shen, K. Sun, and G. Qiu. Deep feature consistent variational autoencoder. *CoRR*, 2016.

[12] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[13] U. Johansson, R. König, and L. Niklasson. The truth is in there - rule extraction from opaque models using genetic programming. 01 2004.

[14] B. Kim, M. Wattenberg, J. Gilmer, C. J. Cai, J. Wexler, F. Viegas, and R. A. Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *ICML*, 2018.

[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR)*, 2014.

[16] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki. Inverse classification for comparison-based interpretability in machine learning. *17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, 2018.

[17] T. Laugel, X. Renard, M.-J. Lesot, C. Marsala, and M. Detyniecki. Defining locality for surrogates in post-hoc interpretablity. *ArXiv*, abs/1806.07498, 2018.

[18] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. December 2015.

[19] F. Locatello, G. Abbati, T. Rainforth, S. Bauer, B. Schölkopf, and O. Bachem. On the fairness of disentangled representations. In *Advances in Neural Information Processing Systems 32*, pages 14611–14624. Curran Associates, Inc., 2019.

[20] F. Locatello, M. Tschannen, S. Bauer, B. Schölkopf, O. Bachem, et al. Disentangling factors of variations using few labels. 2019.

[21] S. Lundberg and S. Lee. A unified approach to interpreting model predictions. *Computing Research Repository (CoRR)*, 2017.

[22] R. Luss, P. Chen, A. Dhurandhar, P. Sattigeri, K. Shanmugam, and C. Tu. Generating contrastive explanations with monotonic attribute functions. *CoRR*, abs/1905.12698, 2019.

[23] K. Martin, A. Liret, N. Wiratunga, G. Owusu, and M. Kern. Developing a catalogue of explainability methods to support expert and non-expert users. In M. Bramer and M. Petridis, editors, *Artificial Intelligence XXXVI*, pages 309–324, Cham, 2019. Springer International Publishing.

[24] E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh. Disentangling disentanglement in variational autoencoders. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4402–4412, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[25] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, Oct 2019. ISSN 1091-6490. doi: 10.1073/pnas.1900654116. URL `http://dx.doi.org/10.1073/pnas.1900654116`.

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

[27] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 625–632, 2005.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[30] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. *Computing Research Repository (CoRR)*, 2016.

[31] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. 2018.

[32] M. T. C. Ribeiro. LIME software. Available from `https://github.com/marcotcr/lime`, 2020.

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

[34] S. M. Shankaranarayana and D. Runje. Alime: Autoencoder based approach for local interpretability. In *IDEAL*, 2019.

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 2014.

[36] S. Tan, R. Caruana, G. Hooker, and Y. Lou. Distill-and-compare. *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society - AIES '18*, 2018. doi: 10.1145/3278721.3278725. URL `http://dx.doi.org/10.1145/3278721.3278725`.

[37] E. Tjoa and C. Guan. A survey on explainable artificial intelligence (XAI): Towards medical XAI, 2019.

[38] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103, 2008.

[39] S. Welling, H. Refsgaard, P. Brockhoff, and L. Clemmensen. Forest floor visualizations of random forests. *arXiv:1605.09196*, 05 2016.

[40] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015. URL `http://arxiv.org/abs/1512.04150`.

A DBA

This appendix contains additional material related to Section 2. We first present experiments illustrating the behavior of DBA-Tab, and then we provide additional details that were omitted from the description of DBA-Att in Section 2.1.

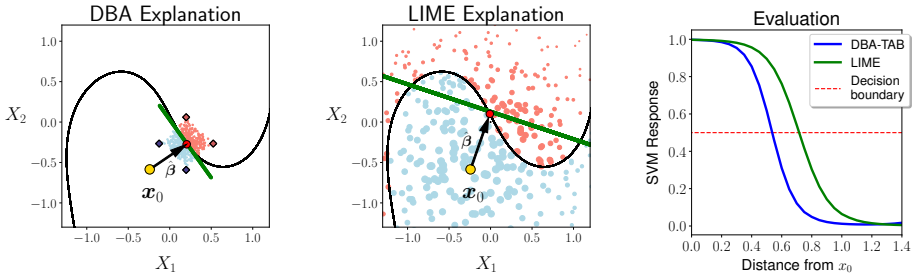
A.1 Experiments with DBA-Tab

In this section we provide additional experiments on tabular data for the simplest version of our method, DBA-Tab. We compare with LIME and report similar aggregate statistics as for the two main experiments in the paper. In the first experiment we illustrate a 2D toy case (the Moons data set), in the second experiment we consider a simplified tabular version of the AIris experiment from Section 3.1, and in the third experiment we provide a real-world example where DBA-Tab and LIME draw opposite conclusions about the importance of a particular feature.

A.1.1 Moons Data: a 2D Toy Example

The Moons distribution is a standard toy example that generates two moon-shaped classes in two dimensions that are not linearly separable. We sample 1000 points from this distribution with noise parameter 0.15, as implemented in Scikit-learn [28]. We split the dataset in $n = 600$ training samples and 400 testing examples, and train a support vector machine (SVM) with radial basis function kernel $e^{-\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}'\|^2}$ and regularization parameter $C = 1.0$ on the standardized training data. DBA-Tab can work directly with the decision boundary for the SVM, but LIME requires probabilities to produce its explanations, which are not directly available from an SVM. We therefore map the SVM margins to a probability by Platt scaling [29] based on 5-fold cross-validation on the training set, as implemented in Scikit-learn. It is known that Platt scaling can change the decision boundary, but in the present case agreement between the classifications based on the probabilities and those of the original SVM is 99.8% on the test set, so the change is minor. The resulting classifier achieves 98% accuracy on the test set.

Experiments and Results For DBA-Tab we use $k = m = 500$ and we take the grid of possible values for r to be $\mathcal{R} = \{0.2, 0.3, \ldots, 1.5, 2, 2.5, \ldots, 5\}$. This $\mathcal{R}$ is different from Section 3 to make sure all $r \in \mathcal{R}$ are large enough to be visualized easily in Figure 6a. The LIME paper [30] does not specify how to sample for tabular data with continuous features, so we follow the approach in the LIME software [32] and



(a) DBA-Tab

(b) LIME. The size of points reflects their weight when fitting LIME's local surrogate model.

