

The Gaussian Discriminant Variational Autoencoder (GdVAE): A Self-Explainable Model with Counterfactual Explanations

Anselm Haselhoff^{1,2} , Kevin Trelenberg¹,
Fabian Küppers³ , and Jonas Schneider³

¹ TrustIn.AI Lab, Ruhr West University of Applied Sciences, Germany

² TML Lab, The University of Sydney, Australia, ³ e:fs TechHub GmbH, Germany
`{name.surname}@hs-ruhrwest.de, {name.surname}@efs-techhub.com`
<https://trustinai.github.io/gdvae>



Fig. 1: FFHQ high-resolution (1024×1024) counterfactuals x^δ for smiling.

Abstract. Visual counterfactual explanation (CF) methods modify image concepts, *e.g.*, shape, to change a prediction to a predefined outcome while closely resembling the original query image. Unlike self-explainable models (SEMs) and heatmap techniques, they grant users the ability to examine hypothetical "what-if" scenarios. Previous CF methods either entail post-hoc training, limiting the balance between transparency and CF quality, or demand optimization during inference. To bridge the gap between transparent SEMs and CF methods, we introduce the GdVAE, a self-explainable model based on a conditional variational autoencoder (CVAE), featuring a Gaussian discriminant analysis (GDA) classifier and integrated CF explanations. Full transparency is achieved through a generative classifier that leverages class-specific prototypes for the downstream task and a closed-form solution for CFs in the latent space. The consistency of CFs is improved by regularizing the latent space with the explainer function. Extensive comparisons with existing approaches affirm the effectiveness of our method in producing high-quality CF explanations while preserving transparency. Code and models are public.

Keywords: Self-explainable generative model · counterfactual explanation · variational autoencoder · Riemannian metric · manifold traversal

1 Introduction

Deep neural networks (DNNs), such as generative adversarial networks (GANs) for image generation [25] and DNN classifiers [46], have achieved notable success. However, they suffer from limited interpretability, often being considered black boxes with decision processes not well understood by humans.

Generative explanation methods identify meaningful latent space directions related to independent factors of variation (*e.g.*, shape). Previous work finds these directions by enforcing disentanglement during training or analyzing the latent space [3, 8, 9, 20, 25, 38, 39, 42]. Explanations are obtained by visualizing the effect of changes in the latent space. Generative models are also used in *counterfactual* (CF) reasoning, which answers questions like, "How can the example be changed to belong to category B instead of A?". This allows users to explore hypothetical "what-if" scenarios [14]. Recent advances combine generative models and classifiers to generate CF explanations, with enhanced techniques focusing on realism and consistency [14, 15, 21, 26, 30, 40, 43]. However, many methods lack transparency, as the CF generation often relies on a separate black-box model, and the classifier itself may not guarantee transparency either.

Self-explainable models (SEMs) provide explanations alongside their predictions without the need for post-hoc training [1, 2, 6, 7, 13]. Many SEMs are based on prototype learning, using these transparent and often visualizable prototypes as a bottleneck in a white-box classifier. This white-box classifier (*e.g.*, linear predictor) is optimized end-to-end. However, generating CFs for these models is only feasible through post-hoc methods, potentially reducing transparency.

To bridge the gap between transparent SEMs and CF methods, we introduce GdVAE, a conditional variational autoencoder (CVAE) designed for transparent classification and CF explanation tasks. Full transparency is achieved with a generative classifier using class-specific prototypes and a closed-form solution for CFs in the latent space, inspired by Euclidean and Riemannian manifold perspectives. The prototype explanations come from the distributions provided by the CVAE's prior network, meaning the classifier has no additional trainable parameters. We solve the inference problem of the CVAE, which involves unknown classes, using expectation maximization that iteratively uses the classifier. Finally, we generate local CF explanations in the latent space using a transparent linear function that supports user-defined classifier outputs, and then use the decoder to translate them back to the input space. Joint training of the classifier and generative model regularizes the latent space for class-specific attributes, enabling realistic image and CF generation. An additional regularizer ensures consistency between query confidence and true confidence of the classified CF.

In summary, our contributions are: (i) We introduce a SEM for vision applications, based on a CVAE, with an intrinsic ability to generate CFs; (ii) We offer global explanations in the form of prototypes directly utilized for the downstream task, visualizable in the input space; (iii) We provide transparent, realistic, and consistent local CF explanations, allowing users to specify a desired confidence value; (iv) We conduct a thorough comparative analysis of our method, analyzing performance, consistency, proximity, and realism on common vision datasets.

2 Related Work

Since our work is a SEM with integrated visual CF explanations, we begin by outlining the categorization criteria. Subsequently, we review generative and CF explanations, as well as prototype-based SEMs tailored for vision tasks. Generative models naturally serve as an integral component of an explainer function used for generating CF images. Typically, this function is learned through probing the classifier and optimizing it for specific properties. In CF research, while various properties are discussed, *realism*, *proximity*, and *consistency* stand out as widely accepted criteria. To simplify, CFs should resemble natural-looking images (*realism*), make minimal changes to the input (*proximity*), and maintain query confidence consistency with the classifier’s predictions when used as input (*consistency*) [4, 14, 26, 43]. Similarly, in prototype-based SEMs, transparency is crucial, characterized by the visualization of prototypes (PT) in the input space and their utilization in a white-box classifier [13]. To align our work with CF methods and SEMs, we adopt the following predicates.

1. *Realism*: CFs should stem from the data manifold with a natural appearance.
2. *Consistency*: The explainer function $\mathcal{I}_F(x, \delta) : (\mathbb{R}^N, \mathbb{R}) \rightarrow \mathbb{R}^N$ should be conform with the desired classifier output $F(x^\delta) \approx F(x) - \bar{\delta} = \delta$, where $\bar{\delta}$ is the desired perturbation of the output function, δ the desired output, and $x^\delta = \mathcal{I}_F(x, \delta)$ the CF for the input x [43].
3. *Proximity*: The CF should minimally change the input.
4. *Transparency*: A model should use explanations (e.g., prototypes) as intrinsic parts of a white-box predictor, and they should be visualizable in input space.

Generative Explanations (a). The first group of approaches aims to explain pre-trained generative models (e.g., GANs). Directions for interpretable control can be derived through unsupervised [11, 22, 37, 49] or supervised [15, 42, 52] analysis of generative models. GANalyze [15] employs a pre-trained classifier to learn linear transformations in the latent space, whereas [42] directly use a linear classifier in the latent space to define the direction. Except for UDID [49], all mentioned methods use linear explainer functions for manifold traversal. Most of these methods, due to their linear explainer function, provide transparency in latent space manipulation. Transparent classification and CF generation aren’t their primary focus, though they can generate CFs without optimizing for factors like *realism*. Our method aligns with these post-hoc methods by using a transparent linear explainer function for CF generation. In contrast, our approach excels by more effectively regularizing the latent space through end-to-end training.

Visual Counterfactual Explanations (b). The second category of methods focuses on CF generation, optimizing *realism*, *proximity*, and *consistency*. EBPE [43] and its extension [14] explain pre-trained classifiers by using a GAN to generate CF images with user-defined confidence values. Similarly, works like [21, 23, 24, 26, 30, 40], train generators with a simpler consistency task, where the user pre-defines the class label only, without specifying the confidence. DiME [24] optimizes CFs iteratively, incurring significant computational costs. Unlike other methods, C3LT [26] only manipulates the latent space with neural networks,

Table 1: Comparison of explanation methods. "Design" column groups approaches according to the headings: (a), (b), and (c). The symbol \sim indicates that most methods use a transparent linear function for latent space traversal and may not be explicitly designed for generating CFs. Explanations are categorized into Counterfactuals ("CF") and Prototype-based ("PT"). \dagger : some works [7, 50] use alternating optimization.

| Design | Approach | Transparency | Explanation CF PT | Optimization |
|--------|----------------------------------|--------------|---------------------------|----------------------|
| (a) | [11, 15, 22, 37, 42, 49, 52] | \sim | \sim | post-hoc |
| (b) | [14, 21, 23, 24, 26, 30, 40, 43] | | \checkmark | post-hoc |
| (c) | [7, 13, 50] | \checkmark | \checkmark | end-to-end \dagger |
| | GdVAE (ours) | \checkmark | \checkmark \checkmark | end-to-end |

similar to methods in the first category, requiring access to a pre-trained generative model. A different line of research [16, 47] seeks to replace image regions based on distractor images of the CF class. In [30] and [21], the classifier and generator are closely coupled during training to enforce a latent space that encodes class-specific information. StyleEx [30], like [24], requires time-consuming inference-time optimization and classifier probing to identify influential coordinates for each input image. In contrast, ECINN [21] is unique in its use of a transparent linear explainer function and an invertible model. Our method is closely related to ECINN, with the distinction that they require a post-hoc analysis of the training data to determine the parameters of the explainer function. Consequently, unlike our model, they approximate the true decision function of their classifier for CF generation, resulting in a loss of transparency. In contrast, all the other methods described employ complex DNNs for CF generation and the classifier, limiting their transparency. Our approach mirrors these CF generation processes but stands out with a transparent, linear explainer function analytically linked to our white-box classifier's decision function.

Self-explainable Models (c). The classifier and CF generation of our GdVAE are closely tied to the same prototypical space. A line of works that comprises this prototype-based learning can be found in SEM research [2, 7, 13, 17, 36, 50, 51]. In [13], a categorization of SEMs was introduced, and our specific focus is on methods prioritizing the *transparency* property [7, 13, 50]. To maintain interpretability, these SEMs employ similarity scores that measure the likeness between features and prototypes within the latent space. Afterwards, these scores are employed within a linear classifier, which encodes the attribution of each prototype to the decision. Unlike ProtoPNet [7] and TesNET [50], ProtoVAE [13] uses end-to-end training, utilizing a model capable of decoding learned prototypes, resulting in a smooth and regularized prototypical space.

Our GdVAE employs one prototype per class with a linear Bayes' classifier, implicitly utilizing Mahalanobis distance instead of a 2-norm-based similarity. Unlike ProtoVAE, our SEM enhances transparency and CF generation, unifying these research areas effectively. Refer to Tab. 1 for an overview.

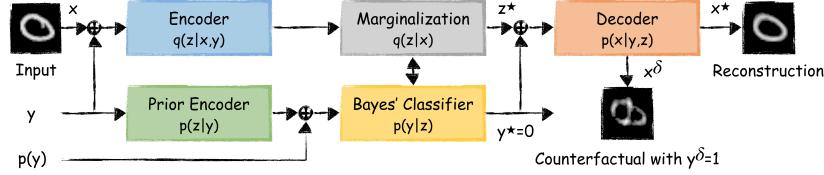


Fig. 2: The GdVAE has three branches: 1.) *Feature Detection & Reconstruction*: The encoder, akin to a recognition network in a CVAE, generates latent code z . During inference, with an unknown class y , the marginal $q(z|x)$ acts as a feature detection module. The decoder reconstructs the input image x using samples z^* from the marginal and y^* from the classifier. 2.) *Prior Encoder & Classifier*: The prior encoder learns the latent feature distribution independently of the input image, providing necessary distributions for the generative classifier. 3.) *Explanation*: During inference, the model generates a class prediction y^* and a latent variable z^* . The user requests a CF by defining a desired confidence value and uses a linear function $z^{\delta} = I_f(z^*, \delta)$ to modify z^* to z^{δ} . The CF x^{δ} is obtained by transforming z^{δ} to image space using the decoder. The CF illustrates crossing the decision boundary, showing features of digits 0 and 1.

3 Method

Notation. We address a supervised learning problem with input samples $x \in \mathbb{R}^N$ (e.g., images) and class labels $y \in \{1, \dots, K\}$. The latent variable $z \in \mathbb{R}^M$ is used for both autoencoding and classification. Model parameters θ and ϕ define the neural networks (NNs) for probabilistic models. For example, we use a Gaussian posterior $q_{\phi}(z|x, y) = \mathcal{N}(\mu_z(x, y; \phi), \Sigma_z(x, y; \phi))$, with $\mu_z(x, y; \phi)$ and $\Sigma_z(x, y; \phi)$ as NNs. In discussions involving encoders and decoders, we omit the class input y for simplicity and employ shorthand notations for encoders and decoders, such as $h(x) = \mu_z(x; \phi)$ and $g(z) = \mu_x(z; \theta)$. We express a probabilistic classifier for discrete variables as $p_{\theta}(y|z)$, which can be transformed into discriminant functions, denoted as $f^{(i)}(z) = \log p_{\theta}(y = i|z)$. For the two-class problem we can use a single discriminant $f(z) = f^{(c)}(z) - f^{(k)}(z)$, where positive values correspond to class c and negative values to class k . The following explanation methods are discussed solely for the two-class problem. The composition of the encoder $h(x)$ and the discriminant $f(z)$ can be used as an input-dependent discriminant function $F(x) = (f \circ h)(x)$. Similarly, we can obtain CF images by generating CFs in the latent space with respect to $f(z)$ and using the decoder to transform them into the image space $I_F(x, \delta) = (g \circ I_f)(z, \delta)$.

Overview. The GdVAE enhances an autoencoder with an integrated generative classifier. We consider a generative model $p_{\theta}(x, y, z) = p_{\theta}(x|y, z)p_{\theta}(y, z)$ with two distinct factorizations for $p_{\theta}(y, z) = p_{\theta}(z|y)p_{\theta}(y) = p_{\theta}(y|z)p_{\theta}(z)$, defining coupled processes. The first factorization establishes a class conditional prior $p_{\theta}(z|y)$ for the latent variable z and delineates an autoencoder (M1), while the second integrates a discriminative classifier $p_{\theta}(y|z)$ (M2) using the latent variable. Later, we'll employ a generative classifier using the prior encoder's mean values as decision prototypes that will benefit from the discriminative learning signal. See an overview and description in Fig. 2.

3.1 Autoencoding and Generative Classification

Model Distributions. *CVAE including a class prior (M1):* For the first factorization of $p_\theta(x, y, z)$ we assume the observed variable x to be generated from the set of latent variables z and y through the following process

$$y \sim p_\theta(y) = \text{Cat}_y(\pi(\theta)), \quad (1)$$

$$z|y \sim p_\theta(z|y) = \mathcal{N}(\mu_z(y; \theta), \Sigma_z(y; \theta)), \quad (2)$$

$$x|y, z \sim p_\theta(x|y, z) = \mathcal{N}(\mu_x(y, z; \theta), \Sigma_x(y, z; \theta)), \quad (3)$$

with categorical distribution $\text{Cat}_y(\pi(\theta)) = \prod_{k=1}^K \pi(\theta)_k^{\mathbb{1}\{y=k\}}$, where π is a probability vector and $\mathbb{1}\{\cdot\}$ is the indicator function. This process defines a CVAE [45] with an added class prior $p_\theta(y)$, capturing class frequency. Thus, we capture both a prior encoder $p_\theta(z|y)$ and a class prior, which are used by our classifier.

GDA model with latent prior (M2): The second factorization of $p_\theta(x, y, z)$ describes our classification model, where the target class y (observable during training) is generated by the latent code z according to our second process

$$z \sim p(z) = \mathcal{N}(0, I), \quad (4)$$

$$y|z \sim p_\theta(y|z) = \text{Cat}_y(\tau(z; \theta)), \quad (5)$$

$$x|y, z \sim p_\theta(x|y, z) = \mathcal{N}(\mu_x(y, z; \theta), \Sigma_x(y, z; \theta)). \quad (6)$$

Instead of using a separate NN to estimate τ , we reuse M1’s distributions to obtain the categorical distribution $p_\theta(y|z) = \eta p_\theta(z|y)p_\theta(y)$, where η is a normalization constant in the context of Bayes’ theorem. In addition to this coupling, both models are jointly trained using a unified learning objective.

Learning Objective. Our generative models feature non-conjugate dependencies, making it intractable to maximize the conditional log-likelihood. Thus, we employ a surrogate posterior $q_\phi(z|x, y)$ to approximate the true posterior $p_\theta(z|y)$ [27]. The surrogate, also called the recognition model, adapts the latent code distribution based on x . Instead of maximizing the log-likelihood $\log p_\theta(x, y)$ of our model, we use the evidence lower bound (ELBO) to define our loss. The resulting per sample loss for the GdVAE is $\mathcal{L}^{gd} = \alpha \mathcal{L}^{M1} + \beta \mathcal{L}^{M2}$, with

$$\mathcal{L}^{M1} = -\mathbb{E}_{z, y \sim q_\phi} [\log p_\theta(x|y, z)] + KL(q_\phi(z|x, y) || p_\theta(z|y)) - \log p_\theta(y), \quad (7)$$

$$\mathcal{L}^{M2} = -\mathbb{E}_{z, y \sim q_\phi} [\log p_\theta(x|y, z)] + KL(q_\phi(z|x, y) || p(z)) - \mathbb{E}_{z \sim q_\phi} [\log p_\theta(y|z)]. \quad (8)$$

α and β control the balance between M1 and M2, and KL denotes the Kullback-Leibler divergence. The derivation of the loss and ELBO can be found in the Supplement. Note that during inference, we cannot directly sample from the encoder $q_\phi(z|x, y)$ since the class y is unknown. Instead, we conduct ancestral sampling by first sampling from $q_\phi(y|x)$ and afterwards from $q_\phi(z|x, y)$ to approximate $q_\phi(z|x)$. To ensure coherence between the training and inference processes, we compute the expectations relative to $q_\phi(z|x)$ and $q_\phi(z, y|x)$ during training, respectively. This alignment enhances the accuracy of predictions.

Marginalization. The training process is straightforward when labels are observable, and we can directly sample from the conditional encoder $q_\phi(z|x, y)$. Likewise, during inference with the model, we require an estimate of z given x and y . The challenge here is that y is unknown during inference.

Therefore, we draw inspiration from semi-supervised learning [28], employ a factorized probabilistic model $q_\phi(z, y|x) = q_\phi(z|x, y)q_\phi(y|x)$ and perform a marginalization $q_\phi(z|x) = \sum_{y=1}^K q_\phi(z|x, y)q_\phi(y|x)$. In practice, besides the conditional encoder $q_\phi(z|x, y)$, a classifier $q_\phi(y|x)$ is needed. To avoid the need for sampling in the image space [45], we initialize the classifier with the class prior $p_\theta(y)$ and iteratively refine both the classifier and the latent feature model. This expectation-maximization (EM)

approach is detailed in Algorithm 1, with a proof in the Supplement. In contrast to a standard EM for a Gaussian mixture model (GMM), where we usually estimate mean and covariance values, we employ the GMM to generate S data samples $z^{(s)}$. Subsequently, we perform a soft assignment using the fixed classifier $p_\theta(y|z)$ and, akin to [12], reestimate $q_\phi(y|x)$. The closer our estimate aligns with the true class of x , the more samples $z^{(s)}$ we obtain from the correct class, as $q_\phi(z|x, y)$ is weighted by $q_\phi(y|x)$.

