Contrastive Explanation using Supervised Generative Models

Poels, Yoeri

Award date:
2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Data Mining Research Group

Contrastive Explanation using Supervised Generative Models

Master Thesis

Yoeri Poels

Supervisor:
dr. Vlado Menkovski

Feb 2021

Abstract

In this thesis, we consider explaining the classification of high-dimensional data by using Machine Learning (ML). The classification of data entails assigning datapoints to groups, according to some common criteria. We aim to explain data by explaining this process. As studying so-called worked examples is an effective learning-method, we can use these explanations to teach people how to distinguish between data classes. Additionally, explaining the classification of data provides one with an understanding of how the explanation-model perceives it. This understanding provides trust when using this model to classify data.

We follow the human tendency of explaining events in a contrasting manner: When asked why something happens, humans tend to (implicitly) compare the event to some other event that did not occur. As such, we seek to design a method generating explanations of why a datapoint was assigned to a given class, opposed to some other class. These explanations are constructed according to the assumption that classes are defined by higher-level features that reside in datapoints.

To design an explanation-generation method, we first survey the field of explanation in ML. Next, to formalize the desired characteristics of an explanation-method concerning classification-processes, we explore literature concerning the human perception of explanation and categorization. The primary outcomes concern the notion of contrastive explanation and the need to explain classifications according to predefined underlying concepts.

Following these studies, we propose Variational Autoencoder-based Contrastive Explanation (VAE-CE), a framework considering both the classification of data and the explanation thereof. VAE-CE is based on the VAE framework[70], a generative model focused on modeling the distribution of its training data according to a higher-level feature space, the latent space. We extend the VAE to disentangle class-specific features from class-irrelevant features. Furthermore, we condition this model such that individual class-features are represented by individual latent dimensions. To achieve the latter, we propose pair-based dimension conditioning, a method that encourages individual latent dimensions in VAEs to shape themselves after features. This feature-shape is determined using pairs of images indicating feature-changes.

VAE-CE explains datapoints by providing their classification and a meaningful interpolation to an alternative class. These interpolations aim to convey class-distinguishing features by transforming the data from one class to another, changing a single feature at each step. To evaluate such explanations, we propose a method for quantifying their quality based on the alignment to a ground-truth explanation, denoted as the explanation alignment cost (*eac*). We validate VAE-CE quantitatively and qualitatively, both with respect to our predefined goals and in comparison to other VAE-based models.

Empirical evaluation of VAE-CE shows that it can generate contrastive explanations clearly conveying the class-distinguishing features. In an introductory study, VAE-CE outperforms comparable VAE-conditioning methods with respect to both the internal representation of features and the ability to generate explanations. These results suggest that VAE-CE has potential as both an explanation- and representation-method, although the question remains how well the approach scales to more complex problems.

Contents

Chapter 1

Introduction

The field of Machine Learning (ML) considers the study of providing learning capability to computers without explicit programming[5, 139]. Put simply, it considers designing methods that learn how to solve tasks from experience or data. ML-methods have displayed state-of-the-art performance in a wide range of domains, ranging from image recognition[38] to predicting protein folds[60]. This strong performance indicates that ML can be applied to difficult problem-domains, and that it can work with high-dimensional or complex data. In this thesis, we consider the use of ML to explain such high-dimensional data. We aim to explain this data by explaining its classification process.

The classification of data considers grouping datapoints according to some common criteria. We seek to design a method that explains this process by generating explanations of classification decisions. This explanation is done for two reasons. To start, one can use explanations of a process to teach people about said process: By observing explanations of classifications, one can learn how to distinguish different groups or classes (the *worked example effect*[152]). Additionally, a model generating classification-explanations builds trust to the end-user[97, 65], if we were to use it to classify data.

The explanations we create are *contrastive*. We follow the human tendency of explaining an event by (implicitly) comparing it to some alternative event that did not take place[85]. As such, we create explanations that explain why a datapoint belongs to a given class in contrast to some other class. Furthermore, we assume that categorical data is constructed according to some (higher-level) generating process. In simpler terms, we assume that categories are defined by underlying concepts or features[132]. Consequently, our explanations should convey the distinguishing features between the assigned class and the alternative, counterfactual class.

Our approach builds upon (Artificial) Neural Networks (NNs)[80, 41], an ML-method (structurally) inspired by biological neural networks. More specifically, we extend the Variational Autoencoder (VAE)[70, 124]. The VAE is a NN-framework focused on learning the probability distribution of its training data by constructing a higher-level feature space, denoted as the latent space. We introduce Variational Autoencoder-based Contrastive Explanation (VAE-CE), a framework considering both the classification of data and the explanation thereof.

VAE-CE considers explaining a datapoint by predicting its class and conveying the higher-level features that must change in order to classify the datapoint as another class. This other class can be specified or automatically inferred (e.g., the second most likely class). The feature-difference is depicted by interpolating from the source datapoint to an *exemplar* of the alternative class, changing a single feature at each step. Only the class-relevant features of the exemplar are depicted in this interpolation, the remaining features are left as-is. To be able to create such an interpolation, our model must represent datapoints such that we transform their individual class-features.

To build the representation model (the VAE), we employ existing literature regarding the conditioning of a VAE's latent space. We seek to disentangle the class-relevant and class-irrelevant information, splitting the latent space into two subspaces. Additionally, we introduce a new method to regularizing a VAE's latent space, denoted as pair-based dimension conditioning. This

method focuses on representing changes of some presupposed shape (the features) in single dimensions, utilizing supervision in the shape of image-pairs exhibiting single feature-changes (or a lack thereof). As such, the definition of what construes a meaningful concept is outsourced to the person constructing the model, allowing them to characterize the concepts suitable for their target audience. Note that we do not assume 'full' supervision, that is, a labeling of all semantically meaningful features. Rather, we focus on supervision exemplifying the *shape* of these features.

Generating explanations then considers interpolating through the latent space. We describe a method for traversing the latent space of VAE-CE that optimizes the quality of the resulting explanation. This method aims to maximize the quality of all intermediate states and steps (each step should change a single feature) while minimizing the explanation length. Additionally, to be able to compare VAE-CE to other approaches, we propose a method for the quantitative analysis of methods generating interpolation-based explanations. This method considers aligning candidate-explanations to a ground-truth explanations, and is denoted as the explanation alignment cost (*eac*).

In order to validate our approach, we conduct an introductory study. We discuss the quality of the generated explanations, and compare both the explanation- and representation-quality of our method to those of similar methods. Furthermore, we validate the use of VAE-CE's subcomponents.

1.1 Research Questions

To address the objective of this thesis, we define a main research question we seek to answer:

How do we use machine learning to explain the classification of complex data?

We focus on explaining data by explaining their classification process. As we must model the data's classification-process in order to explain it, one can also consider this objective as designing a method that provides explanations of its representation of classes.

To answer the main research question, we decompose it into five subquestions (SQs). These questions consider a literature study (SQ1 and SQ2), the design of our approach (SQ3), and the evaluation of this approach (SQ4 and SQ5). These subquestions, the approaches to solving them, and the respective (sub)chapters considering them, are as follows:

SQ1 *What machine learning-based approaches exist when considering the explanation of complex data?*
We survey the ML-literature and provide a summarized overview of current methods considering the explanation of complex data, which considers methods focused on model explanation and methods focused on data explanation or representation. This overview provides us with a starting point for designing our method. (Chapter 2)

SQ2 *How do we define explanation in the context of our goal?*
We explore explanation- and categorization-related literature in order to formulate and ground the desired characteristics of the explanations we seek to generate. This literature spans across various domains, allowing us to consider (categorical) explanation from the perspectives of computer science, cognitive science, psychology, and philosophy. (Chapter 3)

SQ3 *How do we generate explanations using machine learning?*

SQ3.1 *How do we create and condition a model to represent data in a suitable manner for explanation generation?*
We select a suitable base-model and argue its applicability. We describe an approach for conditioning this model as desired, building upon existing work, and additionally employing a novel approach for conditioning models to represent data in the desired manner. (Chapter 4, Section 5.2)

SQ3.2 *How do we generate explanations using this model?*
We propose a method for using this model in order to generate contrastive explanations of the desired structure. (Section 5.3)

SQ4 *How can we quantitatively evaluate the quality of explanations?*
We propose a method for the quantitative evaluation of interpolation-based explanation methods, which is based on the use of synthetic data with a known generative process. (Section 5.4)

SQ5 *How well does our method represent and explain data?*

SQ5.1 *What is the quality of the generated explanations in terms of conveying class-distinguishing features and selecting suitable exemplars?*
We assess the quality of our explanation method by inferring explanations and exemplars, and discussing how they match up to our notion of good explanation. (Sections 7.1.1 and 7.2.1)

SQ5.2 *How does the explanation-quality of our method compare to that of similar approaches?*
We consider a set of different VAE-disentanglement methods and compare the explanations generated by all methods, both quantitatively and qualitatively. (Sections 7.1.1 and 7.2.1)

SQ5.3 *How do our method's subcomponents contribute to the explanation-generation process?*
We evaluate the value of the overarching components of our method by generating explanations with them removed, and discussing the change in explanation quality. (Sections 7.1.1 and 7.2.1)

SQ5.4 *How does the representation-quality of our method compare to that of similar approaches, and how does this quality relate to the explanation-quality?*
We evaluate the internal representations of all compared methods both quantitatively and qualitatively, and discuss the relation of these representations to the explanation-quality. (Sections 7.1.2 and 7.2.2)

1.2 Contributions

The main contributions of this thesis can be summarized as follows:

- We provide a summarized overview of literature considering explanation in machine learning, with a focus on the explanation of high-dimensional data. (Chapter 2)
- We propose a method for encouraging a specified structure in individual latent dimensions of VAEs. This structure is guided by pairs of images indicating changes that should (not) correspond to a change in a single dimension. (Section 5.2.2)
- We propose a method for generating contrastive explanations of a datapoints' class assignment, built upon a foundation of explanation- and categorization literature. This method considers conditioning a VAE to class-features in individual dimensions in a subpart of the latent space, and uses this space to generate interpolations depicting class-relevant features. (Sections 5.2 and 5.3)

- We propose a method for quantifying the quality of contrastive explanation methods. The explanation structure we assume is an interpolation from one datapoint to another, where each pair of consecutive images should depict a single meaningful change. This evaluation method is based on computing the minimal alignment cost to a ground-truth explanation. (Section 5.4)

1.3 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 provides an overview of related work in the domain of explanation in machine learning. Chapter 3 provides a foundation of explanation in the context of our approach. Chapter 4 describes the existing ML-work we build upon. Chapter 5 details our proposed method, and Chapters 6 and 7 provide empirical analysis verifying the validity of our method. Finally, Chapter 8 states conclusions and denotes potential future work.

Chapter 2

Related Work

In this chapter, we provide an overview of related work in the domain of explanation in machine learning, alongside a discussion of how our work fits in this domain. Explanation in ML can be decomposed into a wide-ranging set of topics, each coming with its own set of assumptions and accompanying methods. As the entire domain is incredibly broad, we consider covering it in its entirety (in detail) to be outside of the scope of this work. Therefore, we restrict our focus to short summarizations of prominent or closely related work only; for more extensive and more detailed overviews of explanation-related machine learning literature, we refer to literature surveys such as [46, 40, 3, 14, 130, 158].

First off, we consider related work in the context of high dimensional data (e.g., images), as explaining this type of data is the main focus of our work. This constraint makes Neural Network[80, 41] based approaches most interesting, as they are both powerful when it comes to handling such data[82, 9] and are prominent in literature[25]. Nonetheless, we will touch upon other potentially relevant approaches as well. To structure this chapter, we propose a (simplified) taxonomy of the different subcategories of explanation in machine learning. Note that there is quite often overlap between these subcategories, and that these subcategories are also not necessarily focused only on explanation but rather provide an avenue for creating explanations. As such, consider this taxonomy not as a formal description of the structure of said literature, but rather as a means to allow us to discuss said literature in this chapter.

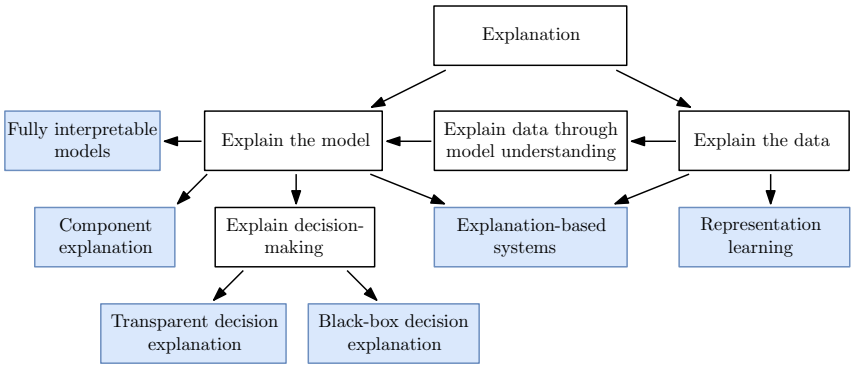


Figure 2.1: Simplified taxonomy of explanation in machine learning.

The taxonomy is depicted in Figure 2.1. We first divide the type of explanation by its purpose: Explaining a machine learning model or explaining data using machine learning methods. We then divide both of these categories into a set of subdomains, where each leaf-node (blue) depicts a topic we consider in this chapter.

A distinction not made in this taxonomy is the notion of scope: Distinguishing between local and global explanations, where local explanations consider explaining individual decisions or datapoints, and global explanations consider explaining entire models or datasets. In the per-topic overviews we will consider (where applicable) both local and global explanation methods. The type of explanation (e.g., contrastive) is also described on a per-method basis, as methods often

differ in how precisely explanation is approached. Finally, it is worth noting that a significant part of the literature we consider does not directly align with our goal, explaining the classification of data (focused on the data itself, rather than a classification model). However, under the assumption that a model can accurately capture the characteristics of data, it is worth considering model-explanation literature as well: Explanation of data can be done by interpreting a model processing the data we seek to explain (e.g., understanding a classification model could help us understand the class-related characteristics of data).

Fully interpretable models: One approach to model explanation rests on using models that are fully interpretable. An issue that arises is defining what 'fully interpretable' means. For now, we propose a rather simple definition: A model is considered fully interpretable when the size of its internal state (i.e., the number of parameters) is not severely (orders of magnitude) larger than the input size, and the transformations made to reach decisions (with respect to the input) are simple (e.g., linear combinations of features). Some prominent[168] examples of methods fitting this description are decision trees[137], (fuzzy) rule-based systems[92], logistic regression classifiers[71], and k-nearest neighbor classifiers[24].

We recognize two drawbacks when applying these methods to problems concerning high-dimensional data. For one, while the methods are theoretically interpretable, we can likely not get a meaningful understanding of them because of their lack of abstraction with respect to the input. Most of these methods provide an almost direct mapping from input features to an output classification, but as the input domain grows (e.g., images with many pixels), individual pixels become less significant. To make sense of large inputs consisting of individually insignificant components, a level of abstraction is desirable[146, 79]. Furthermore, the lack of abstraction could result in very large models, which does not aid their interpretability. Additionally, these fully interpretable methods are generally less powerful than more complex, uninterpretable methods (such as NNs)[82, 9].

An approach to overcoming these issues lies in combining the aforementioned methods with handcrafted features extracted from input data. This approach keeps the property of interpretability and adds a level of abstraction that can make the method both more understandable and more accurate (e.g., [35, 163]). A drawback of this approach is its reliance on the problem domain (as it might not be possible to create such features) and that it requires expert knowledge (and effort) to engineer such features. Considering these drawbacks, we primarily focus on NN-based approaches when considering complex data. It is worth noting that there are other approaches that outperform these fully-interpretable methods (such as support vector machines[23]); however, as they are not necessarily more interpretable and usually less powerful than NNs, we do not examine general-purpose interpretability work surrounding these methods.

Component explanation: Empirical analysis shows that Neural Networks often embed understandable features in their components, and that the interaction of such features can be studied and understood[111]. This notion makes the explanation of NN-components a promising avenue for understanding models and the data they model. The explanation of NN-components primarily comes down to understanding a Neural Network on three levels: On that of individual layers, on that of individual channels (a feature or filter in a convolutional layer), and on that of individual neurons. To discuss work on gathering understanding about NN-components, we adhere to four types of visualization method, as outlined by [119]. These methods are as follows: *Activation Maximization*, *Deconvolutional Neural Networks*, *Network Inversion*, and *Network Dissection*.

Activation Maximization entails visualizing how components work by synthesizing inputs that maximize their respective activations. One approach to creating such inputs concerns directly optimizing images to maximize the components' activations. This optimization process must be sufficiently regularized in order to provide useful (non-noise) images; an overview of such regularization methods is provided by [112]. Alternatively, one can employ a generator network[42] to synthesize images optimized for a components' activation in a controlled manner[105, 106].

Deconvolutional Neural Networks attempt to explain components by visualizing what part of an input image induces their activation. This process works by employing an additional deconvolution network that maps the embedded features back to input space[172]. This approach is similar to decision-explanation methods such as Grad-CAM[141], which also focuses on projecting saliency on input images; however, such methods primarily focus on class activations (final layer neurons) rather than intermediate neurons.

Network Inversion considers recovering an image from an activation map (at some level in the network) in order to visualize what kind of information is retained at the activation map's associated layer. Similar to activation maximization, the two main methods for recovering/generating the image are based on direct image optimization (with regularization)[93] and by using a generator network[34].

Network Dissection considers using semantic concepts to try and explain neurons: The methods involve finding correlations between neurons and predefined concepts by measuring neurons' responses to images containing the aforementioned concepts. The identified correlations are then illustrated using a mapping, for which various types of mapping have been proposed: One can map neurons to concepts directly[10], map neurons to some composition of concepts[104], or map neurons to some concept-based feature vector[37].

Transparent decision explanation: It is worth recognizing the parallels between transparent decision explanation and component explanation, as decision explanation is virtually equivalent to explaining a specific set of neurons that can be found at the end of the network. As such, methods mentioned in the previous paragraph can also be applied to try and explain decisions: To get an idea of what a model considers a class to be, one could synthesize inputs maximizing the activation for the associated output-neuron. However, the notion of decision explanation allows for extra bias in methods attempting to do so, as it is now focused on the network as a whole. As such, we consider methods more tailored towards this goal. The decision explanation of a Neural Network then comes in two flavors: Either one tries to explain a model's decision-making process in its entirety, *global explanation*, or one tries to explain individual decisions made by a model, *local explanation*.

Global explanation methods concern transforming a Neural Network into a more interpretable model, aided by access to the NN's internal state. One approach to do so concerns extracting (fuzzy) rule systems by parsing the weights/biases of a network's neurons[6]. Another approach is to extract decision trees mimicking the inner workings of a NN[177]. A significant drawback of the aforementioned methods is the fact that they are trying to stay faithful to the entire network: With non-trivial NNs, this leads to enormous proxy models that are not usable in practice. To get around this limitation, methods utilizing samples in order to prune irrelevant states have been proposed, such as [110] for greedy rule generation and [155] for a hierarchical partitioning of the input space. The question of whether such methods scale to high-dimensional data remains, as they were created with simpler data in mind. Consequently, *local explanation* is a more promising (and popular) avenue to explore when it comes to models working with high-dimensional data.

For *local explanation*, the focus lies on the explanation of why a NN classified a certain datapoint as it did. One approach for doing so is the evaluation of the influence that input features have on the classification decision. To do so, one can either send a signal forward through a network and—with respect to the classification decision—project it back to input space to evaluate saliency, done in methods such as Grad-CAM[141] or DeepLIFT[147]. Alternatively, one can approach it from the perspective of neuron-contribution, evaluating input neurons with respect to an output (class) neuron, and project saliency as such, done in methods such as Layer-wise Relevance Propagation[8] and Integrated Gradients[151]. In contrast to highlighting all relevant features, [115] propose a method for highlighting features that are most relevant with respect to

another category, i.e., producing a contrastive explanation. Similarly, [118] extends Grad-CAM to highlight the saliency of an image with respect to its classification decision contrasted against another chosen category. A different approach to evaluating decisions rests on finding correlations of predefined concepts with decisions, accomplished using Concept Activation Vectors[66]. This approach provides us with an idea of what concepts a model considers to be associated with a class.

Black-box decision explanation: In contrast to previously mentioned works, one can also explain a model's decision-making process without assuming access to said model's internal structure and state. Such methods are based on understanding how the model works with the given data, i.e., how changes in data change model behavior. The advantage of such an approach is its generalizability between models. Additionally, decision-processes irrelevant with respect to realistic data are ignored. On the other hand, no notion of the abstraction a model has learned can be exploited: The decision-making process must entirely be recovered by the explanation method. This model explanation is again done on both the *global* and *local* level.

Global explanations are created by generating some simpler, interpretable proxy model sharing much of the original model's behavior. Examples of methods for creating such proxy models are [58], generating (fuzzy) rule systems, and [73], extracting decision trees. It remains a question whether such models are applicable to high-dimensional data, as the respective model classes are not particularly suited for it. Furthermore, if one aims to explain data, there is little utility in creating a proxy model for a black-box model processing the data: One could directly model the data itself.

When generating *local explanations*, one is concerned with explaining the decision boundaries of individual datapoints. A common approach to doing so involves perturbing input samples in order to assess what input features impact the classification decision. A method employing this strategy is LIME[125], which explains images by generating a mapping indicating whether pixels contributed to a decision or not. This mapping is created by weighting the classifications of the perturbed samples according to the perceived distance of said samples. Other examples of local explanation methods are Anchors[126], providing rule-based explanations of a decision, and LORE[45], generating a decision tree-based explanation. Local Foil Trees[130, 162] denotes a method that also creates decision tree-based explanations, but rather than creating explanations focused on one specific classification, it creates contrastive explanations. Another contrastive explanation-based method is [28], which considers finding some datapoint's minimum features necessary to either yield the same decision or to change the decision, through optimization of the input. One can also identify these features by using a generative model and traversing its latent space to find a datapoint flipping the classification decision[36].

A drawback of these model-agnostic methods is that how the input space is explored strongly influences the created explanation, as we cannot inspect the original model's internal state to clarify a decision (and obtain the actual reason). Furthermore, model-agnostic explanations likely reside in input space, which is often not satisfactory in terms of understandability: An abstraction to more meaningful concepts is likely desirable.

Explanation-based systems: One can also choose to make explanation an explicit focus of the method. For methods doing so, the line of model explanation versus data explanation is somewhat blurry, as often the model's decision process is explained by describing how the model perceives the data. These explanation-based systems primarily fall into two categories: Models that provide an explicit explanation of a decision or of some datapoint, and models that are structured such that some (understandable) intermediate representation is used to make decisions. We refer to these categories as *explanation-producing systems* and *explainable systems*, respectively.

Explanation-producing systems are focused on generating an explanation of either the data or of some decision/classification the model made. A prominent example of generating an explanation of data is the notion of (neural) image captioning, introduced by [160]: Through images and their captions, a model is taught how to describe (or explain) the contents of an image. This method was extended by [170], improving performance and giving extra insights into the captioning by using an attention mechanism. The image captioning approach was specialized towards discriminative descriptions by [48], providing a textual description of a classification, along with the (discriminative) features that led to this classification. This approach was extended by [49] to ground the explanation: words in the explanation are linked to pieces of the image. Additionally, they extend the method such that it can also generative contrastive explanations, which describe why the image was not classified as some other specified class.

Explainable systems do not necessarily provide an explanation but are structured such that the ability to explain (part of) its decision-making is embedded in their architecture. Systems explicitly evaluating the importance of input features, denoted as Attention-based models[21], fall under this category (although research suggests that this attention only noisily predicts the importance of input components[144]). An example of the use of an attention mechanism in the visual domain is [169], in which the model explicitly uses or discards certain pieces of an image. Recent work surrounding the use of Transformers in the image domain[32] also shows the added benefit of its self-attention mechanism with respect to model understanding, allowing for an inspection as to what parts of an image were used for a classification.

Alternatively, one can explicitly design a model to be interpretable. One can build an interpretable intermediate state by pre-training on a large set of concepts, which can then be used to make understandable classification decisions[175]. A similar method was proposed by [4], which focuses on learning concepts intrinsically rather than using an external concept-corpus. Another approach considers making decisions by linking images to other samples[17]: A model dissects images into parts, which are matched to other images (of some class) in order to assign a class in an understandable manner.

Representation learning: We can also approach data explanation from the perspective of representation learning. This field concerns methods for learning (useful) representations of data: In our case, we seek to create a method for learning understandable representations of data in order to explain or describe said data. To discuss these methods, we divide them into two groups: *Problem-specific* and *problem-agnostic* methods.

Problem-specific approaches concern methods for learning representations where one can assume a very specific type of data and generative process. An example of such a method is [148], which considers an approach to extracting hierarchical image templates from images. The method rests on the assumption that images are decomposable into many patches and that said patches can be described by four categories (sketch, texture, flatness, and color). Furthermore, it is assumed that such patches are often shared between images. Images can then be represented as combinations of such patches, which, in turn, can be used for downstream tasks such as (few-shot) image classification. Furthermore, it provides us with an understandable representation we can use to explain datapoints. Another example is [77], which considers a method for the representation of character images as probabilistic programs, described as the Bayesian program learning (BPL) framework. The method is based on strong hierarchical and structural assumptions about character images: Images of characters consist of a combination of line strokes. By exploiting these assumptions, images can be transformed into probabilistic programs, which in turn can be used for downstream tasks. Moreover, it provides us with more interpretable representations of the data. BPL exceeds at tasks such as one-shot learning[77, 78], as the method's strong inductive bias allows for useful representations to be learned with very few datapoints. Both aforementioned methods are unsupervised; they rely on their strong inductive biases rather than external information.

Problem-agnostic approaches concern methods with more general assumptions about a data's structure and generative process. One such method is Principal Component Analysis (PCA)[166], which rests on the assumption that the data's features are linearly uncorrelated: The method's result is a set of vectors, referred to as principal components, which transform the original data. These principal components are linearly uncorrelated and (in order of component) explain the greatest variance in the data. A related domain is that of Independent Component Analysis (ICA)[61], which concerns extracting independent factors from a linear mix of factors. Autoencoders[72] are a type of NN that bear great similarities to PCA. This class of models seeks to compress data to a set of latent variables in an unsupervised manner: A NN is trained to compress data to a specified number of latent dimensions, which are then used to try and recreate the original datapoint. While not explicitly optimizing dimensions for lack of correlation or for great variance, the target of reconstruction causes models to learn latent dimensions closely corresponding to such factors[131]: In practice, a linear autoencoder is similar to PCA. The potential complexity of NNs allows for the learning of more complex (nonlinear) representations.

The Variational Autoencoder (VAE)[70, 124] extends upon the autoencoder framework by encouraging the latent dimensions to match some prior distribution, often a standard factorized normal distribution. The VAE is a probabilistic generative model, as it captures the distribution of the dataset it models, and one can sample from it according to the knowledge of the assumed prior distribution. The Adversarial Autoencoder (AAE)[94] shares the prior-based latent space approach with the VAE but accomplishes it using a different method (adversarial matching to the prior distribution, rather than a KL divergence term). To further control the properties of the latent representations in (variational) autoencoders, many methods have been proposed. One objective for shaping the latent space is the interpretability of interpolations (when interpolating between two datapoints according to their latent representations). To achieve realistic interpolations, a critic network that evaluates interpolations (and tries to recover their mixing coefficient) can be applied[13]. Another approach concerns using metric learning on class labels to structure a latent space that separates data by class, making latent traversal more class-relevant[30] (additionally, they use this latent space to produce contrasting explanations). To encourage interpolations to traverse high-density regions in the latent space, [64] propose using a gamma distribution as the prior distribution.

Another objective concerns the disentanglement of the latent space, which can be defined as the notion that single latent dimensions are sensitive to changes in single generative factors, while being invariant to changes in others[51, 12]. Such disentangled representations can be encouraged to form both with and without supervision, although the question has been raised whether we can expect good disentangled representations without strong model inductive biases or (implicit) supervision[86]. Unsupervised methods generally extend the regularization term with extra assumptions about the shape of the latent space. Examples are β-VAE[51] (increasing the KL-divergence penalty), TCVAE[18] (extending β-VAE to only emphasize penalty on a component of the KL divergence) and FactorVAE[67] (encouraging factorized dimensions using a discriminator). Supervised approaches come in many different types. One approach is the use of pairs or groups of data, where often the assumption is made that common information in such a group should be encoded to a specific part of the latent space, and the remaining information should be stored in another part. Examples of this are ML-VAE[15], GVAE[55] and CycleVAE[57]. Another approach concerns utilizing labeled features, which comes with several applications: the labeling can be used to enforce some shared part that is consistent throughout images sharing the label-value[39], one can use labeling to encourage information that can be used to predict said labeling to be encoded in certain parts of the latent space[174], and additionally, one can condition priors of pieces of the latent space in order to encourage label-related information to be stored in those pieces[56]. Another supervision-based approach considers enforcing a given structure in the latent space, for example, encouraging a monotonic relationship between images' label-values and latent dimensions[114], or encouraging pairs of input images to follow some supplied transformation in the latent space[157, 50].

2.1 Our Approach

This work positions itself somewhere in between explanation-based systems and representation-learning based approaches. As our method generates explanations it is necessarily explanation-producing. The avenue to creating such explanations concerns learning a meaningful representation of our data. To conclude, we note three concurrently developed works that bear close similarities with certain aspects of our method:

To disentangle individual dimensions of VAEs, [87] design a method using weak supervision, which (in their case) is the use of image pairs (or groups) containing some unlabeled common factors. We assume a similar type of supervision to create a useful representation of our data but focus on image pairs containing all but one common factor (i.e., a special case of their assumption). The approaches to disentangling the latent space using this supervision are fundamentally different. Their approach rests on heuristically finding the common factors and enforcing them to be shared through the averaging of their encodings, whereas ours rests on using a discriminator to encourage single factor changes to occur in each latent dimension.

The approach to shaping latent-dimensions according to an auxiliary model is also proposed by[113]. The main difference is that we optimize for our model to shape latent dimensions such that they are considered desirable according to the auxiliary model (a discriminator), whereas their approach aims at maximizing the causal influence of the latent dimensions on the auxiliary model (a classifier). Furthermore, their focus lies on shaping class-specific latent dimensions, whereas we attempt to model the underlying generative factors that determine a class.

To learn a disentangled latent space, [176] propose a similar approach, that is, synthesizing image pairs that differ in a single latent dimension, and optimizing some objective according to these pairs. The main difference is that their method is focused on unsupervised disentanglement, where they aim to maximize the mutual information between these synthesized image pairs and the varied latent dimension. Conversely, our method focuses on imposing some structure in the latent space according to supervision. We synthesize image pairs (according to a single varied latent dimension) to optimize for this desired structure, employing a (pre-trained) auxiliary discriminator that evaluates the change in the image pair as good or bad.

Chapter 3

Explanation Foundations

In this chapter, we lay the foundation for our work's purpose. Rather than using our own intuition of what constitutes 'good explanation', we ground what good explanation is in corresponding literature. We start by defining and structuring explanation in a general case. Thereafter, we zoom in on our objective, the explanation of some classification (or categorization) process. Finally, we provide a summarized overview of the entire process.

3.1 Defining Explanation

To ground the desiderata of explanation, we consider work on the human perspective of explanation, as the purpose of explanation concerns the human understandability of some process. An overview of this perspective is provided by [101], considering work in the domains cognitive science, philosophy, and psychology. This survey is done through the lens of explanation in Artificial Intelligence (AI), which extends to explanation in ML (as ML is considered to be a subset of AI). Consequently, we can build upon this work when defining explanation in the context of our method. In this subsection, we define what an explanation is, and we characterize its general structure.