(c) SVM probabilities as we move in direction $\hat{\boldsymbol{\beta}}/\|\hat{\boldsymbol{\beta}}\|$ from $\boldsymbol{x}_0$

Figure 6: Comparison of DBA-Tab and LIME solutions for the Moons data

sample from a multivariate normal distribution $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\Sigma})$, where $\hat{\boldsymbol{\mu}}$ is the empirical mean of the train set and $\hat{\Sigma} = \text{diag}(s_1^2, s_2^2)$ is a diagonal matrix based on the empirical variances s_1^2 and s_2^2 of X_1 and X_2 in the training data. For LIME we sample $m = 500$ points. We further use the standard weights $\exp(-\|\boldsymbol{x} - \boldsymbol{x}_0\|^2/\sigma^2)$ for the default choice $\sigma = 0.75\sqrt{d}$ for dimension $d = 2$. The standard LIME implementation uses the LASSO to preselect a small subset of the features before fitting its surrogate model, but since this is very low-dimensional toy data, we omit this preselection step.

We first illustrate the difference between DBA-Tab and LIME by explaining a single data point, as shown in Figure 6. We see that DBA-Tab samples in a region centered on the decision boundary, whereas LIME generates samples centered on $\boldsymbol{x}_0$ and then weighs them based on their distance from $\boldsymbol{x}_0$. The local surrogate model for DBA-Tab fits the part of the decision boundary that is closest to $\boldsymbol{x}_0$, whereas the surrogate model for LIME produces an average over a larger region of the decision boundary that is not sufficiently local to be approximated well by a linear model. It may be possible in this example to get a better approximation of the decision boundary by decreasing σ in an ad hoc manner depending on the distance of $\boldsymbol{x}_0$ to the decision boundary, but this would go against the spirit of LIME and it does not seem likely that such tuning would be possible in general. Another effect, which is not very visible in the figure, is that LIME is biased to avoid mistakes on the class of $\boldsymbol{x}_0$. This happens because samples from the same class as $\boldsymbol{x}_0$ tend to be closer to $\boldsymbol{x}_0$ than samples from the other class, and therefore receive a higher weight. As Figure 6c shows, the DBA-Tab explanation also corresponds to a direction that crosses the decision boundary faster than the explanation for LIME.

Table 3: Moons evaluation statistics, averaged over the whole training set

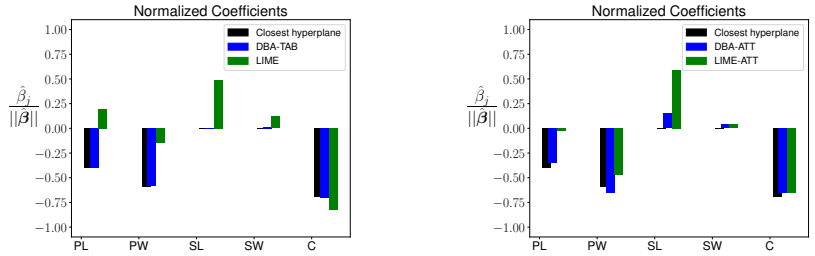
	DBA Fidelity	LIME R^2-Fidelity	Class Balance	DB Distance
DBA-Tab	92.9%	-	49.5%	**0.67**
LIME	-	33.3%	49.4%	0.81

The general pattern that DBA-Tab points more directly at the decision boundary is confirmed by Table 3, which shows aggregate statistics when explaining all points from the training data. We see that the mean fidelity for DBA-Tab is still high, especially compared to LIME, but a little lower than in the AIris and CelebA experiments from Section 3, which suggests that the decision boundary of the SVM in the current experiment is locally less linear. Class balance of the generated samples of both methods is close to 50%. We note however that this does not mean that the LIME sample equally represents both classes, because samples from the class of $\boldsymbol{x}_0$ are generally closer and therefore receive a higher weight than samples from the other class.

A.1.2 A Tabular Simplification of AIris

The AIris experiment from Section 3.1 jointly evaluates all components of DBA-Att, which includes the CNN that is being explained, the VAE and the annotators. Here we provide a greatly simplified tabular version of the experiment, which strips away as many of the complications as possible and only evaluates DBA-Tab. Since the results are similar, we conclude that the simplified experiment captures the essential parts of what is going on.

In our tabular simplification, no images are generated. Instead, we directly observe the five parameters: the observations $\boldsymbol{x} = (\text{PL}, \text{PW}, \text{SL}, \text{SW}, \text{C})$ come from the same uniform distribution as in Section 3.1 and are assigned to classes A and B according to the same rule from (1). In fact, we do not resample, and use exactly the same latent parameters that were used for the sample from Section 3.1 to obtain a train set of size $n = 4000$ and a test set of size 2000.



(a) Explanations for DBA-Tabband LIME in simplified experiment from Appendix A.1.2

(b) Explanations for DBA-Att and LIME-Att from full experiment from Section 3.1 (repetition of Figure 3b without the other methods)

Figure 7: Comparison of explanations from Section 3.1 to explanations from Appendix A.1.2

Experiments and Results We compare DBA-Tab and LIME to explain the true class boundary, with settings as similar to Section 3.1 as possible. Before running the explanation methods, we standardize all features as described in Appendix C.2. For DBA-Tab we set $k = 1000, m = 500$ and $\mathcal{R} = \{0.1, 0.2, \ldots, 0.9, 1, 1.5, 2, \ldots, 9.5, 10\}$. We instantiate LIME with $m = 500$ and $\sigma = 0.75\sqrt{d}$ for $d = 5$. There is one notable limitation: the true class assignments are deterministic, so we are feeding LIME

with class probabilities that are always either 0 or 1. This is allowed for LIME [30, Section 3.2], but it is different from our approach in Section 3.1 where we were feeding LIME the non-deterministic class probabilities produced by a CNN.

Figure 7 shows the explanations of DBA and LIME for a single data point. It compares the results when running DBA-Tab and LIME directly on the latent parameters (simplified experiment from current section) to running DBA-Att and LIME-Att on the corresponding image (full experiment from Section 3.1). The results for DBA are very similar in both cases: it recovers the true coefficients well. Performance appears to be slightly better in Figure 7a compared to Figure 7b, as might be expected given the simplified setup in which the latent parameters are directly accessible. For LIME, we also see strong similarities between the two figures. Surprisingly, its performance appears to be slightly worse in the simplified setting of Figure 7a compared to the harder case of Figure 7b. As we will see below, this is not representative of its general behavior when explaining other cases.