The algorithm yields the classifier $q_\phi(y|x)$, used in the learning objective to estimate $q_\phi(z|x)$. We perform ancestral sampling, initially drawing samples from $q_\phi(y|x)$, then from $q_\phi(z|x, y)$ to approximate $q_\phi(z|x)$ (see Algorithm 1).

Generative Classifier. The generative classifier is built upon a Gaussian discriminant analysis model (GDA) [18] and does not have any additional parameters. Its purpose is to transform the features z from the recognition network and marginalization process into an interpretable class prediction.

During the training of the entire GdVAE, the prior network learns the class-conditional mean $\mu_z(y; \theta) = \mu_{z|y}$ and covariance $\Sigma_z(y; \theta) = \Sigma_{z|y}$ as the parameters of our distribution $p_\theta(z|y) = \mathcal{N}(\mu_z(y; \theta), \Sigma_z(y; \theta))$. We assume conditional independence and decompose the likelihood as $p_\theta(z|y) = \prod_{j=1}^M p_\theta(z_j|y)$. In practice, this results in a diagonal covariance matrix $\Sigma_{z|y} = \text{diag}(\sigma_{z_1|y}^2, \dots, \sigma_{z_M|y}^2)$. We use this distribution to determine the likelihood values for the GDA classifier. The class prior $p_\theta(y)$ can be learned either jointly or separately as the final component of the GDA model. Thus, we use the mean values as class prototypes and the covariance to measure the distance to these prototypes.

To infer the class, we apply Bayes' theorem using the detected feature z from the recognition model $p_\theta(y = i|z) = \eta p_\theta(z|y = i)p_\theta(y = i)$, with the normalizer η . For the explanation method, we further assume equal covariance matrices Σ_z

Algorithm 1 An EM-based classifier

```

 $q_\phi(y|x) \leftarrow p_\theta(y)$ 
for iterations  $t \in \{1, \dots, T\}$  do
    E-Step: Ancestral sampling for GMM
     $z^{(s)} \sim q_\phi(z|x) = \sum_{y=1}^K q_\phi(z|x, y)q_\phi(y|x)$ 
    E-Step: GDA classifier
     $p_\theta(y|z^{(s)}) \leftarrow \eta p_\theta(z^{(s)}|y)p_\theta(y)$ 
    M-Step: Assign mean confidence to  $q$ 
     $q_\phi(y|x) \leftarrow p_\theta(y|z) = \frac{1}{S} \sum_{s=1}^S p_\theta(y|z^{(s)})$ 
end for
return  $q_\phi(y|x)$ 

```

(independent of y), yielding linear discriminants $f^{(i)}(z) = w^{(i)T} z + b^{(i)}$, where the weight and bias are given by $w^{(i)} = \Sigma_z^{-1} \mu_{z|i}$ and $b^{(i)} = -\frac{1}{2} \mu_{z|i}^T \Sigma_z^{-1} \mu_{z|i} + \log p_\theta(y = i)$. For two classes we get $f(z) = f^{(c)}(z) - f^{(k)}(z) = w^T z + b$.

3.2 Counterfactual Explanations (CF)

Instead of directly employing a DNN to define an explainer function $x^\delta = \mathcal{I}_F(x, \delta)$, we generate CFs in the latent space and visualize the outcome using the decoder $\mathcal{I}_F(x, \delta) = g(\mathcal{I}_f(z, \delta))$. Since the discriminant $f(z) = w^T z + b$ of our classifier is linear by construction, we will see that the optimal explainer function is also linear $\mathcal{I}_f(z, \kappa) = z + \kappa \bar{w}$, where the latent vector is adjusted in the direction of $\bar{w} \in \mathbb{R}^M$. Here, $\kappa \in \mathbb{R}$ —a tuning knob for data traversal—represents the strength of the manipulation. Our proposed CF methods are shown in Fig. 4a.

1.) *Local counterfactuals:* A local explanation should meet both consistency and proximity properties. Therefore, the optimal CF z^δ minimizes the distance to the current instance z while ensuring the decision function matches the requested value δ . This involves solving the following constrained optimization problem

$$\mathcal{I}_f(z, \delta) = \arg \min_{z^\delta} \text{dist}(z^\delta, z), \quad \text{subject to } f(z^\delta) = \delta, \quad (9)$$

where $\text{dist}(\cdot, \cdot)$ is a distance metric that guarantees *proximity* and the constraint ensures *consistency*. Regardless of whether we choose the common L2-norm [24, 43] or a Riemannian-based metric (Mahalanobis distance) induced by VAEs [5], the solution to Eq. (9) is a linear explainer function

$$\mathcal{I}_f(z, \delta) = z^\delta = z + \kappa \bar{w}, \quad \text{with } \kappa = \frac{\delta - w^T z - b}{w^T \bar{w}}, \quad (10)$$

where $w = \Sigma_z^{-1}(\mu_{z|c} - \mu_{z|k})$ is the gradient direction of our discriminant. In this approach, any negative value of δ would lead to a change in the class prediction, and $\delta = 0$ corresponds to both classes having equal probability. To simplify user interaction, one can specify the value in terms of a probability using the logit function, such that $\delta = \log \frac{p_c}{1-p_c}$ with $p_c = p(y = c | z^\delta)$.

Using the L2-norm, we obtain the intuitive solution where $\bar{w} = w$ (local-L2). The CF is generated by using the shortest path (perpendicular to the decision surface) to cross the decision boundary (see Fig. 4a). The theoretical analysis on Riemannian manifolds [5] shows that samples close in the latent space with respect to a Riemannian metric lead to close images in terms of the L2-norm, thus optimizing proximity. A Riemannian-based solution using the Mahalanobis distance is $\bar{w} = \Sigma_z w$ (local-M). Training with a spherical covariance $\Sigma_z = \sigma^2 I$ instead of $\Sigma_z = \text{diag}(\sigma_{z_1}^2, \dots, \sigma_{z_M}^2)$ yields equivalent functions and therefore equal empirical results for both Riemannian and L2-based CFs. Proofs, assumptions, and implications for non-linear methods are provided in the Supplement.

2.) *Global counterfactuals:* The second CF approach is to move directly in the direction of the prototype of the opposing class, termed the *counterfactual prototype*. In this scenario, we take a direct path from our current input z to the

CF prototype $\mu_{z|k}$, defining the direction as $\bar{w} = (\mu_{z|k} - z)$, and reuse the local explainer function from Eq. (10).

The local approach minimizes input attribute changes (*proximity*), while global explanations gradually converge to common CF prototypes to reveal the overall model behavior for a category of examples. Both methods maintain the *consistency* property in the latent space. For *realism*, we argue that transitioning directly to the CF prototype or minimizing a distance function is the most effective way to stay within the data distribution, resulting in a natural appearance.

Consistency Loss. Our explainer function implicitly assumes that the encoder and decoder act as inverses of each other. Consequently, it is imperative to ensure that a reconstruction $x^\delta = g(z^\delta)$, based on the latent representation $z^\delta = \mathcal{I}_f(h(x), \delta)$, results in a similar latent representation when encoded once more, i.e., $h(x^\delta) \approx z^\delta$. This alignment is crucial to ensure that the classifier provides the desired confidence when a CF is used as input. To enforce this property, similar to [30, 43, 44], we introduce a tailored consistency loss

$$\mathcal{L}^{con} = \mathbb{E}_{p(\delta)} [KL(q_\phi(z|x^\delta) || q_\phi(z^\delta|x))] , \quad (11)$$

where the term addresses classification consistency for generated CF inputs. Essentially, we are probing latent values between the distributions $p_\theta(z|y=c)$ and $p_\theta(z|y=k)$ to optimize for the consistency property. $q_\phi(z^\delta|x)$ is obtained by applying the linear transformation of the explainer function $\mathcal{I}_f(z, \delta)$ to $q_\phi(z|x)$. In other words, we simply shift the mean value and keep the variance. We use both global and local explainer functions to generate training samples. $p(\delta)$ defines the desired perturbation of the latent variable and we use $p(\delta) = \mathcal{U}(-\varepsilon, \varepsilon)$, where ε can be specified in terms of a probability. The final loss is then given by $\mathcal{L} = \mathcal{L}^{gd} + \gamma \mathcal{L}^{con}$, where γ controls the impact of the consistency regularizer.

4 Experiments

The empirical evaluation aims to validate the performance of our model, focusing on two components: the predictive performance of the GdVAE and the quality of the CFs. We present quantitative results of the predictive performance and CFs in Secs. 4.1 and 4.2, along with qualitative results in Sec. 4.3. In the Supplement, we conduct a hyperparameter investigation covering all method parameterizations. This includes exploring the model balance between M1 and M2, consistency loss, and presenting additional quantitative and qualitative results.

Datasets and Implementation. We employ four image datasets: MNIST [31], CelebA [32], CIFAR-10 [29], and the high-resolution dataset FFHQ [25]. Our neural networks are intentionally designed to be compact. For CelebA, the encoder has five convolutional layers and one linear layer for $\mu_z(x, y; \phi)$ and $\Sigma_z(x, y; \phi)$, which define the distribution $q_\phi(z|x, y)$. The decoder’s architecture is symmetrical to that of the encoder. Prior encoders use fully connected networks with four layers to compute $\mu_z(y; \theta)$ and $\Sigma_z(y; \theta)$, defining our distribution $p_\theta(z|y)$. All baseline methods employ identical backbones as the GdVAE, and when feasible, publicly available code was adjusted to ensure a fair comparison. See the Supplement for details on datasets, models, and metrics.

Table 2: Predictive performance: Importance sampling (IS), ProtoVAE, and a black-box baseline. Classifier accuracy (ACC) and mean squared error (MSE) of reconstructions (scaled by 10^2) are reported. Mean values and standard deviations are from four training runs with different seeds. \dagger : incl. ProtoVAE’s augmentation and preprocessing.

| Method | MNIST | | CIFAR-10 | | CelebA - Gender | |
|-----------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
| | ACC% \uparrow | MSE \downarrow | ACC% \uparrow | MSE \downarrow | ACC% \uparrow | MSE \downarrow |
| IS [45, 48, 54] | 99.0 \pm 0.08 | 1.04 \pm 0.01 | 55.0 \pm 0.59 | 2.45 \pm 0.03 | 94.7 \pm 0.44 | 1.77 \pm 0.08 |
| Ours | 99.0 \pm 0.11 | 1.10 \pm 0.04 | 65.1 \pm 0.78 | 1.71 \pm 0.02 | 96.7 \pm 0.13 | 0.91 \pm 0.01 |
| Baseline | 99.3 \pm 0.04 | 1.12 \pm 0.02 | 69.0 \pm 0.54 | 1.45 \pm 0.01 | 96.7 \pm 0.26 | 0.82 \pm 0.00 |
| ProtoVAE [13] | 99.1 \pm 0.17 | 1.51 \pm 0.23 | 76.6 \pm 0.35 | 2.69 \pm 0.02 | 96.6 \pm 0.24 | 1.32 \pm 0.10 |
| Ours \dagger | 98.7 \pm 0.05 | 0.93 \pm 0.01 | 76.8 \pm 0.91 | 1.18 \pm 0.02 | 96.8 \pm 0.04 | 0.71 \pm 0.01 |

4.1 Evaluation of Predictive Performance

Methodology. For a trustworthy SEM, performance should align with the closest black-box model [13]. Thus, the goal of this evaluation is not to outperform state-of-the-art results on specific datasets but to offer a relative comparison for the GdVAE architecture and various training methods. In all approaches, both the classifier and autoencoder are jointly trained, sharing the same backbone.

Baselines. First, optimal performance for the selected architecture is established using a black-box model, comprising a jointly trained CVAE and classifier as the *baseline*. Next, GdVAE’s inference method is evaluated against the leading CVAE technique, *importance sampling (IS)* [45, 48, 54]. Lastly, *ProtoVAE* [13] is referenced as a prototype-per-class VAE benchmark.

Results. The results in Tab. 2 indicate good generalization in classification and reconstruction across MNIST and CelebA. The GdVAE’s EM-based inference achieves performance close to the optimal baseline with a separate classifier, except for CIFAR where there is a four-percentage-point gap in accuracy. Comparing our EM and the IS approach suggests that our method is more efficient for higher-dimensional images, benefiting from sampling in the lower-dimensional latent instead of image space. With data augmentation and normalization from ProtoVAE, GdVAE achieves comparable results to ProtoVAE.

Takeaway: The inference procedure of our SEM closely matches the performance of a discriminative black-box model. Furthermore, our method consistently delivers competitive results to state-of-the-art approaches, particularly when applied to higher-dimensional images. The class-conditional GdVAE offers better reconstructions compared to ProtoVAE, the only unconditional model.

4.2 Quantitative Evaluation of CF Explanations

Methodology. The experiments aim to evaluate the quality of CFs regarding *realism*, *consistency*, and *proximity*. *Realism*, as defined in [14, 26] or data consistency [43], refers to the CF images being realistic and capturing identifiable concepts. To measure realism, we employ the Fréchet Inception Distance (FID) [14, 26, 43] as a common metric. Akin to [26], *proximity* is assessed using the mean squared error (MSE) between the CF and the query image.

Table 3: Evaluation of CF explanations using Pearson correlation (ρ_p), ACC, and MSE (scaled by 10^2) for consistency, Fréchet Inception Distance (FID) for realism, and MSE (scaled by 10^2) for proximity. Mean values and standard deviations are from four runs with different seeds. The first and second best results are **bolded** and underlined.

| Method | Consistency | | | Realism | Proximity |
|--------------------|------------------------|------------------|------------------|------------------|---------------------|
| | $\rho_p \uparrow$ | $ACC\% \uparrow$ | $MSE \downarrow$ | $FID \downarrow$ | $MSE \downarrow$ |
| MNIST - Binary 0/1 | GANalyze [15] | 0.84±0.04 | 5.5±1.3 | 6.75±1.27 | <u>54.89±4.19</u> |
| | UDID [49] | 0.85±0.01 | 1.2±0.3 | 8.82±0.18 | 38.89±2.01 |
| | ECINN [21] | 0.93±0.02 | 33.0±7.5 | 1.76±0.81 | 87.25±12.63 |
| | EBPE [43] | 0.97±0.01 | 44.6±4.3 | 0.50±0.13 | 108.94±13.61 |
| | C3LT [26] | 0.89±0.03 | 3.6±0.8 | 6.32±1.39 | 57.09±10.78 |
| | Ours (local-L2) | <u>0.95±0.00</u> | <u>42.9±2.7</u> | 0.95±0.11 | 91.22±11.04 |
| | Ours (local-M) | <u>0.95±0.01</u> | 44.6±2.5 | <u>0.87±0.13</u> | 89.91±5.58 |
| CelebA - Smiling | Ours (global) | <u>0.97±0.01</u> | 54.2±4.0 | <u>0.55±0.13</u> | <u>125.45±11.32</u> |
| | GANalyze [15] | 0.78±0.03 | 15.2±3.3 | 5.42±0.97 | 147.43±19.49 |
| | UDID [49] | 0.86±0.06 | 15.8±9.2 | 4.22±2.17 | 178.23±75.84 |
| | ECINN [21] | 0.72±0.21 | 21.3±9.6 | 5.68±4.32 | 95.35±14.48 |
| | EBPE [43] | 0.94±0.01 | 41.9±3.1 | 1.22±0.16 | 191.67±20.51 |
| | C3LT [26] | <u>0.90±0.01</u> | 11.8±5.5 | 3.94±0.66 | 101.46±11.56 |
| | Ours (local-L2) | 0.81±0.04 | 25.0±4.9 | 3.65±1.06 | 85.52±2.37 |
| CIFAR-10 | Ours (local-M) | 0.82±0.05 | <u>25.7±5.1</u> | <u>3.51±1.05</u> | <u>85.56±2.39</u> |
| | Ours (global) | <u>0.89±0.01</u> | <u>45.9±12.3</u> | <u>2.08±0.54</u> | <u>128.93±4.94</u> |
| | - | - | - | - | 5.81±0.53 |

The *consistency* property, also known as compatibility [43] or importance [14], is evaluated using mean squared error (MSE), accuracy (ACC), as well as the Pearson correlation coefficient. We create CFs for every image by requesting confidences within the range $p_c \in [0.05, 0.95]$, with a step size of 0.05. The metrics compare the expected outcome of the classifier p_c (desired probability score of CFs) with the actual probability \hat{p}_c obtained from the classifier for the CF.

Baselines. We employ methods from different designs (see Sec. 2) as baselines with shared backbones. To ensure a fair comparison, we slightly modify methods that originally tackle the simpler consistency task [21, 26] or those intended for unsupervised scenarios [49], aligning them with the consistency defined in Sec. 2. First, we apply generative explanation methods, including GANalyze [15] and UDID [49], while utilizing our pre-trained GdVAE as an autoencoder and classifier. Second, we adapt post-hoc CF methods to be compatible with our GdVAE architecture. We adapt the method from ECINN [21] to approximate our classifier. C3LT [26] is trained to generate CFs for our GdVAE model using a non-linear explainer function instead of our linear one. Finally, EBPE [43] is adjusted to train an encoder and decoder based on the GdVAE architecture and the pre-trained classifier. These approaches are compared to our CF methods.