Explanation on itself considers two things: The act of explaining, the *process*, and the concept of an explanation, the *product*[88]. One can further distinguish between two types of explanatory process[101]: The cognitive process, which concerns the determination or creation of an explanation, and the social process, which considers the transfer of this knowledge. Furthermore, on the level of explanation as both a process and a product, a commonly made distinction is that of scientific explanation versus ordinary or everyday explanation[135, 47]. Scientific explanation is more rigorous than its everyday counterpart, focused on precise hypothesis formulation and experimental or theoretical justification, although there is no consensus on a precise definition[138]. On the other hand, everyday explanation concerns the (informal) transfer of knowledge concerning some event or phenomenon[84].

In this work, we focus on explanation as a product (much like the majority of ML-explanation literature[130]) and constrain ourselves to the creation of everyday explanations. As such, the cognitive process of our explanation can be regarded as the model's inner workings when generating the explanation, whereas the social process considers how the information is conveyed, the shape of the explanation.

To be more precise, we define an everyday explanation as the information that characterizes the causal history of some phenomenon or event (following the work of [84]), i.e., the answer to the question *'why did event* $\boldsymbol{x}$ *occur?'*[1]. Furthermore, we subscribe to the notion of contrastive explanation[85]: The idea of explaining the causes of an event by considering some hypothetical event that did not occur. This real event is referred to as the *fact*, whereas the hypothetical non-occurring event is referred to as the *foil*. Many works argue that people always ask why-questions with respect to some (perhaps implicit) foil[85, 54]. Moreover, empirical research has reinforced

[1]Note that this question on its own is ambiguous, as 'why' can refer to both the cause (*'how come?'*) and the purpose (*'what for?'*)[27].

the idea that humans explain events according to their causes and that we select these causes by contrasting the fact against some foil[95, 121, 122, 130].

Following these definitions, we observe that creating good explanations comes down to identifying and presenting the relevant causes (as providing every possible cause is both unreasonable and counterproductive with respect to explanation quality[101, 53]). The notion of contrasting two events serves as the primary discriminator for this process, allowing us to select causes by evaluating whether they occur in only one of the two events we compare. Furthermore, we can select (combinations of) causes according to their coherence or simplicity, as these aspects are significant criteria for explanation quality[120, 154]. Additionally, [101] remarks that there is a large set of (cognitive) biases humans are subject to when creating and evaluating explanations. As the applicability of these biases depends on the event being explained, we consider covering all of these biases to be outside of the scope of our work: We refer to [101] for a complete overview. Finally, we note that explanation is a social process[52]: We must recognize the assumed prior knowledge we base explanations off, how these assumptions align with our target audience, and formulate our causes according to these aspects.

To conclude, note that we must tread the notion of 'cause' with care. Through the eyes of an ML-model, the cause for a decision is (the state of) some set of input features. However, as we do not know the generative process that produced this input datapoint, we cannot guarantee that these features hold any causal relation to the true decision. In other words, the model could be making (and explaining) decisions by considering correlations rather than causes.

3.2 Explaining a Classification

The purpose of our method is to generate explanations of some classification process. As such, we can specify the definition and structure of the explanations we seek to generate in more detail by considering work on the evaluation of categorical knowledge. The underlying assumption is that shaping explanations according to this psychological categorization process will aid the understandability (and consequently the quality) of our generated explanations. To this end, we consider work on categorization, defined as the process of organizing knowledge by placing things into categories[132]. Note that we do not consider work on categorization as a biological or physical process, that is, the cognitive neuroscience perspective of the classification process as it happens in the brain (e.g., [128, 127]). Shaping explanations according to this physical process does not necessarily aid the human evaluation of explanation quality, and as such, we do not take it into account.

Categorization models generally consider three aspects: The representation of categorical knowledge, the process of matching stimuli (the things we categorize) to this knowledge, and the process of assigning a category according to this matching process[74]. As we aim to specify the desired shape of (categorical) explanations, we will primarily focus on the first aspect: We provide an overview of prominent approaches to representing categorical knowledge. Note that this is not a formal survey of categorization theories and models but rather a peek into this domain in order to formulate the desired characteristics of the explanations we create. For an exhaustive and detailed overview of this field, we refer to surveys such as [74, 159, 117].

We start by restricting our focus to supervised categorization methods only. In this scope, supervision entails the idea of categorizing according to predefined categories, whereas unsupervised categorization considers the idea of creating and defining categories[117]. Furthermore, we subscribe to the idea of describing objects and categories according to their underlying attributes[132]. We then discuss categorization from two main perspectives: According to attributes only and according to both attributes and (the similarity to) stimuli.

The first perspective rests on formulating categories according to their attributes only. We consider two main approaches resting on this notion: Rule-based categorization[107] and decision boundary-based categorization[7]. Rule models consider defining category membership according to necessary and sufficient features: The presence and absence of attributes (this approach is also referred to as the classical view of categorization[149]). Category membership can then be assigned to stimuli according to some (fuzzy) match to these rules. On the other hand, decision boundary models operate not by defining the content of categories but by defining the factors or attributes that distinguish categories. Categorization is done by evaluating at what side of these decision boundaries we can place the to-be-categorized stimuli.

The second perspective considers both attributes and stimuli: Rather than defining conditions or boundaries to categorize novel stimuli, we consider the (attribute-based) similarity to (often known) stimuli. This approach comes in two forms: Prototype-theory[99] and exemplar-theory[107]. In the former, prototype-based categorization, we consider similarity with respect to some prototypes of categories and assign category membership according to some aggregation of these similarities. These prototypes are generally a summary representation of a category, often coming in the form of either an average or a particularly distinct representation of a category. Note that prototypes are not necessarily stimuli we have observed before but rather resemble such stimuli. The latter, exemplar-based categorization, considers similarity to (virtually) all known stimuli rather than to prototypical examples. These known stimuli are referred to as exemplars. Category membership is assigned according to some aggregation of these similarities.

A remark about categorization as a whole is that we must be precise about the categories we consider, alongside their (assumed) context. As previously mentioned, we consider categories according to their attributes: These attributes can often be considered as categories on some other level in the categorization-taxonomy[132]. Although some 'basic' categories are generally deemed more representative[133] (e.g., one is more likely to categorize something as a bird than a sparrow when asked for no further clarification), a lack of clarity often introduces uncertainties in what constitutes a desirable explanation of some classification or categorization process.

Finally, it is important to note that the aforementioned theories often do not attempt to characterize a completely common process: The definition of 'concept' depends on the problem context[90], and as such, there is no one universally correct categorization method when seeking some description of a concept. A categorization-explanation problem could (albeit implicitly) seek an answer in the shape of one specific theory. Additionally, empirical evidence suggests that the correctness and utility of the different theories often does not differ significantly and is dependant on the categorization task at hand[31, 91, 134].

3.3 Our Approach

We provide a summarized description of the desired characteristics of our method's output. In other words, we consider what explanation is, how we seek to structure them, and consider further guidelines for explanation in the context of explaining a classification or categorization process. This summarized overview is as follows:

- We focus on explanation as a *product*, and seek to create *everyday* explanations.
- An everyday explanation considers the information that characterizes the causal history of some phenomenon or event.
- We consider explanations to be contrastive: We explain an event (the *fact*) according to some hypothetical event (the *foil*) that did not occur.
- Explanation is done by identifying and presenting the appropriate causes that highlight the difference between the fact and the foil.

- Two significant criteria we consider with respect to explanation quality are their coherence and simplicity[2].
- When explaining a classification process, we consider the fact to be the assignment of some label (category) to a datapoint (stimuli). We consider the foil to be the assignment of some other label to this same datapoint.[3]
- The causes we present are in the shape of features (attributes) of the datapoint.
- We consider two main approaches to present these features:
  - As features in isolation: Classifications are explained according to rules or decision boundaries.
  - As features and other examples: Classifications are explained according to similarities to prototypical examples, or according to similarities to arbitrary datapoints (exemplars).

[2]The simplicity of an explanation can be expressed quantitatively using metrics such as the number of bits of information necessary to describe an explanation, or the number of steps in the explanation.

[3]This definition is a 'restriction' we place on the foil event. A categorization foil could also consider events such as 'assigned any other class'; however, for the sake of practicality, we do not explicitly consider such cases in this work.

Chapter 4

Background

In this chapter, we define and motivate the prior work that VAE-CE builds upon. In Section 4.1, we describe the type of ML-model we use: The (Artificial) Neural Network. In Section 4.2, we provide a detailed description of the Variational Autoencoder. Finally, in Section 4.3, we detail the prior work we employ for conditioning and structuring the VAE.

4.1 Neural Networks

(Artificial) Neural Networks[80] are an ML-model (structurally) inspired by biological neural networks. The strength of NNs lies in their ability to map nonlinear systems: NNs can be viewed as powerful function approximators. A NN learns its parameters by (iteratively) minimizing its error with respect to some objective function, the loss function. In this subsection we provide a primer for NNs, giving an introductory overview of their structure and training procedure. As this work is focused on applying NNs for explanation rather than fundamentally about NNs as a model, we do not cover all aspects of NNs in detail, and refer to works such as [80, 41] for a more complete and detailed overview.

A Neural Network consists of a large set of separate units, referred to as (artificial) *neurons*[96]. These neurons transmit signals to other neurons in various structures, loosely mimicking the structure of real neurons and synapses in the brain. More formally, a NN is a directed graph of individual neurons, usually organized in a set of layers. The values of neurons, their *activations*, are calculated according to their incoming neurons. This calculation concerns two steps: Computing the input value (or state) of a neuron and applying an activation function to this value. The calculation of input value z for some neuron x and model parameters θ is as follows:

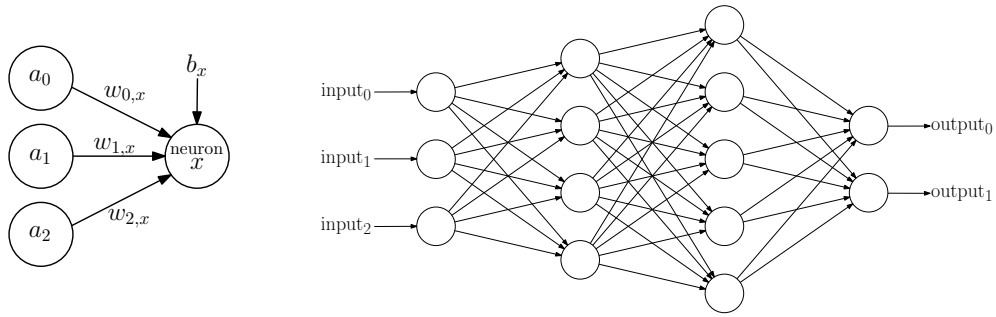
$$z(x;\theta) = \sum_i (w_{i,x} \cdot a_i) + b_x \tag{4.1}$$

where indices i denote the incoming neurons, w denotes their edge weights, a denotes their activation values, and b denotes the neuron's bias value. The resulting input value z is put through an activation function. Some of the most popular activation functions[108, 98] are as follows:

$$ReLU(z,\theta) = max(0,z) \quad \Big| \quad sigmoid(z,\theta) = \frac{1}{1+e^{-z}} \quad \Big| \quad Softmax(z,\theta) = \frac{e^z}{\sum_j e^{z_j}} \tag{4.2}$$

where for the Softmax function, indices j consider the neurons in the same layer. An illustration of a neuron and its determining factors is depicted in Figure 4.1a.

The power of NNs lies in combining these neurons, as individual neurons can only represent simple relationships. How neurons are structured and connected is denoted as a model's *architecture*. As previously mentioned, neurons are most often organized in layers and are connected according to layer connections: Neurons are generally only connected to other neurons residing in layers connected to their own layer. A model's architecture generally consists of an input layer, a number of hidden layers, and an output layer.



(a) A single neuron and its determining factors.

(b) A small (feedforward) Neural Network with two hidden layers.

Figure 4.1: A Neural Network on the micro- and macro-scale.

Two prominent types of layer, when considering problems with regards to high-dimensional data (e.g., images), are fully connected layers[103] and convolutional layers[81]. In fully connected layers, each neuron is connected to all the neurons in its connecting layers. An example of a small NN with two fully connected hidden layers is depicted in Figure 4.1b. Convolutional layers use learned, translationally invariant feature detectors. These feature detectors are 'translated' along the layer input in order to generate an activation map, a map indicating where a feature was (not) found. The functionality of convolutional layers rests on the assumption that in certain types of data, such as images, the same features can occur at many positions in some (input) layer. By learning translationally invariant filters, a network does not have to repeatedly learn the same weights when detecting a feature at multiple positions, resulting in a much more efficient network. These two types of layer form the main building blocks for the NNs we consider; for a more extensive overview of NN layers and architectures, we refer to [80, 41].

Training NNs comes down to finding suitable values for the neurons' weights and biases, the model's parameters. This goal is achieved by optimizing the parameters according to an objective function, the *loss function*. First, all parameters are initialized with random values. The actual training process then consists of propagating datapoints through the model and minimizing their error values. These errors are calculated by comparing the outputs to the desired outcomes using the loss function. To minimize this loss, all parameters are iteratively updated by subtracting (a fraction of) their gradients with respect to the computed error (i.e., the error is *backpropagated* to the neurons through automatic differentiation[165, 136]). This process is denoted as follows:

$$\begin{aligned} w &\leftarrow w - \alpha \frac{\partial}{\partial w} \mathcal{L} \\ b &\leftarrow b - \alpha \frac{\partial}{\partial b} \mathcal{L} \end{aligned} \tag{4.3}$$

where w and b denote the weight and bias of the to-be-updated neuron, α denotes the learning rate (the fraction of the gradient we use for the update), and $\mathcal{L}$ denotes the loss function. $\mathcal{L}$ is necessarily dependant on w and b, along with the data that was propagated. This data can also come with labels, which often denote a datapoint's class or some present feature(s). In contrast to the aforementioned input-datapoints, labels are often only used in the loss computation (and not propagated through the model). The use of such labels is denoted as *supervision*: We refer to training on only the datapoint as *unsupervised* learning, and training on input data along with some label(s) as *supervised* learning. This notion is somewhat flexible and is mostly used to indicate that some additional information was provided. For example, training models on data grouped by class is also considered supervised learning, even though no explicit labels are provided.

The entire learning process is referred to as *Gradient Descent*[1], and is often put into practice by calculating the loss using small batches of datapoints at a time. The exact tuning of all parameters is usually done using an optimizer, which allows for variably-sized parameter updates, a variable learning rate. A prominent example of such an optimizer is Adam[69, 140].

4.2 The Variational Autoencoder

The Variational Autoencoder[70, 124] is a model framework focused on learning the probability distribution of its training data through variational Bayesian inference. It models latent variables that describe the data (through an encoder) and models the data given these latent variables (through a decoder). We use the VAE as the baseline of our method for two reasons. First, modeling a dataset's probability distribution provides a starting point for decomposing the data into a representation we can understand and explain. Second, its generative capabilities (through the decoder) allow us to synthesize data in order to illustrate the data's structure (and explain said data). In this subsection, we define the Variational Autoencoder's architecture, alongside its objective and training procedure.

To provide intuition into the Variational Autoencoder we first consider its architecture, which closely resembles that of 'traditional' autoencoders[72]. An autoencoder consists of an inference model, the encoder, and a generative model, the decoder. Its general objective is to use the encoder to compress datapoint x to some small vector z (the latent space). This vector z is then used by the decoder to (attempt to) reconstruct the original data, creating reconstruction $\widetilde{x}$. To optimize this model, we minimize a reconstruction error; for example, the squared error between x and $\widetilde{x}$. The autoencoder's general structure and (example) objective are depicted in Figure 4.2.

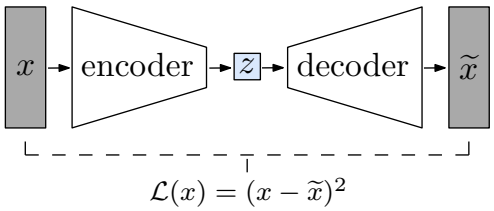


Figure 4.2: The architecture and objective of an autoencoder. Datapoint x gets compressed to z, which the decoder uses to create $\widetilde{x}$, a reconstruction of x. The objective is to minimize the reconstruction error.

Due to the lack of restrictions on the z-space of an autoencoder, we have no guarantees regarding its structure; the z-space's only objective is to retain as much information as possible when paired with its encoder and decoder. In practice, an autoencoder with no restrictions generally does not learn interpretable or useful representations[68, 12]. In contrast to inferring an arbitrary (information-retaining) encoding, the VAE approaches the principle of modeling latent variables from the (Bayesian) perspective of modeling x. We assume that x is generated according to some unknown variable z and seek to model this relation, i.e., approximate the true posterior distribution $p(z|x)$ of the data (using the encoder). The process we seek to model is depicted in Figure 4.3.

[1]There are other approaches to computing a NN's parameters (e.g., [83, 102]); however, gradient descent is by far the most common method[2].

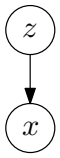


Figure 4.3: The directed graphical model of the assumed generating process of x.

To express this unknown variable, we model the z-space of our model as a probability distribution: The encoder infers the distributions' parameters (e.g., mean z_μ and standard deviation z_σ) from some datapoint x. To reconstruct x, we take a sample z from this distribution and put it through the decoder. The general architecture of the VAE is depicted in Figure 4.4 (alongside its training objective and sampling process, which will be described in subsequent paragraphs).

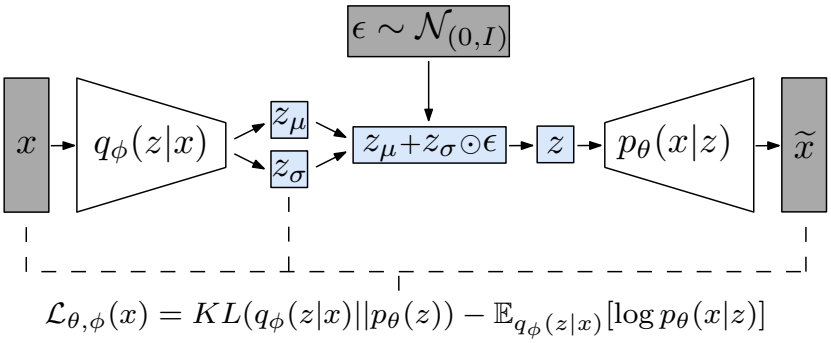


Figure 4.4: The Variational Autoencoder's architecture, along with its sampling process and training objective. Datapoint x gets encoded to z_μ and z_σ, from which is sampled using the reparameterization trick. The resulting z is decoded to $\widetilde{x}$, a reconstruction of x. The objective is to simultaneously regularize the latent space and minimize the reconstruction error.

In order to train the VAE, we seek to approximate the true posterior $p_\theta(z|x)$ (infer the 'correct' latent space z) and maximize the likelihood $p_\theta(x)$ of the data we create (generate realistic data). However, both these terms are intractable, we cannot directly optimize for them. Instead, we maximize a lower bound on these terms, the Evidence Lower Bound ($ELBO$). The motivation for using the $ELBO$ term as objective, along with its derivation and a discussion of its implications, is given in Appendix A. The $ELBO$ term as a loss function (thus negated, as we minimize the loss and maximize the $ELBO$) is as follows:

$$\mathcal{L}_{ELBO} = \mathcal{L}_{\theta,\phi}(x) = \underbrace{KL(q_\phi(z|x)||p_\theta(z))}_{\text{Regularization cost}} \underbrace{-\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{Reconstruction cost}} \tag{4.4}$$

$q_\phi(z|x)$ denotes the distribution of latent variable z given data x, inferred by the encoder (parameterized by ϕ). $p_\theta(x|z)$ denotes the distribution of data x given latent variable z, inferred by the decoder (parameterized by θ). $p_\theta(z)$ denotes the assumed prior distribution of the latent variable z, most often a standard factorized normal distribution, $\mathcal{N}(0, I)$. θ and ϕ denote the parameters for the decoder and encoder model, respectively. To give some intuition, we could consider $q_\phi(z|x)$ as the encoder, and $p_\theta(x|z)$ as the decoder.

As denoted in Equation 4.4, we can understand the objective by considering it as two components: A regularization cost and a reconstruction cost. The first term, $KL(q_\phi(z|x)||p_\theta(z))$, minimizes the Kullback-Leibler divergence[75] between $q_\phi(z|x)$ and $p_\theta(z)$. The KL divergence measures how well a distribution maps to another (according to their relative entropy). Consequently, we can interpret this term as to what extent $q_\phi(z|x)$ matches the chosen prior distribution $p_\theta(z)$: It is a regularization term on the shape of the latent space. Assuming a standard factorized normal

distribution for $p_\theta(z)$, we can compute this KL divergence analytically[70] (denoted as the regularization term in Equation 4.5).

The second term, $-\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$, considers the (negated) expectation of the distribution of data x given our z space. We can view this as the negative log-likelihood of inferred (decoded) data, given our inferred (encoded) latent space. This term is intractable but can be approximated using Monte Carlo estimation. To do so, we infer a latent distribution $\mathcal{N}(z_\mu, z_\sigma)$ from some datapoint x using the encoder. We then sample z from this distribution and compute $-\log p_\theta(x|z)$ by decoding this sample and comparing the reconstructed datapoint $\widetilde{x}$ to our original x. Consequently, we can interpret this term as a quality measure of the encoder and decoder's combined reconstructions: It is a reconstruction penalty (denoted as the reconstruction term in Equation 4.5, assuming squared error as penalty function).

Using an analytical computation of the KL divergence and single-sample Monte Carlo estimation of our (negated) expectation (using squared error as reconstruction cost), we can compute the loss function as follows:

$$\mathcal{L}_{ELBO} = \mathcal{L}_{\theta,\phi}(x) = \underbrace{\frac{1}{2}(z_\mu^2 + z_\sigma^2 - \log z_\sigma^2 - 1)}_{\text{Regularization cost}} + \underbrace{(x - \widetilde{x})^2}_{\text{Reconstruction cost}} \tag{4.5}$$

Directly minimizing this loss function with gradient descent poses an issue: We cannot take gradients with respect to the second component as the reconstruction of $\widetilde{x}$ relies on sampling the z variable from our inferred distribution, which is non-differentiable. To get around this limitation, we employ a technique called the *reparametrization trick*[70]. We do not sample our z directly from the inferred distribution:

$$z \sim q_\phi(z|x) = \mathcal{N}(z_\mu, z_\sigma) \tag{4.6}$$

Rather, we introduce an auxiliary variable ϵ, which is sampled from the assumed prior distribution $p_\theta(z)$, e.g., $\epsilon \sim \mathcal{N}(0, I)$. We then 'sample' z according to ϵ, by shifting ϵ according to our inferred mean z_μ and scaling it according to our inferred standard deviation z_σ:

$$z = z_\mu + z_\sigma \odot \epsilon \tag{4.7}$$

The resulting variable z gives us a deterministic variable estimating the random variable we seek to model, allowing us to take (estimated) gradients in order to optimize our model. The complete process is visualized in Figure 4.4. In short, we optimize the VAE's parameters θ and ϕ jointly by minimizing $\mathcal{L}_{\theta,\phi}(x)$ (as expressed in Equations 4.4 and 4.5) using gradient descent, employing the reparametrization trick. For a step-by-step description, we refer to Algorithm C.1 in the appendix.

4.3 Conditioning the VAE's Latent Space

The VAE serves as the base-model for our method, allowing us to represent data as a relatively low-dimensional latent variable z (shaped according to the chosen prior distribution). We seek to extend the VAE by conditioning this latent space further, exercising more control over its structure. In doing so, we can shape it to better fit data-explanation. In this subsection we define and motivate methods to this end, while navigating the landscape of VAE adaptations.

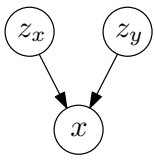
The notion of learning disentangled representations in VAEs[2] considers conditioning the latent variable z to 'disentangle' its (individual) dimensions. While there is no single formalized definition of disentanglement, one can define it as the notion that single latent dimensions are sensitive

[2]The study of disentangled representation learning is not exclusive to VAEs. Disentanglement in Generative Adversarial Networks (GANs)[42, 19, 63] often bears substantial similarity to that in VAEs; however, for this section we restrict ourselves to VAE-based approaches, as GAN-based methods serve more as inspiration rather than a foundation (i.e., we cannot implement them as-is in a VAE).

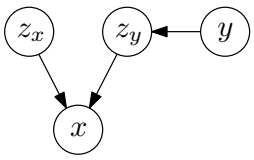
to changes in single generative factors, while being invariant to changes in others[12, 51, 86]. We consider methods addressing this principle, as learning a (more) disentangled representation allows for better use of the latent factors in explanation. Following Chapter 3, we recognize that categorization depends on class-based attributes. The ability to isolate (or even individually disentangle) these attributes consequently aids our ability to generate good explanations. To discuss work on disentanglement, we first distinguish between two overarching approaches: Unsupervised methods and supervised methods.

Unsupervised methods most often approach disentangling z by adapting the regularization term[86, 158] (the first term on the right-hand side in Equation 4.4). This adaptation ranges from increasing the weight of this term[51] to auxiliary models encouraging some desired structure, e.g., a factorized z-variable[67]. While these methods can successfully impose some additional structure, [86] show that they are unreliable in practice. In a large-scale study, they observe that learning disentangled representations using these methods is unreliable, often depending more on hyperparameters and the random seed than model choice. Additionally, they pose that we cannot identify well-disentangled models without supervision. In other words, while additional unsupervised objectives *can* add some desired structure to z, they are unreliable in practice, and identifying their success is seemingly impossible without additional supervision. On account of these observations, we primarily focus on structuring the latent space of our model according to supervised objectives.

Disentanglement objectives guided by supervision usually concern separating the latent space into two subspaces, where one stores the 'supervised' information, and the other stores the remaining information. In order to discuss this separation, we denote z_y as the partition we seek to condition according to this additional information, and z_x as the partition containing the remaining information necessary to (re)generate x. The assumed underlying generating process is depicted in Figure 4.5a.



(a) The assumed generating process for group-based disentanglement.



(b) The assumed generating process for label-based disentanglement.

Figure 4.5: The directed graphical models of the assumed generating processes of x. We assume the information we supply (through groups or labels) is generated according to z_y, whereas z_x denotes the latent dimensions considering the remaining information necessary to generate x. y denotes the assumed 'generating' label.

To discuss supervision-based disentanglement, we discern two approaches: Grouping-based methods and label-based methods. Grouping-based methods encourage separation in the latent space according to some relation present in groups (often pairs) of datapoints. A prime example of this notion is grouping data according to a common factor and encouraging this common factor to be stored in a specific part of the latent space (e.g., [15, 57]). Label-based methods encourage separation in the latent space according to label-information. These methods often encourage information that can be used to predict labels to be stored in a specific part of the latent space (e.g., [174, 56]). The difference in the assumed generating processes is illustrated in Figure 4.5.

In order to contrast these two approaches, we underline some key assumptions they rely on (and differ in). For one, the 'level' of supervision is not equivalent: While we can generate groups according to labels (by grouping datapoints together according to their label), this relationship does not extend the other way; there is no guarantee that we can infer labels from groups. For

example, there could be many independent groups of data that could share a label in some unknown, underlying process. Alternatively, it could be that there is no (straightforward) labeling of the data possible at all.

Additionally, even when considering the same presupposed information (generating groups according to labels), we pose that the resulting conditioning on z is not necessarily equivalent. The key observation is differentiating between *common* information and *distinguishing* information being stored in z_y. Assuming no additional information, groups constructed according to labels can only guarantee to contain information present in all examples of some class (label-value), resulting in the disentanglement method likely only encouraging this common information to be stored in z_y. If class-distinguishing information is only present in a small subset of some class' examples, we cannot expect these examples to be grouped together often. Consequently, we cannot expect this information to be stored in z_y. However, when conditioning z_y according to supplied labels, a sufficiently large model can pick up on any information in x that can be used to determine its label (as is apparent in the [more extreme] case of overfitting[150]). Note that a formal evaluation of this claim is beyond the scope of this work; rather, we use this notion to decide what prior work we base our method on.

As our method aims to explain the classification of data, we are interested in separating the information relevant to this process from the remaining information available in our data; we want to separate the *distinguishing* information. Consequently, we employ label-based disentanglement. The approach to doing so rests on the use of classification objectives, following the works of [174, 56, 29]. z_y is encouraged to contain information related to the supplied label y, whereas z_x is discouraged from containing such information. This objective corresponds to the generating process described in Figure 4.5b. To encourage this objective, two auxiliary classifiers are introduced. One attempts to recover y from z_y, denoted as $c_{\psi_0}(y|z_y)$, whereas the other attempts to recover y from z_x, denoted as $c_{\psi_1}(y|z_x)$. Both these classifiers denote discrete probability distributions, i.e., their outputs consist of a vector of size n (where n is the number of classes), where each element denotes the probability of the class corresponding to that index (with the total summing up to 1).

To finalize the architecture description, we note that empirical results indicate that better generalization is reached when generating these two components of z using different encoders[56]. Therefore, we introduce separate encoders inferring z_y and z_x, denoted by $q_{\phi_0}(z_y|x)$ and $q_{\phi_1}(z_x|x)$, respectively. An overview of the entire model architecture is depicted in Figure 4.6.

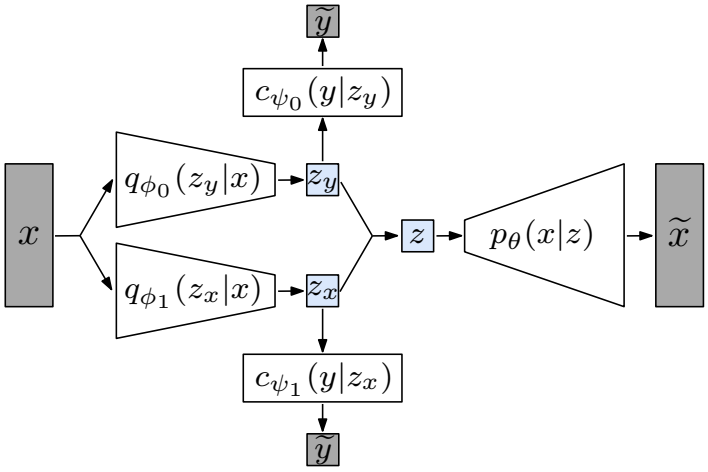


Figure 4.6: The architecture of the disentangled VAE. Datapoint x is encoded by two separate encoders into z_x and z_y, which are concatenated to reconstruct $\widetilde{x}$. We omit the sampling procedure for clarity.

To optimize z_y, we train its classifier $c_{\psi_0}(y|z_y)$ simultaneously with the encoder, using categorical cross-entropy as loss:

$$\mathcal{L}_{classy} = \mathcal{L}_{\psi_0,\phi_0}(x,y) = -\mathbb{E}_{q_{\phi_0}(z_y|x)} \sum_i^n y_i \log(c_{\psi_0}(y_i|z_y)) \tag{4.8}$$

where x denotes the datapoint we propagate, z_y the latent space we condition, y the ground-truth label (represented as a one-hot encoding[3]), n the number of classes, and $c_{\psi_0}(y_i|z_y)$ the probability of class y_i given z_y, supplied by a classifier parameterized by ψ_0. Note that we also train the encoder with this loss, as z_y is determined by encoder $q_{\phi_0}(z_y|x)$.