Table 4: Simplified Tabular AIris evaluation statistics, averaged over 50 test points

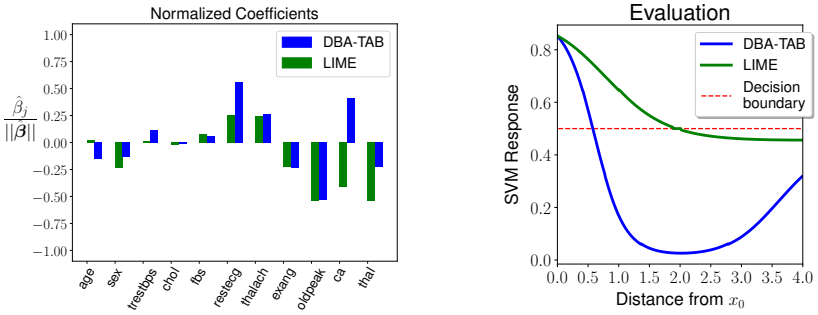
	DBA Fidelity	LIME R^2-Fidelity	Global Fidelity	Class Balance	Decision Boundary Distance	Cosine Similary-	Cosine Similarity+
DBA-Tab	95%	-	-	**50.1%**	**0.7**	**0.906**	**0.998**
LIME	-	33.9%	-	51.2%	0.9	0.665	0.773

Aggregate results are reported in Table 4, which is the analogue of Table 1 from the full experiment, evaluated on the same 50 test cases. For both methods we see that the Cosine Similarities are similar to the results from the full experiment, with small improvements between 0.05 and 0.09. Fidelity is slightly down for DBA but still high. For LIME fidelity is still low, but much better than in Section 3.1. It is not clear that the distance to the decision boundary in Table 4 can be directly compared to the Latent Distance in Table 1, but they are comparable nevertheless. Based on the similarities with the results from Section 3.1, we conclude that the results in the full AIris experiment are driven for a large part by the behavior of the explanation methods and not, for instance, by peculiarities of the CNN or VAE.

A.1.3 UCI Heart Disease Data: Opposite Conclusions from DBA and LIME

In this experiment we use the heart disease data from the UCI repository [6] to give a real-world example of a case where DBA-Tab and LIME lead to opposite conclusions on the importance of one of the features. This shows that it can really make a difference whether we explain the local decision boundary or the classifier probabilities around $\boldsymbol{x}_0$.

The heart disease data consist of 303 instances of patients that are labeled according to whether they suffered a heart disease or not. There are 13 features of mixed types, describing the health conditions of the patients: age, sec, cp, trestbps, chol, fbs, restecg, thalach, exang, slope, oldpeak, ca, thal. Since the linear surrogate models of both DBA-Tab and LIME become difficult to interpret on categorical features with more than two possible values, we simplify the setting as follows: we merge categories for features restecg and thal to make them binary, and we omit features cp and slope for which there



(a) Explanations

(b) SVM probabilities as we move in direction $\hat{\boldsymbol{\beta}}/\|\hat{\boldsymbol{\beta}}\|$ from $\boldsymbol{x}_0$

Figure 8: Comparison of DBA-Tab and LIME on a single patient from the heart disease data

is no natural way to merge categories. In case of restecg we merge categories 1 and 2, which both code for an abnormal electrocardiographic measurement, and in case of thal we merge categories 6 and 7, which both code for defects related to a blood disorder called thalassemia. All resulting features were standardized based on their means and standard deviations, which were estimated on all available data. We randomly split the data into a train set (242 instances) and a test set (61 instances), and train an SVM with the same parameters as in the Moons experiment from Appendix A.1.1, to reach a test accuracy of 85.24%. As in the Moons data set, we obtain probabilistic classifications by Platt scaling, which match the original SVM classifications in 99.6% of the cases in the whole data (train and test set together).

Experiments and Results Parameters for DBA-Tab are the same as in Section 3 and Appendix A.1.2. For LIME, we set the kernel width equal to $\sigma = 0.75\sqrt{d}$ for dimension $d = 11$ and generate $m = 500$ samples.

Figure 8a shows the resulting explanations on a single patient. We see that the explanations generally agree, except that DBA-Tab considers feature ca (the number of major vessels (0-3)) to have a large positive influence on the probability of heart disease, whereas LIME considers the same feature to have a large negative influence. This of course is possible, because the two explanation methods have different goals: explaining the local decision boundary versus explaining the classifier probabilities around $\boldsymbol{x}_0$. Figure 8b shows that, indeed, the LIME explanation is much worse than DBA-Tab at indicating the direction to the closest decision boundary point. So in this case the two goals are incompatible.

A natural follow-up question is whether it is typical that the difference between DBA-Tab and LIME is this large. As a small comfort, it turns out the distances to decision boundary are on average much more similar, as can be seen in the aggregate statistics from Table 5.

Table 5: Heart disease evaluation statistics, averaged over the whole data set (both train and test set)

	DBA Fidelity	LIME R^2-Fidelity	Class Balance	Decision Boundary Distance
DBA-Tab	94.3%	-	**49.2%**	**1.18**
LIME	-	46.4%	45.8%	1.30

A.2 Additional Details for DBA-Att: Modification to VAE Training

As mentioned in Section 2.1, we adjust the training procedure of the VAE to favor preservation of class probabilities. Let $c(\boldsymbol{x})$ be the probability that a classifier c assigns to class +1. (This can be replaced by binary classifications $f(\boldsymbol{x})$ if no probabilities are available.) Then what we want is for $c(\boldsymbol{x})$ and $c(\boldsymbol{x}')$ to be as close as possible when $\boldsymbol{x}'$ is the result of mapping $\boldsymbol{x}$ to the latent space and back using the VAE. During the training of the VAE we therefore monitor the stability of the probabilities on a hold-out set:

$$\frac{1}{n}\sum_{i=1}^{n} |c(\boldsymbol{x}_i) - c(\boldsymbol{x}'_i)|. \tag{2}$$

We initially tried to add this stability as an extra term to the objective function that is being minimized to train the VAE, but we found that this was harmful to the linearity of the resulting latent space. We therefore settled on an alternative solution, which was to keep the standard training procedure, but to calculate the *probability stability* (2) once per epoch and finally output the VAE parameters that minimize (2) during training. This significantly improved label stability both for AIris and for CelebA, without harming the linearity of the latent space.