Results. The results in Tab. 3 reveal performance across diverse datasets in binary classification challenges. Considering that the GdVAE is the sole transparent model, it is essential to bear in mind that most models operate on the GdVAE’s pre-regularized latent space (Fig. 3) when interpreting the results. Consequently, with the exception of EBPE, these methods face a less complex

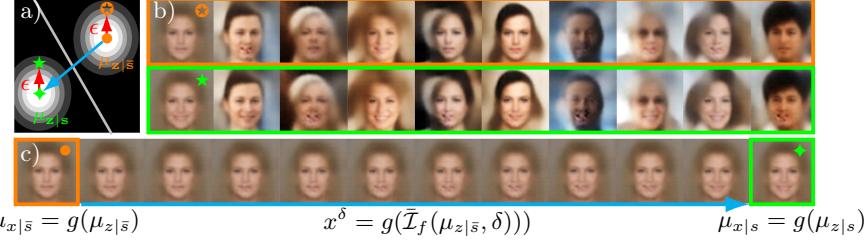


Fig. 3: Regularized latent space. a) Distribution $p_\theta(z|y)$ with class-conditional mean values for not-smiling (orange, \bullet) and smiling (green, \blacklozenge), where $y = \bar{s} = \text{not-smiling}$ and $y = s = \text{smiling}$. b) Reconstructed random samples for not-smiling (top, orange \star) and smiling (bottom, green \star), arranged in ascending order of their Mahalanobis distance from left to right. In each column, the Mahalanobis distance is made consistent by adding the same random vector ϵ (red vector in a) to the mean of both classes, aligning samples along isocontours. c) The global explainer function interpolates between class-conditional means along the straight-line path (cyan arrow in a).

task and should approximate the "true" linear direction of our local CFs post-GdVAE training. It becomes evident that our global CF method exhibits a higher degree of consistency with the classifier, albeit falling short in terms of realism when compared to the local approaches. This divergence is expected as the global method converges toward the mean representation of the CF prototype, producing relatively blurred representations with a notable distance from the query image (poor proximity). However, global CFs effectively uncover the model's overarching decision logic through its prototypes (see Fig. 3c).

Specifically, on the MNIST dataset, our local methods achieve the best or second-best results in all consistency metrics, producing CFs with well-calibrated confidence values. The notably high accuracy values indicate that our methods generate CFs covering the entire confidence range, effectively capturing samples near the decision boundary. However, the realism metric is affected due to the absence of MNIST images near the decision boundary, notably those representing shared concepts of digits 0 and 1 (see Fig. 2). In summary, a favorable trade-off between consistency, realism, and proximity is achieved by EBPE, ECINN, and our local methods. A distinct perspective arises when considering the CelebA dataset, where our local methods excel in achieving optimal results for both proximity and realism, maintaining a low FID score. In terms of consistency metrics like ACC and MSE, our local method (local-M) ranks among the top two performers. Global CFs are distinguished with a separating line in Tab. 3, indicating their deviation from query images by consistently approaching the same prototypes, which effectively reveals biases (e.g., toward female prototypes in CelebA, see Sec. 4.3).

Takeaway: Our SEM, featuring both linear local and global explanations, yields results that stand on par with leading post-hoc explanation techniques such as C3LT and EBPE. Moreover, our model slightly outperforms ECINN across various metrics, with ECINN serving as an optimized post-hoc variant

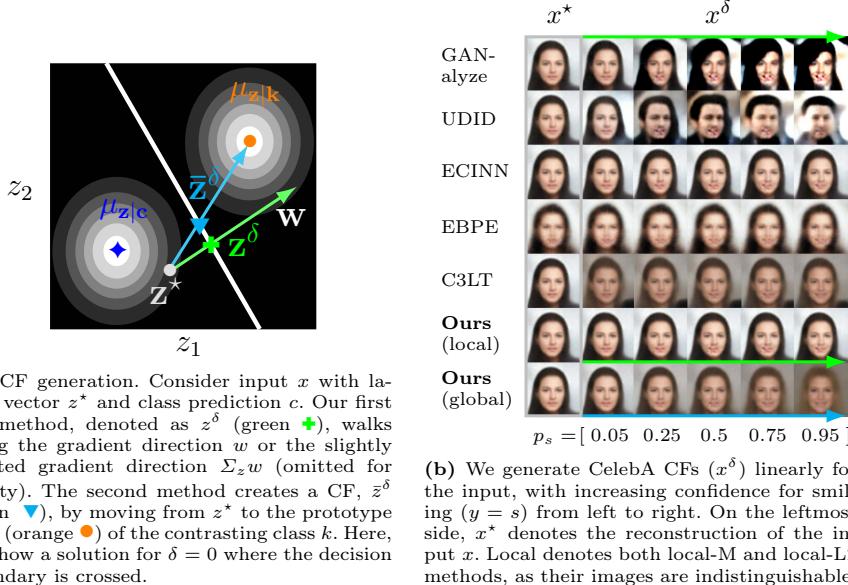


Fig. 4: Left: Counterfactual generation. Right: Counterfactual examples.

of our local CFs. The findings suggest a trade-off between consistency and realism, as no single method excels in all metrics. Notably, as realism decreases (higher FID), consistency (correlation) increases, resulting in different working points for each CF method. For further insights regarding this trade-off between consistency and realism, please refer to the Supplement.

4.3 Qualitative Evaluation

Prototypical Space and Bias Detection. The prototypical space of the GdVAE is shown in Fig. 3. This section’s results reinforce GdVAE’s transparency through easily comprehensible global explanations and latent space visualization. We achieve this by displaying the decoded prototypes and interpolating between them through our global explainer function (see Fig. 3c). The transparent classifier’s prototypes directly uncover biases without the need for quantitative analysis of counterfactuals on simulated datasets, as shown in prior work (e.g., [43]).

Illustrated in Fig. 3c, the classifier’s decision on smiling is shaped by female prototypes, revealing a potential bias or data imbalance not observed in our local CFs and other CF methods. The gender bias is exposed by evaluating the smile classifier across hidden attributes (Tab. 4), indicating reduced performance with increased uncertainty in males. Finally, we leverage the generative capabilities of the CVAE

Table 4: CelebA bias.

| Attr. | ACC% \uparrow | MSE \downarrow |
|--------|-----------------|------------------|
| male | 89.9 ± 1.37 | 0.92 ± 0.03 |
| female | 91.1 ± 0.33 | 0.88 ± 0.02 |

structure, generating samples for various classes. The results are presented in Fig. 3b and organized based on their distance to the corresponding prior. Due to regularization, the latent space preserves the identity of individuals when generating samples for different classes using the same random vector.

CF Explanations. Regarding visual quality, Fig. 4b directly compares our approach with other tested methods, using a random CelebA image. Our local method achieves visual quality comparable to state-of-the-art approaches, yielding results akin to ECINN. C3LT demonstrates smooth outcomes reminiscent of our global method, while EBPE preserves image concepts like our local method but with slight reconstruction variations. The alignment of C3LT with our global CFs, despite its expected approximation of the direction of our local CFs, highlights the advantage of our analytic link between the classifier and CFs.

High-resolution CFs. We showcase our classifier’s scalability in more complex scenarios, such as higher resolutions, by embedding it within a pre-trained StyleGAN architecture on the FFHQ dataset. Local CF explanations for smiling with $p_s = 0.99$ are depicted in Fig. 1. Similar to findings with CelebA data, our method retains the background while altering only pertinent attributes.

5 Conclusion

In this paper, we present a novel self-explainable model capable of delivering counterfactual explanations alongside transparent class predictions. Our approach uses a linear classifier in the latent space that utilizes visualizable prototypes for the downstream task. With the known linear structure, we can provide an analytical solution to generate counterfactual images. Our extensive experiments substantiate our method’s ability to yield results that are on par with state-of-the-art approaches in terms of consistency, proximity, and realism while maintaining transparency. Furthermore, we illustrate how prototypes offer insight into decision logic and aid in identifying classifier bias. We see our method as a significant step toward the integration of self-explainable models and counterfactual explanation techniques. In contrast to previous work that requires post-hoc analysis for generating counterfactuals, our transparent model constrains the shared latent space to support consistency, proximity, and realism. Finally, resembling C3LT, our approach scales and seamlessly integrates with larger network architectures, as demonstrated on the FFHQ dataset.

Acknowledgments

The authors thank e:fs TechHub GmbH, Germany, and Tongliang Liu’s Trustworthy Machine Learning Lab at the University of Sydney for their support. Special appreciation goes to Muyang Li for insightful discussions and Niclas Hüwe for assistance with implementation.

References

1. Agarwal, R., Frosst, N., Zhang, X., Caruana, R., Hinton, G.E.: Neural additive models: Interpretable machine learning with neural nets. In: NeurIPS (2021)
2. Alvarez-Melis, D., Jaakkola, T.S.: Towards robust interpretability with self-explaining neural networks. In: NeurIPS (2018)
3. Bau, D., Zhu, J., Strobelt, H., Zhou, B., Tenenbaum, J.B., Freeman, W.T., Torralba, A.: GAN dissection: Visualizing and understanding generative adversarial networks. In: ICLR (2019)
4. Black, E., Wang, Z., Fredrikson, M.: Consistent counterfactuals for deep models. In: ICLR (2022)
5. Chadebec, C., Allassonniere, S.: A geometric perspective on variational autoencoders. In: NeurIPS (2022)
6. Chang, C.H., Caruana, R., Goldenberg, A.: NODE-GAM: Neural generalized additive model for interpretable deep learning. In: ICLR (2022)
7. Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.K.: This looks like that: Deep learning for interpretable image recognition. In: NeurIPS (2019)
8. Chen, R.T.Q., Li, X., Grosse, R.B., Duvenaud, D.K.: Isolating sources of disentanglement in variational autoencoders. In: NeurIPS (2018)
9. Ding, Z., Xu, Y., Xu, W., Parmar, G., Yang, Y., Welling, M., Tu, Z.: Guided variational autoencoder for disentanglement learning. In: CVPR (2020)
10. Do, M.: Fast approximation of kullback-leibler distance for dependence trees and hidden markov models. IEEE Signal Processing Letters **10**(4), 115–118 (2003)
11. Esser, P., Rombach, R., Ommer, B.: A disentangling invertible interpretation network for explaining latent representations. In: CVPR (2020)
12. Falck, F., Zhang, H., Willetts, M., Nicholson, G., Yau, C., Holmes, C.C.: Multi-facet clustering variational autoencoders. In: NeurIPS (2021)
13. Gautam, S., Boubekki, A., Hansen, S., Salahuddin, S.A., Jenssen, R., Höhne, M.M., Kampffmeyer, M.: Protovae: A trustworthy self-explainable prototypical variational model. In: NeurIPS (2022)
14. Ghandeharioun, A., Kim, B., Li, C.L., Jou, B., Eoff, B., Picard, R.: DISSECT: Disentangled simultaneous explanations via concept traversals. In: ICLR (2022)
15. Goetschalckx, L., Andonian, A., Oliva, A., Isola, P.: Ganalyze: Toward visual definitions of cognitive image properties. In: ICCV (2019)
16. Goyal, Y., Wu, Z., Ernst, J., Batra, D., Parikh, D., Lee, S.: Counterfactual visual explanations. In: ICML (2019)
17. Guyomard, V., Fessant, F., Guyet, T., Bouadi, T., Termier, A.: Vcnet: A self-explaining model for realistic counterfactual generation. In: Amini, M.R., Canu, S., Fischer, A., Guns, T., Kralj Novak, P., Tsoumakas, G. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 437–453. Springer International Publishing, Cham (2023)
18. Haselhoff, A., Kronenberger, J., Küppers, F., Schneider, J.: Towards black-box explainability with gaussian discriminant knowledge distillation. In: CVPRW (2021)
19. Hauberg, S.r., Freifeld, O., Black, M.: A geometric take on metric learning. In: NeurIPS (2012)
20. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-VAE: Learning basic visual concepts with a constrained variational framework. In: ICLR (2017)
21. Hvilsted, F., Iosifidis, A., Assent, I.: Ecinn: Efficient counterfactuals from invertible neural networks. In: BMVC (2021)

22. Härkönen, E., Hertzmann, A., Lehtinen, J., Paris, S.: Ganspace: Discovering interpretable gan controls. In: NeurIPS (2020)
23. Jacob, P., Zablocki, É., Ben-Younes, H., Chen, M., Pérez, P., Cord, M.: STEEX: steering counterfactual explanations with semantics. In: ECCV (2022)
24. Jeanneret, G., Simon, L., Jurie, F.: Diffusion models for counterfactual explanations. In: ACCV (2022)
25. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (2019)
26. Khorram, S., Fuxin, L.: Cycle-consistent counterfactuals by latent transformations. In: CVPR (2022)
27. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
28. Kingma, D.P., Mohamed, S., Jimenez Rezende, D., Welling, M.: Semi-supervised learning with deep generative models. In: NeurIPS (2014)
29. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research) <http://www.cs.toronto.edu/~kriz/cifar.html>
30. Lang, O., Gandelsman, Y., Yarom, M., Wald, Y., Elidan, G., Hassidim, A., Freeman, W.T., Isola, P., Globerson, A., Irani, M., Mosseri, I.: Explaining in style: Training a gan to explain a classifier in stylespace. In: ICCV (2021)
31. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998). <https://doi.org/10.1109/5.726791>
32. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: ICCV (2015)
33. Louizos, C., Swersky, K., Li, Y., Welling, M., Zemel, R.S.: The variational fair autoencoder. In: ICLR (2016)
34. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: NeurIPS (2017)
35. Lundstrom, D.D., Huang, T., Razavyayn, M.: A rigorous study of integrated gradients method and extensions to internal neuron attributions. In: ICML (2022)
36. Parekh, J., Mozharovskyi, P., d'Alché Buc, F.: A framework to learn with interpretation. In: NeurIPS (2021)
37. Plumerault, A., Borgne, H.L., Hudelot, C.: Controlling generative models with continuous factors of variations. In: ICLR (2020)
38. Ren, X., Yang, T., Wang, Y., Zeng, W.J.: Learning disentangled representation by exploiting pretrained generative models: A contrastive learning view. In: ICLR (2021)
39. Rhodes, T., Lee, D.: Local disentanglement in variational auto-encoders using jacobian l_1 regularization. In: NeurIPS (2021)
40. Samangouei, P., Saeedi, A., Nakagawa, L., Silberman, N.: Explaingan: Model explanation via decision boundary crossing transformations. In: ECCV (2018)
41. Seitzer, M.: pytorch-fid: FID Score for PyTorch. <https://github.com/mseitzer/pytorch-fid> (August 2020), version 0.3.0
42. Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of gans for semantic face editing. In: CVPR (2020)
43. Singla, S., Pollack, B., Chen, J., Batmanghelich, K.: Explanation by progressive exaggeration. In: ICLR (2020)
44. Sinha, S., Dieng, A.B.: Consistency regularization for variational auto-encoders. In: NeurIPS (2021)
45. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: NeurIPS (2015)

46. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: ICML (2019)
47. Vandenhende, S., Mahajan, D., Radenovic, F., Ghadiyaram, D.: Making heads or tails: Towards semantically consistent visual counterfactuals. In: ECCV (2022)
48. van de Ven, G.M., Li, Z., Tolias, A.S.: Class-incremental learning with generative classifiers. In: CVPRW (2021)
49. Voynov, A., Babenko, A.: Unsupervised discovery of interpretable directions in the gan latent space. In: ICML (2020)
50. Wang, J., Liu, H., Wang, X., Jing, L.: Interpretable image recognition by constructing transparent embedding space. In: ICCV (2021)
51. Wang, Y., Wang, X.: Self-interpretable model with transformation equivariant interpretation. In: NeurIPS (2021)
52. Yang, C., Shen, Y., Zhou, B.: Semantic hierarchy emerges in deep generative representations for scene synthesis. IJCV **129**(5), 1451–1466 (2021). <https://doi.org/10.1007/s11263-020-01429-5>
53. Yu, C., Wang, W.: Diverse similarity encoder for deep gan inversion. arXiv preprint arXiv:2108.10201 (2022), <https://arxiv.org/abs/2108.10201>
54. Zhang, C., Zhang, K., Li, Y.: A causal view on robustness of neural networks. In: NeurIPS (2020)

The Gaussian Discriminant Variational Autoencoder (GdVAE): A Self-Explainable Model with Counterfactual Explanations

Supplementary Material

Table of Contents

The Supplement is structured as follows:

- Limitations and societal impacts are discussed in Appendix A.
- In Appendix B, you can find the ELBO derivation (Eqs. (7) and (8)), as well as proofs for the EM-based approach (Algorithm 1) and the optimality of linear explainer functions (Eq. (10)).
- Detailed information about the models used, the training process, and experimental specifics, such as compute resources, hyperparameters, dataset, and asset details, can be found in Appendix C.
- Appendix D delves into the metrics employed for predictive performance analysis and CF quality assessment.
- Additional results related to hyperparameter tuning, trade-off between consistency and realism (Figs. 9 and 10) as well as supplementary qualitative results are presented in Appendix E.

A Limitations and Societal Impacts

A.1 Limitations

We acknowledge several limitations concerning the GdVAE and the evaluation methodology:

- Our datasets for quantitative evaluation were selected to balance computational efficiency, alignment with previous studies, and adherence to the Reproducibility Checklist, emphasizing the provision of central tendency and variation (e.g., mean, standard deviation). Unlike prior CF research, our focus on analyzing both central tendency and variability has led to experiments that are four times more costly (see Appendix C.4). By doing so, we acknowledge the limitations imposed by our dataset selection criteria, yet we argue that these limitations are counterbalanced by the gains in the reproducibility of our experimental results.
- Our method, unlike [30] and [14], enables simultaneous manipulation of all class-related attributes but lacks fine-grained control over individual image attributes. Incorporating multiple prototypes, as in [13], could enhance GdVAE’s predictive performance and capability to manipulate individual image attributes.

- Our quantitative evaluation of CF methods is currently limited to binary classification problems, and future evaluations of multi-class problems are needed to advance CF literature. Unfortunately, most visual CF methods (Tab. 1) evaluate CFs for binary tasks like CelebA, where attributes are not mutually exclusive. Those that use multi-class settings do not scale well to a high number of classes without modification, requiring the training of a model for each binary CF class pair [26, 40]. Other approaches require time-consuming inference-time optimization [24, 30] or an additional image of the CF class to guide CF generation [16, 47]. Our method can be expanded to multiple classes. For multiple classes, our CFs (Eq. (10)) can be generated without changes by choosing a reference class. Ambiguity between logits and softmax makes user interaction less convenient, yet setting $\delta < 0$ achieves CFs inducing class flips. Results for MNIST and CIFAR, presented in Figs. 15 and 16, demonstrate CFs for the simpler consistency task by swapping the logits of the predicted and counterfactual classes. This strategy is effective for high-confidence class predictions, such as with MNIST, though it formally only changes the class without necessarily switching to the specified CF class.
- Our approach utilizes a class-conditional encoder model $q_\phi(z|x, y)$. Significantly reduced computational costs can be achieved by employing unconditional models, such as $q_\phi(z|x)$. In [12], a clustering solution is presented, directly leveraging unconditional encoder and decoder models. However, further analyses are needed to conclude the performance implications.
- The EM-based method described in Algorithm 1 is an integral part of the error backpropagation process. Consequently, using a high number of iterations results in very deep computational graphs, which can introduce challenges, including issues like vanishing gradients and other forms of instability.
- All our experiments employ relatively simple network architectures, and we strive to maintain uniform training configurations as closely as possible. For instance, our baselines share the same backbone architecture, are trained with loss functions as faithful as feasible to the original implementations, and undergo training for a duration of 24 epochs. These limitations may result in methods like EBPE performing below their full potential, considering that in their original versions, both EBPE and its extension, DISSECT, were trained for 300 epochs. Results after additional training epochs are in Tab. 12.