To optimize z_x, we train its classifier $c_{\psi_1}(y|z_x)$ and the encoder separately. The classifier is trained to predict y, again using categorical cross-entropy as loss:

$$\mathcal{L}_{classx} = \mathcal{L}_{\psi_1}(x,y) = -\mathbb{E}_{q_{\phi_1}(z_x|x)} \sum_i^n y_i \log(c_{\psi_1}(y_i|z_x)) \tag{4.9}$$

sharing the same variable notation as Equation 4.8, besides optimizing z_x instead of z_y, using classifier $c_{\psi_1}(y|z_x)$ instead of $c_{\psi_0}(y|z_y)$ and using encoder $q_{\phi_1}(z_x|x)$ instead of $q_{\phi_0}(z_y|x)$. Note that we optimize parameters ψ_1, only training the auxiliary classifier.

To condition the information stored in z_x, we optimize its encoder such that z_x cannot be used to classify well. We consider two methods for doing so: Optimizing for the negative cross-entropy[16] (reversing the loss from Equation 4.9) or optimizing for an uninformative probability distribution[174]. These two objectives are denoted in Equations 4.10 and 4.11, respectively:

$$\mathcal{L}_{advx} = \mathcal{L}_{\phi_1}(x,y) = \mathbb{E}_{q_{\phi_1}(z_x|x)} \sum_i^n y_i \log(c_{\psi_1}(y_i|z_x)) \tag{4.10}$$

$$\mathcal{L}_{advx} = \mathcal{L}_{\phi_1}(x,y) = -\mathbb{E}_{q_{\phi_1}(z_x|x)} \sum_i^n \frac{1}{n} \log(c_{\psi_1}(y_i|z_x)) \tag{4.11}$$

sharing the same variable notation as Equation 4.9. Note that we optimize parameters ϕ_1, only training the encoder.

To denote the entire loss function, we adjust the $ELBO$ loss (Equation 4.4) according to the two subspaces z_x and z_y, and split it into a reconstruction and regularization term. These functions are as follows:

$$\mathcal{L}_{ELBO} = \mathcal{L}_{\theta,\phi_0,\phi_1}(x) = \mathcal{L}_{rec} + \mathcal{L}_{kl_0} + \mathcal{L}_{kl_1} \tag{4.12}$$

$$\mathcal{L}_{rec} = \mathcal{L}_{\theta,\phi_0,\phi_1}(x) = -\mathbb{E}_{q_{\phi_0}(z_y|x),q_{\phi_1}(z_x|x)}[\log p_\theta(x|z_y,z_x)] \tag{4.13}$$

$$\mathcal{L}_{kl_0} = \mathcal{L}_{\phi_0}(x) \quad = KL(q_{\phi_0}(z_y|x)||p_\theta(z)) \tag{4.14}$$

$$\mathcal{L}_{kl_1} = \mathcal{L}_{\phi_1}(x) \quad = KL(q_{\phi_1}(z_x|x)||p_\theta(z)) \tag{4.15}$$

The entire loss function for the label-disentangled VAE then is as follows:

$$\mathcal{L}_{DVAE} = \mathcal{L}_{\theta,\phi_0,\phi_1,\psi_0,\psi_1}(x,y) = \mathcal{L}_{rec} + \beta_0\mathcal{L}_{kl_0} + \beta_1\mathcal{L}_{kl_1} + \alpha\mathcal{L}_{classy} + \gamma\mathcal{L}_{advx} + \mathcal{L}_{classx} \tag{4.16}$$

consisting of Equations 4.13, 4.14, 4.15, 4.8, 4.10 or 4.11, and 4.9, respectively. We introduce hyperparameters β_0 and β_1 to control the regularization term (as proposed by [51]), and to balance

[3] A one-hot encoding represents categorical data as a vector of size n, where n denotes the number of categories. This vector consists of a 1 at the place of the given class and 0s at remaining positions, e.g., $class = 0$ with $n = 4$ gives $[1, 0, 0, 0]$. We can interpret this as a categorical probability distribution of the class, often used when optimizing some discrete probability, e.g., a classifier's prediction.

the two subspaces. Additionally, we introduce terms α and γ to balance the disentanglement-objectives against the rest of the model. Note that while $\mathcal{L}_{advx}$ and $\mathcal{L}_{classx}$ consider the same classifier, the former optimizes only the encoder parameters, whereas the latter optimizes only the classifier parameters. For a step-by-step description of the entire training process, we refer to Algorithm C.2 in the appendix.

Balancing the two latent subspaces with auxiliary parameters is done to avoid a degenerate solution where z_x contains no information at all, which is referred to as posterior collapse[89]. This issue can arise because the auxiliary objectives encourage for little information to be stored there, which, along with the KL divergence term regularizing the space, can lead to it matching the prior too closely and storing no information. By setting $\beta_1 < \beta_0$, we encourage the model to make use of z_x as it is 'cheaper' to store information there. Another approach to combating this issue rests on maximizing the mutual information between latent dimensions and the inferred (or original) datapoints[123, 143], however, we found that such an approach can lead to degenerate solutions where (in the case of inferred data) the VAE merely learns to store information in arbitrary pixels. Adjusting such an approach to avoid this pitfall is left for future work.

Two final notes we make on the disentanglement process are as follows. First, while we discussed this process in the context of two subspaces, these methods extend to arbitrarily many subdivisions: We can consider z_y as the subspace we seek to condition by some label y, whereas we can consider z_x as a concatenation of all other subspaces, i.e., arbitrarily many other label-conditioned subspaces. Second, while we discussed the auxiliary classifiers as depending on z_y or z_x, they do not necessarily have to depend on a sampled value: One can infer label-predictions according to both a sampled value (z_y or z_x), or according to the mean of the distribution (μ_y or μ_x).

Chapter 5

Method

In this chapter, we describe and motivate our method to generating contrastive explanations of classifications, VAE-CE. The method rests on two main components: A conditioned VAE that represents the target data in a desirable manner, and a method for traversing this model's latent space in order to generate contrastive explanations. Additionally, we describe and motivate a method for evaluating the explanation-quality of an explanation-generation method. In Section 5.1, we describe and motivate VAE-CE in its entirety. In Section 5.2, we describe the representation-model, alongside a novel technique for imposing structure in VAEs. In Section 5.3, we describe the method for using this model to generate contrastive explanations. Finally, in Section 5.4, we introduce and define a method for the quantitative evaluation of systems generating interpolation-based contrastive explanations.

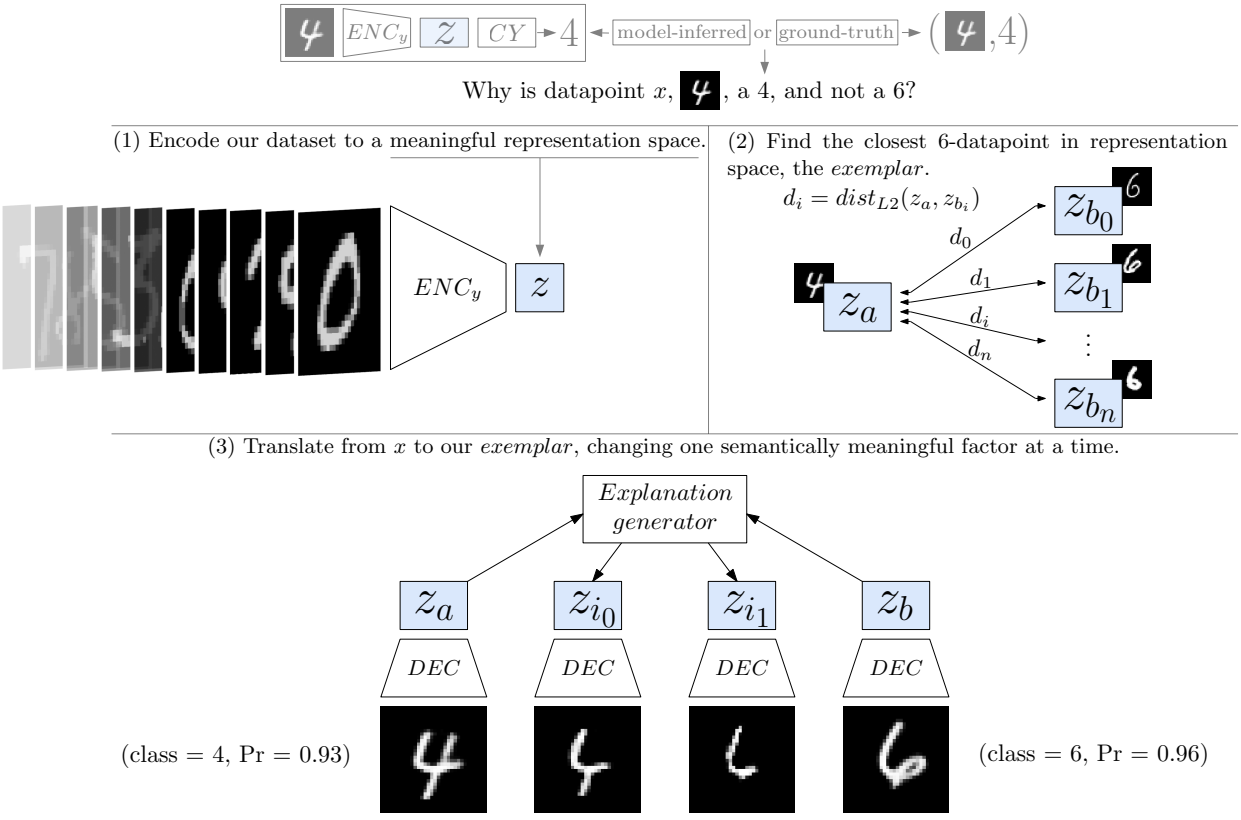


Figure 5.1: A simplified overview of the inference process of VAE-CE; how we generate contrastive explanations. First, we *encode* a dataset to a semantically meaningful space. To then explain why datapoint x did not belong to some foil-class, we traverse this space in order to find an *exemplar* datapoint of this foil-class. We then employ the *explanation generator* paired with the *decoder* in order to create the transition from our datapoint to this exemplar, optimizing for high explanation-quality. Example generated using the MNIST database[81].

5.1 Motivation

In this section, we provide an overview of the method and its underlying assumptions. We start by considering its purpose, from which we motivate its individual components. Finally, we discuss and motivate the supervision used for training VAE-CE.

We define our method's purpose as creating explanations of the classification (or categorization) process of data. In other words, the purpose is to create descriptions of why some datapoint belongs to a particular class. Following prior work on the ideas of explanation and categorization (as described in Chapter 3), we define the goal of our method as generating an output that conveys why a categorization was made, in contrast to some hypothetical alternative decision, i.e., '*Why does datapoint* $\boldsymbol{x}$ *belong to class* $\boldsymbol{a}$ *rather than class* $\boldsymbol{b}$*?*' (a *contrastive* explanation).

To be able to explain this decision, we subscribe to the notion of defining classes by their $features$. As such, the generated explanation should convey the (higher-level) features that distinguish datapoint x from examples of class b. To identify and visualize these features, we adhere to the idea of *exemplar*-based categorization. First, we find the datapoint belonging to class b with the minimal distance to datapoint x (of class a) in some semantically meaningful space. In order to identify this exemplar, we consider both its (Euclidean) distance in this class-specific latent space, alongside the model's confidence that this datapoint belongs to the counterfactual class (to avoid explaining towards some badly interpreted [or outlier] datapoint). We then convey the class-distinguishing features by visualizing the transition from x to this exemplar's class-features, changing one semantically meaningful feature at a time. This explanation-inference process is depicted in Figure 5.1.

Generated explanations depict the distinguishing features between a datapoint and some exemplar of the counterfactual class b. As such, they do not directly involve class a. Rather, we either assume that class a is prior knowledge, making the goal to explain why this example belongs to class a rather than class b, or we first infer class a using our model, making the goal to explain why x was classified as class a rather than class b. If class b not is provided, we select the 2^{nd} most probable class (according to the inferred class probabilities). In this case, the resulting explanation depicts the class-features considered most similar by the model (to those of a).

Given the explanation-goal and its accompanying process, we identify the overarching components necessary to generate explanations. For one, we seek to operate in the space of semantic meaning. As such, we need a model to map between data-space (pixels) and this meaningful representation-space, which is fulfilled by a (class and feature-conditioned) Variational Autoencoder. Additionally, we require a classification model dependant on this representation space, as we seek to classify and consequently explain data according to this space. The VAE, the employed conditioning process used to achieve the desired representation space, and its latent space classifier, are described in Section 5.2.

To generate an explanation, we interpolate between the to-be-explained datapoint and the identified exemplar. As traversing a complex, high-dimensional latent space is non-trivial[20, 171], we describe an approach to traversing this space, denoted as the 'explanation generator' component. The main objectives of this approach entail generating an explanation that is realistic, considers one feature change per step, and is simplistic. To realize these characteristics, the explanation generator builds upon three models: The aforementioned VAE (for operating in a space with semantic meaning) and two 'discriminator' models, which evaluate the realism (of an image) and change quality (between a pair of images), respectively. Both these models are also part of the conditioning process of the VAE, further illustrating the coherence of the representation-model and the explanation generator. These discriminator models are defined in Section 5.2, whereas the explanation generator is defined in Section 5.3.

To finalize the overview of VAE-CE, we describe and motivate the assumed supervision with regards to the data we explain. As we cannot expect a method to provide useful results without some (inductive) biases[167], we explicitly discuss these assumptions. First, we assume access to class labels, as these determine the ground-truth process we seek to explain. Additionally, we assume access to information that defines the type of feature we consider, through pairs of images containing a single feature change (the 'positive' pairs) or pairs containing either no change, or more than one feature change (the 'negative' pairs). The underlying assumption is that these change-pairs exhibit the structure of features we deem meaningful for explanation. A downside is that as we do not distinguish between different feature structures, we cannot consider features that have little in common separately (leading to a potentially worse representation for data concerning significantly distinctive features).

A common context for this type of supervision is data where we can identify features according to some assumption of what *a* feature is, but labeling these features (grouping them) brings greater difficulty. Following the identification of distinct features in a single datapoint, pairs can be created by augmenting the datapoint such that a single feature differs. Another example is data with some temporal coherence: Assuming the observations' features change over time (one at a time), we can create change-pairs according to these progressions, without having to label all features.

We choose to assume supervision regarding features as the desired feature shape is highly dependant on the problem context and the assumed prior knowledge (as noted in Chapter 3). As such, embedding some feature-shape in the model's inductive bias narrows its applicability down to only problems fitting this assumed feature-shape.

To conclude, we note a set of alternative approaches and why we did not consider them. For one, to explain a classification, one can consider single latent dimensions as features and communicate their respective values. We pose that such an approach requires stronger assumptions about the latent space (that we do not wish to make). By explaining according to synthesized data, we can restrict our focus to the relationship between transformations in the latent space and transformations in data-space, whereas directly explaining according to latent features requires knowledge about what dimensions mean and how they depend on their context.

Regarding supervision, we do not assume 'full' supervision, a label for each feature, for two reasons. For one, it is sometimes significantly more challenging to create a full feature-labeling rather than grouping images with single-differences. Note that the opposite can also be the case, e.g., when labeling features is trivial but few single-change pairs exist and augmenting data is non-trivial. Additionally, if we assume that classes are determined by their features, and we have labels of all features, we can directly generate explanations according to this labeling (reducing the problem to only inferring labels, i.e., feature-classification). While there is still merit in designing such a method, we deem the overall utility of designing a method according to this level of supervision to be lower.

Lastly, we choose not to assume supervision regarding explanations in their entirety, e.g., providing the model with datapoints and ground-truth explanation. Such an approach requires the model to learn the notion of explanation autonomously, whereas we can explicitly embed this notion through the method design (inductive bias). Additionally, such supervision is likely more expensive to gather. It both requires more steps per 'sample' (an entire explanation versus class-labels and change-pairs) and presumably requires more samples overall (to capture the increase in problem-complexity).

5.2 The Underlying Models

In this section, we describe and motivate the underlying models we employ to generate explanations. In subsection 5.2.1, we provide an overview of all individual models, alongside their relations to both the representation model and the explanation generator. In Subsection 5.2.2, we define and motivate a new approach with regards to conditioning VAEs according to image-pair based supervision (which can also be applied in a more general case). In Subsection 5.2.3, we describe how we implement this new method in our representation model, and provide a detailed description and training procedure of the entire representation model and its external components.

5.2.1 All Components

To start, we provide an overview of the representation model along with all auxiliary models employed in its conditioning process. A subset of these models is also used in the explanation generation process. An overview of all these models and their relations is depicted in Figure 5.2.

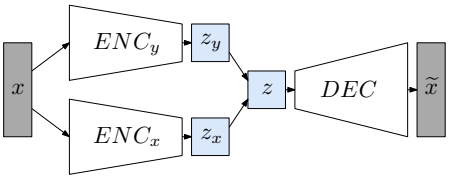
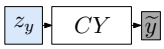
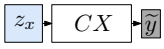
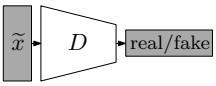
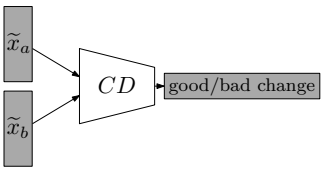
1 	The main representation model. ENC_y and ENC_x encode the datapoint to a class (z_y) and non-class (z_x) specific space. We also refer to these two encoders by the distributions they infer: $q_{\phi_0}(z_y\|x)$ and $q_{\phi_1}(z_x\|x)$, respectively. z_y is additionally conditioned to structure its dimensions according to a presupposed feature shape, giving us the desired semantically meaningful representation. DEC decodes a concatenation of z_x and z_y back to a datapoint. We also refer to DEC as the distribution it infers, $p_\theta(x\|z)$ or $p_\theta(x\|z_x, z_y)$. The entire model is based on a (disentangled) VAE.
2 	Classifier used to condition z_y to represent class-information. Additionally used to infer class-predictions and to identify suitable exemplars. Also referred to by its inferred distribution, $c_{\psi_0}(y\|z_y)$.
3 	(Adversarial) classifier used to condition z_x to represent non class-specific information. Also referred to by its inferred distribution, $c_{\psi_1}(y\|z_x)$.
4 	Discriminator model used to encourage realism in generated samples, used when conditioning z_y to be shaped according to some feature-structure. Additionally used by the explanation generator to find realistic intermediate states.
5 	Change-discriminator model used to encourage dimensions in z_y to concern a presupposed feature-structure (inferred from supervised pairs). Additionally used by the explanation generator to find an interpolation containing realistic changes. Whereas all other models are trained simultaneously, CD is (pre-)trained in isolation, using the positive and negative change pairs.

Figure 5.2: An overview of all model components. White indicates a NN-model, gray indicates an input or output, and blue indicates a latent space.

The motivation behind these components is as follows. The representation model (1) lets one work with data in a semantically meaningful space (denoted as z_y), which allows for the identification of distinguishing features and for the generation of explanations. This model is built upon the disentangled VAE, as described in Section 4.3. As such, the representation model considers two latent spaces: z_y, containing class-distinguishing information, and z_x, containing the remaining information embedded in the data. Additionally, we seek for z_y to not only contain class-related information but to also exhibit a shape similar to some presupposed notion of what features are. To

achieve this shape, z_y is further conditioned, as described in Subsection 5.2.2. The classifier models (2&3) follow from the label-disentanglement method. The z_y-classifier provides the classification-ability to the model, allowing one to infer predictions on (novel) data. Additionally, it is used to identify suitable exemplars, as described in Subsection 5.3. The discriminator models (3&4) follow from the feature-shape conditioning method, and are additionally used to identify both realistic interpolation states and realistic changes in these states. These applications are described in Sections 5.2.2 and 5.3. A description of the entire training process of all model components is given in Section 5.2.3.

5.2.2 Pair-Based Dimension Conditioning

In this subsection, we propose a new method for conditioning the latent space of a VAE. The purpose of this method is to encourage individual latent dimensions to exhibit a presupposed effect in data-space, based on (supervised) image pairs. As such, this method can be considered as an additional regularizer on the latent space. The underlying assumption of this approach is that the supervised image pairs exhibit the result of a single feature (latent dimension) being changed, and that different features follow the same general structure. We condition the latent dimensions in an attempt to make each dimension represent a change of such structure. An overview of this process is depicted in Figure 5.3. In the remainder of this subsection, we discuss this process (alongside its auxiliary models) in detail.

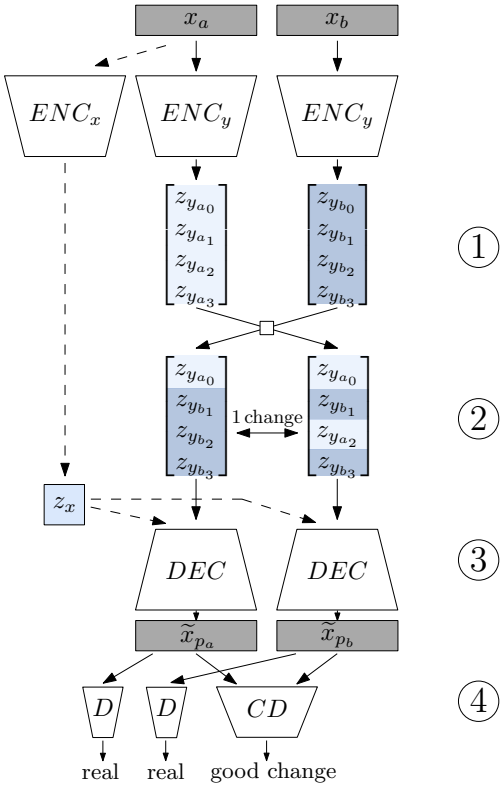


Figure 5.3: The pair-based conditioning process applied to a disentangled VAE (The method does not require two latent spaces, as indicated by the dashed lines: The secondary latent variable can be omitted). (1) We encode a pair of images and sample their latent variables z_{y_a} and z_{y_b}. (2) We construct two latent variables based on these pairs, differing in a single dimension. (3) The two latent variables are decoded (using a shared variable for the remaining latent dimensions, if applicable) to $\widetilde{x}_{p_a}$ and $\widetilde{x}_{p_b}$. (4) The reconstructed pair of images is put through the change discriminator (CD), optimizing for a 'good' change, and the images are individually put through a regular discriminator (D), optimizing the quality of the generated samples.

The approach employs a pre-trained auxiliary model to enforce the desired structure. This model, denoted as the 'Change Discriminator' (CD), takes a pair of datapoints as input and infers whether this pair exhibits a desirable change or not. In our method, we construct CD by pre-training a binary classifier using pairs of images that either indicate a good change (a single feature change between the images) or a bad change (no or multiple feature changes). We note that this implementation is not a necessity: Any differentiable model (such that we can backpropagate its error into the VAE) could be applied, given that it can distinguish between good and bad change-pairs. As such, we do not consider this model in detail in this subsection. For details about CD in the context of the entire method, we refer to Subsection 5.2.3.

To ensure that the inferred samples are of good quality, we employ an additional model, referred to as the 'Discriminator' (D). Since we infer samples that do not directly correspond to real datapoints, we cannot use the VAE reconstruction error to optimize for image quality. Additionally, as these samples are synthesized in order to maximize the probability of a 'good change' in CD, we run the risk of training the VAE to create some sort of 'adversarial attack'[153] on CD. That is, the VAE could learn to make small perturbations to its output (that do not coincide with realistic changes) in order to fool CD to classify them as desired. By evaluating whether a synthesized image is 'real' or 'fake' according to this auxiliary discriminator (and training the VAE according to this error), we attempt to circumvent these issues.

This discriminator is implemented as a binary classifier and is trained during the conditioning process, simultaneously with the VAE. For every training step, it is supplied with both fake images (generated by the VAE during the conditioning process) and real samples. Consequently, D closely resembles the discriminator of a GAN[42]. To train this binary classifier, we optimize the binary cross-entropy:

$$\mathcal{L}_D = \mathcal{L}_\omega(\widetilde{x}) = -[r \log(D(\widetilde{x})) + (1 - r) \log(1 - D(\widetilde{x}))] \tag{5.1}$$

where $\widetilde{x}$ depicts the (potentially synthesized) datapoint we evaluate, r concerns the 'class' of the datapoint ($r = 0$ being fake and $r = 1$ being real), $D(\widetilde{x})$ indicates the inferred probability of the datapoint being real, and ω denote D's (to-be-optimized) parameters.

Following the definition and training process of these two models, we can present the entire conditioning process. This process consists of four stages.

(1) We encode two (arbitrarily picked) datapoints x_a and x_b to the latent space we wish to condition by first inferring the latent distributions' parameters and then sampling from these distributions. Note that if we seek to optimize only a subpart of the latent space (e.g., in a disentangled model), we share the remaining latent dimensions between the two samples we consider. In this method, we optimize subpart z_y (and consequently share z_x). We denote the inferred latent variables as z_{y_a} and z_{y_b}, respectively.

(2) We construct two latent variables that share all but one dimension, created by combining z_{y_a} and z_{y_b} stochastically. We denote these variables as $z_{y_{pa}}$ and $z_{y_{pb}}$, respectively. Each individual dimension comes from either z_{y_a} and z_{y_b}, which can be denoted as follows:

$$\sum_i^n (z_{y_{pa_i}} \in \{z_{y_{a_i}}, z_{y_{b_i}}\}) \tag{5.2}$$

$$\sum_i^n (z_{y_{pb_i}} \in \{z_{y_{a_i}}, z_{y_{b_i}}\}) \tag{5.3}$$

where n denotes the number of dimensions. Additionally, all but one dimension are shared, which

can be denoted as follows:

$$\sum_{i \in N \setminus \{dim\}} (z_{y_{pa_i}} = z_{y_{pb_i}}) \tag{5.4}$$

$$z_{y_{pa_{dim}}} \neq z_{y_{pb_{dim}}} \tag{5.5}$$

where N denotes the set of indices (e.g., $N = \{0, 1, 2\}$ if $n = 3$) and dim denotes the index of the differing latent dimension.

(3) We map the constructed pair (and its shared latent dimensions) back to data-space. That is, we synthesize $\widetilde{x}_{p_a}$ by propagating $(z_{y_{pa}}, z_x)$ through the decoder, and synthesize $\widetilde{x}_{p_b}$ by propagating $(z_{y_{pb}}, z_x)$ through the decoder.

(4) We optimize for the change-quality and realism objectives. To evaluate the change-quality objective, we propagate the pair ($\widetilde{x}_{p_a}$ and $\widetilde{x}_{p_b}$) through CD and take the loss with respect to its inferred change quality. However, we do not compute this loss in isolation. As the two latent variables and their distinguishing dimensions are created stochastically, we have no knowledge of the magnitude of their difference. Consider the case where the distinct dimension's values are almost equal: It is to be expected that the inferred image pair displays little difference. On the other hand, if the latent variable's difference is large, we can be more confident that the resulting pair should exhibit a (single, meaningful) change. As such, we scale the CD-loss according to the (absolute) difference in these two dimensions. Note that this difference cannot become arbitrarily large or small as these dimensions are still regularized by the $ELBO$ objective, ensuring no degenerate solution where latent dimensions' ranges are compressed in order to minimize the loss. The change quality loss can then be denoted as follows:

$$\mathcal{L}_{PAIR} = \mathcal{L}_{\theta,\phi_0,\phi_1}(\widetilde{x}_{p_a}, \widetilde{x}_{p_b}) = |z_{y_{pa_{dim}}} - z_{y_{pb_{dim}}}| \cdot -\log(CD(\widetilde{x}_{p_a}, \widetilde{x}_{p_b})) \tag{5.6}$$

where $CD(\widetilde{x}_{p_a}, \widetilde{x}_{p_b})$ denotes the inferred probability of this pair exhibiting a 'good' change, dim indicates the differing dimension, and θ, ϕ_0, ϕ_1 denote the variables of the VAE we seek to condition. $\widetilde{x}_{p_a}$ and $\widetilde{x}_{p_b}$ are created according to the latent spaces $z_{y_{pa}}$ and $z_{y_{pb}}$ (and a shared z_x), which in turn are constructed according to a pair of input datapoints. This construction process ensures we can take gradients with respect to all VAE parameters (we do not denote this process in the equation for clarity). Note that this equation is equivalent to the (scaled) binary cross-entropy when only considering the positive class.

We evaluate each datapoint's realism individually in order to encourage them to look realistic and to not fool CD through some adversarial attack. To do so, we propagate both datapoints through D and compute the loss with respect to the inferred probability of the datapoints being real:

$$\mathcal{L}_{REAL} = \mathcal{L}_{\theta,\phi_0,\phi_1}(\widetilde{x}_{p_a}, \widetilde{x}_{p_b}) = -[\log(D(\widetilde{x}_{p_a})) + \log(D(\widetilde{x}_{p_b}))] \tag{5.7}$$

where $D(x)$ denotes the inferred probability of x being a real datapoint, and θ, ϕ_0, ϕ_1 denote the variables of the VAE we seek to condition. Again, this equation is equivalent to binary cross-entropy with respect to the positive class.

The entire training objective of the pair-based dimension conditioning approach (training both the VAE and D) can then be denoted as follows:

$$\mathcal{L}_{DIM} = \alpha_r \mathcal{L}_{REAL} + \alpha_p \mathcal{L}_{PAIR} + \mathcal{L}_D \tag{5.8}$$

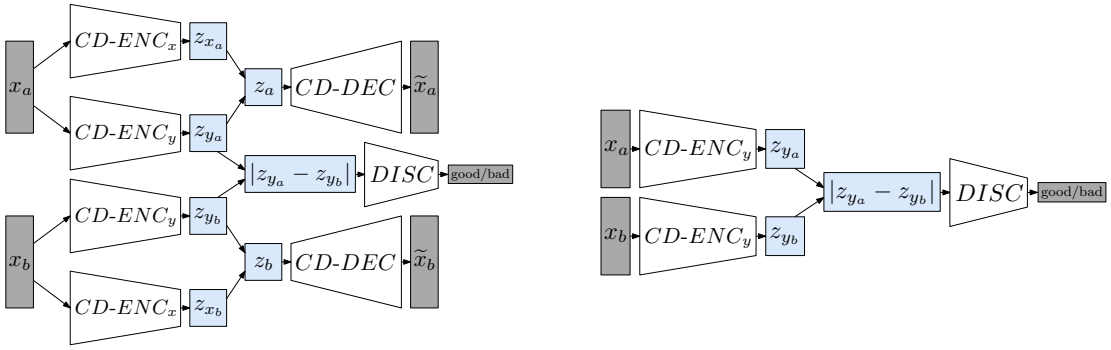
where α_r and α_p denote the coefficients for the realism and change-quality objectives, respectively. $\mathcal{L}_{DIM}$ is minimized simultaneously with the regular (disentangled) VAE objective. While in isolation this function only encourages one dimension to exhibit a specific change-structure, by stochastically constructing the latent-variable pairs and their differing dimensions we encourage the latent space as a whole to behave in this manner (we can refer to this notion as 'amortized' optimization, similar to maximizing the $ELBO$ in the VAE[129]).

5.2.3 Training the Components

In this subsection, we describe the training process of all underlying models. We start by defining CD and its training process, as this model is trained in isolation. Thereafter, we describe the (combined) training process of the remaining components, i.e., how we combine the pair-based dimension conditioning with the disentangled VAE approach.

CD takes a pair of datapoints (images) as input and infers the probability that this pair depicts a desirable change. To train this model, we assume to have access to image pairs depicting such desirable changes, alongside pairs that depict undesirable differences. Then, we aim to train a binary classifier distinguishing these two categories.