B Other Methods

In this section, we provide the details of how we implemented the LIME-Att and CEM-MAF methods.

B.1 LIME-Att

We provide a variant of LIME [30] and ALIME [34] that works in the latent space of a VAE and can explain based on user-supplied annotations, in the same manner as DBA-Att. This procedure, which we call LIME-Att, can be viewed as a modification of DBA-Att in which we replace the DBA-Tab part by LIME. We keep the VAE and annotators the same as in DBA-Att, so we can use the same latent space and attribute space, as shown in Figure 1b.

The sampling procedure described in the LIME paper [30] is not applicable when sampling in the latent space, because it requires mappings back and forth between the vectors $\boldsymbol{z}$ and an interpretable binary representation, which is not available in the latent space. We therefore use the approach used in the LIME software [32] to sample from continuous features, as already described in Appendix A.1.1, except that now we apply it to the latent representations $\boldsymbol{z}$ instead of the inputs $\boldsymbol{x}$. The sample points are

then augmented with their corresponding predicted probabilities, which are obtained through the mapping $\boldsymbol{z} \mapsto \boldsymbol{x} \mapsto c(\boldsymbol{x})$. LIME-ATT further measures distance in the latent space: the weight of a sample point $\boldsymbol{z}$ is $\pi_{z_0}(\boldsymbol{z}) = \exp\left(-\|\boldsymbol{z} - \boldsymbol{z}_0\|^2/\sigma^2\right)$ where $\boldsymbol{z}_0$ is the latent space representation of $\boldsymbol{x}_0$. Following the LIME software [32], we always set $\sigma = 0.75\sqrt{l}$ where l is the dimensionality of $\mathcal{Z}$. Finally LIME-ATT fits a linear surrogate model g using weighted least squares in $\mathcal{A}$ by mapping the generated samples $\boldsymbol{z}$ to corresponding attribute vectors $\boldsymbol{a}$ using the annotators. The attributes of a_j in $\boldsymbol{a}$ are standardized based on their means and standard deviations in the LIME-ATT sample.

A notable difference between our approach and the LIME paper [30] is that we do not impose sparsity of the linear surrogate model g in a pre-selection step with the LASSO, which is not needed because the number of user-specified attributes is small enough to be interpretable in all our experiments. The LIME software [32] further offers the possibility to add L_2-penalization, which we omit for the same reason.

B.2 CEM-MAF

The Contrastive Explanation Method with Monotonic Attribute Functions (CEM-MAF) [22] is an extension of CEM [5], which explains the prediction for $\boldsymbol{x}_0$ by generating a contrastive example $\boldsymbol{x}_\mathrm{c}$ that is similar to $\boldsymbol{x}_0$ but differs in an informative way. In the CEM framework there are two types of contrastive examples. The first type is *pertinent positive*, in which case $\boldsymbol{x}_\mathrm{c}$ corresponds to removing as many features from $\boldsymbol{x}_0$ as possible while maintaining the same class: $f(\boldsymbol{x}_\mathrm{c}) = f(\boldsymbol{x}_0)$. The second type is *pertinent negative*, which means that $\boldsymbol{x}_\mathrm{c}$ corresponds to adding as few features to $\boldsymbol{x}_0$ as possible in order to change the class $f(\boldsymbol{x}_\mathrm{c}) \neq f(\boldsymbol{x}_0)$. Although CEM and CEM-MAF do not produce pertinent negatives that lie on the decision boundary, the idea is similar in spirit to our DBA approach of identifying the fastest direction from $\boldsymbol{x}_0$ to get to a point on the decision boundary. Other similarities are that CEM-MAF can learn user-specified attributes from user annotations and can measure distance in the latent space of a VAE. We therefore compare to the CEM-MAF pertinent negative method.

CEM-MAF requires a VAE (or GAN) to map back and forth between inputs $\boldsymbol{x}$ and latent representations $\boldsymbol{z}$. In our experiments we use the same VAE for both DBA-ATT and CEM-MAF for a fair comparison. Let $c(\boldsymbol{x})$ be the probability that $\boldsymbol{x}$ is in class $+1$ according to a probabilistic binary classifier c. Then CEM-MAF for pertinent negatives minimizes the following objective:

$$\min_{\boldsymbol{z}_\mathrm{c} \in \mathcal{X}} \quad F_{C,\kappa}(\boldsymbol{x}_\mathrm{c}) + \eta \|\boldsymbol{x}_\mathrm{c} - \boldsymbol{x}_0\|_2^2 + \nu \|\boldsymbol{z}_\mathrm{c} - \boldsymbol{z}_0\|_2^2 + A_{\gamma,\mu}(\boldsymbol{z}_\mathrm{c}), \tag{3}$$

where $\boldsymbol{x}_\mathrm{c}$ is the reconstruction in input space that corresponds to the latent representation $\boldsymbol{z}_\mathrm{c}$. Here the term $F_{C,\kappa}(\boldsymbol{x}_\mathrm{c})$ encourages $\boldsymbol{x}_\mathrm{c}$ to be classified as the opposite class of $\boldsymbol{x}_0$. If $c(\boldsymbol{x}_0) \geq 0.5$, i.e. $\boldsymbol{x}_0$ is classified as class $+1$, it is defined as

$$F_{C,\kappa}(\boldsymbol{x}_\mathrm{c}) = C \cdot \max\Big\{c(\boldsymbol{x}_\mathrm{c}) - (1 - c(\boldsymbol{x}_\mathrm{c})), -\kappa\Big\}$$

If $\boldsymbol{x}_0$ is classified as -1, then both occurrences of $c(\boldsymbol{x}_\mathrm{c})$ should be replaced by $1 - c(\boldsymbol{x}_\mathrm{c})$. The second and third term in (3) respectively minimize the distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}_\mathrm{c}$ in the input space and in the latent space. Finally, the last term enforces addition (and not removal) of interpretable attributes:

$$A_{\gamma,\mu}(\boldsymbol{z}_\mathrm{c}) = \gamma \sum_{i=1}^{p} \max\left\{h_i(\boldsymbol{x}_0) - h_i(\boldsymbol{x}_\mathrm{c}), 0\right\} + \mu \sum_{i=1}^{p} |h_i(\boldsymbol{x}_\mathrm{c})|.$$

There is a function $h_i : \mathcal{X} \to \mathbb{R}$ for each interpretable attribute $i = 1, \ldots, p$. A higher value $h_i(\boldsymbol{x})$ indicates stronger presence of attribute i in input $\boldsymbol{x}$, and in typical usage they range from 0 to 1. Each function h_i is learned from user-supplied annotations by a separate neural network. Thus these functions are similar to the annotators in DBA-Att, except that they are non-linear and operate in the input space instead of the latent space.