A.2 Societal Impacts

Our GdVAE is not inherently associated with specific applications that directly cause negative societal impacts. However, it does possess the potential for misuse in unethical ways. For instance, given our generative models, there exists the possibility of their modified use for generating deep fake images with altered attributes. Furthermore, although our SEM provides insights into the model, it does not guarantee the detection of all fairness issues, existing biases, or results that align with attribution-based explanations. Our methods complement existing fairness ([33]) and explainability ([34, 35]) enhancement techniques rather than replacing them.

B Proofs

B.1 Variational Lower Bound of the Joint Log-Likelihood

In this section, we derive the Evidence Lower Bound (ELBO) for our main paper's loss function, which is represented by Eqs. (7) and (8). We define the key variables as follows with x representing the input data (e.g., an image), y corresponding to the target class, and z representing our latent variables. The general ELBO for the model is derived by

$$\begin{aligned} \log p_\theta(x, y) &= \log \int p_\theta(x, y, z) dz = \log \int p_\theta(x, y, z) \frac{q_\phi(z|x, y)}{q_\phi(z|x, y)} dz \\ &= \log \mathbb{E}_{q_\phi(z|x, y)} \left[\frac{p_\theta(x, y, z)}{q_\phi(z|x, y)} \right] \\ &\geq \mathbb{E}_{q_\phi(z|x, y)} \left[\log \frac{p_\theta(x, y, z)}{q_\phi(z|x, y)} \right] \text{(ELBO)} \\ &= \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(x, y, z)] - \mathbb{E}_{q_\phi(\cdot)} [\log q_\phi(z|x, y)] \\ &= \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(x|y, z)] + \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(y, z)] - \mathbb{E}_{q_\phi(\cdot)} [\log q_\phi(z|x, y)] \\ &= -\mathcal{L}^{M1,2}(\theta, \phi; x, y), \end{aligned}$$

with $q_\phi(\cdot) = q_\phi(z|x, y)$. The loss function for the first model M1 including a CVAE with an additional class prior $p_\theta(y, z) = p_\theta(z|y)p_\theta(y)$ is given by

$$\begin{aligned} \mathcal{L}^{M1} &= -\mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(x|y, z)] - \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(y, z)] + \mathbb{E}_{q_\phi(\cdot)} [\log q_\phi(z|x, y)] \\ &= -\mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(x|y, z)] - \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(z|y)] \\ &\quad - \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(y)] + \mathbb{E}_{q_\phi(\cdot)} [\log q_\phi(z|x, y)] \\ &= -\mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(x|y, z)] + KL(q_\phi(z|x, y) || p_\theta(z|y)) - \log p_\theta(y). \end{aligned}$$

The loss for the second model M2 with a classifier and latent prior $p_\theta(y, z) = p_\theta(y|z)p_\theta(z)$ is given by

$$\begin{aligned} \mathcal{L}^{M2} &= -\mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(x|y, z)] - \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(y, z)] + \mathbb{E}_{q_\phi(\cdot)} [\log q_\phi(z|x, y)] \\ &= -\mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(x|y, z)] - \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(y|z)] \\ &\quad - \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(z)] + \mathbb{E}_{q_\phi(\cdot)} [\log q_\phi(z|x, y)] \\ &= -\mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(x|y, z)] + KL(q_\phi(z|x, y) || p_\theta(z)) - \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(y|z)]. \end{aligned}$$

The loss function $\tilde{\mathcal{L}}^{gd} = \tilde{\mathcal{L}}^{gd}(\theta, \phi; x, y)$ for a joint training of the conditional variational autoencoder and classifier can be obtained by combining $\mathcal{L}^{M1} = \mathcal{L}^{M1}(\theta, \phi; x, y)$ and $\mathcal{L}^{M2} = \mathcal{L}^{M2}(\theta, \phi; x, y)$. We then obtain

$$\begin{aligned} \tilde{\mathcal{L}}^{gd} &= \alpha \mathcal{L}^{M1} + \beta \mathcal{L}^{M2} \\ &= -(\alpha + \beta) \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(x|y, z)] + \alpha (KL(q_\phi(z|x, y) || p_\theta(z|y)) - \log p_\theta(y)) \\ &\quad + \beta (KL(q_\phi(z|x, y) || p_\theta(z)) - \mathbb{E}_{q_\phi(\cdot)} [\log p_\theta(y|z)]), \end{aligned} \tag{12}$$

with $q_\phi(\cdot) = q_\phi(z|x, y)$.

B.2 Variational Expectation Maximization for the Marginalization Process (Algorithm 1)

In this section, we employ variational expectation maximization (EM) to provide a proof for the iterative algorithm, as depicted in Algorithm 1. This algorithm serves as a fundamental component for marginalization. It's worth noting that this EM process is nested within the overarching variational optimization of the GdVAE. Consequently, we switch the roles of distributions, with $p(\cdot)$ representing the variational distribution and $q(\cdot)$ signifying the model distribution.

Consider the variable pair (x, z, y) within the model distribution $q_\phi(x, z, y)$, where only x is observable. The objective of variational EM is to optimize the model parameters ϕ by maximizing the marginal likelihood $q_\phi(x)$. This optimization is achieved by leveraging a lower bound on the marginal likelihood, using the 'variational' distribution $p_\theta(z, y|x)$. This lower bound is defined by

$$\begin{aligned}\log q_\phi(x) &\geq \mathbb{E}_{p_\theta(z, y|x)}[\log q_\phi(x, z, y)] - \mathbb{E}_{p_\theta(z, y|x)}[\log p_\theta(z, y|x)] \\ &= -\mathbb{E}_{p_\theta(z, y|x)}\left[\log \frac{p_\theta(z, y|x)}{q_\phi(z, y|x)}\right] + \log q_\phi(x).\end{aligned}$$

By rearranging the lower bound, we obtain the function $\mathcal{J}(p, \phi)$, which serves as the objective for the EM algorithm. This function is meant to be maximized and is defined by

$$\begin{aligned}\Rightarrow 0 &\geq \mathcal{J}(p, \phi) = -KL(p_\theta(z, y|x)||q_\phi(z, y|x)) \\ &= -\mathbb{E}_{p_\theta(z, y|x)}\left[\log \frac{p_\theta(z, y|x)}{q_\phi(z, y|x)}\right] = -\mathbb{E}_{p_\theta(z, y|x)}\left[\log \frac{p(y|z)p_\theta(z|x)}{q_\phi(y|x)q_\phi(z|x)}\right].\end{aligned}$$

We further assume conditional independence and apply the following factorizations $q_\phi(z, y|x) = q_\phi(y|z, x)q_\phi(z|x) = q_\phi(y|x)q_\phi(z|x)$, with $q_\phi(z|x) = \sum_{y=1}^K q(z|x, y)q_\phi(y|x)$. Likewise, we define $p_\theta(z, y|x) = p(y|z)p_\theta(z|x)$. The parameters θ and ϕ were omitted for the distributions assumed to remain constant throughout the EM procedure. These distributions include $p(y|z)$, representing the classifier employing the prior encoder, and $q(z|x, y)$, representing the recognition model of the GdVAE. The iterative EM procedure for step $t \in \{1, \dots, T\}$ can be expressed as follows:

E-Step: Choose a distribution $p = p_\theta(z, y|x)$ that maximizes $\mathcal{J}(p, \phi)$ for fixed ϕ^t .

- Since $p(y|z)$ is fixed the optimum is given by choosing $p_\theta(z|x) = q_{\phi^t}(z|x)$.

M-Step: Choose parameters ϕ^{t+1} that maximize $\mathcal{J}(p, \phi)$ for fixed $p = p_\theta(z, y|x)$.

- The optimum is given by choosing $q_{\phi^{t+1}}(y|x) = \mathbb{E}_{q_{\phi^t}(z|x)}[p(y|z)]$. This choice defines $q_{\phi^{t+1}}(z|x) = \sum_{y=1}^K q(z|x, y)q_{\phi^{t+1}}(y|x)$ as well.

Proof for the M-Step: We can simplify the optimization process by assuming that both $q_\phi(z|x)$ and $p_\theta(z|x)$ are Gaussian mixture models with the same number, denoted as K , of mixture components. Drawing inspiration from [10], we can next

derive a lower bound, denoted as $\mathcal{L}(p, \phi)$, for $\mathcal{J}(p, \phi)$ through the application of the *log-sum* inequality and by inserting $p_\theta(z|x) = q_{\phi^t}(z|x)$ into $\mathcal{J}(p, \phi)$

$$\begin{aligned}\mathcal{J}(p, \phi) &= - \int \sum_{y=1}^K p(y|z) q_{\phi^t}(z|x) \cdot \log \frac{p(y|z) \sum_{y^*=1}^K q(z|x, y^*) q_{\phi^t}(y^*|x)}{q_\phi(y|x) \sum_{y^*=1}^K q(z|x, y^*) q_\phi(y^*|x)} dz \\ &\geq \mathcal{L}(p, \phi) = - \int \sum_{y=1}^K p(y|z) q_{\phi^t}(z|x) \cdot \log \frac{p(y|z) q(z|x, y) q_{\phi^t}(y|x)}{q_\phi(y|x) q(z|x, y) q_\phi(y|x)} dz \\ &= -\mathbb{E}_{q_{\phi^t}(z|x)} \left[\sum_{y=1}^K p(y|z) \log \frac{p(y|z) q_{\phi^t}(y|x)}{q_\phi(y|x) q_\phi(y|x)} \right].\end{aligned}$$

To maximize $\mathcal{L}(p, \phi)$, we employ the Lagrange multiplier $\lambda \in \mathbb{R}$ and set the derivative with respect to a specific class c to zero

$$\begin{aligned}0 &= \frac{\partial}{\partial q_\phi(c|x)} \left[\mathcal{L}(p, \phi) - \lambda \left(\sum_{y=1}^K q_\phi(y|x) - 1 \right) \right] = \mathbb{E}_{q_{\phi^t}(z|x)} \left[2 \frac{p(c|z)}{q_\phi(c|x)} \right] - \lambda, \\ &\Rightarrow q_{\phi^{t+1}}(y|x) = \mathbb{E}_{q_{\phi^t}(z|x)} [p(y|z)].\end{aligned}$$

We determine the value of λ by solving this equation for K classes, incorporating the property $\sum_{y=1}^K q_\phi(y|x) = 1$. Subsequently, we arrive at the solution $q_{\phi^{t+1}}(y|x) = \mathbb{E}_{q_{\phi^t}(z|x)} [p(y|z)]$ or, equivalently, the parameter ϕ^{t+1} that maximizes $\mathcal{J}(p, \phi)$ while keeping $p_\theta(z, y|x)$ fixed. This optimization is performed for a single input x , and we approximate the expectation through Monte Carlo integration. Therefore, we sample $z^{(s)}$ from $q_{\phi^t}(z|x)$ and calculate $q_{\phi^{t+1}}(y|x) = \mathbb{E}_{q_{\phi^t}(z|x)} [p(y|z)] \approx \frac{1}{S} \sum_{s=1}^S p_\theta(y|z^{(s)})$.

B.3 Optimality of Linear Explainer Functions (Eq. (9) and Eq. (10))

Euclidean Space ($V := I$). Our SEM is regularized to produce a linear separating hyperplane due to the linear classifier we employ. This results in a linear path for CF generation, where the classifier's gradient vector w describes the *shortest path* for CF generation (see Fig. 4a). This *latent space closeness* is also used as an L2-based metric $\text{dist}_I^2(z^{(1)}, z^{(2)})$ to measure CF proximity [43] or to directly optimize a perceptual loss for CF generation [24].

Riemannian Manifolds ($V := \Sigma_z^{-1}$). A generalized perspective on the distance function arises from considering the theoretical analysis on Riemannian manifolds presented in [5, 19]. According to [19], a Riemannian manifold, denoted as the pair (\mathcal{M}, v_z) , can be understood as a smoothly curved space \mathcal{M} (e.g., latent space) equipped with a Riemannian metric v_z . The Riemannian metric is an inner product $v_z(a, b) = \langle a, b \rangle_z = a^T V(z)b$ on the tangent space $\mathcal{T}_z \mathcal{M}$ for each $z \in \mathcal{M}$. The metric tensor $V(z)$ is a positive definite matrix that induces a distance measure. Assuming a constant metric tensor or single metric learning,

the Mahalanobis distance is obtained [5, 19]

$$\text{dist}_V^2(z^{(1)}, z^{(2)}) = \left\| z^{(1)} - z^{(2)} \right\|_V^2 = (z^{(1)} - z^{(2)})^T V (z^{(1)} - z^{(2)}). \quad (13)$$

If the metric tensor is chosen to be the identity matrix $V := I$, we obtain the L2-based Euclidean distance. To define a smooth continuous Riemannian metric with a metric tensor in every point z , [19] propose

$$V(z) = \sum_{k=1}^K \pi_k(z) V_k, \quad (14)$$

where V_k are pre-trained metric tensors, *e.g.*, obtained by a Large Margin Nearest Neighbor classifier, which are associated with the mean of each class. π_k are weights that change smoothly with z , where each $\pi_k > 0$ and $\sum_{k=1}^K \pi_k = 1$. As an example of a smooth weight function, they use the following

$$\pi_k(z) \propto \exp \left(-\frac{\rho}{2} \left\| z - z^{(k)} \right\|_{V_k}^2 \right), \quad (15)$$

with the constant ρ and class means $z^{(k)}$. Based on this continuous Riemannian metric (see Eq. (14)), [5] propose using the trained covariance matrices from a VAE’s encoder $q_\phi(z|x) = \mathcal{N}(\mu_z(x; \phi), \Sigma_z(x; \phi))$ to define the metric. Specifically, they employ a weighted linear combination of $V_k = \Sigma_z^{-1}(x^{(k)}; \phi)$, where the training data, or a subset thereof, is used to approximate the metric in the latent space. In addition to Eq. (14), there is a supplementary additive component which is approximately zero. They further observe that, due to the Evidence Lower Bound (ELBO) objective, variables that are close in the latent space with respect to $V(z)$ will also produce samples that are close in the image space in terms of L2 distance, which is crucial for ensuring counterfactual proximity.

Optimization and Assumptions. In our analysis, we assume that the regularizer included in the ELBO induces a surrogate posterior $q_\phi(z|x, y)$ that closely approximates the true posterior $p_\theta(z|y) = \mathcal{N}(\mu_z(y; \theta), \Sigma_z(y; \theta))$. Given this approximation, we may consider using pre-trained metric tensors from the GDA (Gaussian discriminant analysis) classifier instead of those derived from a Large Margin Nearest Neighbor classifier [19]. By doing so, as referenced in Eq. (14), we obtain a Riemannian metric

$$V(z) = \sum_{k=1}^K \pi_k(z) \Sigma_z^{-1}(y=k; \theta), \quad (16)$$

with $\pi_k(z) = p_\theta(y=k|z)$. In our experiments, we use only two classes and have chosen the covariance to be independent of the class y in order to obtain linear discriminants. Consequently, this results in a constant metric tensor, effectively employing a single metric

$$V(z) = \pi_1(z) \Sigma_z^{-1} + (1 - \pi_1(z)) \Sigma_z^{-1} = \Sigma_z^{-1} = V. \quad (17)$$

Having defined two types of distance metrics, the L2 and Mahalanobis distances, we can now optimize the objective

$$\mathcal{I}_f(z, \delta) = \arg \min_{z^\delta} \text{dist}_V^2(z^\delta, z), \quad \text{subject to } f(z^\delta) = \delta. \quad (18)$$

To minimize the objective $\text{dist}_V^2(z^\delta, z)$, we use a Lagrange multiplier $\lambda \in \mathbb{R}$ and set the derivatives with respect to the counterfactual z^δ and λ to zero

$$\mathcal{L}(z^\delta) = (z^\delta - z)^T V (z^\delta - z) + \lambda(f(z^\delta) - \delta), \quad (19)$$

$$\frac{\partial \mathcal{L}(z^\delta)}{\partial z^\delta} = 2(z^\delta - z)V + \lambda w = 0, \quad \Rightarrow z^\delta = z + \frac{\lambda}{2}V^{-1}w, \quad (20)$$

$$\frac{\partial \mathcal{L}(z^\delta)}{\partial \lambda} = 0, \quad \Rightarrow f(z^\delta) = w^T z^\delta + b = \delta. \quad (21)$$

By combining Eqs. (20) and (21), we arrive at the solution

$$w^T z^\delta = w^T z + \frac{\lambda}{2} w^T V^{-1} w = \delta - b, \quad (22)$$

$$\Rightarrow z^\delta = z + \kappa V^{-1} w, \quad \text{with } \kappa = \frac{\delta - w^T z - b}{w^T V^{-1} w}. \quad (23)$$

Given the assumptions used, we obtain a linear explainer function to generate counterfactuals, regardless of whether we choose the common L2-based metric $V = I$ (Euclidean space) or the Riemannian metric $V = \Sigma_z^{-1}$. Training with $\Sigma_z = \sigma^2 I$ instead of $\Sigma_z = \text{diag}(\sigma_{z_1}^2, \dots, \sigma_{z_M}^2)$ results in equivalent explainer functions, thus yielding equal empirical results for both L2-based and Riemannian-based metrics. Consequently, a linear function is optimal, and thus there is no superior solution for generating counterfactuals under these conditions. If the assumption that $q_\phi(z|x, y) \approx p_\theta(z|y)$ is not met, a non-linear CF method (*e.g.*, C3LT) may better fulfill the proximity property in the image space.

C Training and Model Details

The software to train and perform inference with our GdVAE model is available in the supplemental code.

C.1 Datasets

MNIST [31]. The MNIST dataset comprises two sets: a training set with 60,000 labeled examples and a test set with 10,000 labeled examples. Each example is a 28x28 pixel grayscale image representing a handwritten digit ranging from 0 to 9. For the predictive performance analysis (Tab. 2) we use all classes and for the evaluation of counterfactuals (Tab. 3) we exclusively utilize the digits 0 and 1 for the training and to generate counterfactuals (denoted by "MNIST-Binary 0/1"). We adhere to the standard data splits for both training and testing.