As CD is used to evaluate changes in 'interpolated' samples, it likely has to classify pairs of images quite distinct from its training set. In an attempt to deal with this issue, we construct CD in a heavily regularized manner. Rather than directly training an end-to-end classifier, we construct CD as an *additional* disentangled VAE and classify image pairs by taking the absolute dimension-wise difference of their (class-relevant) latent embeddings. This difference is then put through a discriminative model, which predicts whether this difference results from a 'good' or a 'bad' pair. The entire model is depicted in Figure 5.4.



(a) The classification pass of training CD. We encode two datapoints and evaluate whether the difference in z_y corresponds to a good or bad change.

(b) Components used when inferring a prediction from CD (as used in the context of conditioning another VAE).

Figure 5.4: Training and using CD.

Training this model according to its discriminative objective is done by minimizing the binary cross-entropy between the inferred labels and the true labels. This is denoted as follows:

$$pred(x_a, x_b) = DISC(|q_{c\phi_0}(z_{y_a}|x_a) - q_{c\phi_0}(z_{y_b}|x_b)|) \tag{5.9}$$

$$\mathcal{L}_{CDISC} = \mathcal{L}_{c\omega,c\phi_0}(x_a, x_b) = -[r \log(pred(x_a, x_b)) + (1-r)\log(1 - pred(x_a, x_b))] \tag{5.10}$$

where $q_{c\phi_0}$ denotes the encoder of the class-space ($CD\text{-}ENC_y$), x_a and x_b denote the input-pair, r denotes the label ($r = 0$ being a bad change and $r = 1$ being a good change) and $DISC$ denotes the latent-difference discriminator. $c\phi_0$ and $c\omega$ denote the (to-be-updated) parameters of $CD\text{-}ENC_y$ and $DISC$, respectively. During this pass, we also optimize the $ELBO$, as we also seek to regularize the latent dimensions such that they follow they assumed prior and represent the given datapoint. This addition make the total loss function of the classification-pass as follows:

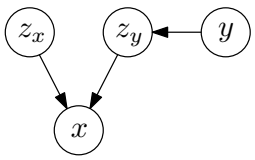
$$\mathcal{L}_{CD\text{-}CLASS} = \mathcal{L}_{c\theta,c\phi_0,c\phi_1,c\omega}(x_a, x_b) = \mathcal{L}_{rec} + \beta_0\mathcal{L}_{kl_0} + \beta_1\mathcal{L}_{kl_1} + \alpha_c\mathcal{L}_{CDISC} \tag{5.11}$$

consisting of Equations 4.13, 4.14, 4.15 and 5.10, respectively. $c\theta, c\phi_0, c\phi_1, c\omega$ denote the parameters of the decoder ($CD\text{-}DEC$), the two encoders ($CD\text{-}ENC_y$ and $CD\text{-}ENC_x$) and the discriminator ($DISC$), respectively. β_0 and β_1 serve to control the latent space regularization, and α_c serves as the coefficient for the classification objective.

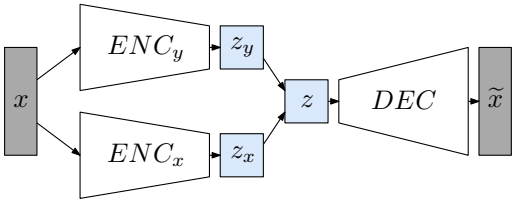
To train CD in its entirety, we alternate between this discrimination objective and the disentanglement objective (as described in Chapter 4). We do not train all objectives in one 'pass' as the datasets are not shared: The pairs do not necessarily depict examples of a class; rather, we only assume they depict some change. As such, we encourage the disentanglement in a separate pass, taking datapoints and labels from the 'regular' dataset. Training CD then works as follows: First, loss $\mathcal{L}_{CD\text{-}CLASS}$ is computed according to a pair of input points (x_a, x_b) and their change-quality label y. Thereafter, loss $\mathcal{L}_{DVAE}$ (Equation 4.16) is computed according to a datapoint x_{cl} and its class-label y_{cl}. Both losses are summed (with each subcomponent weighted according to the chosen hyperparameters), after which the parameter gradients are computed with respect to this error. Parameters are then updated according to these gradients by the optimizer, which concludes the training step. For a step-by-step description of this process, we refer to Algorithm C.3 in the appendix.

To conclude, we describe the training process of the remaining components. The assumed structure of the data we seek to explain, along with the representation model, is depicted in Figure 5.5. The training loop consists of two main steps, training the representation model and its auxiliary models at once. We optimize the model as a label-disentangled VAE (Section 4.3), training the main representation model and the latent space classifiers CY and CX. Additionally, we optimize z_y's dimensions to exhibit meaningful features (Section 5.2.2), training the main representation model and D.

The training loop works as follows. First, $\mathcal{L}_{DVAE}$ (Equation 4.16) is computed according to *two* pairs of input points and class-labels, (x_a, y_a) and (x_b, y_b). We compute the loss according to two pairs as we re-use the latent (class-space) embeddings of x_a and x_y, that is, z_{y_a} and z_{y_b}. The second step considers computing $\mathcal{L}_{DIM}$ (Equation 5.8). Using the aforementioned embeddings z_{y_a} and z_{y_b}, we create the single-difference latent variables and compute $\mathcal{L}_{DIM}$ accordingly. In order to train D, we additionally sample two real datapoints (coming from either the class-example or the change-pair dataset) and optimize its loss function $\mathcal{L}_D$. The respective losses are summed (with subcomponents weighted accordingly), whereafter the respective gradients are computed. Parameters are updated according to these gradients by the optimizer, which concludes the training step. All models (bar CD) are trained according to these loss functions (or subcomponents thereof; D according to $\mathcal{L}_D$, CY according to $\mathcal{L}_{classy}$ and CX according to $\mathcal{L}_{classx}$). For a step-by-step description of this training process, we refer to Algorithm C.4 in the appendix.



(a) The directed graphical model depicting the assumed generating process of the data we seek to explain. Additionally, we assume individual z_y dimensions to control individual features in x, which is approximated by the method described in Subsection 5.2.2.



(b) The main components of the representation model, equivalent to the disentangled VAE as described in Section 4.3.

Figure 5.5: An overview of the representation model.

5.3 The Explanation Method

In this section, we describe and motivate the approach to generating explanations, using the models described in Section 5.2. We generate an explanation according to an input-datapoint and a counterfactual class. If this counterfactual class is not supplied, we select the 2^{nd} most probable class according to classification component CY. Generating the explanation considers the class-specific space (z_y[1]). First, we use this space to identify an exemplar-datapoint. Then, we generate an interpolation from the input-datapoint to a datapoint depicting the class-features of the exemplar. The interpolation-path is identified such that it depicts realistic intermediate states and correct changes. In Subsection 5.3.1, we describe the approach to identifying the exemplar of the foil-class we seek to contrast our to-be-explained fact datapoint with. In Subsection 5.3.2, we describe the method for generating the interpolation between the chosen datapoint and the identified exemplar, producing the desired explanation.

5.3.1 Identifying the Exemplar

The identification of an exemplar is based on the assumptions outlining what a good exemplar is, with regards to explaining a classification process. We pose that ideally, one should select the exemplar according to two criteria: The representativeness of the exemplar (with respect to its class) and the semantic similarity of the exemplar to the fact datapoint. In this subsection, we argue these principles and describe how we approach them in our method.

Representativeness can be defined as the degree to which the datapoint's features are similar to its parent population and how well the datapoint reflects the underlying generating process[62]. As such, explaining using a more representative datapoint provides us with an explanation better reflecting the class-features. In an attempt to capture this property, we use the z_y-classifier, CY, as a proxy: We select the exemplar by setting a threshold on the probability estimate of the counterfactual class. The underlying assumption is that datapoints with a low probability estimate do not explicitly contain the class-relevant features, according to the explanation-model's perception (or alternatively, the datapoint is an outlier). As such, explanations incorporating these low-probability datapoints likely do not give an accurate representation of the relevant features we wish to highlight. Additionally, when provided access to class labels, we discard datapoints where the inferred class and the true class are not equivalent, as using the model to explain a datapoint it did not 'understand' does not seem promising.

The second criterion considers the semantic similarity of the fact-datapoint to the exemplar. We argue the value of this property as the difference to a more similar foil-datapoint likely better outlines the relevant class-distinguishing features (as there are fewer [irrelevant] differences). Additionally, as fewer feature-differences have to be conveyed, the resulting explanation is likely shorter. To evaluate this similarity, we consider the distance in the semantically meaningful space as a proxy-measure: We select the datapoint with the minimal Euclidean distance in z_y (that meets the aforementioned representativeness criterion) as our exemplar.

Another approach to selecting an exemplar considers selecting the exemplar resulting in the most desirable explanation. However, as such an approach likely constitutes generating all explanations and comparing them (which is also a challenge on its own), it rules out the use of computationally-expensive explanation-generation methods. As such, we do not consider it.

5.3.2 Traversing the Latent Space

When interpolating in a high-dimensional space (which is translated back to data-space using a complex model), the path we take has a significant effect on the intermediate states' quality and

[1]When generating explanations, we use the inferred mean rather than a random sample from the inferred distribution. As such, we use z_y to denote the inferred mean in this section.

their coherence. Finding a desirable path is non-trivial[20, 171]; there is no straightforward approach that necessarily gives an optimal interpolation-path. As such, we propose an approach for traversing this space aimed at maximizing the quality of the intermediate states and steps while minimizing the total length. In this section, we describe this approach in detail.

The explanation aims to convey the class-distinguishing features between datapoint a and exemplar datapoint b. As such, each step of the explanation aims to depict the change of one feature that differs between a and b. We assume that this feature-difference resides in the difference between z_{y_a} and z_{y_b}. As such, we keep the remaining dimensions fixed to z_{x_a}, while switching the remaining dimension-values from z_{y_a} to z_{y_b}. For each interpolation step, we allow multiple such dimension-values to be switched, as there is no guarantee that every dimension-difference depicts a feature-change (i.e., small differences are likely—but not necessarily—meaningless). Finally, we consider all orders of changing (groups of) dimensions; as we do not assume individual dimensions to be fully disentangled, the order we follow has a significant effect on the quality and coherence of the intermediate samples.

An interpolation-explanation is defined as the sequence of images generated (using DEC) from the latent-interpolation denoted as (z_y, z_x). z_y is the interpolation state starting at z_{y_a} and ending at z_{y_b}, consisting of some combination of their dimension values, i.e., $\sum_i^n (z_{y_i} \in \{z_{y_{a_i}}, z_{y_{b_i}}\})$. z_x is fixed to z_{x_a}. Each step in the interpolation then considers changing (at least) one dimension-value in z_y until $z_{y_a} = z_{y_b}$. We do not consider exploring all paths, as the number of sequences following this description becomes unreasonably large rather quickly[2]. Rather, we construct the explanation by finding the path that maximizes the per-state realism and the per-step change quality. A disadvantage of this approach is that we do not consider the coherence on a larger scale, i.e., many steps apart.

To find the optimal explanation following these properties, we construct a graph that embodies all paths, weighted according to the cost of every interpolation-step. By computing the shortest path in this graph, we find the path that optimizes the selected properties. We denote this graph as the interpolation graph, where each node considers the state of z_y, and each node is connected to the states with a number of dimension-values changed from $z_{y_{a_i}}$ to $z_{y_{b_i}}$ (while not changing any back). An example of the interpolation graph for $n = 3$ dimensions is depicted in Figure 5.6.

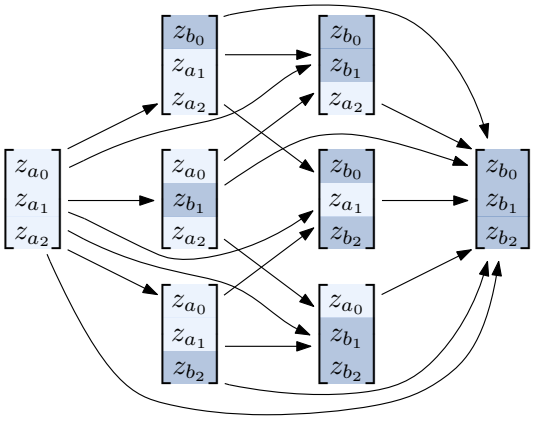


Figure 5.6: The interpolation graph when considering the transition between two latent variables of size 3, z_a and z_b (edge weights omitted for clarity).

[2]The exact number of such paths, given n latent dimensions, is equal to the n^{th} Ordered Bell number[100], which denotes the number of weak orderings of a set. A weak ordering indicates an order where multiple elements may share a place in the order, which is equivariant to the ordered dimension-wise interpolation in which multiple dimensions can be switched at once. This number, denoted as $a(n)$, does not follow a simple equation. Rather, we note that $n! < a(n) < (n+1)^n$ (for $n > 1$)[178] to illustrate the size-complexity.

Edges are weighted according to the cost of choosing this node in the interpolation, which is computed as the (weighted) sum of the change-quality between the source and target state of the edge and the realism of the target state. These values are computed by inferring a sample according to each node's state (using DEC) and inferring the probability of the change being desirable and the datapoint being real, using CD and D, respectively. As we seek to minimize the overall cost, we take the complement of these probabilities for weighting the edges. Additionally, we adjust the total edge-weight according to the number of dimensions changed in order to avoid sacrificing too much explanation-quality for simpler explanations.

The main bottleneck of finding an explanation lies in constructing the graph, as we must synthesize a datapoint for each node, infer the probability of the datapoint being fake for each node, and infer the probability of the change being bad for each edge. All these operations are non-trivial, i.e., they require data to be propagated through different model components. The number of nodes, $|V|$, is 2^n, and the number of edges, $|E|$, is $3^n - 2^n$ (we refer to Appendix B for the derivation). Accordingly, this method does not scale to arbitrarily high-dimensional z_y-spaces. Given that all edge weights are computed, the shortest path can be found in linear time. As the graph is directed and acyclic (as we only change dimensions from z_{y_a} to z_{y_b}), we can compute the shortest path by topologically sorting the vertices and iterating over them in the sorted order, which has a time complexity of $\Theta(|V| + |E|)$[156].

To conclude, we infer the class-probabilities of datapoints a and b in order to place the identified interpolation (the shortest path in the interpolation graph) in context. The resulting explanation then consists of the class predictions (and the associated probability estimates) of datapoint a and exemplar b, alongside the translation from a to b depicting a single feature change at a time. For a step-by-step description of the entire explanation-generation process, we refer to Algorithm C.5 in the appendix.

5.4 The Explanation-Quality Evaluation Method

In this section, we describe and motivate an approach for quantitatively evaluating the explanation-ability of a method creating interpolation-based contrastive explanations. This approach is based on aligning generated explanations with ground-truth explanations. Consequently, we refer to it as the explanation alignment cost (*eac*). We introduce this approach as we are not aware of another method for quantitatively evaluating the explanation-quality according to the specific explanation definition and structure we assume. As quantifying the explanation-quality is inherently biased with respect to one's assumptions of what a good explanation is, we propose an evaluation-method aligning with the goal we seek to reach.

The *eac* considers evaluating the quality of an explanation, given that we explain from datapoint a towards b. As such, it does not evaluate the quality of the selected exemplar, only that of the resulting explanation. Furthermore, as these explanations attempt to model some ground-truth generative factors (features) in data, we necessarily require access to such a process in order to evaluate an explanation's correctness. The to-be-evaluated method is trained on a synthetic dataset generated by a known process and evaluated accordingly.

To evaluate an explanation from datapoint a to b, we identify an optimal alignment with respect to some ground-truth explanation of the same contrasting process. That is, every state in the generated explanation has to be mapped to (at least) one state in a ground-truth explanation, and vice versa. We constrain this mapping such that both aligned sequences are increasing (if state x is mapped to state y in the other sequence, no state after x can be mapped to a state before y, for any arbitrary x and y). To find this alignment we consider the Dynamic Time Warping (DTW) algorithm[142], which computes an optimal alignment meeting these requirements in $O(nm)$ time (where n and m denote the length of each explanation). To evaluate the cost of each

individual state-to-state mapping, a cost function for the difference between two datapoints must be selected, e.g., the (per-pixel) squared error. The total explanation cost can then be quantified as the sum of these costs, i.e., the alignment cost.

Generating ground-truth explanations is done according to the true generating process. By interpolating from a to b while changing only one class-specific generating factor at each step and changing no more factors than necessary (e.g., changing the same factor repeatedly or changing non-class specific factors), we create an explanation perfectly coinciding with our sketched goal. An issue to overcome is that no preference of order (of the dimensions we change) is assumed. As such, any ground-truth explanation (changing dimensions in arbitrary order) is equally valid. Consequently, we must evaluate our generated explanation against all these orders when computing its true deviation from a perfect explanation. A significant downside of this fact is that there exist $m!$ such orders, given m differences in the generating features. Even though m generally is significantly smaller than the data's dimensionality, it still puts a significant bound on the complexity of the process we seek to explain and evaluate.

5.4.1 Computing the Explanation Alignment Cost

To summarize, the evaluation process works as follows. First, the to-be-evaluated method is trained according to a synthetic dataset with a known generating process. To then evaluate its explanation-generation quality, we generate datapoints a and b according to two sets of (class-specific) generating factors. We use these datapoints as input for the method we evaluate, i.e., we generate an interpolation-explanation of the difference between a and b. Next, we evaluate this explanation. We generate all possible ground-truth explanations using the ground-truth generating factors, and compute the alignment cost between the generated explanation and the true explanations. The smallest alignment cost to any such ground-truth explanation represents the explanation cost: The lower this cost, the higher the explanation-quality. This explanation-evaluation process is repeated for a large number of generating-factor pairs in order to quantify the explanation-generation capability with respect to a significant proportion of the data.

Chapter 6

Experimental Setup

In this chapter, we describe the setup for an introductory evaluation of VAE-CE. This experiment is based on the subquestions of explanation- and representation-quality we wish to evaluate. We evaluate these questions using synthetic data and the MNIST database[81]. The questions, alongside the evaluations we conduct to answer them, are as follows:

SQ 5.1 *What is the quality of the generated explanations in terms of conveying class-distinguishing features and selecting suitable exemplars?*
We generate explanations using VAE-CE and discuss whether they depict a single factor-change per interpolation-step. Additionally, we evaluate whether VAE-CE selects exemplars containing the most common factors and whether these exemplars are likely to be correctly classified.

SQ 5.2 *How does the explanation-quality of our method compare to that of similar approaches?*
We consider alternative approaches to disentangling individual features in a VAE's latent space. These approaches are incorporated in the baseline class-disentangled DVAE (Chapter 4) to disentangle z_y's dimensions. As the type of supervision differs, only a subset of these methods is applied to MNIST. The comparison considers the explanation alignment score (*eac*) for the synthetic data. We qualitatively compare the generated explanations for both datasets.

SQ 5.3 *How do our method's subcomponents contribute to the explanation-generation process?*
We evaluate the effect of generating explanations without the explanation generator and without the pair-based dimension conditioning. We compare the difference in *eac* for the synthetic data.

SQ 5.4 *How does the representation-quality of our method compare to that of similar approaches, and how does this quality relate to the explanation-quality?*
We consider metrics quantifying the representation-quality of all conditioning-approaches we compare. These metrics are the KL divergence (*kl*), the model-accuracy (*acc*), the accuracy of a logistic regression qualifier trained on the latent embeddings (*l-acc*), and the mutual information gap (*mig*). As the last metric requires access to the ground-truth generating factors, it is only evaluated for the synthetic dataset. Additionally, we conduct qualitative analysis (using reconstructed and synthesized data) to support these metric-evaluations.

In the remainder of this chapter, we discuss the datasets, evaluations, and models in detail. In Section 6.1, we describe the datasets (and the respective augmentations) we consider. In Section 6.2, we describe the considered metrics and evaluations. Finally, in Section 6.3, we discuss the methods we compare, alongside their implementations.

6.1 Datasets

We consider two datasets for the evaluation of our method. We design a synthetic dataset in order to have complete control over the generating process, in order to objectively evaluate the learned representations and generated explanations. Additionally, we consider the MNIST database of handwritten digits[81], to evaluate our method in a more realistic setting not influenced by our assumptions about data. These datasets share various characteristics: They consider the same number of classes, the same image resolution, and we assume the same underlying feature-shape.

6.1.1 Synthetic Data

The synthetic dataset is constructed according to the general process we seek to model. That is, each datapoint is determined by class-relevant and class-irrelevant factors. We assume that the class-relevant factors altogether are determined by the class, and individually follow a presupposed structure. This structure is defined as the occurrence of lines, where each line is determined by its orientation, length, and relative position in the image. We assume eight (discrete) variables determining whether a specific line occurs in the data or not. These lines are depicted in Figure 6.1.

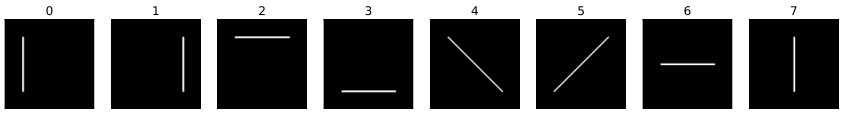


Figure 6.1: The underlying factors determining the synthetic datapoints' class. These factors (lines) are determined by their orientation, length, and position in an image.

Using these factors we create ten classes, where each class consists of some combination of lines. We ensure that no lines always co-occur to ensure that each factor has individual meaning (i.e., two lines always co-occurring are not meaningful individually with respect to the class). Furthermore, we introduce one class with two varying compositions (split 50/50), in order to evaluate the identification of the exemplars (i.e., the variant with the most shared attributes should be chosen). These classes, alongside the lines they consist of, are depicted in Figure 6.2.

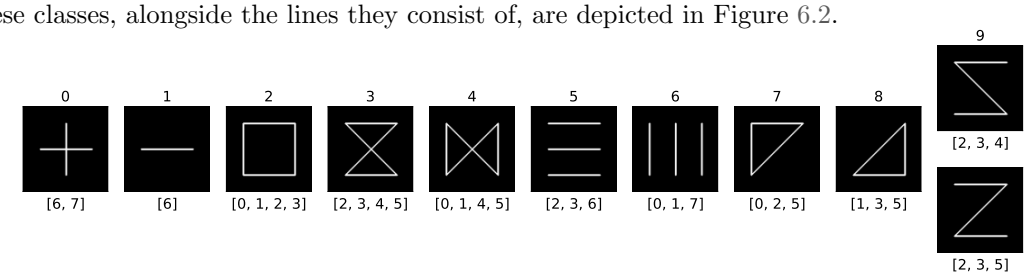


Figure 6.2: The ten classes in the synthetic dataset. The value above depicts the class index, whereas the value below depicts the indices of the lines constructing the class.

Datapoints are generated by taking these class-specific factors as a base shape and adding non-trivial noise. The noise-process seeks to mimic that of handwritten shapes (such as MNIST-digits) and consists of two components: Distortion (as shapes are often distorted when drawn) and line-width variation (as line-thickness in hand-drawn shapes is not uniform). We refer to Appendix D for a detailed description of this generation procedure. Examples of synthetic datapoints are depicted in Figure 6.3.

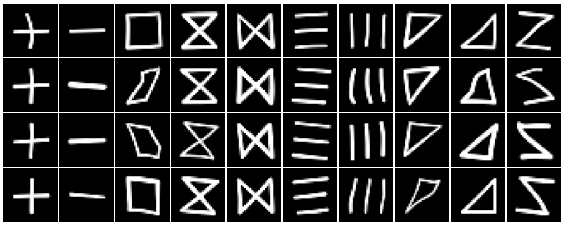


Figure 6.3: 40 synthetic datapoints, 4 examples of each class.

The training set consists of 10 000 32×32-pixel images for each class, whereas the test set consists of 1000 such images for each class (independently sampled). Model selection is done according to the explanation-quality metric (which samples directly from the generative process). Consequently, we do not consider an explicit validation set for tuning the model architecture and hyperparameters.

Change-pairs are created by generating images using two shapes at a time. These two shapes are based on classes (Figure 6.2) with some line(s) hidden, to keep the setting semi-realistic[1]. Examples of such pairs are depicted in Figure 6.6. Other feature-based supervision (used by methods we compare VAE-CE to) can be created according to knowledge of the generative process. For each type of supervision we generate the same number of samples, 100 000 (equivalent to the number of training samples).

(a) Positive change-pairs. Each pair depicts a single line-change.

(b) Negative change-pairs. Each pair depicts either no or multiple line-changes.

Figure 6.4: Examples of synthetic change-pairs. Each pair indicates either the change of a single generating factor (positive) or no/multiple such changes (negative).

6.1.2 MNIST

The MNIST database of handwritten digits[81] consists of real-world examples of handwritten digits, collected from American Census Bureau employees and American high school students. These digit-samples are size-normalized and centered in fixed-size grayscale images of 28×28 pixels and are labeled according to their class (i.e., a value from 0-9). For ease of implementation[2], we pad these images to 32×32 by adding black pixels around the borders. Examples of the MNIST dataset are depicted in Figure 6.5.

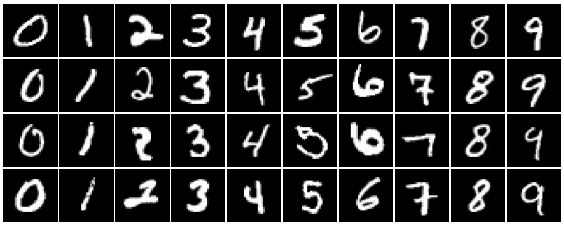


Figure 6.5: 40 samples from the MNIST dataset, 4 examples of each class.

The training set consists of 60 000 (32×32-padded) images in total, around 6000 images for each class ($\pm$ a few hundred). The test set consists of 10 000 images, again relatively balanced (about 1000 images per class). The images from the train and test set come from different writers. Note that model selection is done according to external factors: We do not consider an explicit validation dataset for determining the model architecture and hyperparameters.

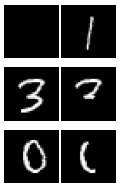
To create change-pairs exhibiting (un)desirable changes, we augment digit-samples according to our presupposed notion of what a relevant factor is, that is, a continuous line. This process considers reducing images to individual lines and clustering pixels according to these lines. A detailed description of this augmentation process is provided in Appendix E. Using this line-split, we create pairs that either exhibit a single line-change (positive pairs) or no/multiple changes (negative pairs) by hiding sets of lines in a single image (or pairing different images for the negative pairs).

[1]The assumption is that when modeling real-world data, the change-pairs likely consist of (augmentations of) real datapoints rather than novel datapoints following the generating process.

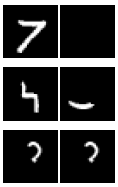
[2]Creating networks consisting of both convolutional (the encoder) and transposed convolutional (the decoder) layers is easier if we work with dimensions that are a power of two, as we can halve such a dimensionality repeatedly.

As this augmentation process is only a rough approximation of what we presume the true generating factors to be, the resulting pairs are rather noisy. Examples of change-pairs are depicted in Figure 6.6. For more change-pair examples, we refer to Appendix F. Again, we create as many augmented samples as there are real samples; 60 000 in this case.

Creating a labeling of all lines (i.e., inferring the entire generating process) is a more challenging task than augmenting individual images to create change-pairs. As such, we do not compare VAE-CE to methods requiring 'full' supervision when evaluating the model-performance on MNIST, as we cannot (easily) generate such supervision.



(a) Positive change-pairs. Each pair depicts a single line-change.



(b) Negative change-pairs. Each pair depicts either no or multiple line-changes.

Figure 6.6: Examples of MNIST change-pairs. Each pair indicates either the change of a single generating factor (positive) or no/multiple such changes (negative).

6.2 Considered Evaluations

In this section, we provide an overview of the evaluations we make. We describe the metrics we consider for quantitative analysis, alongside the approaches we employ for qualitative analysis. We motivate the use of these metrics and approaches with respect to the overarching questions of representation- and explanation-quality we seek to answer. Note that all these analyses are conducted with respect to (test) data that has not been used in either the training process (parameter optimization) or the model-selection process (hyperparameter selection).

6.2.1 Quantitative Evaluation

Explanation alignment cost (*eac*) considers our method to evaluating the explanation quality, as described in Section 5.4. We quantify the quality of a generated explanation according to its alignment cost to a ground-truth explanation. The cost function (for mapping states) we adhere to is the (per-pixel) squared error with a small added value (to ensure that in the case of a perfect match, we promote the shortest alignment). This cost is denoted as follows:

$$cost = (x_a + x_b)^2 + \epsilon \tag{6.1}$$

where x_a denotes a state of a ground-truth explanation, x_b denotes the mapped state of the generated explanation, and $\epsilon = 0.001$. As computing the *eac* relies on access to the ground-truth generating factors, we only evaluate it using the synthetic dataset. We generate 90 (a, b) pairs of images (one for each class combination, both ways), where a denotes the starting point of our explanation and b denotes the exemplar we explain towards. Note that a and b can differ in non-class specific factors as well. The final *eac* denotes the average alignment cost of all interpolations.

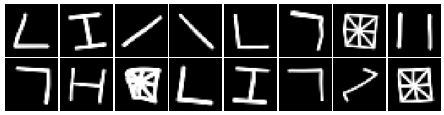
Exemplar quality denotes a small experiment conducted on the synthetic dataset, considering exemplars of class 9. Class 9 consists of factors [2, 3, 4] or [2, 3, 5]: factor 4 and 5 differ. Classes 7 and 8 both contain factor 5 but not factor 4, whereas the remaining classes include either both factors 4 and 5, or neither. We evaluate whether datapoints of class 7 and 8 are assigned to exemplars of class 9 with factor 5 (rather than 4) significantly more often than the other classes are.

The aim of this experiment is to evaluate whether exemplars generally are selected according to the highest number of common factors. This experiment is conducted on a set of 2000 test-datapoints.

Reconstruction error (*rec*) is used as a proxy of the data likelihood, how well the model captures the distribution of the dataset. This metric is evaluated to quantify the (negative) impact of the added regularization terms. *rec* is evaluated by reconstructing datapoints using mean latent-representations and taking the squared error of these reconstructions:

$$rec = (x - \widetilde{x})^2 \tag{6.2}$$

where x and $\widetilde{x}$ denote the input sample and its reconstruction, respectively. The total *rec* represents the averaged error over the entire test set. Additionally, we evaluate this metric with respect to different datasets with (assumed) shared underlying factors, to get an idea of how well the representations generalize to similar problem-domains. In the case of the synthetic data-model, we consider shapes consisting of unseen combinations of factors. In the case of the MNIST-model, we consider EMNIST Letters[22], a dataset extending MNIST so as to also include the Latin alphabet. EMNIST-samples follow the same preprocessing steps as MNIST and originate from the same source. Examples of these auxiliary test-sets are depicted in Figure 6.7.



(a) Unseen combinations of synthetic generating factors.

(b) EMNIST Letters.

Figure 6.7: Datasets with similar generating processes to the synthetic data (a) and to MNIST (b).

KL divergence (*kl*) is used to evaluate the general structure of the latent distribution. A larger divergence indicates that the latent distribution differs more from the prior distribution, i.e., the latent distribution is taking some arbitrary shape. Such an arbitrary shape leads to difficulty when sampling or interpolating in the latent space; we are more likely to traverse meaningless regions since we cannot assume it is shaped similar to the prior distribution. As we assume the prior distribution to be a standard factorized normal distribution ($\mathcal{N}(0, I)$), we compute the KL divergence analytically as follows:

$$kl = \frac{1}{2}(z_\mu^2 + z_\sigma^2 - \log z_\sigma^2 - 1) \tag{6.3}$$

where z denotes the latent distribution we evaluate. We sum and average this error over the entire test set. *kl* is evaluated separately for both the class-relevant and class-irrelevant latent subspaces.