The objective (3) is non-convex. Optimization with standard stochastic optimization procedures therefore does not always give the same solution. It depends on hyperparameters C, κ, η, ν, γ and μ, which the user needs to tune to optimize convergence. An automatic tuning procedure is available for C, which increases computational cost substantially. The algorithm also bears the unavoidable computational cost of the training and tuning of p (convolutional) neural networks with many parameters, to serve as annotators. By comparison, the annotators in DBA-Att take much less computation and fewer user-annotations, because they only require fitting linear models with logistic regression.

Tuning the hyperparameters for CEM-MAF is non-trivial and requires elaborate computationally expensive experimentation. We select hyperparameters for each experiment individually to guarantee that the optimizer always converges to a small objective value. To make experiments for many explanations and algorithms computationally feasible, we do not tune C automatically but instead find for each experiment the smallest C that yields a small objective value for all inputs that we explain. We select values for the other hyperparameters according to the authors' suggestions [22] and by observing convergence.

C AIris Experiment

In this section, we provide additional information about the AIris experiment reported in Section 3.1. We first discuss how the images were generated. Then, in Section C.2, we show how standardizing the features changes the coefficients of the hyperplanes from (1). In Section C.3 we provide details on how we trained the VAE, CNN and annotators. Section C.4 contains details for LIME-Att, CEM-MAF and Global. And finally, Section C.5 shows additional results on the best and worst performance of DBA-Att as well as a PCA projection of its sample S onto two dimensions.

C.1 Image Generation: Details and Examples

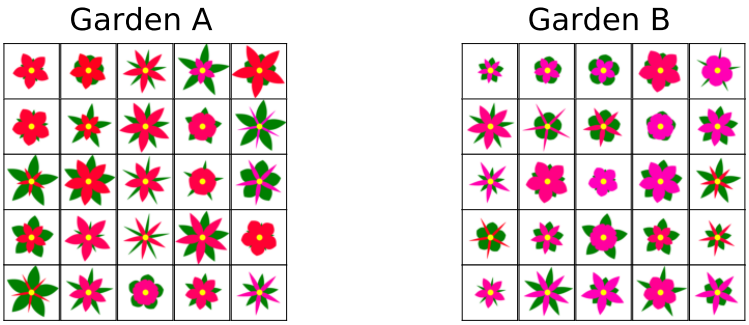
The image generation program for the AIris data generates images of flowers using third-order connected Bezier curves. It takes as input the parameters from Table 6, and creates flowers with an equal number of petals and sepals using standard open source visualization software [12]. In order for the flowers to be realistic we had to bound the ranges of their shape parameters in the intervals shown in Table 6. Color of the petals is manipulated by a continuous mixing parameter C that interpolates between red and magenta.

Flowers are generated by uniformly sampling parameters over their range of allowed values, and splitting them into classes A and B according to the non-linear rule (1). We sample 4000 images for training as well as 2000 test images. The proportion of class A in the training set is 47.43%, so the two classes are approximately balanced. Figure 9 shows examples for the two resulting "gardens". A first impression is that color (C) separates the classes, which is true since C is an non-zero coefficient of the

Table 6: AIris parameters

		range	mean	sd
Petal length	PL	$[0.3, 0.7]$	0.5	0.115
Petal width	PW	$[0.1, 0.7]$	0.4	0.173
Sepal length	SL	$[0.3, 0.7]$	0.5	0.115
Sepal width	SW	$[0.1, 0.7]$	0.4	0.173
Color	C	$[0.1, 0.8]$	0.45	0.202

first hyperplane. However more features are responsible for separating the gardens and it is hard for a human eye to capture the ground truth.



(a) Examples from class A (b) Examples from class B

Figure 9: Examples of AIris data

C.2 Standardizing Features and Coefficients of the True Hyperplanes

As mentioned in a footnote [2] in Section 3.1, the coefficients of the hyperplanes from (1) need to be standardized before comparing them to the explanations produced by the explanation methods. This works as follows: we first consider standardized features $\widetilde{\mathrm{PL}}, \widetilde{\mathrm{PW}}, \widetilde{\mathrm{SL}}, \widetilde{\mathrm{SW}}, \widetilde{\mathrm{C}}$, which are obtained from the parameters by subtracting the mean and dividing by the standard deviation of the parameter. The ranges for the parameters are specified in Table 6, which also shows their means and standard deviations, where we use that the standard deviation of a uniform distribution on an interval $[a, b]$ is $(b-a)/\sqrt{12}$. The true decision boundary from (1) then becomes the rule to assign to class A if

$$0.038\widetilde{\mathrm{PL}}+0.057\widetilde{\mathrm{PW}}+0.067\widetilde{\mathrm{C}} < .0545 \quad \text{and} \quad 0.038\widetilde{\mathrm{PL}}+0.057\widetilde{\mathrm{PW}}+0.038\widetilde{\mathrm{SL}} > -.062,$$

In particular, the coefficients of the hyperplanes are obtained from the unstandardized coefficients by multiplying by the standard deviations of the corresponding features.

C.3 Global Mappings: VAE, CNN and Annotators

Here we provide details on how we trained the VAE, CNN and annotators for the AIris experiment. These are the global mappings between spaces in Figure 1b.

C.3.1 VAE

We trained a convolutional variational autoencoder (CVAE) that is a slight modification of the CVAE of Hou et al. [11]. The objective function being optimized has two terms: reconstruction loss (implemented with binary cross-entropy) and Kullback-Leibler divergence, weighted by factors 0.5 and 1.2 respectively. The CVAE is implemented in TensorFlow 2.2. The latent space has 10 dimensions.