CIFAR-10 [29]. The CIFAR-10 dataset comprises over 60,000 images, along with annotations for ten classes. Each example is a 32x32 pixel image with 3 color channels. We adhere to the standard data splits for both training and testing. This dataset is used to analyze the predictive performance in Tab. 2.

CelebA [32]. The CelebA dataset comprises over 200,000 face images, along with annotations that cover a range of attributes, including gender, age, and facial expression. In our analysis, we employ a center crop of 128x128 pixels for the images and resize them to 64x64 pixel images with 3 color channels. For counterfactual evaluation (Tab. 3), we use the attributes of "smiling" (labeled as 1) and "not smiling" (labeled as 0), while the "gender" attribute is used to assess predictive performance (Tab. 2). We adhere to the standard data splits for both training and testing. For licensing details and information on human subject data collection, please see the reference [32].

FFHQ [25]. The Flickr-Faces-HQ (FFHQ) dataset comprises over 70,000 high-resolution (1024x1024 pixels) images of human faces, curated for diversity in age, ethnicity, and accessories. The dataset was developed by NVIDIA. For counterfactual generation (Fig. 1), we use the attributes of "smiling" (labeled as 1) and "not smiling" (labeled as 0) available at <https://github.com/DCGM/ffhq-features-dataset/>. For licensing details and information on human subject data collection, please see the reference [25].

C.2 GdVAE Details

GdVAE Training. The training algorithm is outlined in the supplemental code. The fundamental GdVAE training procedure, does not include the consistency loss. This particular training method is employed in Tab. 2 and in Tab. 3 the consistency regularizer is incorporated. During the training process, we employ both the global and local-L2 explainer functions to generate counterfactual (CF) examples. To generate samples, we follow a random selection procedure, sampling from either explainer function with equal probability. Fig. 5 illustrates the inputs to the loss functions.

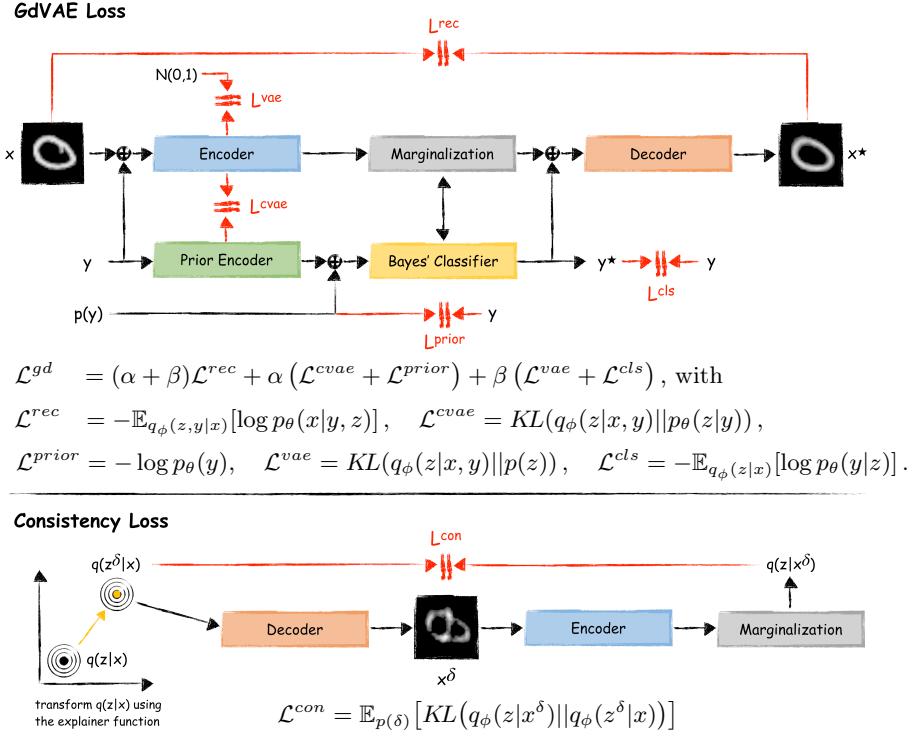


Fig. 5: Diagram illustrating the GdVAE model components and their interactions, highlighting the key elements that contribute to loss function computation.

The hyperparameters used for training and architectural decisions to replicate the results from the paper are summarized in Tab. 5. The architectures are presented in Tabs. 6 to 10. We maintain consistent hyperparameters ($\alpha/\beta = 1$, $\gamma = 1$, $T = 3$, $S = 20$) across all datasets. Baseline models adopt dataset-specific settings for improved results, as indicated in Tab. 12.

GdVAE Architectures. The architecture of the neural network for the CelebA GdVAE is inspired by the encoder and decoder configurations presented in [9]. When dealing with CelebA, we use 4×4 kernels for every convolutional layer, with a stride of 2 and padding set to 1. However, the last convolutional layer deviates from this pattern and employs the default stride of 1 with no padding. In between these layers, Rectified Linear Units (ReLU) are applied.

For more detailed model specifications related to CelebA, please consult Tab. 7. The notation "FC ($\times 2$)" signifies that two distinct fully connected networks are in use for both the mean and log variance calculations. In our architecture, "Conv2d" stands for 2D convolution, while "ConvT2d" corresponds to 2D transposed convolution. The stride and padding settings for these operations are represented as "s" and "p", respectively. These operations are implemented using the torch.nn package in PyTorch.

It's worth noting that the prior encoder and label embeddings remain consistent across all models and datasets, as displayed in Tab. 6. Architectural details for other datasets are provided in Tabs. 8 to 10. For FFHQ we use the pre-trained StyleGAN architecture based on [53]. Our GdVAE model, as shown in Table 10, is integrated between StyleGAN's encoder and decoder. For a comprehensive understanding of the hyperparameters used in the experiments, please refer to Tab. 5 in the case of experiments detailed in Tabs. 2 and 3. An ablation study regarding the hyperparameters can be found in Appendix E.1.

Table 5: Hyperparameters and architectural decisions for our GdVAE model configurations on MNIST, CIFAR-10, and CelebA. Values specific to binary classification scenarios (MNIST - binary 0/1, CelebA) that include the use of consistency loss are enclosed in parentheses, if they differ from the parameters used in other experiments. The ADAM optimizer is consistently employed across all datasets.

| | MNIST | CIFAR-10 | CelebA |
|---|--------|----------|--------|
| Batch size | 64 | 64 | 64 |
| Learning rate | 0.0005 | 0.0005 | 0.0005 |
| Epochs | 24 | 24 | 24 |
| Number classes K | 10(2) | 10 | 2 |
| Latent dimension M | 10 | 64 | 64 |
| Latent dimension L | 10(4) | 10 | 4 |
| Samples S in Algorithm 1 | 20 | 20 | 20 |
| Samples for \mathbb{E} in Eq. (11) | (10) | - | (10) |
| ϵ for $p(\delta) = \mathcal{U}(-\epsilon, \epsilon)$ | (2.94) | - | (2.94) |
| Samples O for \mathbb{E} in Eqs. (7) and (8) | 1 | 1 | 1 |
| Iterations T in Algorithm 1 | 3 | 3 | 3 |

Table 6: Prior encoder and label embeddings. In binary classification tasks, we opt for a class-independent choice of $\Sigma_z(y; \theta) = \Sigma_z(\theta)$. Consequently, we employ a distinct encoder network for the covariance $\Sigma_z(\theta)$, which is obtained by utilizing a constant input, $y = 1$.

| Prior Encoder $p_\theta(z y) = \mathcal{N}(\mu_z(y; \theta), \Sigma_z(y; \theta))$ |
|--|
| Input: K values |
| FC: L values \Rightarrow FC: L values \Rightarrow FC: L values |
| FC ($\times 2$): $\text{dim}(z) = M$ values |
| Encoder Label Embedding for y in $q_\phi(z x, y)$ |
| Input: K values \Rightarrow FC: 1 channel, $W \times H$ image |
| Decoder Label Embedding for y in $p_\theta(x y, z)$ |
| Input: K values \Rightarrow FC: 1 value |

Table 7: Architecture for CelebA. We center crop and resize the images to have a width and height of $(W \times H) = (64 \times 64)$ and keep the three channels $C = 3$. The class label y is incorporated into the input image as a fourth channel through a label embedding (Tab. 6). The latent space's dimensionality is defined as $\dim(z) = M = 64$. We utilize $q_\phi(z|x, y) = \mathcal{N}(\mu_z(x, y; \phi), \Sigma_z(x, y; \phi))$, with diagonal covariance matrix $\Sigma_z(x, y; \phi) = \text{diag}(\sigma_{z_1|x,y}^2, \dots, \sigma_{z_M|x,y}^2)$.

| Encoder $q_\phi(z x, y)$ | Decoder $p_\theta(x y, z)$ |
|--|---|
| Input: $4 \times 64 \times 64$ | Input: $\dim(z) + 1 = M + 1$ values |
| Conv2d: 32 channels, 4×4 kernel, s=2, p=1 | FC: 256 channels, 1×1 image |
| Conv2d: 32 channels, 4×4 kernel, s=2, p=1 | ConvT2d: 64 channels, 4×4 kernel |
| Conv2d: 64 channels, 4×4 kernel, s=2, p=1 | ConvT2d: 64 channels, 4×4 kernel, s=2, p=1 |
| Conv2d: 64 channels, 4×4 kernel, s=2, p=1 | ConvT2d: 32 channels, 4×4 kernel, s=2, p=1 |
| Conv2d: 256 channels, 4×4 kernel | ConvT2d: 32 channels, 4×4 kernel, s=2, p=1 |
| FC: $2 \cdot \dim(z) = 2 \cdot M$ values | ConvT2d: 3 channels, 4×4 kernel, s=2, p=1 |
| Output: M values for $\mu_z(x, y; \phi)$ and $\log \Sigma_z(x, y; \phi)$ | Output: $3 \times 64 \times 64$ image for $\mu_x(y, z; \theta)$ |

Table 8: Architecture for CIFAR-10.

| Encoder $q_\phi(z x, y)$ | Decoder $p_\theta(x y, z)$ |
|--|---|
| Input: $4 \times 32 \times 32$ | Input: $\dim(z) + 1 = M + 1$ values |
| Conv2d: 64 channels, 4×4 kernel, s=2, p=1 | FC: 256 channels, 1×1 image |
| Conv2d: 128 channels, 4×4 kernel, s=2, p=1 | ConvT2d: 256 channels, 2×2 kernel |
| Conv2d: 256 channels, 4×4 kernel, s=2, p=1 | ConvT2d: 256 channels, 2×2 kernel, s=2 |
| Conv2d: 256 channels, 2×2 kernel, s=2 | ConvT2d: 128 channels, 4×4 kernel, s=2, p=1 |
| Conv2d: 256 channels, 2×2 kernel | ConvT2d: 64 channels, 4×4 kernel, s=2, p=1 |
| FC ($\times 2$): $\dim(z) = M$ values | ConvT2d: 3 channels, 4×4 kernel, s=2, p=1 |
| Output: M values for $\mu_z(x, y; \phi)$ and $\log \Sigma_z(x, y; \phi)$ | Output: $3 \times 32 \times 32$ image for $\mu_x(y, z; \theta)$ |

Table 9: Architecture for MNIST.

| Encoder $q_\phi(z x, y)$ | Decoder $p_\theta(x y, z)$ |
|--|---|
| Input: $2 \times 28 \times 28$ | Input: $\dim(z) + 1 = M + 1$ values |
| Conv2d: 64 channels, 6×6 kernel, s=2 | FC: 256 channels, 4×4 image |
| Conv2d: 128 channels, 5×5 kernel | ConvT2d: 128 channels, 5×5 kernel |
| Conv2d: 256 channels, 5×5 kernel | ConvT2d: 64 channels, 5×5 kernel |
| FC ($\times 2$): $\dim(z) = M$ values | ConvT2d: 1 channel, 6×6 kernel, s=2 |
| Output: M values for $\mu_z(x, y; \phi)$ and $\log \Sigma_z(x, y; \phi)$ | Output: $1 \times 28 \times 28$ image for $\mu_x(y, z; \theta)$ |

Table 10: Architecture for FFHQ.

| Encoder $q_\phi(z x, y)$ | Decoder $p_\theta(x y, z)$ |
|--|---|
| Input: $2 \times 96 \times 96$ | Input: $\dim(z) + 1 = M + 1$ values |
| Conv2d: 32 channels, 4×4 kernel, s=2, p=1 | FC: 256 channels, 1×1 image |
| Conv2d: 32 channels, 4×4 kernel, s=2, p=1 | ConvT2d: 128 channels, 3×3 kernel, s=1, p=0 |
| Conv2d: 64 channels, 4×4 kernel, s=2, p=1 | ConvT2d: 64 channels, 4×4 kernel, s=2, p=1 |
| Conv2d: 64 channels, 4×4 kernel, s=2, p=1 | ConvT2d: 64 channels, 4×4 kernel, s=2, p=1 |
| Conv2d: 128 channels, 4×4 kernel, s=2, p=0 | ConvT2d: 32 channels, 4×4 kernel, s=2, p=1 |
| Conv2d: 256 channels, 3×3 kernel, s=1, p=0 | ConvT2d: 32 channels, 4×4 kernel, s=2, p=1 |
| FC: $2 \cdot \dim(z) = 2 \cdot M$ values | ConvT2d: 1 channel, 4×4 kernel, s=2, p=1 |
| Output: M values for $\mu_z(x, y; \phi)$ and $\log \Sigma_z(x, y; \phi)$ | Output: $1 \times 96 \times 96$ feature map for $\mu_x(y, z; \theta)$ |

C.3 Details of Baseline Models and Assets

Black-Box Baseline. The optimal baseline for evaluating the predictive performance of the GdVAE in Tab. 2 involves training a discriminative classifier and a CVAE jointly. Unlike the GdVAE architecture, this classifier employs a separate neural network as its backbone. This backbone network replicates the structure of the CVAE’s encoder (refer to Tabs. 7 to 9), allowing the CVAE to learn an optimal representation for reconstruction, while the discriminative classifier learns an independent representation for classification.

The classifier leverages the CVAE encoder without the additional class input to generate a latent representation z . Subsequently, this latent vector is processed by a one-layer fully connected neural network, which uses a softmax function to map z to the class output.

Importance Sampling (IS) [45, 48, 54] for Generative Classification. An alternative approach to our EM-based inference is presented in [48], where they use separate VAEs for different classes and perform an importance sampling strategy to obtain a generative classifier. We extend this approach to CVAEs and use the importance sampling strategy as a baseline in Tab. 2. The importance sampling strategy for CVAEs to approximate the likelihood $p_\theta(x|y)$ is given by

$$p_\theta(x|y) = \mathbb{E}_{z \sim p_\theta(z|y)}[p_\theta(x|y, z)] = \int p_\theta(x|y, z)p_\theta(z|y)dz \quad (24)$$

$$\approx \frac{1}{S} \sum_{s=1}^S \frac{p_\theta(x|y, z^{(s)})p_\theta(z^{(s)}|y)}{q_\phi(z^{(s)}|x, y)}, \quad (25)$$

where the samples $z^{(s)}$ are drawn from $q_\phi(z|x, y)$ and the likelihood is afterwards used in a Bayes’ classifier $p_\theta(y|x) = \eta p_\theta(x|y)p_\theta(y)$. The drawback of this approach is that in addition to the encoder it requires to invoke the decoder model for each sample and since the dimensionality of the input space $x \in \mathbb{R}^N$ is typically larger than the dimensionality of the latent space $z \in \mathbb{R}^M$, more samples are required for a good approximation.

This baseline method is denoted by importance sampling (IS) in Tab. 2. The architecture and learning objective remain unchanged, but instead of using Algorithm 1, importance sampling in the image space is employed. For a fair comparison, we utilize $S = 60$ for the importance sampling method. This is in line with the GdVAE, which uses $S = 20$ samples but conducts $T = 3$ iterations, resulting in a total of 60 samples as well.

ProtoVAE [13] (<https://github.com/SrishtiGautam/ProtoVAE>). We utilized the original implementation as detailed in [13]. In our adaptation, we substituted the backbone network with the GdVAE backbone and configured the model to work with a single prototype. Furthermore, we explored various hyperparameter settings, such as adjusting the loss balance to match our reconstruction loss and binary cross-entropy loss. The results are presented in Tab. 13.

GANalyze [15] (<http://ganalyze.csail.mit.edu/>). We utilized the original implementation as detailed in [15]. The code already includes a PyTorch version,

which is mainly used for the implementation of the so called transformer $T(z, \alpha)$. The transformer is equivalent to a linear explainer function

$$\mathcal{I}_f(z, \alpha, k) = z^\alpha = z + \alpha w^{(k)}, \quad (26)$$

where α is the requested confidence for the counterfactual class and $w^{(k)}$ is the direction that is learned for each counterfactual class k . As in the original implementation we use a quadratic loss between the desired confidence and the classifier output and utilize the per sample loss

$$\mathcal{L}_{cf}(x^\alpha, \alpha, k) = [p_\theta(y = k|x^\alpha) - \alpha]^2, \quad (27)$$

where k is the counterfactual class with respect to the class label c of the input image x , $x^\alpha = \mathcal{I}_f(h(x), \alpha, k)$ the counterfactual, and h the encoder. We optimize GANalyze by approximating the expectation in

$$\mathcal{L}^{GANalyze}(w) = \mathbb{E}_{\alpha \sim \mathcal{U}(0,1)}[\mathcal{L}_{cf}(x^\alpha, \alpha, k)] \quad (28)$$

with 10 samples for each training image. GANalyze's primary objective is to learn a vector $w^{(k)}$ for each class, as opposed to a single vector as seen in GdVAE. Instead of using a GAN, we leverage our pre-trained GdVAE as an encoder, decoder, and classifier. The conditional decoder takes the counterfactual class k as its input. Moreover, we have explored various hyperparameter settings. For instance, one such setting involves the use of normalization, a practice not documented in the paper but applied in the code.

This normalization enforces the counterfactual distance from the origin to be identical to the original input $z = h(x)$

$$\mathcal{I}_f^{norm}(z, \alpha, k) = \frac{\mathcal{I}_f(z, \alpha, k)}{\|z\|_2}. \quad (29)$$

The results of various hyperparameter settings are presented in Tab. 12.