Model-accuracy (*acc*) denotes the accuracy of the z_y classification component: The number of correctly classified samples divided by the total number of samples (with respect to class-labels). We evaluate this metric to both evaluate the utility of a model as a classifier and to quantify the likelihood that explanations consider correctly classified datapoints (and consequently convey the true class-distinguishing factors). To quantify the latter better, we also compute the accuracy when only considering datapoints the model is confident about, that is, datapoints with a class probability greater than $t = 0.95$ (as used when identifying an exemplar). We report both the accuracy on this subset and the proportion of data that meets this criterion.

Latent-space logistic regression classifier accuracy (*l-acc*) considers the accuracy of a logistic regression classifier (with respect to class-labels) on the latent space-embeddings. This metric is used to evaluate how much class-information is embedded in both the latent space subdivisions (z_x and z_y). We train a Scikit-learn logistic regression classifier[116] on the embeddings

of the test-data in both latent spaces and evaluate their respective classification accuracies.

Mutual information gap (*mig*)[18] is a metric that seeks to quantify the disentanglement of individual latent dimensions, given access to the ground-truth generating factors. This measure rests on the principle of mutual information of random variables[145]. The extent of disentanglement is measured by computing the gap between the mutual information of a ground-truth factor and the latent factors with the highest and second-highest mutual information (for each ground-truth factor), normalized by the ground-truth factor's entropy. This measure is denoted as follows:

$$mig = \frac{1}{K} \sum_{k}^{K} \frac{1}{H(v_k)} \Big(I(z_{j^{(k)}}; v_k) - \max_{j \neq j^{(k)}} I(z_j; v_k) \Big) \tag{6.4}$$

where K denotes the number of ground-truth factors, H the entropy of a variable, I the mutual information between two variables, v the ground-truth factors, z the latent factors, and $j^{(k)}$ the index of the latent variable with the maximum mutual information with ground-truth factor v_k. We estimate the *mig* following the same procedure as [86]: We encode our test set to the latent representation we seek to evaluate (z_y—the class-relevant factors), of which we consider the mean representation (i.e., we do not sample a value). Each (latent) dimension of this representation is binned into 20 bins, from which we compute the discrete mutual information with the ground-truth factors.

We evaluate a disentanglement measure as a more disentangled space implies a better separation of ground-truth generating factors. As we seek to model these factors in order to generate explanations, better disentanglement is (likely) highly correlated with good explanation. As this metric depends on access to the ground-truth generating factors, we evaluate the *mig* only on the synthetic dataset.

6.2.2 Qualitative Evaluation

The qualitative evaluation considers generating data according to different objectives and discussing how well the different methods stack up in fulfilling the objectives. These evaluations are as follows:

- **Explanation generation**: We generate explanations both according to the explanation-generation process (starting with a single datapoint) and according to input pairs of datapoints. We evaluate the explanation-quality with respect to our presupposed notion of good explanation and compare different representation-models and interpolation-methods. Additionally, we discuss the quality of the identified exemplars: We generate explanations where the target class has distinct samples and evaluate whether VAE-CE identified an exemplar that aligns well with the source-datapoint of our explanation.

- **Data reconstruction**: We reconstruct samples and discuss the observed differences between the original and reconstructed samples, to evaluate the facets of sample-generation the methods differ in.

- **Data generation**: We generate datapoints by sampling (subcomponents of) the latent variable from the prior. The purpose is to evaluate the generation ability of the method as a whole and to evaluate the extent of disentanglement between the different latent subspaces (z_x and z_y).

- **Latent-space traversal**: We encode datapoints and traverse the values of individual dimensions, and synthesize output-data accordingly. The objective is to evaluate the representation-quality of individual dimensions of z_y with respect to the desired dimension-wise structure (individual line changes).

6.3 Comparison Overview

We compare VAE-CE to methods with similar capabilities, i.e., we stay within the domain of VAE-based representation methods. As such, we restrict the focus of our experiments to evaluating the explanation- and representation-quality with respect to our predefined goals, rather than evaluating these assumptions themselves (e.g., by conducting a user experiment with various types of explanations). First, a short overview of all methods we compare is given, after which the implementation of these methods (the model selection) is described. Finally, we discuss how we generate explanations with these methods.

6.3.1 Other Feature-Disentanglement Methods

We use the (class-)disentangled VAE (Section 4.3), denoted as DVAE, as the baseline representation model. The methods considered in this evaluation are combined with this model, further conditioning the class-relevant latent space, z_y. We do not consider the methods in isolation as it would not be a fair comparison: Without a split between class-relevant and class-irrelevant factors, we cannot expect any other method to perform as well, as there is no (reliable) method for interpolating between only class-relevant factors. The methods we include in this evaluation all build upon the principle of disentangling (individual) factors using additional supervision. First, we provide a summarization of these methods and their assumed supervision and denote how we refer to them[3].

DVAE (Disentangled-VAE) considers the (class-)disentangled VAE, the baseline of our method. The only assumed supervision is that of class-labels. We denote its loss function as follows (we refer to Section 4.3 for details):

$$\mathcal{L}_{DVAE} = \mathcal{L}_{rec} + \beta_0 \mathcal{L}_{kl_0} + \beta_1 \mathcal{L}_{kl_1} + \alpha \mathcal{L}_{classy} + \gamma \mathcal{L}_{advx} + \mathcal{L}_{classx} \tag{6.5}$$

LVAE (Label-VAE) considers disentangling individual factors according to a labeling of all feature-values, following the principle of [16, 174, 56, 29]. As such, each datapoint is paired with its feature-values. To disentangle these features, the same label-disentanglement principles as described in Section 4.3 are applied. For each ground-truth feature and its assigned latent dimension, a classifier-pair is added. One classifier learns to predict this feature from the specified dimension, whereas the other classifier is used to encourage all remaining dimensions in z_y not to contain information about the feature. Thus, in the case of eight generating features (the synthetic dataset), we add eight classifier and adversarial classifier pairs. As this type of supervision assumes knowledge of all feature-values, we only consider LVAE when evaluating the synthetic data. The added objectives are those of the extra classifiers. We denote the loss function of LVAE as follows:

$$\mathcal{L}_{LVAE} = \mathcal{L}_{DVAE} + \sum_{i}^{N} \left(\alpha_f \mathcal{L}_{class_i} + \gamma_f \sum_{i} {}_{adv_{ic}} + \mathcal{L}_{class_{ic}} \right) \tag{6.6}$$

where N denotes the number of features we disentangle (for simplicity, we assume this to be equal to the number of latent dimensions in z_y). $\mathcal{L}_{class_i}$ denotes the classification-loss concerning feature i. $\mathcal{L}_{adv_{ic}}$ considers the adversarial loss of the complementing classifier of i, i.e., the classifier attached to all other dimensions in z_y. $\mathcal{L}_{class_{ic}}$ denotes the term training this complementary classifier. We introduce additional hyperparameters α_f and γ_f in order to balance the feature-disentanglement term against the remaining terms.

GVAE (Group-VAE)[55] considers disentangling features by grouping images with common feature-values together. In order to ensure that we have a balanced distribution of common-feature pairs while also having a balanced distribution of class-labels, we apply this principle in a separate training pass. As such, training GVAE considers alternating between the standard DVAE procedure

[3] Even though we adapt all methods, we do not add a prefix (D) to their names so as to not clutter them. Rather, we adhere to the names (if provided) of the regular (non-class-disentangled) versions of these methods.

and the group-based disentanglement procedure. The latter considers, for a group-specific latent dimension, averaging all inferred parameters for the specified shared dimension. By minimizing the $ELBO$ under this constraint, the averaged (shared) dimensions are encouraged to contain the group-specific information, as this information is necessarily shared (through averaging) in the training pass. As such, this method does not introduce additional hyperparameters. Since the dimension-assignment of a group requires knowledge of the full generative process, we only consider GVAE when evaluating synthetic data. The loss is denoted as follows:

$$\mathcal{L}_{GVAE} = \mathcal{L}_{DVAE} \tag{6.7}$$

ADA-GVAE (Adaptive-Group-VAE)[87] considers the same dimension-averaging procedure as GVAE but identifies the common features heuristically. Again, we consider two alternating optimization passes: Optimizing the DVAE objective and optimizing the $ELBO$ while averaging the shared dimensions. To better compare this method to VAE-CE we consider a special case of ADA-GVAE, where we assume that all but one feature are shared: A one feature difference, which is equivalent to a positive change-pair. When considering only a single differing dimension, their common-feature identification process reduces to selecting the dimension with the highest KL divergence (between the two paired datapoints' embeddings) as the differing feature and averaging the remaining latent-parameters. As no auxiliary training objectives are added, we denote the loss as follows:

$$\mathcal{L}_{ADA\text{-}GVAE} = \mathcal{L}_{DVAE} \tag{6.8}$$

VAE-CE (VAE-based Contrastive Explanation) denotes our proposed method, as described in Chapter 5. The assumed supervision is that of datapoints and their class-labels to train the main VAE, and access to positive and negative change-pairs in order to (pre-train) CD. The loss function for training the main representation model is denoted as follows (we refer to Section 5.2.2 for details):

$$\mathcal{L}_{VAE\text{-}CE} = \mathcal{L}_{DVAE} + \alpha_r \mathcal{L}_{REAL} + \alpha_p \mathcal{L}_{PAIR} + \mathcal{L}_D \tag{6.9}$$

6.3.2 Model Implementations

To start, we note that the aim of this study is not to find the perfect configuration and architecture for each method. Rather, we seek to apply all methods such that they are (equally) effective, in order to provide a fair comparison between them. All models are constructed and trained following the same procedure and are optimized to best generate explanations. The model architectures are based on (simple) MNIST-VAE architectures found in prior works (e.g. [161, 123]) and use a dimensionality of 8 for both z_x and z_y. The implementation is done using TensorFlow[1]. We refer to Appendix G for details on all model architectures.

The objective function of each method consists of a sizeable set of sub-objectives, of which many do not necessarily cooperate. As such, bad hyperparameter configurations can lead to solutions where objectives are disregarded. To find suitable hyperparameters, we first identify a non-degenerate solution, which we define as a solution where none of the objectives are ignored. This process considers changing the minimum number of hyperparameters from a shared base-configuration until no objectives are ignored (i.e., no collapsed latent spaces, uninformative classifiers, or all-zero outputs).

Following the starting solution, we search for hyperparameters configurations near this solution. The purpose of this search is to find well-performing configurations for each method. Each model is trained on $\approx 2\,000\,000$ datapoints, and we train four models per configuration (to exclude the stochastic influences apparent when training NNs). We use the configuration-statistics (μ and σ) of each metric-evaluation in order to draw (statistically significant) conclusions about the model performance.

The final model configurations are selected by the minimum explanation alignment cost on a validation set of explanation-pairs. As the *eac* cannot be evaluated for MNIST, we use the hyperparameters identified for the synthetic data when training models on MNIST. Since the synthetic data is similar to MNIST in its general structure (e.g., the type of features, the number of classes), transferring the hyperparameters resulted in sensible models. For a detailed overview of the hyperparameter selection process, alongside minor implementation details, we refer to Appendix G.

6.3.3 Explanation Generation Methods

To evaluate the explanation-quality of all methods, we define two 'naïve' methods of interpolation that can easily be applied to all methods. Additionally, it allows us to quantify the utility of the graph-based explanation generation of VAE-CE. For some datapoint-pair a and b, we change its class-relevant factors z_{y_a} to those of z_{y_b}, while keeping z_{x_a} fixed (similar to our predefined method). The two additional interpolation methods we consider are as follows:

- **Smooth interpolation**: We consider all immediate states of z_y as a convex combination of z_{y_a} and z_{y_b}. All dimensions are adjusted at once, according to a predefined number of steps, in equal proportion for each step. For our experiments, we use five interpolation-states.
- **Dimension-wise interpolation**: We identify all significantly different dimensions of z_y heuristically: If $|z_{y_{a_i}} - z_{y_{b_i}}| > 1$ (the σ of our prior), the dimension is significant and must change separately. All remaining dimensions are changed from z_{y_a} to z_{y_b} at once. The interpolation is constructed by changing all significantly different dimensions from z_{y_a} to z_{y_b} in arbitrary order. The non-significant dimensions are all changed at once (at the same time as the first significant dimension, in the first interpolation-step).

For VAE-CE's explanation generation, we set the explanation parameters to $t = 0.95$ (exemplar threshold), $\alpha = 0.5$ (realism weight), $\beta = 1$ (change-quality weight), and $\gamma = 1$ (normalization exponent). When identifying exemplars, we rely on the model-classifier and do not use additional class-labels.

Chapter 7

Results and Analysis

In this chapter, we provide the results of the experiments described in Chapter 6, alongside a discussion of these results' implications. For both datasets, we first discuss the explanation-quality, after which we discuss the representation-quality and the relation between these evaluations.

For the quantitative measures, we report their mean and (sample) standard deviation ($n = 4$ models per type). When considering the statistical significance of a measure between two models, we employ Welch's t-test[1][164] with a significance level of $\alpha = 0.05$. For the qualitative analysis, we select a single model per model type. This selection is done according to the minimum *eac* in the case of synthetic data and the minimum *rec-gen* ratio in the case of MNIST. In Section 7.1, we consider the synthetic dataset, whereas in Section 7.2, we consider the MNIST database. Finally, in Section 7.3, we report conclusions and provide discussion about the experimental results.

Note that these results do not consider an evaluation of the methods in isolation. Model-selection (hyperparameter tuning) is done according to the *eac* (see Appendix G for details). While the explanation-quality analysis is closely aligned with this objective, the representation-quality evaluations have to be considered through the lens of using the method when optimized for explanation-generation.

7.1 Synthetic Data

7.1.1 Explanation-Quality

Quantitative: *eac*. We first consider the explanation alignment cost, the primary objective we wish to minimize. For all methods, we create explanations according to smooth interpolations (*eac-sm*) and dimension-wise interpolations (*eac-dim*) between two datapoints. For VAE-CE, we also generate explanations using its explanation generator component (*eac-graph*). The results of these evaluations are depicted in Table 7.1

	(Mean ± SD)		
Model	*eac-sm* ↓	*eac-dim* ↓	*eac-graph* ↓
DVAE	27.53 ± 0.293	27.66 ± 0.414	
LVAE	26.95 ± 0.452	25.38 ± 1.56	
GVAE	27.87 ± 0.603	28.90 ± 0.648	
ADA-GVAE	**26.84** ± 0.568	26.22 ± 1.09	
VAE-CE (Ours)	28.96 ± 0.824	**21.78** ± 0.545	**19.92** ± 0.635

Table 7.1: Explanation-quality results on the synthetic data, quantified by the explanation alignment cost (*eac*). We consider smooth interpolations (*eac-sm*), dimension-wise interpolations (*eac-dim*) and VAE-CE's explanation generator (*eac-graph*).

[1] A two-sample t-test hypothesizing equal means, assuming unequal variances. Furthermore, it assumes that our samples are independent and normally distributed. While the former holds (as the random seeds are independently generated), the latter is arguable, as we consider only a small sample size.

VAE-CE performs best overall, with a significant difference ($p = 0.0030$) between its cost of 19.92 and the second best cost of 25.38 (LVAE with dimension-wise interpolation). Additionally, dimension-wise interpolation using VAE-CE results in the second-lowest overall cost, hinting that both the explanation-generation method and the conditioning-approach contribute to the low score. The *eac-dim* and *eac-graph* for VAE-CE are still significantly different, however ($p = 0.0046$).

To evaluate whether VAE-CE's explanation generator is primarily responsible for the score, we also generate graph-based explanations for the other methods. We take D and CD from the best-performing VAE-CE-model and generate explanations using the other methods' representation model. The resulting scores are compared to the best-performing type of naïve interpolation. These results are depicted in Table 7.2.

	(Mean ± SD)		
Model	*min*(*eac-sm*, *eac-dim*) ↓	*eac-graph* ↓	$p(\mu_1 = \mu_2)$
DVAE	27.53 ± 0.293	26.87 ± 0.308	0.0204
LVAE	25.38 ± 1.56	23.86 ± 1.00	0.1600
GVAE	27.87 ± 0.603	26.52 ± 0.716	0.0287
ADA-GVAE	26.22 ± 1.09	23.24 ± 1.15	0.0093
VAE-CE (Ours)	**21.78** ± 0.545	**19.92** ± 0.635	0.0046

Table 7.2: A comparison of the *eac* between the best naïve interpolation (*min*(*eac-sm*, *eac-dim*)) and VAE-CE's explanation generator (*eac-graph*), alongside the significance of these differences.

For every model we observe an improvement when using the explanation generator rather than a naïve method of interpolation (although not statistically significant for LVAE). The benefit of using the explanation generator for any method ($eac = 25.756 \pm 2.39$ without it and $eac = 24.08 \pm 2.69$ with it, $n = 20$ models per group) is also significant ($p = 0.0046$). As such, the explanation-generator approach can likely be used with other conditioning methods as well.

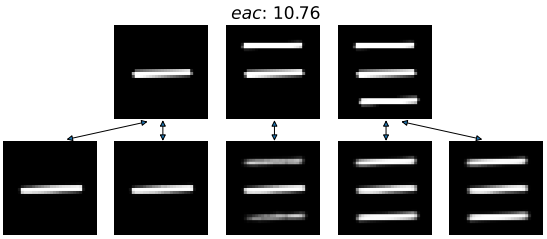
VAE-CE still outperforms all other models; the differences between VAE-CE and the two next lowest scores (23.24 for ADA-GVAE and 23.86 for LVAE) are still significant ($p = 0.0046$ and $p = 0.0011$, respectively). As such, we conclude that VAE-CE likely is better suited to explanation generation than the other methods. Furthermore, it is likely that both the conditioning method (pair-based dimension conditioning) and the explanation generator contribute to this performance.

Qualitative: *explanation generation*. To compare the explanation-quality between the methods, we sample two pairs of datapoints from the test set and generate their respective interpolations. For each method, we depict the interpolation following from the method with the lowest *eac* according to Table 7.1 (we do not consider the explanation generator for all methods, as it is only a part of VAE-CE). The input-pairs are depicted in Figure 7.1, whereas the resulting explanations are depicted in Figure 7.2. For each explanation, the top row depicts the closest ground-truth explanation, whereas the bottom row depicts the computed explanation. The lines indicate the optimal mapping between the two. Note that only class-relevant factors should be changed in the interpolation; the end state should not be fully identical to the target-image.

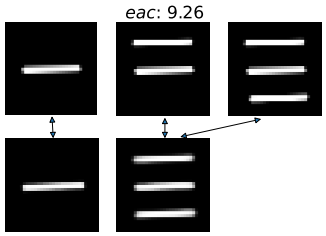
(a) An example of class 1 and of class 5.

(b) An example of class 8 and of class 3.

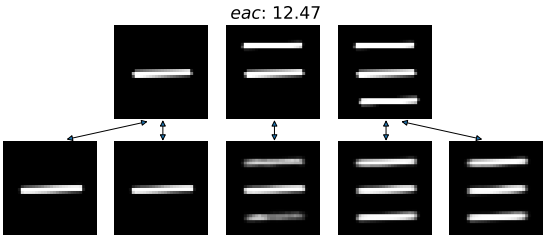
Figure 7.1: Two synthetic-data input-pairs. Explanations are generated by interpolating between these images' class-features.



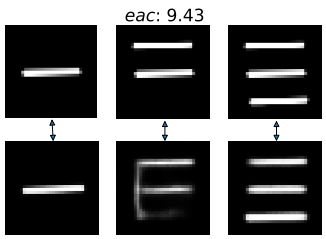
(a) DVAE



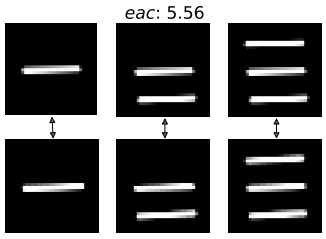
(b) LVAE



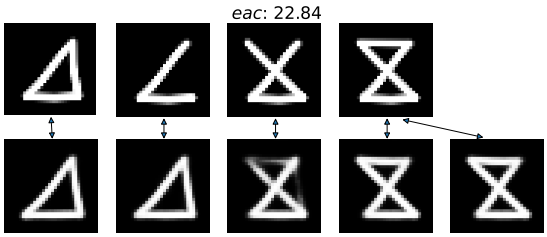
(c) GVAE



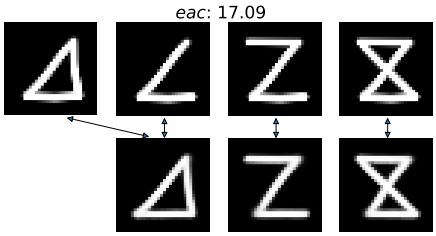
(d) ADA-GVAE



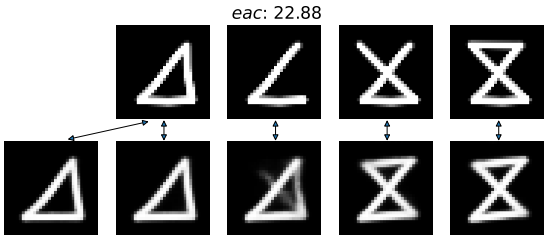
(e) VAE-CE



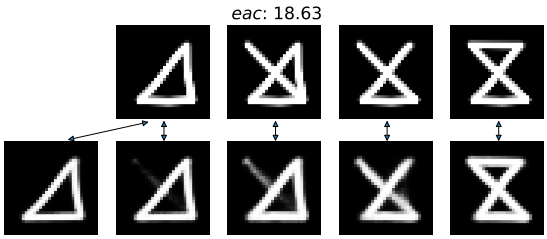
(a) DVAE



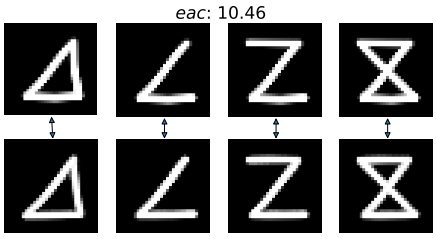
(b) LVAE



(c) GVAE



(d) ADA-GVAE



(e) VAE-CE

Figure 7.2: The generated explanations from the pairs in Figure 7.1, class 1 to 5 (left) and class 8 to 3 (right). The top row indicates the closest ground-truth explanation, whereas the bottom row depicts the created interpolation. The arrows indicate the identified mapping. The *eac* is denoted above each alignment.

For these two examples, VAE-CE is the only approach clearly conveying the generative factors one step at a time. The intermediate states of ADA-GVAE depict line-changes to an extent, although the intermediate states are somewhat blurry. The remaining approaches seem to primarily interpolate between different classes rather than individual lines.

To explore VAE-CE's explanation process in its entirety, we generate another set of explanations. We sample eight datapoints from the test set and provide them as input. The model infers the datapoints' class, the counterfactual class (the second most probable class), and the exemplar. An explanation is then generated from these variables. The input-datapoints are depicted in Figure 7.3, whereas the resulting explanations are depicted in Figure 7.5.

Figure 7.3: Eight synthetic datapoints we generate an explanation for.

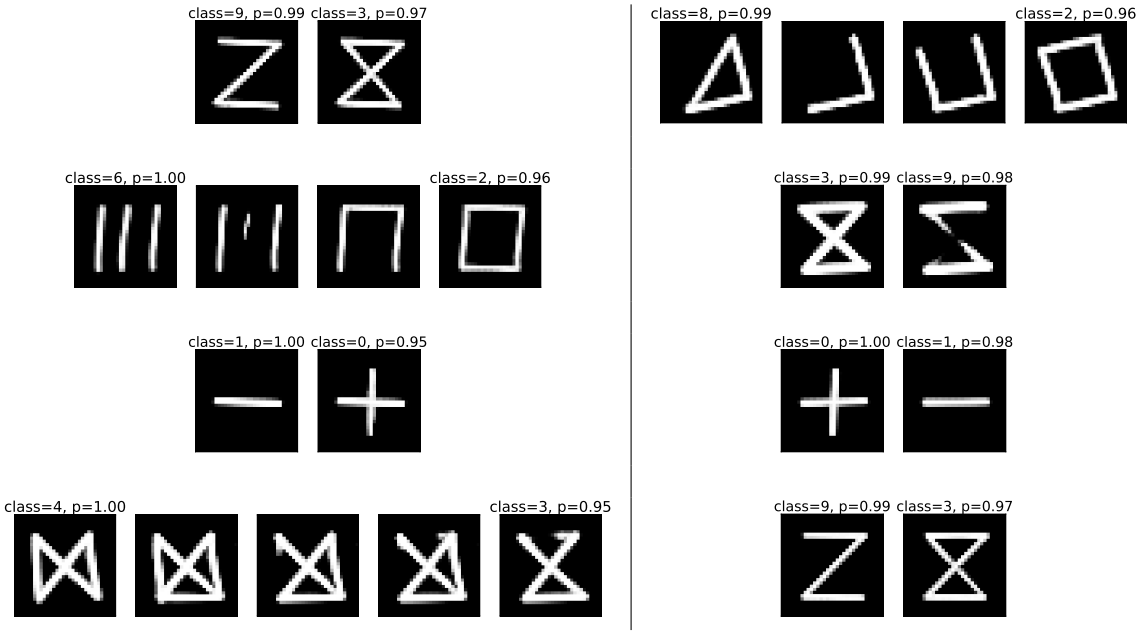


Figure 7.4: Explanations of the images depicted in Figure 7.3, generated using VAE-CE. Each explanation depicts the predicted class, the changes that must be made to classify the datapoint as another class, and the predicted counterfactual class.

Most steps indicate a single line being changed. Some imperfections are apparent: Sometimes, pieces of the line are not drawn (or small line-pieces are drawn when they should not be). The identified counterfactual classes and exemplars seem satisfactory, as classes that are closely related are picked, as is apparent from five explanations only depicting a single factor-change. Additionally, all assigned classes are correct.

Quantitative: ***exemplar quality***. To conclude, we report on the exemplar-selection experiment, conducted using VAE-CE. The probability of selecting an exemplar with factor 5 is $p_{78} = 0.9378 \pm 0.0436$ for classes 7 and 8, and $p_c = 0.7902 \pm 0.0176$ for the remaining classes. The difference between these probabilities is significant ($p = 0.0034$), indicating that if two class-

variants exist, we are significantly more likely to select the variant with more factors in common (compared to the situation where all variants have the same number of common factors).

We compare different classes' probabilities to negate the effect of the overall distribution of factors: $p_c = 0.7902$ shows that we are more likely to pick the factor-5-variant in all cases, even when the number of different factors is equal for both variants. We hypothesize that this preference follows from the fact that more classes exist with factor 5, which in turn distributes datapoints with factor 5 closer to the mean of the distribution. Arbitrary class-embeddings in the latent space are then more likely to reside near a variant with factor 5, resulting in an exemplar-probability significantly greater than random chance (0.5). We consider a formal analysis of this effect to lie outside of the scope of this experiment.

7.1.2 Representation-Quality

Quantitative: *rec* & *kl*. We first consider the *ELBO*-related metrics, the reconstruction error and the KL divergence. Their respective evaluations are depicted in Table 7.3.

	(Mean ± SD)			
Model	*rec* ↓	*rec-gen* ↓	*kl-*z_y ↓	*kl-*z_x ↓
DVAE	11.93 ± 0.281	46.93 ± 0.145	**4.425** ± 0.163	7.540 ± 0.213
LVAE	13.49 ± 1.09	45.19 ± 3.89	10.96 ± 5.39	7.054 ± 0.218
GVAE	**10.19** ± 0.194	40.83 ± 1.09	7.072 ± 0.566	5.738 ± 0.752
ADA-GVAE	10.67 ± 0.335	**23.01** ± 1.04	8.233 ± 1.17	**4.940** ± 1.10
VAE-CE (Ours)	14.51 ± 0.598	32.67 ± 0.608	7.808 ± 0.139	7.944 ± 0.464

Table 7.3: *ELBO*-related metrics on the synthetic data. We consider the reconstruction cost on both the test data (*rec*) and on data consisting of novel combinations of lines (*rec-gen*). The KL divergence is evaluated for the class-specific (*kl-*z_y) and class-irrelevant (*kl-*z_x) latent space.

While *rec* is related to the *eac* (both metrics consider the squared error with respect to ground-truth data), *rec* on its own is not a good predictor of the explanation quality: VAE-CE has the largest reconstruction error and the smallest *eac*. The internal representation is likely more important than the ability to reconstruct each pixel accurately. *kl-*z_y and *kl-*z_x similarly show deviations that seem somewhat arbitrary; they are likely heavily dependant on the hyperparameter weighting rather than necessarily being related to explanation-ability (e.g., DVAE has the [shared] highest weight for *kl-*z_y and has no additional regularization, resulting in the lowest *kl-*z_y).

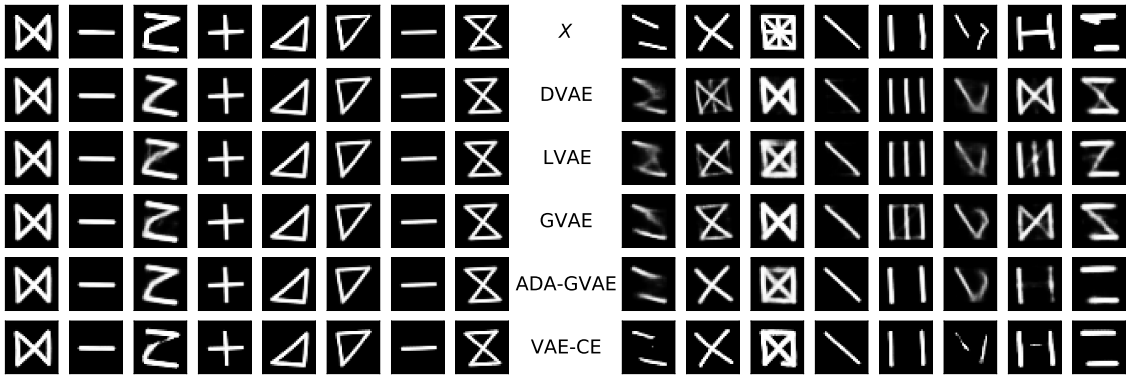
An interesting observation is the large deviations in *rec-gen*, where both change-pair supervised methods produce a significantly lower cost. These two methods also produced some of the lowest *eac*-values, indicating that a method's generalization and explanation-quality could be related. In an attempt to cancel out the reconstruction quality itself (quantifying only the difference when modeling novel data), we evaluate the ratios of *rec-gen* to *rec*. These ratios are depicted in Table 7.4. These values must be treated with care, however. While a small ratio indicates little extra cost when modeling novel data, if the baseline (*rec*) is bad, this ratio is relatively meaningless. We consider these *rec*-values to be similar enough (a ratio of < 1.5 for any model combination) for the comparison to make sense.

	(Mean ± SD)
Model	*rec-gen* ÷ *rec* ↓
DVAE	3.936 ± 0.094
LVAE	3.350 ± 0.070
GVAE	4.008 ± 0.134
ADA-GVAE	**2.157** ± 0.080
VAE-CE (Ours)	2.255 ± 0.080

Table 7.4: Ratio of the reconstruction cost of novel data and that of the test dataset.