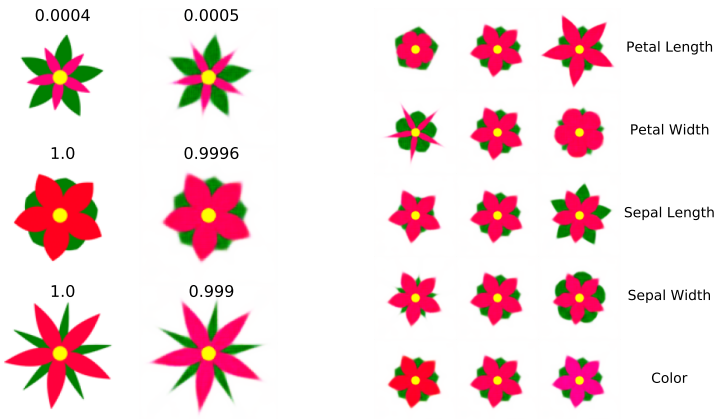
Architecture The architecture we used is a slight modification of the CVAE of Hou et al. [11]. Our inputs and outputs are 128×128 RGB images. The encoder part of the CVAE consists of 5 blocks with each block consisting of a convolutional layer followed by a Batch Normalization layer and a LeakyReLU layer ($\alpha = 0.3$). The convolutional layers of the 5 blocks all have kernel size 4×4, a stride parameter of 1 and filter sizes 32, 64, 128, 256 and 512 with that order. The 5 blocks are then followed by a global average pooling operation and a dense layer with output dimension 20 that is interpreted as producing 10 means and 10 corresponding log-variances. The decoder starts with a dense layer of 4096 neurons and then mirrors the architecture of the encoder with upsampling and nearest neighbor interpolation. Filter sizes for the 5 convolutional layers are 344, 64, 32, 16 and 3, with kernel size 3×3. The first 4 convolutional layers are also followed by Batch Normalization and LeakyReLU operations. The last convolutional layer has a sigmoid activation function, since we optimize for binary cross-entropy. We also experimented with MSE reconstruction loss but achieve better attribute vector quality with the former so we report results with binary cross-entropy reconstruction loss.

Training Details We use the Adam optimizer with learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and zero decay. We train for 200 epochs with a batch size of 500 and store the weights that minimize the probability stability from (2), as discussed in Appendix A.2.

Qualitative Evaluation Figure 10a shows examples of the reconstruction quality when mapping $\boldsymbol{x} \mapsto \boldsymbol{z} \mapsto \boldsymbol{x}'$ as well as the corresponding CNN probabilities $c(\boldsymbol{x})$ and $c(\boldsymbol{x}')$.

C.3.2 CNN

Our CNN architecture is almost the same as the encoder architecture of the VAE, since we noticed good feature extraction behavior under this design. What differs is that the last dense layer of the network consists of 200 neurons. The output layer has a sigmoid activation function and outputs the probability $c(\boldsymbol{x})$ of $\boldsymbol{x}$ being of class A. We optimize using stochastic gradient descent (SGD) with a learning rate of 10^{-4}. The CNN was trained for 200 epochs with a batch size of 128. We use early stopping and store the weights in the epoch that validation binary cross entropy loss is minimized (here epoch 156). This gives high accuracy: 99.33% and 98.75% in the train and test set, respectively.



(a) Original vs reconstruction (b) Quality of attribute vectors

Figure 10: Qualitative evaluation of global mappings. On the left the original images and their CNN probabilities are compared with the reconstructions produced by the CVAE. On the right the original image (middle column) is varied along the direction of each learned attribute.

C.3.3 Annotators

DBA-Att annotators are L_2-penalized logistic regression models which learn the mappings $\mathcal{Z} \to \mathcal{A}$.

Because of the danger of entanglement discussed in Section 4, it is important to verify that the annotators indeed express the concepts that the user had in mind. This can be seen in Figure 10b: the middle column always shows the same example image and the other two columns show the effect of changing one of the attributes by a unit step in the latent space. Concretely, we map the input image $\boldsymbol{x}$ to a latent representation $\boldsymbol{z}$. Then add (or subtract) $\hat{\boldsymbol{\theta}}_j / \|\hat{\boldsymbol{\theta}}_j\|$ to $\boldsymbol{z}$ and map back to a corresponding input $\boldsymbol{x}'$, where $\hat{\boldsymbol{\theta}}_j$ are the coefficients of the annotator for attribute j. The left column corresponds to decreasing an attribute; the right column to increasing it.

C.4 Details for LIME-Att, CEM-MAF and Global

LIME-Att For a general description of LIME-Att see Appendix B.1. We run LIME-Att with $m = 500$ and kernel width $\sigma = 0.75 \cdot \sqrt{10}$, because the dimension of the latent space is 10.

CEM-MAF For a general description of CEM-MAF, see Appendix B.2. We optimize its objective (3) with SGD using a constant learning rate of 0.0001, which seems to yield stable convergence. To save computation, we run 500 SGD iterations for all explanations since CEM-MAF usually converged after about 400 iterations. Before running the experiment for all methods we have tuned CEM-MAF parameters empirically and according to the authors' suggestions [22]. We report results with the best set of parameters found, which is $\{C, \kappa, \eta, \nu, \gamma, \mu\} = \{2500, 5, 0.1, 1, 100, 100\}$.

CEM-MAF returns $\boldsymbol{x}_\mathrm{c}$ which is a contrastive example from the opposite class of $\boldsymbol{x}_0$. Thus the corresponding latent direction $\boldsymbol{\delta} = \boldsymbol{z}_\mathrm{c} - \boldsymbol{z}_0$ points towards the decision boundary in the latent space. It is also possible to obtain a coefficient vector $\hat{\boldsymbol{\beta}}$ from CEM-MAF, which represents the importance of various attributes in distinguishing $\boldsymbol{x}_\mathrm{c}$ from $\boldsymbol{x}_0$. Following its authors [22], we obtain this vector as

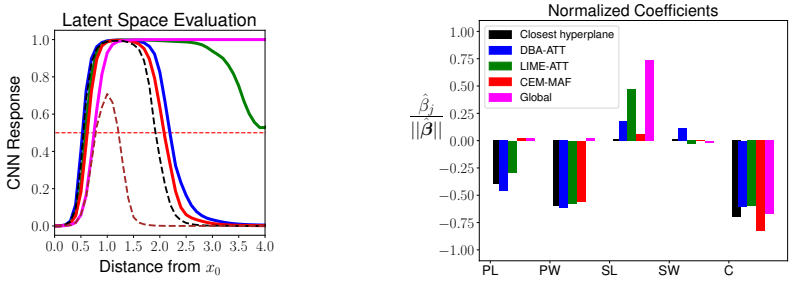
$$\hat{\boldsymbol{\beta}} = \begin{pmatrix} h_1(\boldsymbol{x}_\mathrm{c}) - h_1(\boldsymbol{x}_0) \\ \vdots \\ h_p(\boldsymbol{x}_\mathrm{c}) - h_p(\boldsymbol{x}_0) \end{pmatrix}.$$

We use this vector as the final CEM-MAF explanation in Figure 3b as well as to compute Cosine Similarity+ and Cosine Similarity- measures in Table 1.