UDID [49] (<https://github.com/anvoynov/GANLatentDiscovery>). We utilized the original implementation as detailed in [49]. The code already includes a PyTorch version, which is mainly used for the implementation of the so called latent deformator $A(\alpha e_k)$ and reconstructor $R(x, x^\alpha)$ that are employed for unsupervised latent space analysis. The deformator is a non-linear explainer function

$$\mathcal{I}_f(z, \alpha, k) = z^\alpha = z + A(\alpha e_k), \quad (30)$$

where $e_k \in \mathbb{R}^K$ are standard unit vectors defining the directions, one for each class. $\alpha \sim \mathcal{U}(0, 1)$ defines the strength of manipulation and $A(\cdot) \in \mathbb{R}^{M \times K}$ is defined by a neural network. The primary objective of the reconstructor $R(x, x^\alpha) = (\hat{k}, \hat{\alpha})$ is to compare the original image x and the manipulated version x^α , aiming to replicate the manipulation in the latent space. Consequently, the method seeks to discover disentangled directions. Instead of employing a GAN, we utilize our pre-trained GdVAE as an encoder, decoder, and classifier. Additionally, we

introduce a supervised learning component \mathcal{L}_{cf} to align the method with other supervised CF methods. We therefore employ a per-sample loss defined by

$$\mathcal{L}^{UD}(A) = \mathbb{E}_\alpha \left[\mathcal{L}_{cl}(k, \hat{k}) + \lambda \mathcal{L}_r(\alpha, \hat{\alpha}) + \mathcal{L}_{cf}(x^\alpha, \alpha, k) \right], \quad (31)$$

where $\lambda = 0.25$ represents the weight for the regression task, \mathcal{L}_{cl} corresponds to the cross-entropy function, \mathcal{L}_r pertains to the mean absolute error, and \mathcal{L}_{cf} denotes the supervised loss for the counterfactuals. The supervised loss aligns with the one used in GANalyze (as in Eq. (27)). We use a sample size of 10 for α to approximate the expectation. The results of various hyperparameter settings are presented in Tab. 12. To enhance the method and refine the proximity property, we incorporated a proximity loss $\mathcal{L}_{prox}(x, x^\alpha) = \|x - x^\alpha\|_2^2$, aiming to maintain close resemblance between the counterfactual and query images.

ECINN [21]. ECINN employs an invertible neural network and applies a post-hoc analysis of class conditional means within the latent space to determine an interpretable direction. They make the assumption that the covariance matrix Σ_z (see Sec. 3.1) is the identity matrix. Their post-hoc method is similar to our local-L2 function, but it approximates the true classifier using empirical mean values. Consequently, we view ECINN as an empirical implementation of our method, additionally estimating the empirical covariance.

Furthermore, they create two counterfactuals: one with high confidence for the opposing class and another precisely at the decision boundary. We replicate and extend their approach by using our explainer function (refer to Eq. (10)), wherein we empirically determine class conditional mean values and the covariance matrix after training the GdVAE. Thus, we employ $\bar{w} = \bar{\Sigma}_z^{-1} (\bar{\mu}_{z|k} - \bar{\mu}_{z|c})$, utilizing the empirical means $\bar{\mu}_{z|l}$ and covariance $\bar{\Sigma}_z$ with the predicted class of the GdVAE as label. The formula for the counterfactual at the decision boundary is exactly the one that ECINN would employ.

AttFind [30] (<https://github.com/google/explaining-in-style>). We utilized the original implementation as detailed in [30]. To accommodate our PyTorch-based model, we translated their TensorFlow implementation of the *AttFind* method. It's important to emphasize that our usage of the AttFind method is solely for identifying significant directions within the latent space of our GdVAE. We do not employ their GAN architecture.

The AttFind method operates by iterating through latent variables, evaluating the impact of each variable on the classifier's output for a given image, and subsequently selecting the top-k significant variables. An image is considered explained once AttFind identifies a manipulation of the latent space that leads to a substantial alteration in the classifier's output, effectively changing the classifier's class prediction. We employ their *Subset* search strategy, which focuses on identifying the top-k variables where jointly modifying them results in the most significant change in the classifier's output. This method is limited to producing class flips and is not intended for generating CFs with desired confidence values. Hence, we have labeled this method with AttFind[†] to denote its original purpose of effecting class changes only. The results are presented in Tab. 15, though their accuracy and MSE may not be directly comparable to those in Tab. 3.

EBPE [43] (https://github.com/...by_Progressive_Exaggeration). We utilized the original implementation as detailed in [43]. To accommodate our PyTorch-based model, we translated their TensorFlow implementation. The most significant modification involves replacing the GAN with the CVAE architecture utilized by all approaches. EPBE, aside from GdVAE, uniquely required decoder training, resulting in distinct latent space characteristics and reconstruction quality. EBPE solely uses the pre-trained GdVAE for classification.

The original EBPE version was designed to work with the CelebA dataset, which is more complex than the MNIST dataset. The MNIST classifier we aimed to explain achieved remarkably high accuracy and confidence, resulting in some bins within the EBPE training having no samples for generating images at specific confidence values. When a bin lacked any samples, it was impossible to generate additional images from it. This issue could be addressed through hyperparameter tuning. For hyperparameter analysis, we define the loss

$$\begin{aligned}\mathcal{L}^{EBPE} = & \lambda_{cGAN} \mathcal{L}_{cGAN}(G, D) + \lambda_{rec} \mathcal{L}_{rec}(G) \\ & + \lambda_{cyc} \mathcal{L}_{cyc}(G) + \lambda_{cfClss} \mathcal{L}_{cfClss}(G) + \lambda_{recClss} \mathcal{L}_{recClss}(G).\end{aligned}\quad (32)$$

Here, \mathcal{L}_{cGAN} represents EBPE’s conditional GAN loss, \mathcal{L}_{rec} is the reconstruction loss, \mathcal{L}_{cyc} denotes the cycle loss, \mathcal{L}_{cfClss} is the classification loss for the counterfactual, and $\mathcal{L}_{recClss}$ stands for the classification loss for the reconstruction of the input image. λ represents the weight of the corresponding loss. For further details, please see [43] and the original implementation. Detailed results for different configurations can be found in Tab. 12.

As highlighted in the ‘Limitations’ section, the results showcased in the main paper utilized 24 epochs to maintain uniformity across all methods. For improved EBPE performance with extended training, see the 96-epoch results in Tab. 12.

C3LT [26] (<https://github.com/khorrami/c3lt>). We utilized the original implementation as detailed in [26]. C3LT was initially designed for use with pre-trained models, and we applied this implementation to our GdVAE models.

While C3LT was originally tailored for the simpler consistency task of class modification without explicitly requesting a user-defined confidence, we extended its functionality by introducing a supervised loss term, denoted as \mathcal{L}_{cf} . This loss term serves to align the user-requested confidence level (α) with the predicted confidence ($\hat{\alpha}$), with $\hat{\alpha} = p_\theta(y = k | \mathcal{I}_f(h(x), \alpha, k))$. For \mathcal{L}_{cf} , we experimented with both quadratic (Eq. (27)) and cross-entropy functions to assess their performance. The results of these experiments can be found in Tab. 12.

In conclusion, we leveraged C3LT to facilitate these modifications and added a supervised loss term to ensure alignment between the requested confidence and the model’s predictions. The original version, without the added loss term, is referred to as C3LT[†] in Tab. 15.

C.4 Compute Resources

For training our GdVAE and baseline models, we had the option to utilize two high-performance PCs equipped with multiple GPUs. The first PC featured an Nvidia RTX 3090 and a Titan RTX, both boasting 24 GB of memory each. The second PC made use of two Nvidia A6000 GPUs, each equipped with 48 GB of memory. To establish a reference point for the required computational time, we considered the GdVAE exclusively for the counterfactual tasks, as they represent an upper limit for the models' demands.

For an individual epoch on binary MNIST, it took 5 minutes, and for CelebA, it took 60 minutes on the A6000. In the context of our hyperparameter sweep, we explored $25 = (5+8+6+6)$ different model configurations for MNIST and 12 for CelebA, encompassing parameters like the number of samples S , iterations T , the balance between α/β , and the consistency weight γ . Each of these configurations underwent four separate training runs to calculate the mean and standard deviations, ultimately leading to the training of 100 models for MNIST and 48 for CelebA, respectively.

Consequently, the total computational time required for this parameter analysis sums up to $(5 \text{ min}/\text{epoch} \cdot 100 \cdot 24 \text{ epochs} + 60 \text{ min}/\text{epoch} \cdot 48 \cdot 24 \text{ epochs}) = 81120 \text{ min}$, which is equivalent to 1352 h of GPU time.

D Evaluation Metrics

D.1 Metrics for Predictive Performance in Tab. 2

Accuracy (ACC). In Tab. 2, we assess the predictive performance of the classifier using the accuracy (ACC) metric, defined as

$$ACC = \frac{1}{D} \sum_{d=1}^D \mathbb{1} \left\{ \hat{y}^{(d)} = y^{(d)} \right\}, \quad (33)$$

with the indicator function $\mathbb{1}\{\cdot\}$. Here, \hat{y} is the class prediction, determined as the class with the highest probability according to $\hat{y} = \arg \max_i p_\theta(y=i|z)$. The ground-truth class labels are denoted by $y \in \{1, \dots, K\}$ and we have D test data samples. $y^{(d)}$ represents the d -th sample.

Mean Squared Error (MSE). To evaluate the reconstruction quality in Tab. 2, we calculate the mean squared error (MSE) between the ground-truth input image x and the reconstructed image \hat{x}

$$MSE = \frac{1}{D \cdot N} \sum_{d=1}^D \sum_{i=1}^N \left(x_i^{(d)} - \hat{x}_i^{(d)} \right)^2. \quad (34)$$

We assume images to be vectorized, with $x \in \mathbb{R}^N$. Here, N is defined as the product of the image's width (W), height (H), and number of channels (C), i.e., $N = W \cdot H \cdot C$. $x^{(d)}$ represents the d -th sample.

D.2 Metrics for CF Explanations in Tab. 3

Realism. Realism in counterfactuals is essential, as they should resemble natural data. To assess realism, we employ the *Fréchet Inception Distance (FID)* metric, a standard measure for this purpose [14, 26, 43]. We compute the FID values using the PyTorch implementation from [41].

Consistency. CF explanations aim to influence the behavior of a classifier to obtain desired outcomes. In case of our method the consistency task is to guarantee that the requested confidence value p_c accurately matches to the confidence prediction of the classifier $\hat{p}_c = p_\theta(y = c|h(g(z^\delta)))$ for the CF $z^\delta = \mathcal{I}_f(z, \delta)$. c is consistently assigned to the class label of the original input.

As demonstrated in [43], a method for assessing consistency is to create a plot that compares the expected classifier outcomes with the confidence predictions of the classifier for the generated CF. The optimum is reached when we obtain an identity relationship ($p_c = \hat{p}_c$) between the two quantities. We use kernel density estimates (KDE) to visualize this relationship (Figs. 10 and 17).

Similar to [14], we quantitatively evaluate the existence of a linear relationship using the *Pearson correlation coefficient*. In addition, we utilize the *mean squared error (MSE)* between the desired p_c and the estimated confidence \hat{p}_c

$$MSE = \frac{1}{D^*} \sum_{d=1}^{D^*} \left(p_c^{(d)} - \hat{p}_c^{(d)} \right)^2. \quad (35)$$

Here, we have D^* samples and $p_c^{(d)}$ represents the d -th sample. Given that we request confidence values within the range $p_c \in [0.05, 0.95]$, with a step size of 0.05, we acquire 19 counterfactuals per test image. As a result, $D^* = 19 \cdot D$, where D represents the number of test images.

Accuracy (ACC). Finally, the accuracy metric, as defined in [26] and denoted by "Val", is designed for the simpler consistency task, which assesses only class flips as a binary classification problem. To evaluate continuous confidence requests, we employ 12 bins $b \in \{1, \dots, 12\}$ and treat the bin assignment of the prediction as a multi-class problem. Therefore, we use Eq. (33) with ground-truth bins $b^{(d)}$ and predicted bins $\hat{b}^{(d)}$ for the D^* samples.

In particular, ACC and MSE are used to gauge whether the counterfactuals are predominantly generated at the extreme confidence levels, near one or zero. While the results may not be directly comparable, in the context of methods that focus solely on generating class flips, such as C3LT[†] [26] and AttFind[†] [30], we adopt accuracy with two bins as a reference point, as per the definition in [26] for binary classification. Please note that this task is considerably simpler, resulting in accuracy values in Tab. 15 being close to 100% for methods marked with †.
Proximity. The CF should only change the input in a minimal way $x^\delta = \arg \min_{x'} \rho(x, x')$, with respect to some user defined quantity $\rho(\cdot)$, e.g., $\rho(x, x') = \|x - x'\|_2$ [4].

Mean Squared Error (MSE). In [26] they measure proximity by means of the *L1-norm* between the query image x and the counterfactuals x^δ . We adopt this metric by using Eq. (34) for the D^* counterfactual samples.

E Additional Results

E.1 GdVAE Hyperparameter Analysis

Parameterization of the Loss Function. Before arriving at the final loss, we conducted several initial experiments on the MNIST dataset. Initially, we compared our loss formulation (A) derived from Eq. (12) with the loss formulation (B1) outlined in Sec. 3.1, where the training process is streamlined with the inference process. Additionally, we fine-tuned the reconstruction and classification loss by adjusting the scale of the reconstruction loss using $p_\theta(x|y, z) = \mathcal{N}(\mu_x(y, z; \theta), \Sigma_x(y, z; \theta))$, with $\Sigma_x := \Sigma_x(y, z; \theta) = 0.6^2 \cdot I$ for likelihood calculation (as in [12]), instead of the standard $\Sigma_x = I$. Furthermore, the cross-entropy loss for classification was rescaled in proportion to the image size, using the factor $0.1 \cdot W \cdot H \cdot C$ (B2), where W , H , and C represent the width, height, and channels of the input image, respectively. Finally, using two priors $p(z|y)$ and $p(z)$ is not essential, but they are part of the model. In our probabilistic view, the used factorization (see Sec. 3) justifies including $p(y|z)$ (and thereby $p(z)$), in contrast to works that merely append $p(y|z)$. However, using an uniform prior for $p(z)$, akin to excluding $p(z)$ from the loss, is valid. For MNIST, incorporating the normal prior results in a lower reconstruction error.

The results pertaining to these settings, including classification accuracy (ACC) and mean squared error (MSE) as a metric for reconstruction quality, are presented in Tab. 11. We adopted the B2 setting for all other experiments and datasets without dataset-specific fine-tuning.

Table 11: Evaluation of different loss functions and settings of the GdVAE on the MNIST dataset. We use $\alpha = \beta = 1$, $S = 20$, and $T = 3$. MSE is scaled by a factor of 10^2 . Mean values, including standard deviation, are reported over four training processes with different seed values.

| Setting | Details | $ACC\% \uparrow$ | $MSE \downarrow$ |
|----------------------------------|--|------------------|------------------|
| A: | Eq. (12), $\Sigma_x = I$ | 88.6 ± 1.25 | 1.19 ± 0.04 |
| B1: | Eqs. (7) and (8), $\Sigma_x = I$ | 96.0 ± 0.85 | 1.04 ± 0.02 |
| B2: | Eqs. (7) and (8), $\Sigma_x = 0.6^2 \cdot I$ | 99.0 ± 0.11 | 1.10 ± 0.04 |
| B2 w/o $p(z)$: | Eqs. (7) and (8), $\Sigma_x = 0.6^2 \cdot I$ | 99.0 ± 0.08 | 1.14 ± 0.03 |

How to Parameterize the EM-Based Algorithm? The GdVAE (Algorithm 1) requires user-defined values for the number of iterations (T) and samples (S). To determine the optimal number of iterations, we assess the stability of the training process on the MNIST dataset. The evaluation involves measuring the entropy of $q_\phi(y|x)$ during training epochs and comparing successive epochs. The results, depicted in Fig. 6, illustrate the change in entropy with increasing iterations, averaged over the entire test dataset. The error bars represent the standard error across four training runs with different seeds. Convergence is observed after 3 iterations. Other parameters are set to $\alpha = \beta = S = 1$. Based on the convergence analysis, we fix the value of T to 3 for subsequent experiments.

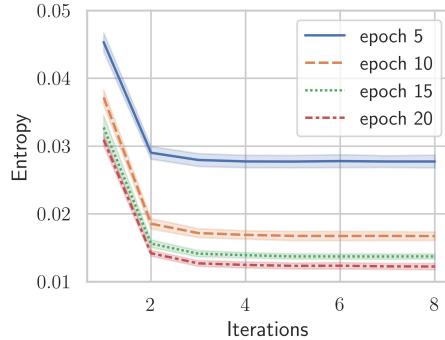


Fig. 6: Evaluation of number of iterations T during the training process on the MNIST dataset. Mean values, including standard error, are reported over four training processes with different seed values.

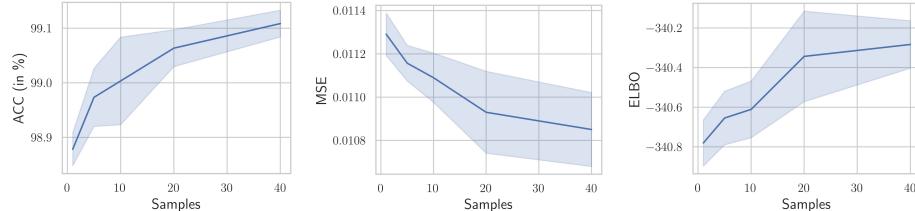


Fig. 7: Quality of models trained on the MNIST dataset with a different number of samples S .

After assessing convergence, we examine the influence of the number of samples S drawn from $q_\phi(z|x)$ on performance. We evaluate classification accuracy (ACC), reconstruction error (MSE), and the average ELBO of M1 and M2 through multiple training sessions with different random seeds using $S \in \{1, 5, 10, 20, 40\}$ and report the average values along with the standard error. The results, depicted in Fig. 7, demonstrate the expected improvement in performance with an increasing number of samples. All model trainings are stable and result in accuracy values $\geq 98\%$ and for $S \in \{20, 40\}$ comparable results are obtained with a small standard error regarding classification accuracy. By selecting $S = 20$, a balanced trade-off between computational load and performance is achieved, as indicated by the evidence lower bound (ELBO).