The ratios of ADA-GVAE and VAE-CE are not significantly different ($p = 0.1359$); however, both are significantly smaller than all other ratios ($p < 0.0001$ for all comparisons). As such, pair-based supervision seems to lead to more generalizing representations. The question remains whether this improvement stems from the different datapoints used in training (many pair-images depict only sub-components of datapoints: A wider range of shapes is 'leaked' into the learning process) or from the inductive biases present in these methods. Additionally, LVAE produces a significantly lower score than DVAE and GVAE, indicating that label-based supervision also leads to better generalization ($p < 0.001$ for both comparisons).

Qualitative: *data reconstruction*. Reconstructions are evaluated using eight datapoints from both the regular and novel test data. These two evaluations are depicted in Figures 7.5a and 7.5b.



(a) Reconstructions of test data. (b) Reconstructions of novel line combinations.

Figure 7.5: Reconstructions of synthetic datapoints. The top row depicts the input datapoints, whereas each other row depicts the model's respective reconstructions of said datapoints.

The regular reconstructions tell a story similar to the *rec* evaluation: While there are slight differences in per-pixel accuracy, the reconstructions are very similar. From the novel-data reconstructions, it is apparent that LVAE, ADA-GVAE, and VAE-CE generalize to an extent, whereas the remaining methods mostly depict either existing classes or blurry combinations of classes. A point of interest is that VAE-CE seems to produce sharper images than the remaining methods. We hypothesize that this sharpness follows from the discriminator component (in the auxiliary conditioning pass), as blurry pixels indicate 'fake' data and are likely adjusted to fool D (similar to GANs).

Quantitative: *l-acc* & *acc*. Next, we evaluate the extent of class-based disentanglement in latent subspaces z_y and z_x (l-acc-z_y and l-acc-z_x), alongside the classification-ability of the models according to z_y (acc). The respective metrics are depicted in Table 7.5.

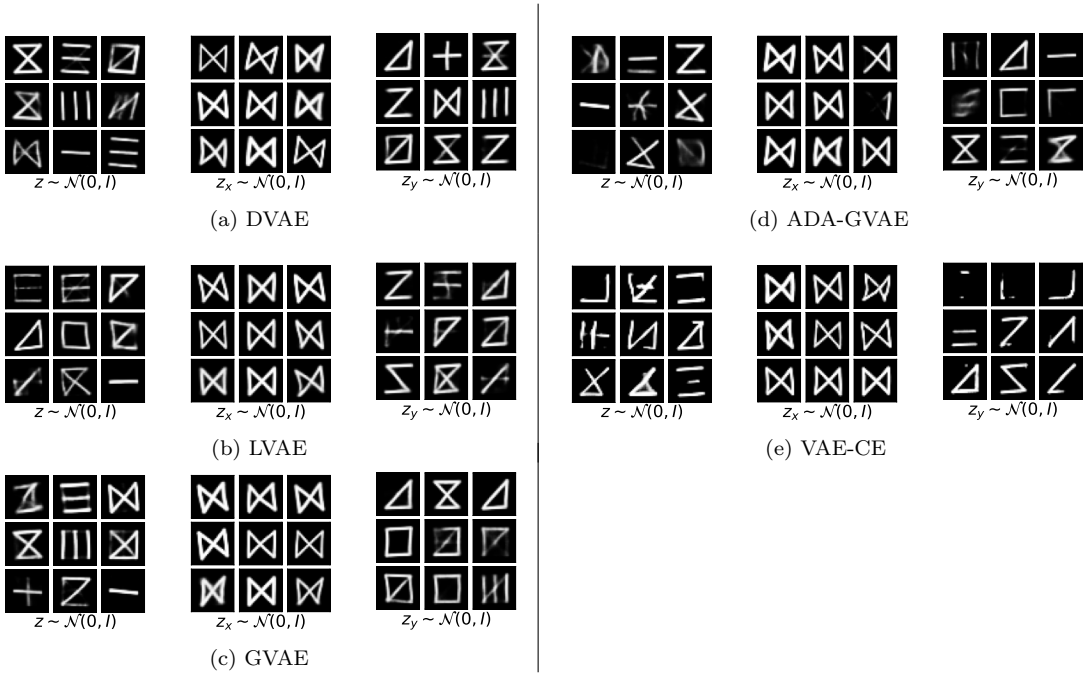
	(Mean ± SD)		
Model	l-acc-z_y ↑	l-acc-z_x ↓	acc ↑
DVAE	**0.9749** ± 0.0006	**0.1700** ± 0.0150	**0.9734** ± 0.0007
LVAE	0.9618 ± 0.0022	0.2080 ± 0.0572	0.9537 ± 0.0080
GVAE	0.9641 ± 0.0007	0.2023 ± 0.0232	0.9621 ± 0.0010
ADA-GVAE	0.9610 ± 0.0011	0.1801 ± 0.0361	0.9585 ± 0.0008
VAE-CE (Ours)	0.9657 ± 0.0004	0.1822 ± 0.0139	0.9629 ± 0.0009

Table 7.5: Latent-space accuracy metrics for the synthetic data. To evaluate the class-based disentanglement, we evaluate the accuracy of a logistic regression classifier trained on the class-relevant (l-acc-z_y) and class-irrelevant (l-acc-z_y) latent space. Furthermore, we report the accuracy of the model's classifier component, attached to z_y (acc).

All models disentangled the class-relevant and irrelevant factors well; little information about the class can be extracted from z_x (given that we have ten classes, we cannot expect $l\text{-}acc\text{-}z_x$ to drop below 0.1). As such, while class-disentanglement is essential to be able to generate good explanations, the difference therein is not substantial between models. The overall classification ability (acc) is similar to that of the logistic regression classifier ($l\text{-}acc\text{-}z_y$), which is unsurprising given that CY only consists of a single layer.

To quantify the probability that the exemplars inferred by VAE-CE are of the correct class, we consider the accuracy of all datapoints with an estimated probability above the chosen threshold of $t = 0.95$. The resulting accuracy is 0.9994 ± 0.0001, indicating that the exemplars are very unlikely to correspond to the incorrect class. The fraction of data still taken into consideration (reaching $p(y) \geq t$) is 0.9072 ± 0.0080; only $\approx 10\%$ of the data is discarded.

Qualitative: *data generation*. To evaluate both the disentanglement and the generative ability of the models, we generate images by sampling z_x, z_y, or z in its entirety from the prior. These images are depicted in Figure 7.6. The non-sampled dimensions are inferred from the first datapoint in Figure 7.5a.



(a) DVAE (b) LVAE (c) GVAE (d) ADA-GVAE (e) VAE-CE

Figure 7.6: Generated images based on synthetic data, using a shared datapoint's encoding as the baseline. For each model, we synthesize images by sampling (from left to right) z, z_x, or z_y from the prior, while keeping the remaining values fixed.

All models generate realistic data. LVAE, ADA-GVAE, and VAE-CE also generate combinations of lines not occurring in the training set. Adjusting only z_x resulted in no class-specific features being changed, with the exception of one image in ADA-GVAE. We do not rule out that such changes can occur in other methods, however, given that these images depict a tiny sample. z_y clearly controls the class-specific features, randomly sampling z_y shows distinct differences in shape. Again, we note that VAE-CE is the only method not to generate any blurry images.

Quantitative: *mig*. We consider the mig to quantify the per-dimension disentanglement of the methods. These values are depicted in Table 7.6. A higher mig seems to coincide with better explanations, as VAE-CE scores the highest, followed by LVAE and ADA-GVAE. We do note that the difference between VAE-CE and LVAE is not significantly different ($p = 0.0543$). As such, we do not exclude the possibility that the stochasticity of training NNs caused this difference.

	(Mean ± SD)
Model	*mig* ↑
DVAE	0.1206 ± 0.0300
LVAE	0.4227 ± 0.0465
GVAE	0.1484 ± 0.0692
ADA-GVAE	0.3402 ± 0.0732
VAE-CE (Ours)	**0.4923** ± 0.0326

Table 7.6: The mutual information gap of dimensions in z_y, measured using the encodings of synthetic test-datapoints.

Quantitative: ***metric correlations***. To quantify the relation of the representation- and explanation-quality, we consider the Pearson correlation coefficients[11] between the representation-metrics and the *eac*. The statistically significant correlations are depicted in Figure 7.7.

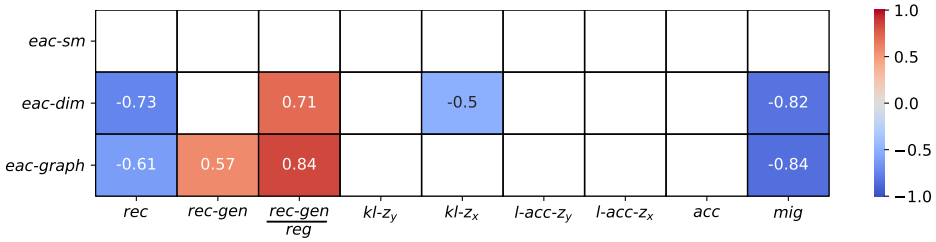
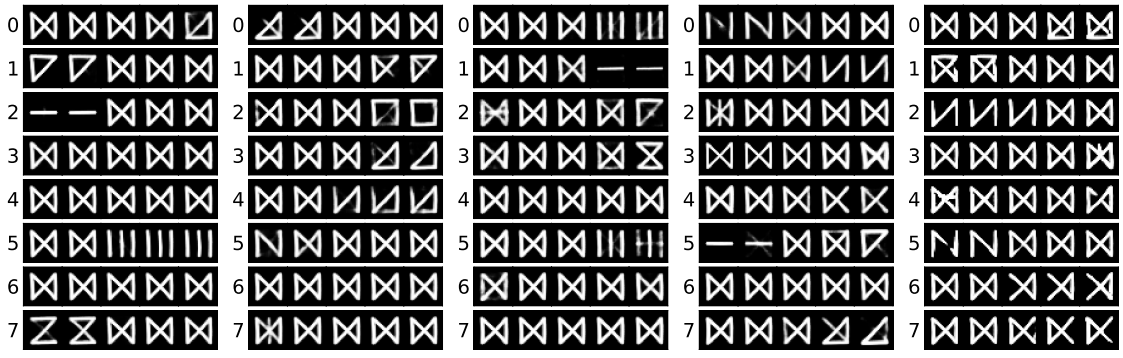


Figure 7.7: Significant correlations between representation-quality metrics and the *eac*, measured using the synthetic dataset.

Both the *mig* and the *rec-gen* ratio show strong correlations with the *eac*. These correlations make sense, given that better disentanglement likely leads to interpolations only changing a single factor per step, and better generalization indicates that models can better produce novel data (intermediate states). Other correlations seem less useful: a negative correlation between *rec* and *eac* indicates that worse reconstructions would result in better explanations. We hypothesize that this correlation is the result of VAE-CE reconstructing relatively badly, as it seems to trade reconstruction quality for more meaningful dimensions. The (albeit weaker) correlation of *eac* with *kl*-z_x is likely also noise, following from the added regularization in z_y (which led to better explanations) causing more information to be stored in z_x.



(a) DVAE (b) LVAE (c) GVAE (d) ADA-GVAE (e) VAE-CE

Figure 7.8: Latent traversals of each dimension of z_y, using a shared datapoint's encoding as the baseline. Each row depicts the synthesized images generated by traversing the corresponding dimension in z_y between [-3, 3] (with a fixed step size) while keeping the remaining values fixed.

Qualitative: ***latent-space traversal***. To conclude, we evaluate the structure of the individual dimensions in z_y. For each model, we encode a datapoint (the same as in Figure 7.6) to its latent

representation and traverse its dimensions one at a time. For each dimension, we interpolate between [-3, 3] (with a fixed step size) and synthesize datapoints from the modified intermediate states. As such, we can evaluate what kind of effect a single dimension-change has on the datapoint. These latent-traversals are depicted in Figure 7.8. DVAE and GVAE both depict entire class-transitions in their traversals, indicating that the dimensions do not coincide with the underlying generative factors. Both LVAE and ADA-GVAE depict individual line-changes in some dimensions, but not in all. For VAE-CE, every dimension depicts a line segment being added or removed. This order of line-disentanglement coincides with the relative *mig* and *eac* scores, supporting the use of these metrics for our explanation purpose.

7.2 MNIST

7.2.1 Explanation-Quality

Qualitative: ***explanation generation***. To compare the explanation-quality of the different methods, we sample four input-pairs and generate their interpolations according to the best-performing interpolation-method (Table 7.1). These input-pairs are depicted in Figure 7.9, whereas the resulting explanations are depicted in Figure 7.10.

(a) 4 to 3. (b) 2 to 8. (c) 0 to 4. (d) 5 to 3.

Figure 7.9: Four MNIST input-pairs. Explanations are generated by interpolating between these images' class-features.

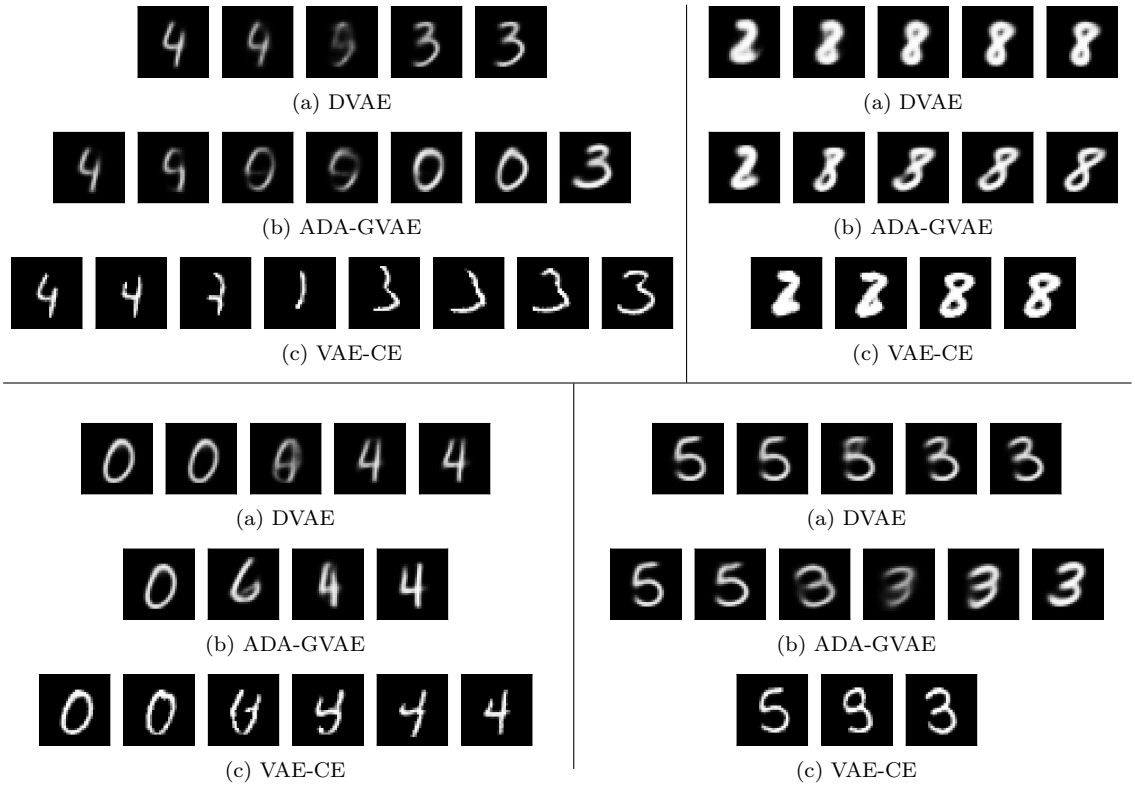


Figure 7.10: The generated explanations from the pairs in Figure 7.9, from 4 to 3 (top left), 2 to 8 (top right), 0 to 4 (bottom left) and 5 to 3 (bottom right).

The explanations for the MNIST pairs are less convincing than those for the synthetic data. While individual line-changes are apparent in VAE-CE's interpolation, it is not necessarily a single change per step: Some steps indicate little to no change, and some steps consider multiple line-changes. For ADA-GVAE, some line-changes are visible, although they are rather blurry. DVAE's interpolations only depict (blurred combinations of) entire classes. While in comparison VAE-CE still creates the interpolations most aligned with the goal of a single line-change per step, they are not without errors.

We also consider explanations generated from datapoints in isolation, using VAE-CE. These datapoints are depicted in Figure 7.11, whereas their explanations are depicted in Figure 7.14.

Figure 7.11: Eight MNIST datapoints we generate an explanation for.

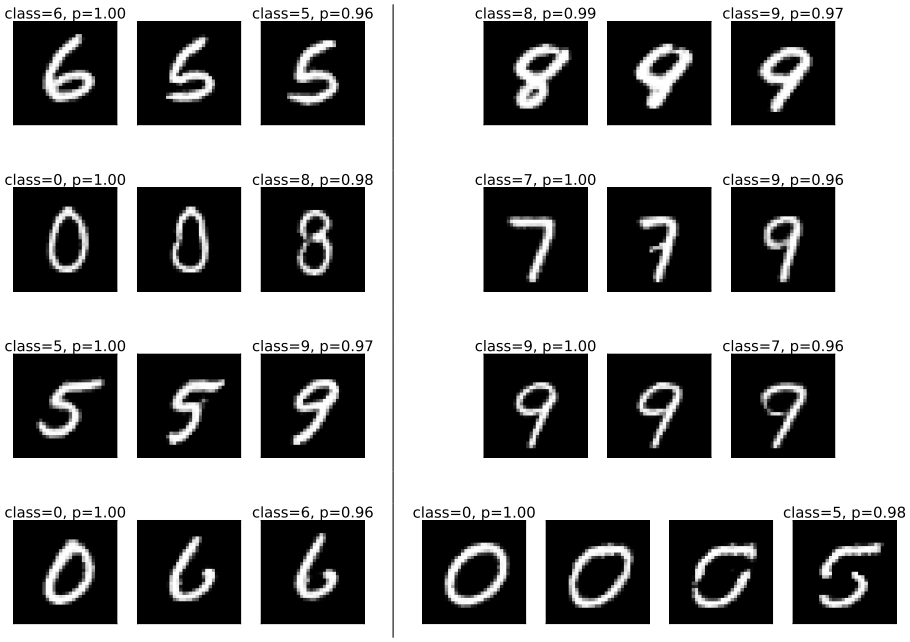
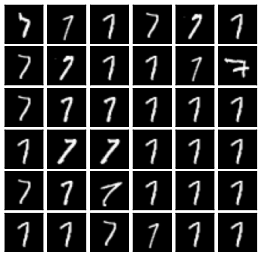


Figure 7.12: Explanations of the images depicted in Figure 7.11, generated using VAE-CE. Each explanation depicts the predicted digit, the changes that must be made to classify the datapoint as another digit, and the predicted counterfactual digit.

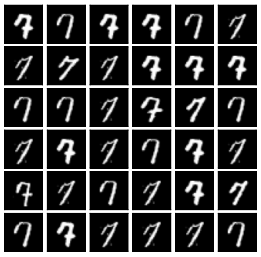
The resulting explanations are less noisy than the previous pair-based explanations, likely because of the exemplar-selection providing a more suitable 'target' to interpolate towards (rather than an arbitrary datapoint that might have little in common). There are still steps not coinciding with single line-changes, although most steps indicate either a single line being deformed or a single line being removed/created. The inferred classes are all correct.

Lastly, we consider the exemplar-selection process in isolation. To evaluate whether VAE-CE selects exemplars according to the most common factors (lines), we generate exemplars according to two queries: Contrasting a one against a seven and contrasting a four against a seven. The underlying assumption is that a seven can be written both with and without a horizontal line in the middle. Ones do not have such horizontal lines, whereas fours often do. As such, we hypothesize that the

latter exemplar-query should result in more examples of sevens with horizontal lines. We identify 36 exemplars for each query. The results are depicted in Figure 7.13.



(a) Exemplars selected when contrasting against ones.



(b) Exemplars selected when contrasting against fours.

Figure 7.13: Identified seven-exemplars. We seek to evaluate whether exemplars identified using ones (left) contain less horizontal lines than exemplars identified using fours (right).

The seven-exemplars contain a line significantly more often when contrasting querying from a four to querying from a one (12 times versus 1), indicating that the common factors seem to play a role in the exemplar selection. We recognize that this experiment is rather limited, as it does not evaluate all classes[2] and only considers a small sample size.

7.2.2 Representation-Quality

Quantitative: *rec* & *kl*. The results of the *ELBO*-related metrics are depicted in Table 7.7.

	(Mean ± SD)			
Model	*rec* ↓	*rec-gen* ↓	*kl-*z_y ↓	*kl-*z_x ↓
DVAE	16.01 ± 0.348	52.93 ± 1.34	**4.341** ± 0.305	8.043 ± 0.254
ADA-GVAE	**13.64** ± 0.191	**37.63** ± 0.471	10.14 ± 1.06	**3.604** ± 0.901
VAE-CE (Ours)	21.67 ± 1.34	58.40 ± 1.38	7.117 ± 0.288	6.646 ± 0.351

Table 7.7: *ELBO*-related metrics on MNIST. We consider the reconstruction cost on both the test data (*rec*) and on EMNIST examples (*reg-gen*). The KL divergence is evaluated for the class-specific (*kl-*z_y) and class-irrelevant (*kl-*z_x) latent space.

The differences in *rec* are similar to those in the synthetic dataset. VAE-CE generates images with the largest reconstruction error, whereas ADA-GVAE denotes a lower error than the baseline, DVAE. Again, we note that this difference is not necessarily the result of the conditioning method, but rather the hyperparameter choice: ADA-GVAE reports a significantly larger *kl-*z_y than both DVAE and VAE-CE ($p = 0.0009$ and $p = 0.008$, respectively), indicating a trade-off between reconstruction quality and prior-regularization. In contrast to the synthetic data, VAE-CE does not report a lower *rec-gen* than DVAE, indicating worse generalization (to EMNIST). To further inspect this property, we consider the ratio of *rec-gen* and *rec*, as depicted in Table 7.8.

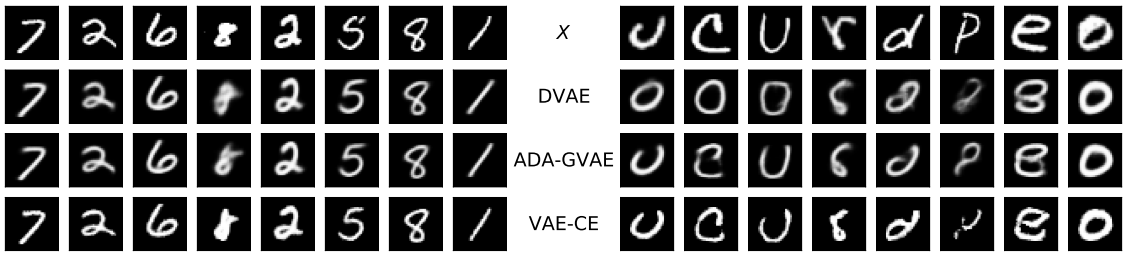
	(Mean ± SD)
Model	*rec-gen* ÷ *rec* ↓
DVAE	3.306 ± 0.058
ADA-GVAE	2.758 ± 0.024
VAE-CE (Ours)	**2.699** ± 0.117

Table 7.8: Ratio of the reconstruction cost of EMNIST versus MNIST.

[2]We only consider these two classes as it is clear that a one does not contain a horizontal line, whereas a four (most often) does. Many other digits contain some horizontal component, but whether that should be transferred over to the seven-exemplar requires further discussion.

The difference between ADA-GVAE and VAE-CE is not significant ($p = 0.393$), whereas the ratio of both is significantly smaller than that of DVAE ($p < 0.001$ for both). This difference is relatively small compared to that in the synthetic dataset, however (≈ 2.2 vs. 3.9 for the synthetic data, and ≈ 2.7 vs. 3.3 for MNIST), indicating that either the models learned less-general representations, or that MNIST and EMNIST differ more in their underlying factors. Additionally, we note that VAE-CE's ratio could be artificially low, given that its *rec* was the largest by a decent margin. To further inspect the generalization property, we consider qualitative analysis.

Qualitative: *data reconstruction*. Eight datapoints were sampled from both MNIST and EMNIST, and reconstructed using each model. These datapoints and their reconstructions are depicted in Figures 7.14a and 7.14b, respectively.



(a) Reconstructions of MNIST samples. (b) Reconstructions of EMNIST samples.

Figure 7.14: Reconstructions of datapoints. The top row depicts the input datapoints, whereas each other row depicts the model's respective reconstructions of said datapoints.

The regular reconstructions are all rather similar, with the only noticeable difference being the sharpness of VAE-CE's reconstructions. EMNIST's reconstructions support the idea that, despite its high *rec-gen*, VAE-CE generalizes better to novel data than the baseline (DVAE). The reconstructions produced by both ADA-GVAE and VAE-CE are not flawless, but they depict line-combinations not apparent in MNIST (e.g., the 'e'). DVAE's reconstructions are closer to blurred versions of digits rather than novel line combinations.

Quantitative: *l-acc & acc*. The class-disentanglement and classification-ability related metrics are depicted in Table 7.9.

	(Mean ± SD)		
Model	*l-acc-*z_y ↑	*l-acc-*z_x ↓	*acc* ↑
DVAE	**0.9940** ± 0.0005	0.1822 ± 0.0094	**0.9911** ± 0.0007
ADA-GVAE	0.9701 ± 0.0015	0.1997 ± 0.0149	0.9645 ± 0.0017
VAE-CE (Ours)	0.9834 ± 0.0017	**0.1817** ± 0.0119	0.9795 ± 0.0019

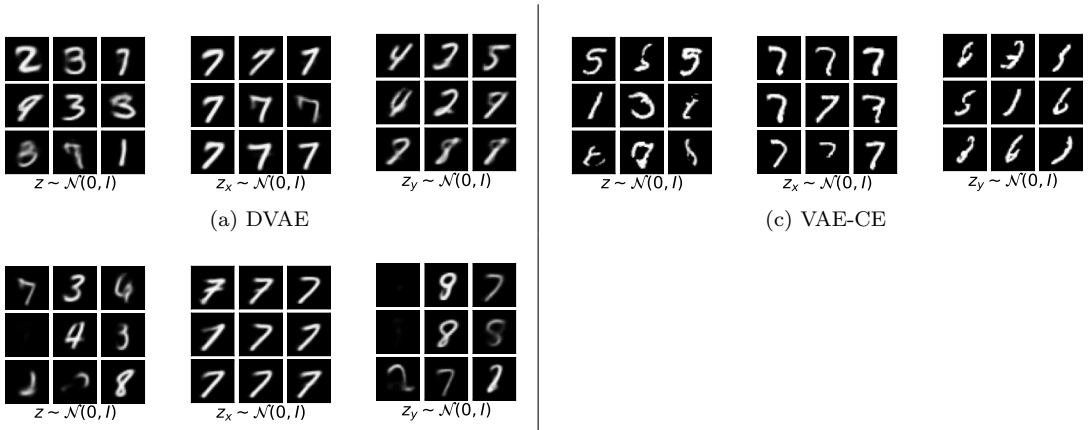
Table 7.9: Latent-space accuracy metrics for MNIST. To evaluate the class-based disentanglement, we evaluate the accuracy of a logistic regression classifier trained on the class-relevant (*l-acc-*z_y) and class-irrelevant (*l-acc-*z_y) latent space. Furthermore, we report the accuracy of the model's classifier component, attached to z_y (*acc*).

All models show a similar level of disentanglement, with little information to be extracted out of z_x. The classification accuracy of DVAE is significantly greater than that of ADA-GVAE and VAE-CE ($p < 0.001$ for both), with a larger difference than in the case of the synthetic data. While isolating the cause of this discrepancy is difficult, it highlights that the additional dimension-conditioning might not necessarily correspond with MNIST's true generating process.

The accuracy-percentages of both ADA-GVAE and VAE-CE are also somewhat low compared to even simplistic NNs[81].

To quantify the probability that VAE-CE's explanations consider correctly-classified datapoints, we also infer the accuracy given that we only consider datapoints above the set confidence threshold of $t = 0.95$. The resulting accuracy is 0.9972 ± 0.0003 and considers 0.9193 ± 0.0114 of the entire dataset. Consequently, the identified exemplars are very unlikely to consider incorrectly classified datapoints.

Qualitative: *data generation*. To qualitatively analyze the latent subcomponents' effect on data, we synthesize images while sampling (subparts of) the latent space. The remaining latent values are inferred from the first datapoint in Figure 7.14a. The synthesized images are depicted in Figure 7.15.



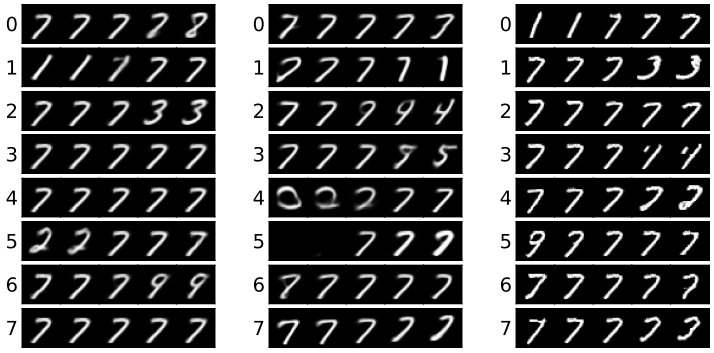
(a) DVAE (c) VAE-CE

(b) ADA-GVAE

Figure 7.15: Generated images based on MNIST, using a shared datapoint's encoding as the baseline. For each model, we synthesize images by sampling (from left to right) z, z_x, or z_y from the prior, while keeping the remaining values fixed.

The class-disentanglement between z_y and z_x is apparent in all models: Images following a randomized z_x depict randomly modified seven-digits, whereas a randomized z_y results in random digit-like samples of various styles. DVAE and ADA-GVAE created images largely in line with (partially blurred) digit-examples, whereas VAE-CE also synthesized images only depicting subparts of digits. Finally, we note that ADA-GVAE generated the occasionally empty image (likely stemming from the change-pair supervision containing empty samples) and that VAE-CE's samples are significantly sharper than other methods' samples.

Qualitative: *latent-space traversal*. To conclude, we examine the effect of individual z_y dimensions on output-images. Given the latent representation of a seven (as used in Figures 7.14a and 7.15), we traverse each z_y-dimension in the range of [-3, 3] while keeping the remaining dimensions fixed. The resulting images are depicted in Figure 7.16. DVAE's z_y-space does not embed individual line-values in its dimensions, as single-dimension traversals resulted in entire class shifts (changing multiple lines). Both ADA-GVAE and VAE-CE approach our desired notion of changing individual lines according to individual dimensions, although the former also learned a few dimensions controlling many factors at once. As such, even if not completely successful, VAE-CE best approaches our desired notion of representing digits according to their underlying lines.



(a) DVAE (b) ADA-GVAE (c) VAE-CE

Figure 7.16: Latent traversals of each dimension of z_y, using a shared datapoint's encoding as the baseline. Each row depicts the synthesized images generated by traversing the corresponding dimension in z_y between [-3, 3] (with a fixed step size) while keeping the remaining values fixed.

7.3 Conclusions

We aimed to answer four questions (SQ5.1-5.4) that consider the explanation- and representation quality of our method (and others). In this section, we summarize our findings with respect to each question, and note the limitations of this experiment.

SQ5.1: For the synthetic data, our method effectively captured the features of the dataset. Explanations depicted individual line-changes at each step and did not change more features than necessary. The exemplar-selection procedure is also more likely to select datapoints that contain more common features. The data used for exemplar-selection considered the majority of the dataset ($> 90\%$) and was very accurately classified ($> 99.9\%$).