Global Finally, Global is trained on the training data with access to the original latent parameters PL, etc. The latent parameters are standardized based on their empirical means and standard deviations. We then use unpenalized logistic regression to fit a global surrogate model to predict the class labels of the CNN.

C.5 Additional Results: Best Case, Worst Case and PCA

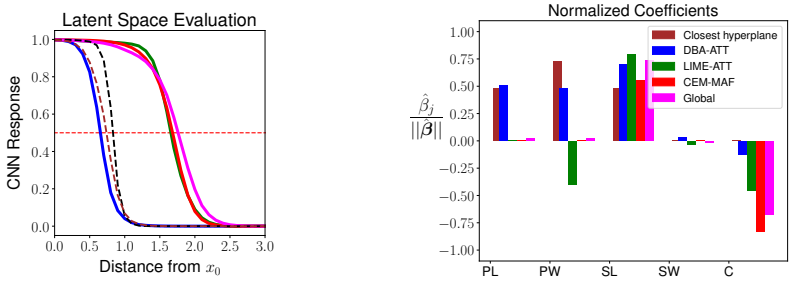
Best-case and Worst-case Examples Here we show the results for two more cases, selected from the 50 test images that were also used in Table 1: one where DBA-Att performs the worst, in Figure 11, and one where DBA-Att shows a large advantage over other methods, in Figure 12.



(a) CNN probabilities moving in direction $\boldsymbol{\delta}/\|\boldsymbol{\delta}\|$ from $\boldsymbol{x}_0$

(b) Explanation coefficients $\hat{\boldsymbol{\beta}}$

Figure 11: Worst-case example



(a) CNN probabilities moving in direction $\boldsymbol{\delta}/\|\boldsymbol{\delta}\|$ from $\boldsymbol{x}_0$

(b) Explanation coefficients $\hat{\boldsymbol{\beta}}$

Figure 12: Best-case example

In all 50 cases DBA-Att found a direction in the latent space that crossed the decision boundary faster or at least as fast as all other methods. The worst-case example in Figure 11 therefore shows a case where the distance to the decision boundary is approximately the same for DBA-Att, LIME-Att and CEM-MAF. Although the distances to the decision boundary are the same, Figure 11b shows that the DBA-Att coefficients still match slightly better with those of the closest hyperplane.

In the best-case example in Figure 12a, we see that DBA-Att succeeds in approximating the closest hyperplane, while other methods fail to approximate any of the hyperplanes. From Figure 12b we can conclude that the other methods average the important features for both hyperplanes in a misleading manner. For instance, all other methods fail to identify the relevance of PL, which is an important feature in both hyperplanes but with opposite sign.

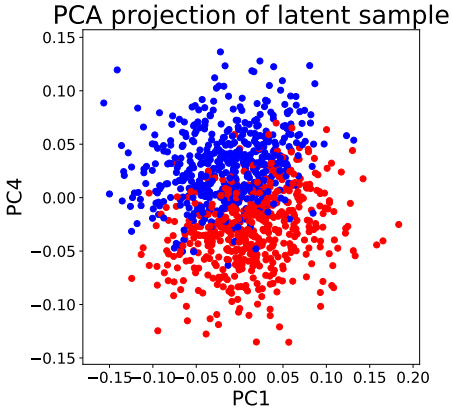


Figure 13: PCA projection of local sample corresponding to the explained example in Section 3.1

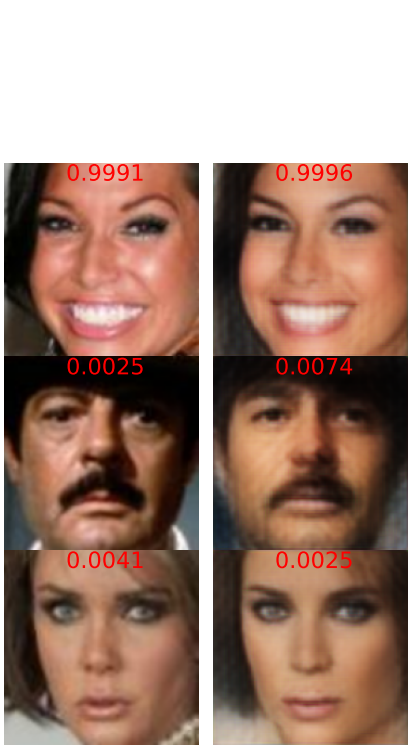
PCA Figure 13 shows the PCA projection of the DBA-Att sample S described in Section 3.1 and corresponding to Figures 2 and 3. We plot the components for which the sample is most separable (first and fourth component) and color by predicted class label of the CNN. This visualization shows that the simulation step of DBA-Att creates a reasonably separable sample on the decision boundary of the CNN, which can be approximated with high fidelity by a linear surrogate model.

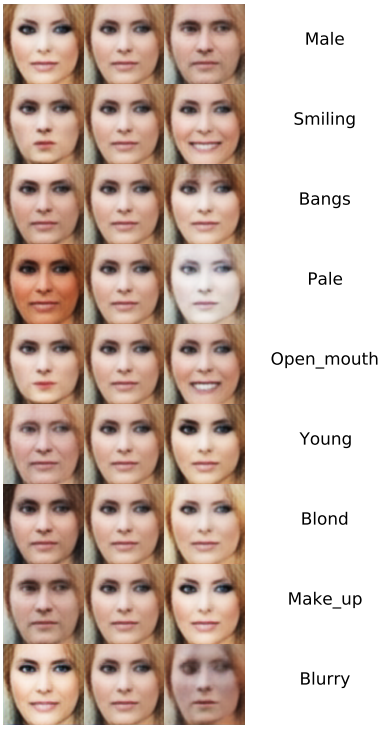
D CelebA

In this section we provide further details regarding the CelebA experiment from Section 3.2.