How to Balance the Loss for Models M1 and M2? The proposed GdVAE model incorporates two types of loss functions: the M1 and M2 losses. Both losses include the reconstruction loss, but differ primarily in their impact on the recognition model $q_\phi(z|x, y)$ and the classifier. While M1 focuses on training the likelihood model for purely generative classification, M2 provides a discriminative training signal for the generative classifier and enforces a standard normal distribution in the latent space. As described in Sec. 3.1, we ensure alignment

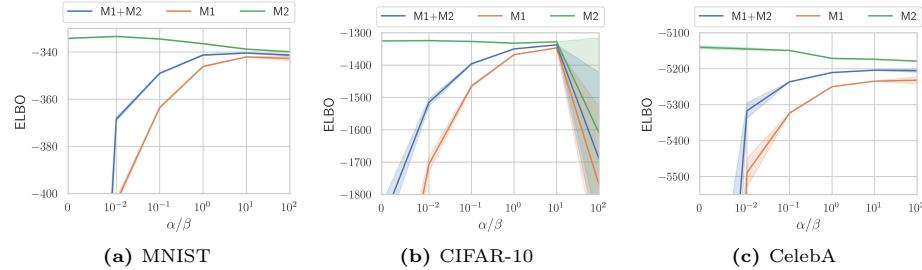


Fig. 8: Analysis of the optimal balance between models M1 and M2 on MNIST, CIFAR-10, and CelebA datasets. Mean values, including standard error, are reported over four training processes with different seed values.

between the training and inference processes, requiring both models to utilize the EM-based classifier for calculating the reconstruction loss using $p_\theta(x|z, y)$. For the EM algorithm to function properly, a prerequisite is that, given an input image x , the recognition model $q_\phi(z|x, y)$ must either have the lowest Kullback-Leibler divergence $KL(q_\phi(z|x, y)||p_\theta(z|y))$ for the correct class or be independent of y (e.g., [12]). The M1 loss does not consider these aspects, as it solely focuses on minimizing $KL(q_\phi(z|x, y)||p_\theta(z|y))$ for the correct class without enforcing it to be smaller than for the other classes. The desired behavior is enforced by adding the M2 loss, which incorporates a discriminative training signal and a KL divergence term that promotes independence. Furthermore, the classifier utilized in M2 relies on the likelihood function learned in M1, highlighting the interdependence between both models.

We analyze the optimal interplay between models M1 and M2 by evaluating the parameterization of the loss function defined by Eqs. (7) and (8), where we select suitable values for α and β . To keep notation concise, we use the ratio α/β instead of the individual values. We assess combinations of α and β from the set $\{0, 1, 10, 100\}$ and present the ELBO results graphically in Fig. 8. The results indicate that assigning equal weight values to the models ($\alpha/\beta = 1$) or using $\alpha/\beta = 10$ generally yields good performance across diverse datasets. To slightly enhance classification accuracy, we opted for $\alpha/\beta = 1$ in all experiments discussed in the main paper. For the MNIST dataset, opting for $\alpha/\beta = 1$ led to an accuracy of 99.0%, as opposed to 98.8% with nearly identical reconstruction error. Similar accuracy improvements were observed for CIFAR-10 (65.1% and 63.4%) and CelebA (96.7% and 96.4%) datasets. These results are used in Tab. 2.

How to Balance the Consistency Loss? The experiments aim to assess the quality of counterfactuals (CF) when varying the impact of the consistency loss. As in the main paper, we employ the Fréchet Inception Distance (FID) to gauge *realism* and Pearson and Spearman’s rank correlation coefficients to gauge *consistency*. Following a similar approach to [43], we also visualize the requested versus the actual response of the classifier using a kernel density estimate plot.

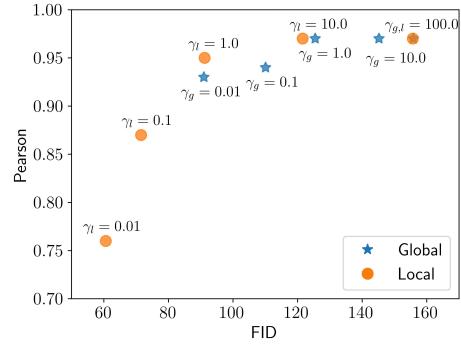


Fig. 9: Pearson correlation against FID scores on the MNIST dataset, where γ_l represents the consistency weight for the local method, and γ_g for the global one.

The ablation study, examining the influence of the consistency loss, is depicted in Fig. 10. We assess weight values λ for the consistency loss from the set $\{0.0, 0.01, 0.1, 1.0, 10, 100\}$. The error bars represent the standard error across four training runs with different seeds. We present results for both our local-L2 and global CF generation methods, as described in the main paper Sec. 3.2. x^* serves as the FID baseline, representing values obtained by applying the encoder and decoder directly to the test data. Thus, the FID score for x^* reflects the reconstructed test set.

The ablation study on the consistency loss in Fig. 10 reveals a trade-off between *consistency* and *realism*. When the consistency parameter exceeds $\lambda = 1$, the FID score of the ground truth reconstructions is significantly affected. Similarly, as the influence of the consistency regularizer increases, the correlation and FID value also increases. In Fig. 9, we visualize this trade-off by directly plotting the Pearson correlation against the FID scores.

The KDE plot uncovers an intriguing observation: even without utilizing the consistency regularizer, reasonable counterfactual examples can be generated with high correlation values ($\rho_p \approx 0.9$). Interestingly, counterfactuals tend to be generated at the extreme ends of the confidence range, near one or zero. Therefore, the method effectively flips the class of the query image, but the confidence values are not well calibrated. With increasing correlation values and improved calibration, the realism measure (FID) yields inferior results. One possible explanation is that more counterfactuals are generated near the decision boundary, where there is less real data available, resulting in a compromised natural appearance. It becomes evident that, for the consistency task [26, 30, 40] in which the user solely pre-defines the class label without specifying the confidence level, generating realistic images with low FID scores is considerably easier. Furthermore, we observe that global CF generation exhibits greater consistency with the classifier but lags behind in terms of realism when compared to the local CF generation process. The KDE plots for each model’s four training runs are displayed in Fig. 17. These are the models used in Tab. 3.

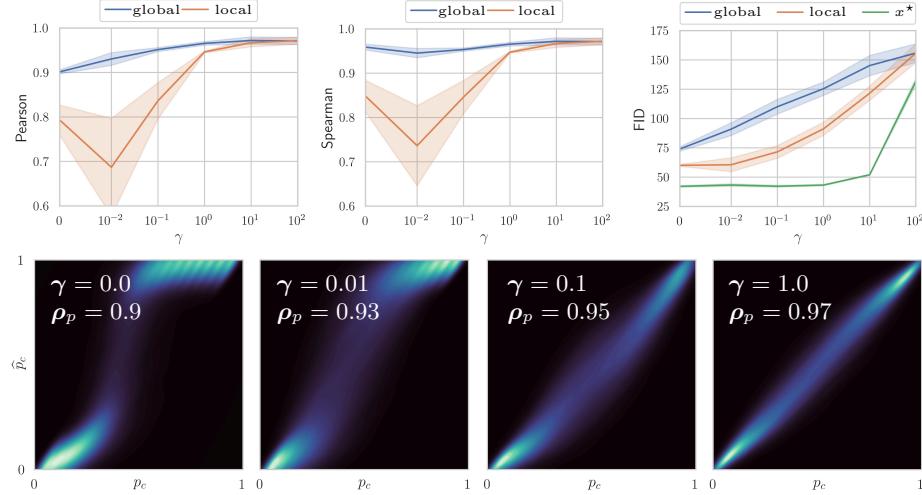


Fig. 10: Consistency of counterfactual examples using the MNIST dataset. Top: Pearson's ρ_p and Spearman correlation coefficients assess the relationship between the requested confidence and the classifier's output for counterfactual examples. Significance in correlation is indicated by a $p - \text{value} \leq 0.001$. Similarly, the Fréchet Inception Distance (FID) quantifies the image quality of the generated counterfactuals in comparison to the real data. Mean values, including standard error, are reported over four training processes with different seed values. Bottom: Consistency of the requested confidence versus the actual classifier confidence is visually depicted using a kernel density estimate (KDE) plot of the observations. The desired confidence output of the classifier for a counterfactual example x^δ is specified by p_c , while the actual confidence acquired by inputting the counterfactual to our classifier is represented by $\hat{p}_c = p(y = c|x^\delta)$. Additionally, the corresponding Pearson correlation coefficient ρ_p is visualized.

E.2 Analysis of Baseline Models

We conducted explorative parameter tuning for all baseline models, initially using the parameterization from the original implementation. Subsequently, we fine-tuned the parameters based on the results to achieve a balanced performance across various metrics and datasets. Results regarding ProtoVAE can be found in Tab. 13 and the results for the counterfactual methods are presented in Tabs. 12 and 15. Additionally, Tab. 14 presents the GdVAE realism results for various ranges of query confidences for comparison with Tab. 12. In the main paper, the FID scores for $p_c \in [0, 1]$ are reported.

ProtoVAE [13]. The hyperparameter analysis for ProtoVAE is detailed in Tab. 13. Setting A represents the original parameterization of the loss function as introduced by [13]. In an effort to enhance results, we modified the loss function based on our GdVAE settings, outlined in Tab. 11. Initially, we adjusted the scale of the reconstruction loss using $p_\theta(x|y, z) = \mathcal{N}(\mu_x(y, z; \theta), \Sigma_x(y, z; \theta))$, with $\Sigma_x := \Sigma_x(y, z; \theta) = 0.6^2 \cdot I$ for likelihood calculation (B1), maintaining the original weighting of the other loss terms. Setting B2 involves adjusting

Table 12: CF explanations across diverse baseline model configurations. Mean values, with standard deviation, are reported across four training runs with different seeds. The configurations used in the main paper’s experiments (Tab. 3) are **bolded**.

| Setup | | Consistency | | | Realism (FID) \downarrow | | |
|---------------------|-----------|----------------------|------------------|------------------|-------------------------------|----------------------------|------------------------|
| | | $\rho_p \uparrow$ | $ACC\% \uparrow$ | $MSE \downarrow$ | $\hat{p}_c \notin [0.1, 0.9]$ | $\hat{p}_c \in [0.1, 0.9]$ | $\hat{p}_c \in [0, 1]$ |
| | | GANalyze [15] | | | | | |
| MNIST Binary 0/1 | A | 0.48 \pm 0.04 | 2.8 \pm 0.8 | 19.33 \pm 1.31 | 79.88 \pm 10.04 | 110.36 \pm 8.54 | 55.77 \pm 7.46 |
| | B1 | 0.84 \pm 0.04 | 5.5 \pm 1.3 | 6.75 \pm 1.27 | 51.65 \pm 5.25 | 96.20 \pm 8.93 | 54.89 \pm 4.19 |
| | B2 | 0.93 \pm 0.03 | 25.9 \pm 16.8 | 2.09 \pm 1.35 | 122.16 \pm 17.74 | 158.01 \pm 27.19 | 132.36 \pm 23.35 |
| UDID [49] | | | | | | | |
| MNIST Binary 0/1 | A | 0.87 \pm 0.02 | 13.1 \pm 5.9 | 5.91 \pm 1.04 | 159.16 \pm 21.48 | 181.54 \pm 29.72 | 155.37 \pm 17.39 |
| | B1 | 0.87 \pm 0.03 | 0.6 \pm 0.2 | 8.47 \pm 0.12 | 46.74 \pm 4.34 | 104.99 \pm 12.26 | 42.98 \pm 3.93 |
| | B2 | 0.85 \pm 0.01 | 1.2 \pm 0.3 | 8.82 \pm 0.18 | 42.01 \pm 1.84 | 104.11 \pm 10.33 | 38.89 \pm 2.01 |
| | B3 | 0.80 \pm 0.01 | 2.9 \pm 0.5 | 10.45 \pm 0.16 | 45.90 \pm 1.01 | 104.46 \pm 7.61 | 42.40 \pm 1.14 |
| | C | 0.79 \pm 0.05 | 4.7 \pm 1.5 | 8.82 \pm 0.18 | 74.71 \pm 21.85 | 106.04 \pm 22.54 | 72.64 \pm 24.42 |
| CelebA Smiling | A | 0.98 \pm 0.01 | 71.5 \pm 6.8 | 0.24 \pm 0.13 | 411.09 \pm 22.61 | 381.26 \pm 24.01 | 370.69 \pm 19.59 |
| | B2 | 0.86 \pm 0.06 | 15.8 \pm 9.2 | 4.22 \pm 2.17 | 146.58 \pm 43.18 | 217.36 \pm 96.52 | 178.23 \pm 75.84 |
| EBPE [43] | | | | | | | |
| CelebA Smiling | A | 0.91 \pm 0.01 | 26.3 \pm 1.3 | 1.64 \pm 0.19 | 347.88 \pm 190.56 | 463.90 \pm 35.91 | 425.76 \pm 15.80 |
| | B1 | 0.86 \pm 0.02 | 20.0 \pm 4.6 | 3.21 \pm 0.92 | 217.41 \pm 0.76 | 252.24 \pm 12.78 | 241.99 \pm 8.13 |
| | B2 | 0.00 \pm 0.01 | 0.0 \pm 0.0 | 32.18 \pm 2.14 | 372.58 \pm 19.57 | n/a | 372.58 \pm 19.57 |
| | B3 | 0.99 \pm 0.01 | 80.8 \pm 3.3 | 0.09 \pm 0.05 | 387.15 \pm 11.34 | 403.74 \pm 9.36 | 391.91 \pm 16.79 |
| | C | 0.94 \pm 0.01 | 41.9 \pm 3.1 | 1.22 \pm 0.16 | 193.99 \pm 20.44 | 191.90 \pm 20.66 | 191.67 \pm 20.51 |
| | D1 | 0.97 | 54.39 | 0.56 | 185.17 | 185.16 | 184.96 |
| | D2 | 0.96 | 53.62 | 0.57 | 148.02 | 146.75 | 146.73 |
| | D3 | 0.96 | 53.71 | 0.57 | 120.45 | 120.14 | 120.00 |
| C3LT [26] | | | | | | | |
| MNIST Binary 0/1 | A | 0.84 \pm 0.08 | 2.0 \pm 0.8 | 4.71 \pm 2.21 | 94.13 \pm 12.06 | 106.90 \pm 14.25 | 92.30 \pm 19.59 |
| | B | 0.89 \pm 0.03 | 3.6 \pm 0.8 | 6.32 \pm 1.39 | 63.49 \pm 8.73 | 96.11 \pm 17.22 | 57.09 \pm 10.78 |
| CelebA Smiling | A | 0.89 \pm 0.04 | 3.5 \pm 2.5 | 3.52 \pm 0.71 | 122.29 \pm 12.06 | 138.82 \pm 15.33 | 133.13 \pm 13.56 |
| | B | 0.90 \pm 0.01 | 11.8 \pm 5.5 | 3.94 \pm 0.66 | 96.65 \pm 4.97 | 113.70 \pm 19.52 | 101.46 \pm 11.56 |

Table 13: Hyperparameter analysis of ProtoVAE. We report classifier’s accuracy (ACC) and mean squared error (MSE) of the reconstructions. MSE is scaled by a factor of 10^2 . Mean values, including standard deviation, are reported over four training processes with different seeds. Configurations for experiments in Tab. 2 are **bolded**.

| | Setting | $ACC\% \uparrow$ | $MSE \downarrow$ |
|----------|-------------------------|------------------|------------------|
| MNIST | A: ProtoVAE [13] | 99.1 \pm 0.17 | 1.51 \pm 0.23 |
| | B1 | 98.0 \pm 0.12 | 1.00 \pm 0.01 |
| | B2 | 94.8 \pm 0.37 | 1.01 \pm 0.01 |
| | B3 | 94.8 \pm 0.31 | 0.99 \pm 0.01 |
| CIFAR-10 | A: ProtoVAE [13] | 76.6 \pm 0.35 | 2.69 \pm 0.02 |
| | B1 | 58.4 \pm 1.31 | 0.92 \pm 0.02 |
| | B2 | 31.2 \pm 2.18 | 0.85 \pm 0.05 |
| | B3 | 30.0 \pm 3.25 | 0.37 \pm 0.02 |
| CelebA | A: ProtoVAE [13] | 96.6 \pm 0.24 | 1.32 \pm 0.10 |
| | B1 | 96.0 \pm 0.22 | 0.76 \pm 0.02 |
| | B2 | 88.9 \pm 0.80 | 0.75 \pm 0.05 |
| | B3 | 85.8 \pm 0.50 | 0.74 \pm 0.03 |

Table 14: FID evaluation of GdVAE’s CF explanations for different confidence ranges.

| | Method | Realism (FID) ↓ | | |
|------------|------------------------|-------------------------------|----------------------------|------------------------|
| | | $\hat{p}_c \notin [0.1, 0.9]$ | $\hat{p}_c \in [0.1, 0.9]$ | $\hat{p}_c \in [0, 1]$ |
| MNIST | Ours (global) | 126.93±8.73 | 140.11±9.04 | 125.45±11.32 |
| | Ours (local-L2) | 91.80±10.35 | 101.25±11.07 | 91.22±11.04 |
| Binary 0/1 | Ours (global) | 118.40±5.15 | 138.79±6.11 | 128.93±4.94 |
| | Ours (local-L2) | 86.01±2.60 | 86.59±2.47 | 85.52±2.37 |
| CelebA | Ours (global) | | | |
| | Ours (local-L2) | | | |
| Smiling | Ours (global) | | | |
| | Ours (local-L2) | | | |

the weight of the reconstruction and classification loss according to our GdVAE learning objective. Finally, setting B3, an extension of B2, incorporates a change in the learning rate from 0.001 to the one used by our GdVAE, which is 0.0005. As intended, we achieved a consistent reduction in reconstruction error; however, this improvement came at the cost of lower classification accuracy. Hence, we retained the parameterization from the original implementation.

GANalyze [15]. All hyperparameter tuning for GANalyze was done on the MNIST binary dataset and the results can be found in Tab. 12. In the initial setting (A), we employed the loss function from the original implementation, which included the normalization of counterfactuals. In the second setting (B1), we recreated the loss as described in the paper, omitting the normalization and achieving enhanced results. To further explore the efficacy of setting B1, we extended the training to 48 epochs (setting B2) instead of the original 24 epochs.

UDID [49]. The hyperparameter tuning for UDID primarily focused on optimizing the proximity loss and selecting the appropriate classification loss. In setting A, we assessed the model with no proximity loss, defined as $\mathcal{L} = \mathcal{L}^{UD} + \gamma \cdot \mathcal{L}_{prox}(x, x^\alpha)$, where $\gamma = 0$. Settings B1, B2, and B3 were evaluated with a proximity loss weighted by $\gamma \in \{0.1, 1.0, 10.0\}$. In the final configuration, C, based on B2, we replaced the mean square error classification loss with cross-entropy loss. For the CelebA dataset, we revisited settings A and B2 from the MNIST experiments. Setting A yielded the highest accuracy and lowest MSE (consistency), but the generated counterfactuals lacked coherence and showed no resemblance to the input data, as indicated by the high FID score. The FID played a crucial role in selecting setting B2, which includes the proximity loss.