These results were partially mirrored for MNIST, but the resulting explanations depicted more incorrect changes. We note that this deficiency can follow from multiple factors, such as the MNIST-supervision being relatively noisy or the architecture not fitting to MNIST (as model-selection was done using the synthetic data).

SQ5.2: For the synthetic data, VAE-CE outperformed all compared methods with respect to the *eac*, hinting that it is a suitable method for explanation-generation. Qualitatively, we argue that VAE-CE's explanations also matched our presupposed notion of good explanation better than the alternative methods, both for the synthetic data and for MNIST.

SQ5.3: We considered explanations generated both without the explanation-generator and without the pair-based dimension conditioning method. The removal of either (or both) resulted in a worse *eac*, indicating that both components play a role. Furthermore, we identified that the explanation-generator alone improves the explanation-quality of all methods.

SQ5.4: Between the compared methods, the representation-quality did not differ significantly for most factors: They had a similar reconstruction ability, generative ability, and class-information split. An interesting difference was found in the generalization of models (measured by reconstructing a novel dataset), where ADA-GVAE and VAE-CE performed best. Both these datasets used change-pairs as additional supervision, hinting that this type of data can provide a strong inductive bias. Furthermore, the *mig* of the different methods strongly correlated with explanation quality. Qualitatively, we argue that VAE-CE best captures the underlying features (lines) in individual dimensions.

7.3.1 Limitations

We note that this experiment was primarily intended to explore the performance of our method, and recognize the following set of shortcomings. First, we only considered a single model architecture and a limited set of hyperparameters per model. It is possible that choices in the model-construction process influenced the relative performance of the methods. Additionally, we only trained a small number of models for each configuration; while many conclusions were still statistically significant, it is desirable to explore a larger sample size. The datasets were also somewhat simplistic, raising the question of how well the approaches scale to more complex data. Furthermore, we did not explore the computational cost of inferring and evaluating explanations. As noted in Chapter 5, both these methods have a large computational complexity. While in our experiments this complexity did not lead to limitations (e.g., both explanation-inference and explanation-evaluation took in the order of seconds per datapoint), a formal analysis of this cost remains future work. Finally, we recognize that VAE-CE is the only method specifically designed with our assumptions of good explanation in mind. As such, the *eac*-comparison is inherently biased towards our approach.

Chapter 8

Conclusions

In this thesis, we examined how to explain the classification of complex data using machine learning, i.e., the explanation of data according to their classification. To answer this question, we considered a set of subquestions. In this chapter, we provide a summarized overview of all outcomes (i.e., the answers to SQ2-5). Additionally, we present potential avenues for future work.

SQ2 *How do we define explanation in the context of our goal?*
To define the explanations we seek to generate, we considered works on explanation and categorization spanning across various domains. In summary, we consider explanations to be *contrastive*: We explain a classification by considering an alternative class and describing why the datapoint was not assigned to that class. This description comes in the shape of class-features, predefined concepts assumed to determine a class. We consider two approaches to describing these differences: By identifying features in isolation, or by evaluating common features between the datapoint and existing examples (or prototypes) of a class.

SQ3 *How do we generate explanations using machine learning?*
We proposed Variational Autoencoder-based Contrastive Explanation (VAE-CE), a framework for generating contrastive explanations using a class-disentangled VAE as a baseline. This VAE is conditioned using a new method, denoted as pair-based dimension conditioning. This method uses pairs of images that either depict a single feature being changed, or no/multiple feature-changes. Using these pairs, we train a classifier that identifies (in)correct changes. This classifier is used to encourage individual latent dimensions to represent individual features.

To create explanations using this representation model, we described a method for identifying a (counterfactual) exemplar to explain towards. Class-distinguishing features are conveyed by interpolating towards this exemplar, depicting a single feature-change at each step. To generate this interpolation, we described a method for traversing a model's latent space. We construct a graph of all dimension-changes, weighted according to the quality of the change, the quality of the intermediate state, and the number of changes. By finding the shortest path in this graph, we generate the explanation maximizing these qualities while minimizing the explanation length.

SQ4 *How can we quantitatively evaluate the quality of explanations?*
We proposed the explanation alignment cost (*eac*), a method for quantifying the performance of explanation-generation methods. This cost is computed using a dataset with a known generative process. The to-be-evaluated method generates an explanation using a set start- and end-datapoint. To evaluate this explanation, all ground-truth explanations corresponding to the input-pair are generated. The *eac* then denotes the minimum cost for aligning the generated explanation to any fitting ground-truth explanation.

SQ5 *How well does our method represent and explain data?*
We conducted an (introductory) study to evaluate both the representation- and explanation-quality of our approach. Empirical evidence suggests that VAE-CE can create explanations matching our predefined goals. For the synthetic dataset, it quantitatively outperformed alternative feature-disentanglement approaches with respect to the *eac*. Furthermore, both overarching components of VAE-CE—the feature-conditioning method and the explanation-generation method—

seem to play a role in creating good explanations. Lastly, the evaluation of the latent space of our model suggests that it can represent features well, often controlling individual features with individual dimensions. The disentanglement of individual features and the generalization-ability of a model seem most correlated with the explanation-quality.

While VAE-CE performed well on the synthetic data, it created less convincing explanations for MNIST. It still outperformed the compared approaches, but the resulting explanations displayed more unnecessary or undesirable changes. Whether this quality-discrepancy follows from the noisy MNIST-supervision, flawed assumptions about MNIST's features, an architectural mismatch or VAE-CE not being as applicable remains an open question.

8.1 Future Work

To conclude, we note future work. Regarding our disentanglement method, we only explored optimizing for 'good' changes using latent representations differing in one factor, constructed from datapoints. Alternative approaches consider sampling these representations directly from the prior distribution (much like a GAN) or optimizing the latent space in an adversarial manner (using representations with many differing dimensions and optimizing for 'bad' changes). With regards to the change discriminator, we propose the exploration of approaches primarily requiring 'positive' change-pairs, such as a simultaneously learned discriminator[42] or an anomaly-detection based method[76].

We propose exploring more efficient methods for generating and evaluating explanations, using heuristic-based search algorithms such as beam search[44]. To evaluate the quality of individual images better, we propose the exploration of perceptual loss functions[59, 33] rather than per-pixel error functions.

With regards to analysis, we note that the conducted empirical evaluation considered only relatively simple data and a limited set of model-configurations. As such, applying VAE-CE to more complex data is a potential avenue for research. Lastly, we note that while our explanation-shape is designed according to prior literature, the precise execution does not necessarily have (empirical) grounding; one could explore the human-perception of the interpolation-approach to explanation (e.g., by conducting a user experiment).

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 45

[2] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018. 18

[3] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018. 5

[4] D. Alvarez-Melis and T. S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7786–7795, 2018. 9

[5] J. Alzubi, A. Nayyar, and A. Kumar. Machine learning from theory to algorithms: an overview. In *Journal of physics: conference series*, volume 1142, page 012012, 2018. 1

[6] R. Andrews, J. Diederich, and A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, 8(6):373–389, 1995. 7

[7] F. G. Ashby and R. E. Gott. Decision rules in the perception and categorization of multidimensional stimuli. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14(1):33, 1988. 14

[8] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015. 7

[9] A. Baldominos, Y. Saez, and P. Isasi. A survey of handwritten character recognition with MNIST and EMNIST. *Applied Sciences*, 9(15):3169, 2019. 5, 6

[10] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3319–3327. IEEE Computer Society, 2017. 7

[11] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 37–40. Springer, 2009. 54

[12] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 10, 18, 21

[13] D. Berthelot, C. Raffel, A. Roy, and I. J. Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 10

[14] O. Biran and C. Cotton. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8, pages 8–13, 2017. 5

[15] D. Bouchacourt, R. Tomioka, and S. Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2095–2102. AAAI Press, 2018. 10, 21

[16] R. Cai, Z. Li, P. Wei, J. Qiao, K. Zhang, and Z. Hao. Learning disentangled semantic representation for domain adaptation. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2060–2066. ijcai.org, 2019. 23, 44, 95

[17] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. Su. This looks like that: Deep learning for interpretable image recognition. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8928–8939, 2019. 9

[18] T. Q. Chen, X. Li, R. B. Grosse, and D. Duvenaud. Isolating sources of disentanglement in variational autoencoders. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2615–2625, 2018. 10, 43

[19] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2172–2180, 2016. 20

[20] Y. Chen, X. Xu, Z. Tian, and J. Jia. Homomorphic latent space interpolation for unpaired image-to-image translation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2408–2416. Computer Vision Foundation / IEEE, 2019. 26, 35

[21] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 577–585, 2015. 9

[22] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017. 42

[23] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 6

[24] P. Cunningham and S. J. Delany. k-nearest neighbour classifiers: 2nd edition (with python examples). *arXiv preprint arXiv:2004.04523*, 2020. 6

[25] I. N. da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves. *Introduction*, pages 3–19. Springer International Publishing, Cham, 2017. 5

[26] E. D. de Jong. Incremental sequence learning. *arXiv preprint arXiv:1611.03068*, 2016. 86

[27] D. C. Dennett. *From bacteria to Bach and back: The evolution of minds*. WW Norton & Company, 2017. 12

[28] A. Dhurandhar, P. Chen, R. Luss, C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 590–601, 2018. 8

[29] Z. Ding, Y. Xu, W. Xu, G. Parmar, Y. Yang, M. Welling, and Z. Tu. Guided variational autoencoder for disentanglement learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 7917–7926. IEEE, 2020. 22, 44

[30] J. v. Doorenmalen and V. Menkovski. Evaluation of cnn performance in semantically relevant latent spaces. In *International Symposium on Intelligent Data Analysis*, pages 145–157. Springer, 2020. 10

[31] S. Dopkins and T. Gleason. Comparing exemplar and prototype models of categorization. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 51(3):212, 1997. 14

[32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 9

[33] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 658–666, 2016. 63

[34] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4829–4837. IEEE Computer Society, 2016. 7

[35] I. El Naqa, P. Grigsby, A. Apte, E. Kidd, E. Donnelly, D. Khullar, S. Chaudhari, D. Yang, M. Schmitt, R. Laforest, et al. Exploring feature-based approaches in pet images for predicting cancer treatment outcomes. *Pattern recognition*, 42(6):1162–1171, 2009. 6

[36] A. Feghahati, C. R. Shelton, M. J. Pazzani, and K. Tang. Cdeepex: Contrastive deep explanations. 2018. 8

[37] R. Fong and A. Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8730–8738. IEEE Computer Society, 2018. 7

[38] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020. 1

[39] A. Gabbay and Y. Hoshen. Demystifying inter-class disentanglement. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 10

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

[41] I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 1, 5, 16, 17

[42] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014. 6, 20, 30, 63

[43] R. L. Graham, D. E. Knuth, O. Patashnik, and S. Liu. Concrete mathematics: a foundation for computer science. *Computers in Physics*, 3(5):106–107, 1989. 78

[44] A. Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012. 63

[45] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018. 8

[46] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018. 5

[47] C. G. Hempel et al. Aspects of scientific explanation. 1965. 12

[48] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016. 9

[49] L. A. Hendricks, R. Hu, T. Darrell, and Z. Akata. Grounding visual explanations. In *European Conference on Computer Vision*, pages 269–286. Springer, 2018. 9

[50] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018. 10

[51] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 10, 21, 23

[52] D. J. Hilton. Conversational processes and causal explanation. *Psychological Bulletin*, 107(1):65, 1990. 13

[53] D. J. Hilton. Social attribution and explanation. 2017. 13

[54] D. J. Hilton and B. R. Slugoski. Knowledge-based causal attribution: The abnormal conditions focus model. *Psychological review*, 93(1):75, 1986. 12

[55] H. Hosoya. Group-based learning of disentangled representations with generalizability for novel contents. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2506–2513. ijcai.org, 2019. 10, 44

[56] M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pages 322–348. PMLR, 2020. 10, 21, 22, 44

[57] A. H. Jha, S. Anand, M. Singh, and V. Veeravasarapu. Disentangling factors of variation with cycle-consistent variational auto-encoders. In *European Conference on Computer Vision*, pages 829–845. Springer, 2018. 10, 21

[58] U. Johansson, R. König, and L. Niklasson. The truth is in there-rule extraction from opaque models using genetic programming. In *FLAIRS Conference*, pages 658–663. Miami Beach, FL, 2004. 8

[59] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 63

[60] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, K. Tunyasuvunakool, O. Ronneberger, R. Bates, A. Žídek, A. Bridgland, C. Meyer, S. A. A. Kohl, A. Potapenko, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, M. Steinegger, M. Pacholska, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. High accuracy protein structure prediction using deep learning. *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction*, 22:22–24, 2020. 1

[61] C. Jutten and J. Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, 24(1):1–10, 1991. 10

[62] D. Kahneman and A. Tversky. Subjective probability: A judgment of representativeness. *Cognitive psychology*, 3(3):430–454, 1972. 34

[63] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4401–4410. Computer Vision Foundation / IEEE, 2019. 20

[64] Y. Kilcher, A. Lucchi, and T. Hofmann. Semantic interpolation in implicit models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 10

[65] B. Kim, E. Glassman, B. Johnson, and J. Shah. ibcm: Interactive bayesian case model empowering humans via intuitive interaction. 2015. 1

[66] B. Kim, M. Wattenberg, J. Gilmer, C. J. Cai, J. Wexler, F. B. Viégas, and R. Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2673–2682. PMLR, 2018. 8

[67] H. Kim and A. Mnih. Disentangling by factorising. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2654–2663. PMLR, 2018. 10, 21

[68] D. P. Kingma. Variational inference & deep learning: A new synthesis. 2017. 18

[69] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 18, 92

[70] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. ii, 1, 10, 18, 20

[71] D. G. Kleinbaum and M. Klein. *Introduction to Logistic Regression*, pages 1–39. Springer New York, New York, NY, 2010. 6

[72] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991. 10, 18

[73] R. Krishnan, G. Sivakumar, and P. Bhattacharya. Extracting decision trees from trained neural networks. *Pattern recognition*, 32(12), 1999. 8

[74] J. K. Kruschke. Models of categorization. *The Cambridge handbook of computational psychology*, pages 267–301, 2008. 13

[75] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951. 19, 76

[76] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim. A survey of deep learning-based network anomaly detection. *Cluster Computing*, pages 1–13, 2019. 63

[77] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 9

[78] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. The omniglot challenge: a 3-year progress report. *Current Opinion in Behavioral Sciences*, 29:97–104, 2019. 9

[79] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017. 6

[80] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 1, 5, 16, 17

[81] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 17, 25, 38, 40, 59

[82] Y. LeCun, C. Cortes, and C. J. Burges. The MNIST database of handwritten digits. Available online: `http://yann.lecun.com/exdb/mnist/` (accessed on 27 November 2020), 2012. 5, 6

[83] F. H.-F. Leung, H.-K. Lam, S.-H. Ling, and P. K.-S. Tam. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural networks*, 14(1):79–88, 2003. 18

[84] D. K. Lewis. Causal explanation. 1986. 12

[85] P. Lipton. Contrastive explanation. *Royal Institute of Philosophy Supplement*, 27:247–266, 1990. 1, 12

[86] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4114–4124. PMLR, 2019. 10, 21, 43

[87] F. Locatello, B. Poole, G. Rätsch, B. Schölkopf, O. Bachem, and M. Tschannen. Weakly-supervised disentanglement without compromises. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6348–6359. PMLR, 2020. 11, 45

[88] T. Lombrozo. The structure and function of explanations. *Trends in cognitive sciences*, 10(10):464–470, 2006. 12

[89] J. Lucas, G. Tucker, R. B. Grosse, and M. Norouzi. Don't blame the elbo! A linear VAE perspective on posterior collapse. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9403–9413, 2019. 24

[90] E. Machery. *Doing without concepts*. Oxford University Press, 2009. 14

[91] W. T. Maddox and F. G. Ashby. Comparing decision bound and exemplar models of categorization. *Perception & psychophysics*, 53(1):49–70, 1993. 14

[92] L. Magdalena. *Fuzzy Rule-Based Systems*, pages 203–218. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. 6

[93] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 5188–5196. IEEE Computer Society, 2015. 7

[94] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 10

[95] D. R. Mandel and D. R. Lehman. Counterfactual thinking and ascriptions of cause and preventability. *Journal of personality and social psychology*, 71(3):450, 1996. 13

[96] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. 16

[97] J. E. Mercado, M. A. Rupp, J. Y. Chen, M. J. Barnes, D. Barber, and K. Procci. Intelligent agent transparency in human–agent teaming for multi-uxv management. *Human factors*, 58(3):401–415, 2016. 1

[98] M. A. Mercioni and S. Holban. The most used activation functions: Classic versus current. In *2020 International Conference on Development and Application Systems (DAS)*, pages 141–145. IEEE, 2020. 16

[99] C. B. Mervis and E. Rosch. Categorization of natural objects. *Annual review of psychology*, 32(1):89–115, 1981. 14

[100] I. Mezo. *Combinatorics and number theory of counting sequences*. CRC Press, 2019. 35

[101] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019. 12, 13

[102] B. Millidge, A. Tschantz, and C. L. Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv preprint arXiv:2006.04182*, 2020. 18

[103] M. Minsky and S. A. Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017. 17

[104] J. Mu and J. Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33, 2020. 7

[105] A. M. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3387–3395, 2016. 6

[106] A. M. Nguyen, J. Yosinski, and J. Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016. 6

[107] R. M. Nosofsky. Exemplar-based approach to relating categorization, identification, and recognition. *Multidimensional models of perception and cognition*, pages 363–393, 1992. 14

[108] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018. 16

[109] S. Odaibo. Tutorial: Deriving the standard variational autoencoder (vae) loss function. *arXiv preprint arXiv:1907.08956*, 2019. 77

[110] K. Odajima, Y. Hayashi, G. Tianxia, and R. Setiono. Greedy rule generation from discrete data and its use in neural network rule extraction. *Neural Networks*, 21(7):1020–1028, 2008. 7

[111] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in. 6

[112] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. 6

[113] M. O'Shaughnessy, G. Canal, M. Connor, C. Rozell, and M. Davenport. Generative causal explanations of black-box classifiers. *Advances in Neural Information Processing Systems*, 33, 2020. 11

[114] A. Pati and A. Lerch. Attribute-based regularization of latent spaces for variational auto-encoders. *Neural Computing and Applications*, pages 1–16, 2020. 10

[115] M. Pazzani, A. Feghahati, C. Shelton, and A. R. Seitz. Explaining contrasting categories. In *IUI Workshops*, 2018. 7

[116] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 42

[117] E. M. Pothos and A. J. Wills. *Formal approaches in categorization*. Cambridge University Press, 2011. 13

[118] M. Prabhushankar, G. Kwon, D. Temel, and G. AlRegib. Contrastive explanations in neural networks. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3289–3293. IEEE, 2020. 8

[119] Z. Qin, F. Yu, C. Liu, and X. Chen. How convolutional neural networks see the world — a survey of convolutional neural network visualization methods. *arXiv preprint arXiv:1804.11191*, 2018. 6

[120] M. Ranney and P. Thagard. Explanatory coherence and belief revision in naive physics. Technical report, PITTSBURGH UNIV PA LEARNING RESEARCH AND DEVELOPMENT CENTER, 1988. 13

[121] B. Rehder. A causal-model theory of conceptual representation and categorization. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29(6):1141, 2003. 13

[122] B. Rehder. When similarity and causality compete in category-based property generalization. *Memory & Cognition*, 34(1):3–16, 2006. 13

[123] A. L. Rezaabad and S. Vishwanath. Learning representations by maximizing mutual information in variational autoencoders. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2729–2734. IEEE, 2020. 24, 45

[124] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org, 2014. 1, 10, 18

[125] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 8

[126] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1527–1535. AAAI Press, 2018. 8

[127] M. Riesenhuber. Object categorization in man, monkey, and machine: Some answers and some open questions. *Object categorization: Computer and human vision perspectives*, pages 216–240, 2009. 13

[128] M. Riesenhuber and T. Poggio. Models of object recognition. *Nature neuroscience*, 3(11):1199–1204, 2000. 13

[129] D. Ritchie, P. Horsfall, and N. D. Goodman. Deep amortized inference for probabilistic programs. *arXiv preprint arXiv:1610.05735*, 2016. 31

[130] M. J. Robeer. Contrastive explanation for machine learning. Master's thesis, 2018. 5, 8, 12, 13

[131] M. Rolinek, D. Zietlow, and G. Martius. Variational autoencoders pursue PCA directions (by accident). In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12406–12415. Computer Vision Foundation / IEEE, 2019. 10

[132] E. Rosch. Principles of categorization. *Concepts: core readings*, 189, 1999. 1, 13, 14

[133] E. Rosch, C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-Braem. Basic objects in natural categories. *Cognitive psychology*, 8(3):382–439, 1976. 14

[134] J. N. Rouder and R. Ratcliff. Comparing categorization models. *Journal of Experimental Psychology: General*, 133(1):63, 2004. 14

[135] D.-H. Ruben. *Explaining explanation*. Routledge, 2015. 12

[136] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. 17

[137] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991. 6

[138] W. C. Salmon. Four decades of scientific explanation.[part 6] conclusion: peaceful coexistence? 1989. 12

[139] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959. 1

[140] R. M. Schmidt, F. Schneider, and P. Hennig. Descending through a crowded valley—benchmarking deep learning optimizers. *arXiv preprint arXiv:2007.01547*, 2020. 18

[141] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 618–626. IEEE Computer Society, 2017. 7

[142] P. Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23):40, 2008. 36

[143] A. Serdega and D.-S. Kim. Vmi-vae: Variational mutual information maximization framework for vae with discrete and continuous priors. *arXiv preprint arXiv:2005.13953*, 2020. 24

[144] S. Serrano and N. A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy, July 2019. Association for Computational Linguistics. 9

[145] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. 43

[146] R. Shivhare and C. A. Kumar. On the cognitive process of abstraction. *Procedia Computer Science*, 89:243–252, 2016. 6

[147] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 2017. 7

[148] Z. Si and S.-C. Zhu. Learning and-or templates for object recognition and detection. *IEEE transactions on pattern analysis and machine intelligence*, 35(9):2189–2205, 2013. 9

[149] E. E. Smith and D. L. Medin. *Categories and concepts*, volume 9. Harvard University Press Cambridge, MA, 1981. 14

[150] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 22

[151] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017. 7

[152] J. Sweller and G. A. Cooper. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and instruction*, 2(1):59–89, 1985. 1

[153] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 30

[154] P. Thagard. Explanatory coherence. *Behavioral and brain sciences*, 12(3):435–502, 1989. 13

[155] J. J. Thiagarajan, B. Kailkhura, P. Sattigeri, and K. N. Ramamurthy. Treeview: Peeking into deep neural networks via feature-space partitioning. *arXiv preprint arXiv:1611.07429*, 2016. 7

[156] C. Thomas H., L. Charles E., R. Ronald L., and S. Clifford. *Introduction to Algorithms, Third Edition.*, volume 3rd ed. The MIT Press, 2009. 36, 83

[157] L. Tonnaer, L. A. P. Rey, V. Menkovski, M. Holenderski, and J. W. Portegies. Quantifying and learning disentangled representations with limited supervision. *arXiv preprint arXiv:2011.06070*, 2020. 10

[158] M. Tschannen, O. Bachem, and M. Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018. 5, 21

[159] J.-P. Vergne and T. Wry. Categorizing categorization research: Review, integration, and future directions. *Journal of Management Studies*, 51(1):56–94, 2014. 13

[160] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3156–3164. IEEE Computer Society, 2015. 9

[161] M. J. Vowels, N. C. Camgöz, and R. Bowden. Nestedvae: Isolating common factors via weak supervision. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9199–9209. IEEE, 2020. 45

[162] J. v. d. Waa, M. Robeer, J. van Diggelen, M. Brinkhuis, and M. Neerincx. Contrastive explanations with local foil trees. *arXiv preprint arXiv:1806.07470*, 2018. 8

[163] X. Wang, J. Jia, J. Yin, and L. Cai. Interpretable aesthetic features for affective image classification. In *2013 IEEE International Conference on Image Processing*, pages 3230–3234. IEEE, 2013. 6

[164] B. L. Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947. 47

[165] P. J. Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982. 17

[166] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 10

[167] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997. 27

[168] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008. 6

[169] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 842–850. IEEE Computer Society, 2015. 9

[170] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2048–2057. JMLR.org, 2015. 9

[171] J.-W. Yan, C.-S. Lin, F.-E. Yang, Y.-J. Li, and Y.-C. F. Wang. Semantics-guided representation learning with applications to visual synthesis. *arXiv preprint arXiv:2010.10772*, 2020. 26, 35

[172] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 7

[173] T. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984. 86

[174] Z. Zheng and L. Sun. Disentangling latent space for VAE by label relevant/irrelevant dimensions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12192–12201. Computer Vision Foundation / IEEE, 2019. 10, 21, 22, 23, 44

[175] B. Zhou, Y. Sun, D. Bau, and A. Torralba. Interpretable basis decomposition for visual explanation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018. 9

[176] X. Zhu, C. Xu, and D. Tao. Learning disentangled representations with latent variation predictability. In *European Conference on Computer Vision*, pages 684–700. Springer, 2020. 11

[177] J. R. Zilke, E. L. Mencía, and F. Janssen. Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, pages 457–473. Springer, 2016. 7

[178] Q. Zou. The log-convexity of the fubini numbers. *Transactions on Combinatorics*, 7(2):17–23, 2018. 35

Appendix A

The VAE Objective

In this appendix, we discuss the motivation for the VAE's objective, along with its derivation and a discussion of its implications.

In order to discuss the VAE objective, we first introduce the Kullback-Leibler (KL) divergence[75]. This divergence quantitively expresses the distance between two probability distributions according to their relative entropy, and is defined as follows:

$$KL(q(x)||p(x)) = \mathbb{E}_{q(x)} \log \frac{q(x)}{p(x)} \tag{A.1}$$

Following this, we motivate and define the objective function for the VAE. The VAE's objective is to model the distribution of some dataset X. We assume that X consists of independent and identically distributed samples of some variable x, and we assume that these samples are generated according to some unobserved continuous random variable z. Thus, we can define the distribution of x as follows:

$$p(x) = \int p(x|z)p(z)dz \tag{A.2}$$

Following these assumptions, we can define two objectives for the VAE:

- Approximating the true posterior $p_\theta(z|x)$ (inferring the 'real' latent space). To do so, we minimize the KL divergence between an approximated distribution we infer with the our encoder, $q_\phi(z|x)$, and the true posterior, $p_\theta(z|x)$:

$$\min_\phi \; KL(q_\phi(z|x)||p_\theta(z|x)) \tag{A.3}$$

- Maximizing the log-likelihood of the distribution we infer with our decoder, $\log p_\theta(x)$ (generating realistic data):

$$\max_\theta \; \log p_\theta(x) \tag{A.4}$$

The terms we seek to optimize for are $p_\theta(z|x)$ and $p_\theta(x)$. Note that we can express $p_\theta(z|x)$ as follows:

$$p_\theta(z|x) = \frac{p_\theta(z)p_\theta(x|z)}{p_\theta(x)} \tag{A.5}$$

As such, we require $p_\theta(x)$ to evaluate both presented objectives. Unfortunately, $p_\theta(x)$ is intractable: Evaluating Equation A.2 considers all possible values of z, i.e., all possible ways z could be constructed by our encoder (which is not reasonable to compute). As such, we cannot directly optimize for our objectives. Instead, we optimize for the Evidence Lower Bound ($ELBO$), a lower bound on the log-likelihood, $\log p_\theta(x)$. We show that maximizing the $ELBO$ maximizes the log-likelihood of our data and minimizes the KL divergence between the approximated and the true posterior. To derive the $ELBO$, we start from the KL divergence between the approximated and

true posterior. The derivation is as follows:

$$KL(q_\phi(z|x)||p_\theta(z|x)) = \mathbb{E}_{q_\phi(z|x)}[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}] \tag{A.6}$$
$$= -\mathbb{E}_{q_\phi(z|x)}[\log \frac{p_\theta(z|x)}{q_\phi(z|x)}] \tag{A.7}$$
$$= -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z|x) - \log q_\phi(z|x)] \tag{A.8}$$
$$= -\mathbb{E}_{q_\phi(z|x)}[\log \frac{p_\theta(z,x)}{p_\theta(x)} - \log q_\phi(z|x)] \tag{A.9}$$
$$= -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z,x) - \log p_\theta(x) - \log q_\phi(z|x)] \tag{A.10}$$
$$= -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z,x) - \log q_\phi(z|x)] + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x)] \tag{A.11}$$
$$= -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z,x) - \log q_\phi(z|x)] + \log p_\theta(x) \tag{A.12}$$
$$= -ELBO + \log p_\theta(x) \tag{A.13}$$

Giving us the following two definitions for the ELBO:

$$ELBO = \log p_\theta(x) - KL(q_\phi(z|x)||p_\theta(z|x)) \tag{A.14}$$
$$ELBO = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z,x) - \log q_\phi(z|x)] \tag{A.15}$$

Equation A.14 defines the $ELBO$ in terms of our original objectives, the intractable terms. As the KL divergence is always positive[109], we can infer that the $ELBO$ is a lower bound on the log-likelihood $\log p_\theta(x)$, meaning we can maximize the $ELBO$ in order to maximize $\log p_\theta(x)$. Furthermore, the gap between the $ELBO$ and $\log p_\theta(x)$ is precisely the KL divergence between the approximated and true posterior, $KL(q_\phi(z|x)||p_\theta(z|x))$. As such, maximizing the $ELBO$ also minimizes this divergence. Additionally, our lower bound approaches the true likelihood as this divergence approaches 0. We conclude that maximizing the $ELBO$ both encourages our decoder to generate realistic data and our encoder to try and approximate the true posterior distribution. In order to better understand the (tractable) $ELBO$, as defined in Equation A.15, we rewrite it as follows:

$$ELBO = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z,x) - \log q_\phi(z|x)] \tag{A.16}$$
$$= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z,x) - \log p_\theta(z) + \log p_\theta(z) - \log q_\phi(z|x)] \tag{A.17}$$
$$= \mathbb{E}_{q_\phi(z|x)}[\log \frac{p_\theta(z,x)}{p_\theta(z)} + \log p_\theta(z) - \log q_\phi(z|x)] \tag{A.18}$$
$$= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z) + \log p_\theta(z) - \log q_\phi(z|x)] \tag{A.19}$$
$$= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z) - \log q_\phi(z|x)] + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \tag{A.20}$$
$$= -\mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(z)] + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \tag{A.21}$$
$$= -\mathbb{E}_{q_\phi(z|x)}[\log \frac{q_\phi(z|x)}{p_\theta(z)}] + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \tag{A.22}$$
$$= -KL(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \tag{A.23}$$

Equation A.23 gives us the desired format for the $ELBO$. When taken as a loss function, we must take the negative $ELBO$, as a loss function is necessarily minimized (whereas we seek to maximize the $ELBO$). As such, we define the loss function, the objective of the VAE, as follows:

$$\mathcal{L}_{\theta,\phi}(x) = \underbrace{KL(q_\phi(z|x)||p_\theta(z))}_{\text{Regularization cost}} \underbrace{-\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{Reconstruction cost}} \tag{A.24}$$

Appendix B

Interpolation-Graph Complexity

In this appendix, we derive the size of the interpolation graph. To do so, we consider the graph as the (discrete, dimension-wise) interpolation of some vector a to some vector b, where $|a| = |b| = n$. We consider nodes as intermediate states in this transition, where each state indicates a vector of size n. For each dimension i in this vector, it either takes the value of a_i or b_i, where the former indicates it has not been changed yet, and the latter indicates it has changed to its 'final' value. As we can change any number of dimensions from a to b at a time, each edge depicts some number of dimensions being changed.