D.1 Global Mappings: CNN, VAE and Annotators

We first provide details on how we trained the VAE, CNN and annotators for CelebA. These are the global mappings between spaces in Figure 1b.





(a) Original vs reconstruction (b) Quality of attribute vectors

Figure 14: Qualitative evaluation of global mappings. On the left the original images and their CNN probabilities are compared with the reconstructions produced by the DFC-VAE. On the right the original image (middle column) is varied along the direction of each learned attribute.

D.1.1 VAE

We use the exact same architecture described in Section C of this appendix, but we modify the objective function as suggested by Hou et al. [11]. They propose to add an extra term called the deep feature consistency (DFC) loss, giving a DFC-VAE. This term involves features of a selected set of layers from a pre-trained large scale architecture and minimizes the reconstruction error on these features. We refer to Hou et al. [11] for further details. For our experiment we employ relu1_1, relu2_1, relu3_1 layers of the pre-trained VGGNet [35]. We further use a latent dimension of 100, batch size 64 and the same optimizer as in AIris. We train for 10 epochs with learning rate 10^{-3} and monitor the probability stability from (2). Figure 14a illustrates that the CVAE achieves good reconstruction quality when mapping $\boldsymbol{x} \mapsto \boldsymbol{z} \mapsto \boldsymbol{x}'$ and that the corresponding CNN probabilities $c(\boldsymbol{x})$ and $c(\boldsymbol{x}')$ are close.

D.1.2 CNN

The employed architecture for the CNN consists of 4 convolutional layers with filter sizes $\{16, 32, 64, 128\}$, kernel size 3×3 and ReLU activation. Each of the layers is followed by a 2-dimensional Max Pooling operation with pool size 2×2. After the last convolutional layer a Global Average Pooling operation follows. Next, there is a dense layer with 64 neurons and ReLU activation which is followed by Batch Normalization. Finally, the output layer yields the probability for the smiling female category through a sigmoid activation function. The network is trained for 10 epochs with learning rate $10^{-}3$ and batch size 64. We use the same Adam optimizer as for the VAE. The resulting CNN achieves train accuracy 93.2% and test accuracy 91.5%.

D.1.3 Annotators

DBA-Att annotators are L_2-penalized logistic regression models which learn the mappings $\mathcal{Z} \to \mathcal{A}$. For CelebA, we train them on a random subset of the training data with 9000 examples with regularization parameter $\alpha = 0.1$. This shows that we do not need the entire training set to train the annotators. In Figure 14b the quality of these vectors can be observed.

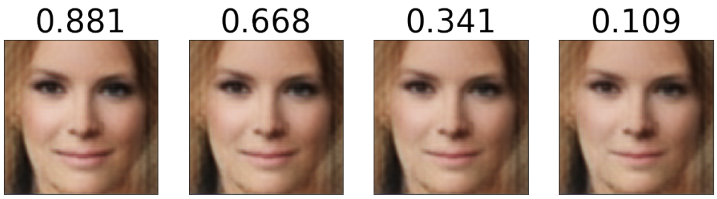
Because of the danger of entanglement discussed in Section 4, it is important to verify that the annotators indeed express the concepts that the user had in mind. This can be seen in Figure 14b: the middle column always shows the same example image and the other two columns show the effect of changing one of the attributes by a unit step in the latent space. Concretely, we map the input image $\boldsymbol{x}$ to a latent representation $\boldsymbol{z}$. Then add (or subtract) $\hat{\boldsymbol{\theta}}_j / \|\hat{\boldsymbol{\theta}}_j\|$ to $\boldsymbol{z}$ and map back to a corresponding input $\boldsymbol{x}'$, where $\hat{\boldsymbol{\theta}}_j$ are the coefficients of the annotator for attribute j. The left column corresponds to decreasing an attribute; the right column to increasing it. In this case we can see that the attributes approximately represent the intended concepts.

D.2 Details for DBA-Att, LIME-Att and CEM-MAF

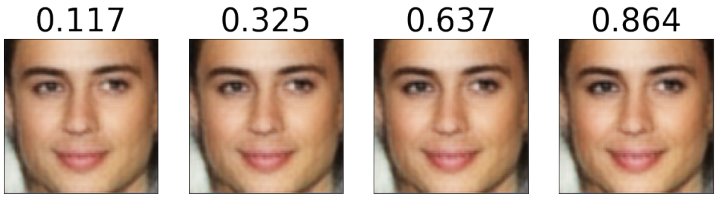
DBA-Att DBA-Att runs with $k = m = 500$ and the same $\mathcal{R} = \{0.1, 0.2, \ldots, 0.9, 1, 1.5, 2, \ldots, 9.5, 10\}$ as in the AIris experiment. To speed up the detection step of the algorithm, we do not select the k closest points from the entire training data, but only from the subset of 9000 instances that were also used to train the annotators.

LIME-Att For a general description of LIME-Att see Appendix B.1. For LIME-Att we use $m = 500$ and $\sigma = 0.75\sqrt{100}$, because the dimension of the latent space is 100.

CEM-MAF We experiment with CEM-MAF hyperparameters as described in Appendix B.2. Reported results are with $\{C, \kappa, \eta, \nu, \gamma, \mu\} = \{500, 5, 1, 1, 100, 100\}$ which is quite similar to the hyperparameters used by Luss et al. [22] in their experiment in which they apply CEM-MAF to CelebA. The objective (3) is optimized in 1000 epochs of SGD using polynomial decay for the learning rate with starting value 10^{-2} and power 0.5. With this set-up, CEM-MAF converged to a PN for all of the 30 explained cases.



(a) Transition from class A to class B



(b) Transition from class B to class A

Figure 15: The path of $\boldsymbol{x}_0$ towards the decision boundary of the CNN when moving in the latent direction $\boldsymbol{\delta}$ proposed by DBA-Att. In both cases, $\boldsymbol{x}_0$ is shown in the left-most image. The numbers above the images are the predicted probabilities of class A for the CNN.

D.3 Additional Result: Visualization of the Path Towards the Decision Boundary

We close with an alternative visualization of latent space evaluations like the one shown in Figure 5c. Figure 15 shows two images $\boldsymbol{x}_0$ and how they change when we move in the direction $\boldsymbol{\delta}$ produced by DBA-Att. We observe in both cases that DBA-Att has found a very minimal variation to make the CNN change its the classification.