AttFind [30]. As AttFind was not part of the main paper’s comparative study, we applied it directly to our GdVAE’s latent space without optimization. Refer to Tab. 15 for the results.

EBPE [43]. To evaluate EBPE, we experimented with different hyperparameters and activation functions in the decoder’s output layer. The initial implementation, designed exclusively for CelebA, used the parameters detailed in Tab. 16 ("Original") as our starting configuration. However, this configuration yielded suboptimal performance on both MNIST and CelebA. For MNIST, setting A from Tab. 16 was chosen for its balance between consistency and realism.

EBPE encountered challenges in generating good reconstructions for the CelebA dataset with our backbone network and the original settings (see Tab. 16). To address this issue, we explored different activation functions to prevent image saturation, particularly as our GdVAE architecture employed a ReLU function in the output layer. In setting A, we employed a Sigmoid function, while method

Table 15: Evaluation of CF explanations for the simple consistency task. Methods marked with a † exclusively induce class changes, rendering accuracy incomparable to other methods. These † methods closely adhere to the original implementation. We use ACC to assess the classification consistency for generated CFs. The Fréchet Inception Distance (FID) is employed to gauge CF realism. Proximity is measured with MSE (scaled by 10^2). Mean values, with standard deviation, are reported across four training runs with different seeds. It is evident that all methods marked with † achieve 100% accuracy in altering the class of the query image. The methods without a † represent the modified versions discussed in the main paper. These methods enable users to pre-define a confidence value when generating CFs.

| MNIST - Binary 0/1 | | | |
|---------------------------|---------------------------------------|------------------------|------------|
| Method | Consistency Realism (FID) ↓ Proximity | | |
| | ACC% ↑ | $\hat{p}_c \in [0, 1]$ | MSE ↓ |
| AttFind [†] [30] | 100.0±0.0 | 120.14±21.88 | 11.92±6.69 |
| C3LT [†] [26] | 100.0±0.0 | 88.67±11.01 | 14.92±0.85 |
| C3LT [26] | 3.6±0.8 | 57.09±10.78 | 5.83±1.47 |
| CelebA - Smiling | | | |
| Method | Consistency Realism (FID) ↓ Proximity | | |
| | ACC% ↑ | $\hat{p}_c \in [0, 1]$ | MSE ↓ |
| AttFind [†] [30] | 100.0±0.0 | 109.18±14.82 | 2.32±1.37 |
| C3LT [†] [26] | 100.0±0.0 | 171.72±7.97 | 8.59±0.84 |
| C3LT [26] | 11.8±5.5 | 101.46±11.56 | 3.97±0.86 |

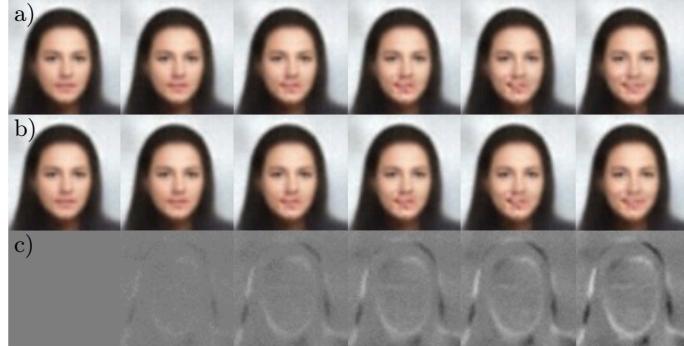
B1 utilized a hyperbolic tangent (\tanh) function—the latter being used in the original GAN model by the authors of EBPE. Hence, all subsequent settings adopted the \tanh function in the decoder architecture. Emphasis was placed on optimizing the reconstruction quality and FID scores by varying the weight of the conditional GAN loss (discriminator loss). Specifically, settings B1, B2, and B3 utilized weights of $\lambda_{cGAN} \in \{0.01, 0.1, 0.001\}$ for the discriminator loss.

Building on these findings, we enhanced the FID of version B1 by employing a higher weight of $\lambda_{rec} = \lambda_{cyc} = 10^3$ for the reconstruction loss in setting C, deviating from the previously used weight of 10^2 . To validate our implementation, we used EBPE to explain a separate discriminative classifier. The objective was to assess whether further improvements could be achieved by extending the training time beyond the 24 epochs that was used by all other methods, as the original implementation employed 300 epochs. Consequently, methods D1, D2, and D3 were executed with 24, 48, and 96 epochs, respectively. To expedite training, we employed a single discriminative classifier instead of the GdVAE, and as a result, standard deviations are not reported. It's evident that EBPE requires more training time compared to the GdVAE, and with four times the training duration, it converges toward similar FID values as our global method. The results of this hyperparameter analysis are shown in Tab. 12.

C3LT [26]. We examined two loss function settings on MNIST and CelebA. Setting A employs the mean squared error, as utilized by GANalyze (Eq. (27)), while setting B is founded on the cross-entropy loss for \mathcal{L}_{cf} . The results are summarized in Tab. 12.

Table 16: Hyperparameters for EBPE.

| MNIST | | | | | | |
|----------|------------------|-----------------|-----------------|-------------------|--------------------|------------|
| Setting | λ_{cGAN} | λ_{rec} | λ_{cyc} | λ_{cfCls} | λ_{recCls} | Activation |
| A | 0.01 | 1 | 10 | 10 | 0.1 | ReLU |
| CelebA | | | | | | |
| Setting | λ_{cGAN} | λ_{rec} | λ_{cyc} | λ_{cfCls} | λ_{recCls} | Activation |
| Original | 1 | 10^2 | 10^2 | 1 | 1 | tanh |
| A | 0.01 | 10^2 | 10^2 | 1 | 1 | Sigmoid |
| B1 | 0.01 | 10^2 | 10^2 | 1 | 1 | tanh |
| B2 | 0.1 | 10^2 | 10^2 | 1 | 1 | tanh |
| B3 | 0.001 | 10^2 | 10^2 | 1 | 1 | tanh |
| C | 0.01 | 10^3 | 10^3 | 1 | 1 | tanh |

**Fig. 11:** a) Local-L2 CFs, b) Local-M CFs, and c) the difference, with white and black indicating a deviation of approximately ±3%.

E.3 Qualitative Results

Additional results: MNIST in Fig. 12, CelebA in Fig. 13, FFHQ in Fig. 14, and multi-class CFs for MNIST and CIFAR in Figs. 15 and 16. All local explanations were generated using the L2-based method. A comparison of local-L2 and local-M explanations is provided in Fig. 11.

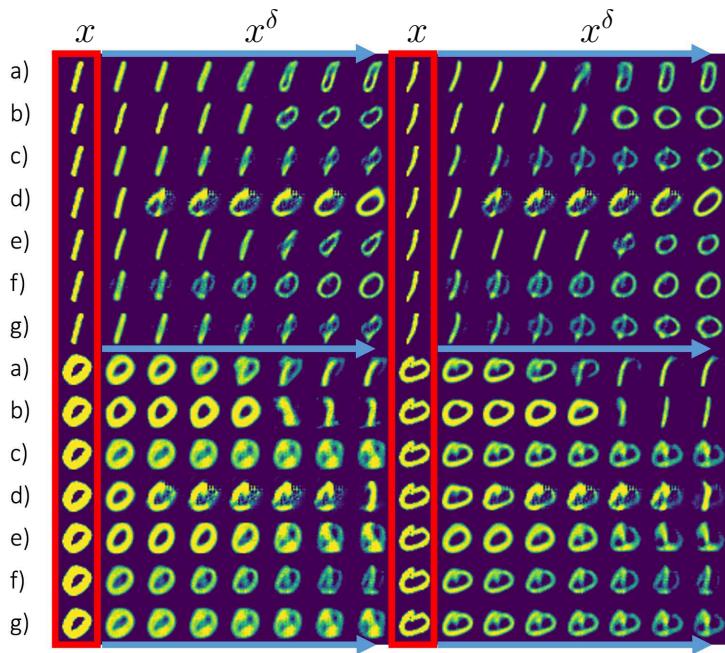


Fig. 12: MNIST CFs. We generate CFs (x^δ) linearly for the input, with decreasing confidence for the true class from left to right. On the leftmost side of each section, x denotes the input. We generate samples for $p_c = [0.99, 0.95, 0.75, 0.5, 0.25, 0.05, 0.01]$. a) GANalyze, b) UDID, c) ECINN, d) EBPE, e) C3LT, f) Ours (global), g) Ours (local).



Fig. 13: CelebA CFs

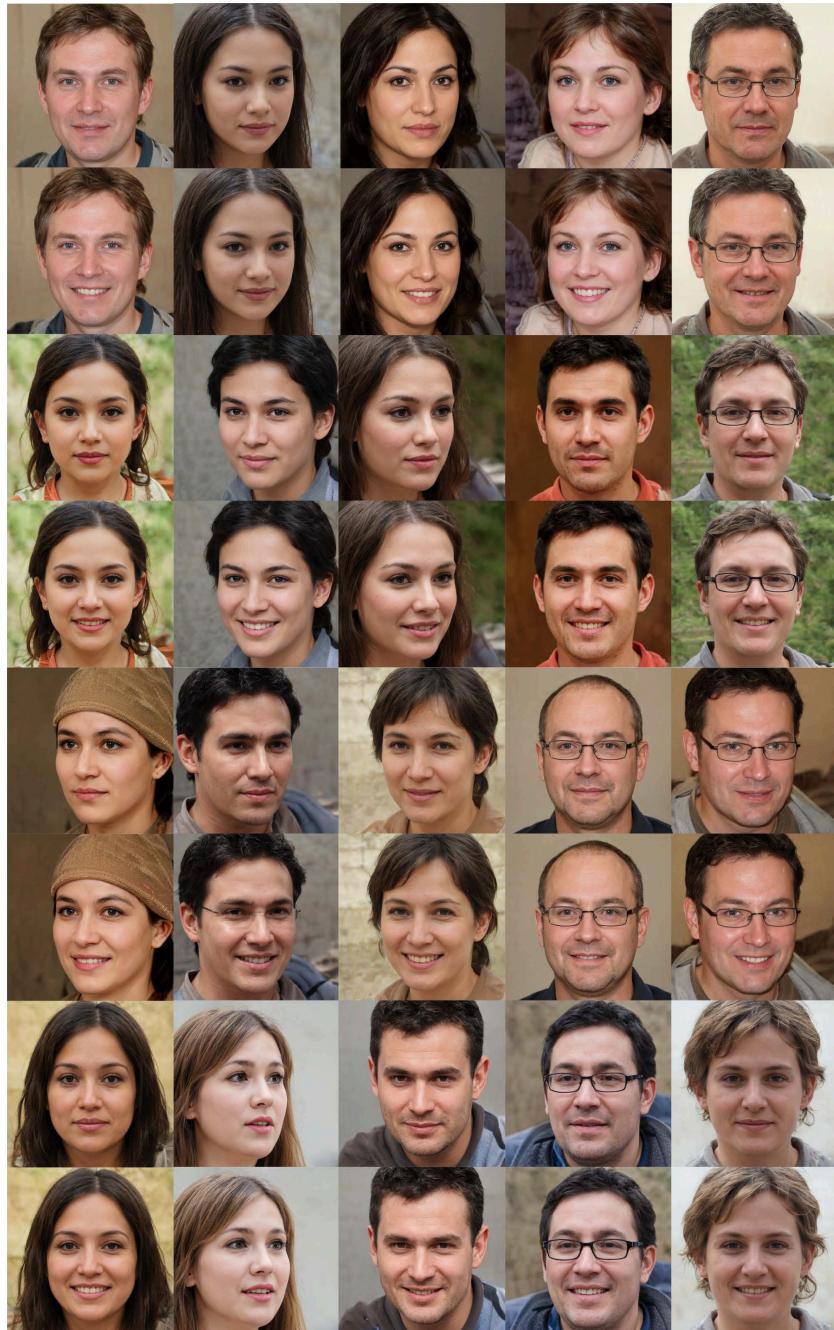


Fig. 14: FFHQ CFs

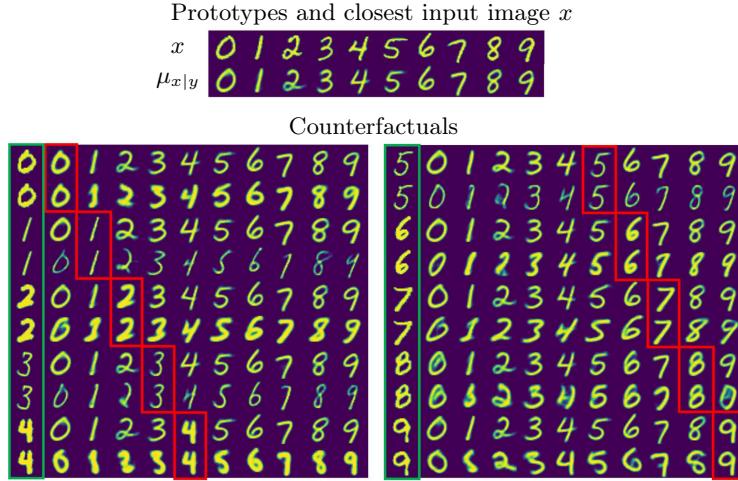


Fig. 15: MNIST multi-class CFs. CFs for the simpler consistency task by swapping the logits of the predicted and counterfactual classes. The green rectangle indicates the input image, while the red rectangles highlight the image reconstructions. Global (top) and local (bottom) CFs are each positioned to the left and right of the reconstructions, respectively. The local CFs maintain key image characteristics, such as line thickness. This CF strategy changes class predictions 100% of the time for the global method, and 99% of the time for the local method.

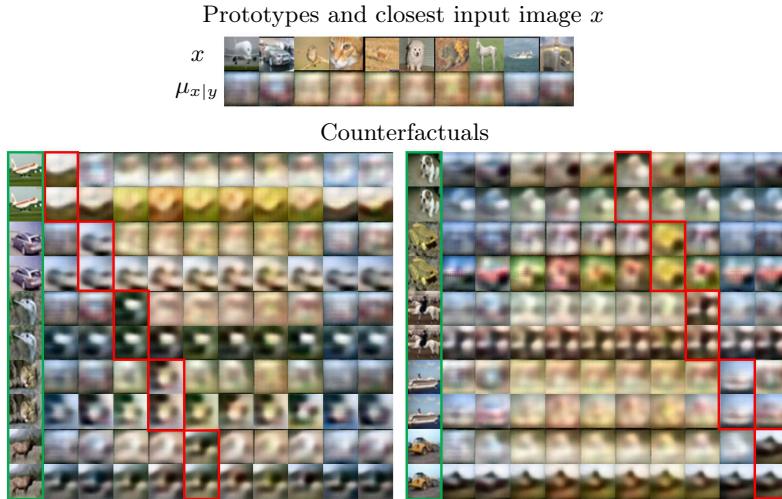


Fig. 16: CIFAR-10 multi-class CFs. CFs by swapping the logits of the predicted and counterfactual classes. The green rectangle indicates the input image, while the red rectangles highlight the image reconstructions. Global (top) and local (bottom) CFs are each positioned to the left and right of the reconstructions, respectively. The CFs are generated primarily through color adjustments and minor shape adaptations. This CF strategy changes class predictions 97% of the time for the global method, and 89% of the time for the local method.

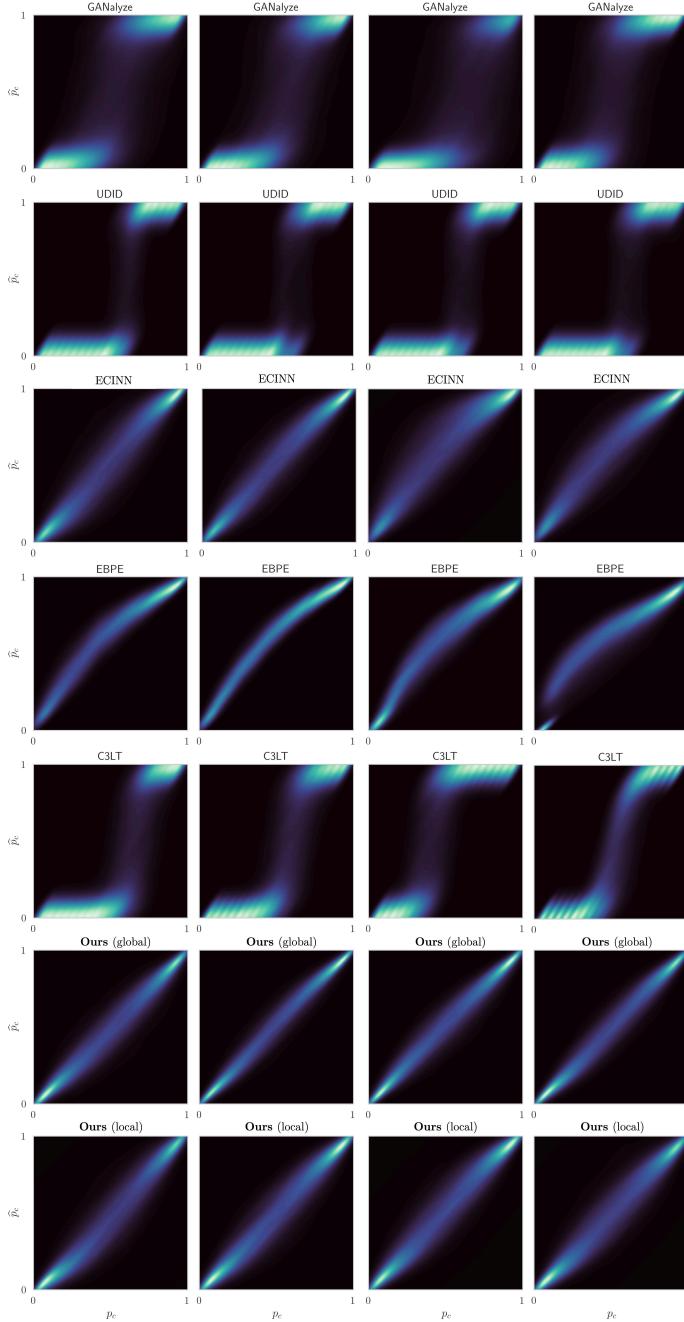


Fig. 17: MNIST KDE plots are employed to visually depict the relationship between the requested confidence value p_c and the predicted confidence by the classifier $\hat{p}_c = p_\theta(y = c|h(g(z^\delta)))$ for the counterfactual $z^\delta = \mathcal{I}_f(z, \delta)$.