The number of nodes, $|V|$, is 2^n. Each node in the graph considers one interpolation state, where for each dimension in this state, we pick the value from either a or b. As such, there are 2^n possible combinations, giving us $|V| = 2^n$.

The number of edges, $|E|$, is $3^n - 2^n$. We can derive this number by reasoning about the number of edges that are attached to each node. Each node has an outgoing edge towards all nodes that (in comparison) have changed some dimensions(s) from a_i to b_i, and have not changed any dimensions from b_i to a_i. As such, given a node with k dimensions already changed to b_i, we have $n - k$ dimensions that are yet to be changed. For each combination of $n - k$ changes we add an edge to the corresponding node, giving $2^{n-k} - 1$ outgoing edges for a node with k already-changed values (we subtract one as we do not consider the case where none of these $n-k$ values are changed). The number of nodes that exist for each such k is the number of combinations of size k (the number of values we changed), given n values to change in total: $\binom{n}{k}$. The total number of edges can then be expressed by summing all values of k. This sum is as follows:

$$|E| = \sum_{k=0}^{n} \left[\binom{n}{k} (2^{n-k} - 1) \right] \tag{B.1}$$

To rewrite this equation, we consider the Binomial Theorem[43]:

$$(x + y)^n = \sum_{k=0}^{n} \left[\binom{n}{k} x^{n-k} y^k \right] \tag{B.2}$$

The resulting derivation is as follows:

$$|E| = \sum_{k=0}^{n} \left[\binom{n}{k} (2^{n-k} - 1) \right] \tag{B.3}$$

$$= \sum_{k=0}^{n} \left[\binom{n}{k} (2^{n-k}) \right] - \sum_{k=0}^{n} \left[\binom{n}{k} \right] \tag{B.4}$$

$$= 3^n - 2^n \tag{B.5}$$

giving us the number of edges, $|E| = 3^n - 2^n$.

Appendix C

Algorithms

In this appendix, we provide pseudocode for all algorithms. These algorithms consider the training procedures for all models and the explanation-generation process. The utilized (loss-calculation) functions are as follows[1]:

$$SE(x_a, x_b) = (x_a - x_b)^2 \tag{C.1}$$

$$KL(\mu, \sigma) = \frac{1}{2}(\mu^2 + \sigma^2 - \log \sigma^2 - 1) \tag{C.2}$$

$$BCE(y, \widetilde{y}) = -[y \log(\widetilde{y}) + (1 - y) \log(1 - \widetilde{y})] \tag{C.3}$$

$$CCE(y, \widetilde{y}) = -\sum_{i}^{n} y_i \log(\widetilde{y}_i) \tag{C.4}$$

Algorithm C.1: VAE training procedure

Input: X – Datapoints we seek to model
Params: ϕ – $q_\phi(z|x)$, the encoder
θ – $p_\theta(x|z)$, the decoder

$\{\phi, \theta\} \leftarrow$ initialize (e.g., uniformly random)
for each epoch **do**
  for each batch in epoch **do**
    $x \sim X$ ▷ `sample batch of data`
    $(\mu, \sigma) = q_\phi(z|x)$ ▷ `infer latent parameters`
    $\epsilon \sim \mathcal{N}(0, I)$ ▷ `reparametrization trick`
    $z = \mu + \sigma \odot \epsilon$
    $\widetilde{x} = p_\theta(x|z)$ ▷ `reconstruct data`
    $\mathcal{L} = KL(\mu, \sigma) + SE(x, \widetilde{x})$ ▷ $\mathcal{L}_{ELBO}$
    Compute $\nabla_{\phi,\theta}\mathcal{L}$ ▷ `gradient computation`
    Update all parameters according to gradients, using optimizer
  end
end

[1]Note that $KL(\mu, \sigma)$ is not the computation of any arbitrary KL divergence. Rather, it is the analytical computation of $KL(q||p)$, where both q and p are assumed to be factorized normal distributions, p is the standard factorized normal distribution $\mathcal{N}(0, I)$, and $q = \mathcal{N}(\mu, \sigma)$.

Algorithm C.2: Disentangled VAE training procedure

Input: (X, Y) – Datapoints and their corresponding class-labels
Args: α – Label-classification disentanglement weight
β_0 – z_y regularization weight
β_1 – z_x regularization weight
γ – Adverse label-classification disentanglement weight
Params: ϕ_0 – $q_{\phi_0}(z_y|x)$, the z_y encoder
ϕ_1 – $q_{\phi_1}(z_x|x)$, the z_x encoder
θ – $p_\theta(x|z_y, z_x)$, the (combined) decoder
ψ_0 – $c_{\psi_0}(y|z_y)$, the z_y classifier
ψ_1 – $c_{\psi_1}(y|z_x)$, the z_x classifier

$\{\phi_0, \phi_1, \theta, \psi_0, \psi_1\} \leftarrow$ initialize (e.g., uniformly random)
for each epoch **do**
  for each batch in epoch **do**
    $\mathcal{L} = 0$ ▷ initialize loss per batch
    $(x, y) \sim (X, Y)$ ▷ sample batch of data+label
    $(\mu_y, \sigma_y) = q_{\phi_0}(z_y|x)$ ▷ infer latent parameters
    $(\mu_x, \sigma_x) = q_{\phi_1}(z_x|x)$
    $\mathcal{L}$ += $\beta_0 KL(\mu_y, \sigma_y)$ ▷ $\mathcal{L}_{kl_0}$
    $\mathcal{L}$ += $\beta_1 KL(\mu_x, \sigma_x)$ ▷ $\mathcal{L}_{kl_1}$
    $\epsilon \sim \mathcal{N}(0, I)$ ▷ reparametrization trick
    $z_y = \mu_y + \sigma_y \odot \epsilon_0$ ▷ ϵ is independent for every z
    $z_x = \mu_x + \sigma_x \odot \epsilon_1$
    $\widetilde{y}_y = c_{\psi_0}(y|z_y))$ ▷ predict class using inferred sample
    $\widetilde{y}_x = c_{\psi_1}(y|z_x))$
    $\mathcal{L}$ += $\alpha CCE(y, \widetilde{y}_y)$ ▷ $\mathcal{L}_{classy}$
    $\mathcal{L}$ += $\gamma[-CCE(y, \widetilde{y}_x) \vee CCE(\frac{1}{n}, \widetilde{y}_x)]$ ▷ $\mathcal{L}_{advx}$ (choice)
    $\mathcal{L}_{cx} = CCE(y, \widetilde{y}_x)$ ▷ $\mathcal{L}_{classx}$ (applied separately)
    $\widetilde{x} = p_\theta(x|z_y, z_x)$ ▷ reconstruct data
    $\mathcal{L}$ += $SE(x, \widetilde{x})$ ▷ $\mathcal{L}_{rec}$

    Compute $\nabla_{\phi_0,\phi_1,\theta,c\psi_0}\mathcal{L}$ and $\nabla_{c\psi_1}\mathcal{L}_{cx}$ ▷ gradient computation
    Update all parameters according to gradients, using optimizer
  end
end

Algorithm C.3: *CD* training procedure

Input: $((X_a, X_b), Y)$ – Change-pairs and their corresponding labels
(X_{cl}, Y_{cl}) – Class-examples and their corresponding labels

Args: α_c – Change-pair classification weight
α – Label-classification disentanglement weight
β_0 – z_y regularization weight
β_1 – z_x regularization weight
γ – Adverse label-classification disentanglement weight

Params: $c\phi_0$ – $CD\text{-}ENC_y$, the z_y encoder model
$c\phi_1$ – $CD\text{-}ENC_x$, the z_x encoder model
$c\theta$ – $CD\text{-}DEC$, the (combined) decoder model
$c\omega$ – $DISC$, the latent-difference discriminator
$c\psi_0$ – $CD\text{-}CY$, the z_y classifier
$c\psi_1$ – $CD\text{-}CX$, the z_x classifier

$\{c\phi_0, c\phi_1, c\theta, c\omega, c\psi_0, c\psi_1\} \leftarrow$ initialize (e.g., uniformly random)

for each epoch **do**
  for each batch in epoch **do**
    $\mathcal{L} = 0$ ▷ `initialize loss per batch`
    $((x_a, x_b), y) \sim ((X_a, X_b), Y)$ ▷ `sample batch of pair+label`
    $(\mu_{y_a}, \sigma_{y_a}) = CD\text{-}ENC_y(x_a)$ ▷ `infer latent parameters`
    $(\mu_{x_a}, \sigma_{x_a}) = CD\text{-}ENC_x(x_a)$
    $(\mu_{y_b}, \sigma_{y_b}) = CD\text{-}ENC_y(x_b)$
    $(\mu_{x_b}, \sigma_{x_b}) = CD\text{-}ENC_x(x_b)$
    $\mathcal{L} \mathrel{+}= \beta_0[KL(\mu_{y_a}, \sigma_{y_a}) + KL(\mu_{y_b}, \sigma_{y_b})]$ ▷ $\mathcal{L}_{kl_0}$
    $\mathcal{L} \mathrel{+}= \beta_1[KL(\mu_{x_a}, \sigma_{x_a}) + KL(\mu_{x_b}, \sigma_{x_b})]$ ▷ $\mathcal{L}_{kl_1}$
    $\epsilon \sim \mathcal{N}(0, I)$ ▷ `reparametrization trick`
    $z_{y_a} = \mu_{y_a} + \sigma_{y_a} \odot \epsilon_0$ ▷ ϵ `is independent for every` z
    $z_{x_a} = \mu_{x_a} + \sigma_{x_a} \odot \epsilon_1$
    $z_{y_b} = \mu_{y_b} + \sigma_{y_b} \odot \epsilon_2$
    $z_{x_b} = \mu_{x_b} + \sigma_{x_b} \odot \epsilon_3$
    $\widetilde{y} = DISC(|z_{y_a} - z_{y_b}|)$ ▷ `infer change-quality prediction`
    $\mathcal{L} \mathrel{+}= \alpha_c BCE(y, \widetilde{y})$ ▷ $\mathcal{L}_{CDISC}$
    $\widetilde{x}_a = CD\text{-}DEC(z_{y_a}, z_{x_a})$ ▷ `reconstruct datapoints`
    $\widetilde{x}_b = CD\text{-}DEC(z_{y_b}, z_{x_b})$
    $\mathcal{L} \mathrel{+}= [SE(x_a, \widetilde{x}_a) + SE(x_b, \widetilde{x}_b)]$ ▷ $\mathcal{L}_{rec}$

    $(x_{cl}, y_{cl}) \sim (X_{cl}, Y_{cl})$ ▷ `sample batch of data+class-label`
    $(\mu_{y_{cl}}, \sigma_{y_{cl}}) = CD\text{-}ENC_y(x_{cl})$ ▷ `infer latent parameters`
    $(\mu_{x_{cl}}, \sigma_{x_{cl}}) = CD\text{-}ENC_x(x_{cl})$
    $\mathcal{L} \mathrel{+}= \beta_0 KL(\mu_{y_{cl}}, \sigma_{y_{cl}})$ ▷ $\mathcal{L}_{kl_0}$
    $\mathcal{L} \mathrel{+}= \beta_1 KL(\mu_{x_{cl}}, \sigma_{x_{cl}})$ ▷ $\mathcal{L}_{kl_1}$
    $\epsilon \sim \mathcal{N}(0, I)$ ▷ `reparametrization trick`
    $z_{y_{cl}} = \mu_{y_{cl}} + \sigma_{y_{cl}} \odot \epsilon_0$ ▷ ϵ `is independent for every` z
    $z_{x_{cl}} = \mu_{x_{cl}} + \sigma_{x_{cl}} \odot \epsilon_1$
    $\widetilde{y}_{cl\text{-}y} = CD\text{-}CY(z_{y_{cl}})$ ▷ `predict class using inferred sample`
    $\widetilde{y}_{cl\text{-}x} = CD\text{-}CX(z_{x_{cl}})$
    $\mathcal{L} \mathrel{+}= \alpha CCE(y_{cl}, \widetilde{y}_{cl\text{-}y})$ ▷ $\mathcal{L}_{classy}$
    $\mathcal{L} \mathrel{+}= \gamma[-CCE(y_{cl}, \widetilde{y}_{cl\text{-}x}) \vee CCE(\frac{1}{n}, \widetilde{y}_{cl\text{-}x})]$ ▷ $\mathcal{L}_{advx}$ `(choice)`
    $\mathcal{L}_{cx} = CCE(y_{cl}, \widetilde{y}_{cl\text{-}x})$ ▷ $\mathcal{L}_{classx}$ `(applied separately)`
    $\widetilde{x}_{cl} = CD\text{-}DEC(z_{y_{cl}}, z_{x_{cl}})$ ▷ `reconstruct the datapoint`
    $\mathcal{L} \mathrel{+}= SE(x_{cl}, \widetilde{x}_{cl})$ ▷ $\mathcal{L}_{rec}$

    Compute $\nabla_{c\phi_0, c\phi_1, c\theta, c\omega, c\psi_0}\mathcal{L}$ and $\nabla_{c\psi_1}\mathcal{L}_{cx}$ ▷ `gradient computation`
    Update all parameters according to gradients, using optimizer
  end
end

Algorithm C.4: Main model training procedure

Input: (X, Y) – Datapoints and their corresponding class-labels
X_r – 'Real' datapoints (from class-examples and change-pairs)

Args: α_r – Realism classification weight
α_p – Change-quality classification weight
α – Label-classification disentanglement weight
β_0 – z_y regularization weight
β_1 – z_x regularization weight
γ – Adverse label-classification disentanglement weight

Params: ϕ_0 – ENC_y, the z_y encoder model
ϕ_1 – ENC_x, the z_x encoder model
θ – DEC, the (combined) decoder model
ω – D, the realism discriminator
ψ_0 – CY, the z_y classifier
ψ_1 – CX, the z_x classifier

$\{\phi_0, \phi_1, \theta, \omega, \psi_0, \psi_1\} \leftarrow$ initialize (e.g., uniformly random)
for each epoch **do**
  for each batch in epoch **do**
    $\mathcal{L} = 0$ ▷ initialize loss per batch
    $(x_a, y_a) \sim (X, Y),\ (x_b, y_b) \sim (X, Y)$ ▷ sample 2 batches of data+label
    $(\mu_{y_a}, \sigma_{y_a}) = ENC_y(x_a)$ ▷ infer latent parameters
    $(\mu_{x_a}, \sigma_{x_a}) = ENC_x(x_a)$
    $(\mu_{y_b}, \sigma_{y_b}) = ENC_y(x_b)$
    $(\mu_{x_b}, \sigma_{x_b}) = ENC_x(x_b)$
    $\mathcal{L} \mathrel{+}= \beta_0[KL(\mu_{y_a}, \sigma_{y_a}) + KL(\mu_{y_b}, \sigma_{y_b})]$ ▷ $\mathcal{L}_{kl_0}$
    $\mathcal{L} \mathrel{+}= \beta_1[KL(\mu_{x_a}, \sigma_{x_a}) + KL(\mu_{x_b}, \sigma_{x_b})]$ ▷ $\mathcal{L}_{kl_1}$
    $\epsilon \sim \mathcal{N}(0, I)$ ▷ reparametrization trick
    $z_{y_a} = \mu_{y_a} + \sigma_{y_a} \odot \epsilon_0$ ▷ ϵ is independent for every z
    $z_{x_a} = \mu_{x_a} + \sigma_{x_a} \odot \epsilon_1$
    $z_{y_b} = \mu_{y_b} + \sigma_{y_b} \odot \epsilon_2$
    $z_{x_b} = \mu_{x_b} + \sigma_{x_b} \odot \epsilon_3$
    $\widetilde{y}_{a\text{-}y} = CY(z_{y_a})$ ▷ predict class using inferred sample
    $\widetilde{y}_{a\text{-}x} = CX(z_{x_a})$
    $\widetilde{y}_{b\text{-}y} = CY(z_{y_b})$
    $\widetilde{y}_{b\text{-}x} = CX(z_{x_b})$
    $\mathcal{L} \mathrel{+}= \alpha[CCE(y_a, \widetilde{y}_{a\text{-}y}) + CCE(y_b, \widetilde{y}_{b\text{-}y})]$ ▷ $\mathcal{L}_{classy}$
    $\mathcal{L} \mathrel{+}= \gamma[\ (-CCE(y_a, \widetilde{y}_{a\text{-}x}) - CCE(y_b, \widetilde{y}_{b\text{-}x})) \vee$ ▷ $\mathcal{L}_{advx}$ (choice)
    $\quad (CCE(\frac{1}{n}, \widetilde{y}_{a\text{-}x}) + CCE(\frac{1}{n}, \widetilde{y}_{b\text{-}x}))\ ]$
    $\mathcal{L}_{cx} = CCE(y_a, \widetilde{y}_{a\text{-}x}) + CCE(y_b, \widetilde{y}_{b\text{-}x})$ ▷ $\mathcal{L}_{classx}$ (applied separately)
    $\widetilde{x}_a = DEC(z_{y_a}, z_{x_a})$ ▷ reconstruct datapoints
    $\widetilde{x}_b = DEC(z_{y_b}, z_{x_b})$
    $\mathcal{L} \mathrel{+}= [SE(x_a, \widetilde{x}_a) + SE(x_b, \widetilde{x}_b)]$ ▷ $\mathcal{L}_{rec}$

    Construct $z_{y_{pa}}, z_{y_{pb}}$ using z_{y_a}, z_{y_b} s.t. 1 dimension differs
    $\widetilde{x}_{p_a} = DEC(z_{y_{pa}}, z_{x_a})$ ▷ reconstruct datapoints with shared z_x
    $\widetilde{x}_{p_b} = DEC(z_{y_{pb}}, z_{x_a})$
    $\mathcal{L} \mathrel{+}= \alpha_r[BCE(1, D(\widetilde{x}_{p_a})) + BCE(1, D(\widetilde{x}_{p_b}))]$ ▷ $\mathcal{L}_{REAL}$
    $\mathcal{L} \mathrel{+}= \alpha_p |z_{y_{pa}} - z_{y_{pb}}| BCE(1, CD(\widetilde{x}_{p_a}, \widetilde{x}_{p_b}))$ ▷ $\mathcal{L}_{PAIR}$

    $x_{r_a} \sim X_r,\ x_{r_b} \sim X_r$ ▷ sample batch of real datapoints
    $\mathcal{L}_D = BCE(0, D(\widetilde{x}_{p_a})) + BCE(0, D(\widetilde{x}_{p_b})) +$ ▷ $\mathcal{L}_D$
    $\quad BCE(1, D(x_{r_a})) + BCE(1, D(x_{r_b}))$

    Compute $\nabla_{\phi_0, \phi_1, \theta, c\psi_0} \mathcal{L}$, $\nabla_{c\psi_1} \mathcal{L}_{cx}$ and $\nabla_\omega \mathcal{L}_D$ ▷ gradient computation
    Update all parameters according to gradients, using optimizer
  end
end

Algorithm C.5: Explanation generation procedure

Input: X – Dataset
Y – Dataset class-labels (optional)
x_a – To-be-explained datapoint
y – Foil-class to contrast against (optional)

Args: t – Classification probability threshold
α – Realism weight
β – Change quality weight
γ – Normalization exponent, $0 \leq \gamma \leq 1$

Output: A sequence of datapoints $(\widetilde{x}_0, \widetilde{x}_1, \ldots, \widetilde{x}_m)$ denoting the interpolation between x_a and an exemplar of class y, and the class-predictions for $\widetilde{x}_0$ and $\widetilde{x}_m$.

$z_{y_a} = ENC_y(x_a)$[1]
if y does not exist **then** ▷ `if foil not provided`
  $y = 2^{nd}$-highest $\in CY(z_{y_a})$ ▷ `contrast against` 2^{nd} `most probable class`
$z_{y_b} = NONE$
$dist_b = \infty$ ▷ `find exemplar`
for $i \in |X|$ **do** ▷ `consider all datapoints`
  $z_{y_i} = ENC_y(X[i])$ ▷ `get datapoint embedding`[2]
  confidence $= CY(z_{y_i})[y]$ ▷ $p(z_{y_i}$ `depicts class` $y)$
  if confidence $< t$ **then** *continue* ▷ `skip data not reaching threshold`
  if Y exists **then** ▷ `if` Y `is supplied`
    if $Y[i] \neq y$ **then** *continue* ▷ `skip incorrectly classified data`
  $dist = SE(z_{y_a}, z_{y_i})$ ▷ `check if` z_{y_i} `is the closest`[3]
  if $dist < dist_b$ **then**
    $dist_b = dist$
    $z_{y_b} = z_{y_i}$
end

Initialize interpolation graph $G = (V, E)$ ▷ `find explanation path`
An interpolation state depicts a dimension-wise combination of z_{y_a} and z_{y_b}
An interpolation step depicts some dimensions being changed from z_{y_a} to z_{y_b}
for each interpolation state z_{y_i} **do** ▷ 2^n ($n =$ `no. dimensions`)
  $V.add(z_{y_i})$
  $\widetilde{x}_i = DEC(z_{y_i}, z_x)$ ▷ `compute reconstruction`
  $r_i = 1 - D(\widetilde{x}_i)$ ▷ `compute` $1 - p(\widetilde{x}_i = real)$
  cache values $\widetilde{x}_i$ and r_i ▷ `store values for later use`
end
for each interpolation step from z_{y_i} to z_{y_j} **do** ▷ $3^n - 2^n$
  $c_{ij} = 1 - CD(\widetilde{x}_i, \widetilde{x}_j)$ ▷ `compute` $1 - p(\widetilde{x}_i, \widetilde{x}_j =$ `good change`$)$
  $k =$ no. dimensions changed
  $w_{ij} = \frac{\alpha r_j + \beta c_{ij}}{k^\gamma}$ ▷ `compute total edge weight`
  $E.add((z_{y_i}, z_{y_j}), w_{ij})$
end
path $= shortest\_path(G,\ source = z_{y_a},\ target = z_{y_b})$) ▷ $\Theta(|V| + |E|)$[156]
interpolation $= [\,]$
$z_x = ENC_x(x_a)$ ▷ `take remaining variables from` x_a
for z_{y_i} in path **do** ▷ `generate explanation`
  interpolation.append(DEC(z_{y_i}, z_x))
end
pred$_a = CY(z_{y_a}).max()$ ▷ `infer predicted classes and`
pred$_b = CY(z_{y_b}).max()$ ▷ `associated probabilities`
return interpolation, pred$_a$, pred$_b$

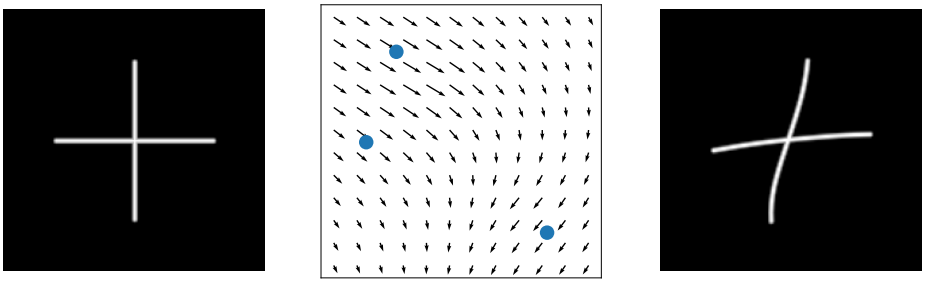
[1] We use the inferred mean during explanation-generation: $ENC_y(x)$ denotes the inferred mean, we ignore the standard deviation.
[2] In the case of multiple explanations we cache these embeddings and predictions (for performance reasons).
[3] Finding the smallest squared error is equivalent to finding the smallest Euclidean distance (which is the minimum-distance target [Section 5.3]).

Appendix D

Synthetic Data Generation

In this appendix, we describe how synthetic datapoints are generated according to the classes and shapes as described in Section 6.1.1. These base-shapes are transformed into datapoints by first distorting them, after which an image is generated by using these distorted shapes and some line-thickness 'field'.

We first consider the shape-distortion procedure. As an example, we consider generating a datapoint of class 0, a +, as depicted in Figure D.1a. This shape is distorted according to a (randomly generated) 'vector field' that determines how much each piece of the shape has to move. Such a field consists of a normalizing force (i.e., no movement) and 3 to 5 'vector' points, each placed at a (uniformly distributed) random position. The directions of these vectors (their x and y component[1]) are uniformly randomly selected and normalized. Furthermore, each point is assigned a random weight ($\sim \mathcal{U}(3, 6)$). The direction of each point in the field is calculated according to a weighted average of all vector-directions and the (0, 0) force. Each weight is determined by $w \cdot \frac{1}{dist+1}$, where w depicts the weight-parameter and $dist$ the euclidean distance of the point to the vector. An example of such a vector field is depicted in Figure D.1b. The shape is distorted by computing the offset of the shape's pieces according to this vector field, scaled according to an image-level weight ($\sim \mathcal{U}(0.3, 0.6)$). In practice, each line consists of a subdivision of 20 evenly-spaced segments, of which the start- and end-coordinates are transformed. The transformed shape is additionally rotated slightly ($\sim \mathcal{N}(0, \frac{\pi}{40})$). The new shape is centered and normalized to keep the size and position of the data constant. The result of this distortion process is depicted in Figure D.1c.



(a) The input shape of class 0.

(b) The vector field. Each blue dot depicts a 'force' in this field.

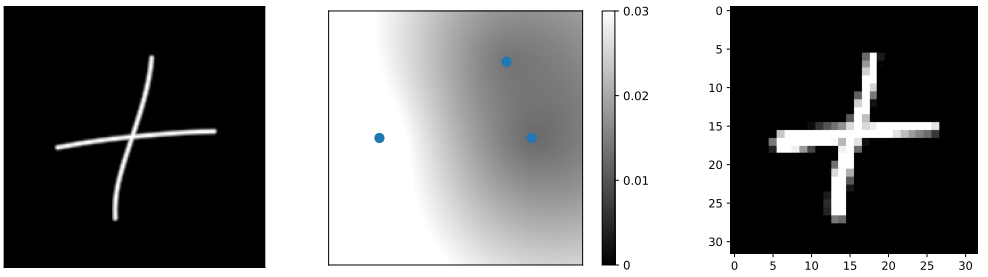
(c) The resulting shape, distorted, rotated and normalized.

Figure D.1: The distortion step of the data generation process.

Next, we generate a line-based image according to the distorted shape. Rather than applying a uniform line-width, we (slightly) vary the width throughout the image. This variation is produced by generating some stroke-thickness 'field' that determines this line-width (as a proportion of the total image) at any given point. This field is generated similarly to the vector field. We sample

[1]Determining a random vector according to uniformly distributed individual components is slightly biased towards diagonal directions, however as we are dealing with noise (and the bias is equal in all directions), this slight bias does not cause any issues.

a base stroke-thickness ($\sim \mathcal{U}(0.001, 0.05)$) and create 3 to 5 random points. Each of these points comes with its own stroke-thickness ($\sim \mathcal{U}(0.001, 0.05)$), weight ($\sim \mathcal{U}(3, 6)$), and decay exponent (exponent $\in \{2, 3, 4\}$). The stroke-thickness at each point in the image is then determined by a weighted average of this base thickness and the thickness of each point. Each point's weight is determined by $w \cdot e^{-exp \cdot dist}$, where w depicts the weight-parameter, exp the decay exponent, and $dist$ the euclidean distance between the stroke-point and the point we compute. An example of such a stroke-field is depicted in Figure D.2c. The final image is created by computing the distance of each pixel to its nearest line and checking with the stroke-field whether the pixel should be drawn as a line or not (i.e., a white pixel or a black pixel). On the edge of lines (in a prespecified range of 0.04), we linearly interpolate between white and black in order to create smooth edges. Following this process we can generate images of arbitrary resolution; an example of a 32x32 image is depicted in Figure D.2.



(a) The shape we draw.

(b) The stroke-thickness field. Each blue dot depicts a randomly generated 'weight' in this field. The legend indicates the width of the stroke (as proportion of the image) at a given point.

(c) The resulting 32 × 32-pixels datapoint, generated using the distorted shape and the stroke-thickness field.

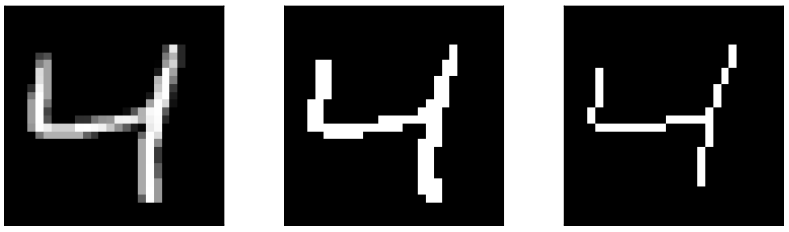
Figure D.2: Creating the final image according to the stroke-thickness.

Appendix E

MNIST Line-Augmentation

In this appendix, we describe how we split MNIST datapoints into individual lines. We seek to identify smooth curves, which we define as a sequence of points where the angle of the consecutive points (with respect to a horizontal line) does not change faster than some predefined limit.

As an example, we split a four into its underlying lines. This datapoint is depicted in Figure E.1a. First, we reduce this image to thin lines, following the same procedure as [26]. This procedure considers first thresholding the image, that is, marking all pixels above a specific value. The threshold is found by starting at 0 and increasing it iteratively until either the number of remaining (white) pixels drops under 50%, the number of 4-connected or 8-connected components changes, or the threshold reaches 250 (or ≈ 0.98 if pixels are denoted as values from $[0, 1]$ instead of $[0, 255]$). The result of this thresholding process is depicted in Figure E.1b. This (binary) image is thinned using the Zhang-Suen thinning algorithm[173], of which the result is depicted in Figure E.1c.



(a) The input image, a four.

(b) The thresholded image.

(c) The thinned image.

Figure E.1: The thinning steps of augmenting MNIST-digits.

To create lines from this representation, we considering grouping neighboring white pixels together. We consider 8-connected pixels as neighbors, i.e., the 8 pixels directly surrounding a given pixel. First, we identify the starting points of lines. We denote a starting pixel as a pixel with only one neighbor, where this neighboring pixel is connected to both the starting pixel and one other pixel. The first condition marks pixels likely to be the start point of a line, whereas the latter ensures that the starting pixel is not an 'outlier' pixel connected to a pixel in the middle of a line. If no such starting pixels can be found, the condition is relaxed to only the first condition. If still no starting pixels can be identified, we mark all pixels with the maximum number of neighbors as starting pixels (as these pixels likely lie on intersections of lines). The starting pixels for the thinned 4-image are depicted in Figure E.2.

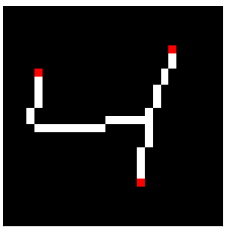


Figure E.2: The starting pixels of the thinned 4, marked as red pixels.

Lines are identified by starting at an (arbitrary) starting pixel and iteratively adding pixels such that the overall line does not curve faster than some set limit ($\frac{\pi}{4}$, in our case). First, an arbitrary neighbor of the starting pixel is added to the line (in most cases, the starting pixel only has one neighbor). The angle between these two points (and a horizontal line) determines the line's current angle. Then, points are iteratively added by checking the last-added point's neighbors and adding the neighbor such that the angle of the line changes the least (ignoring already-added neighbors). The line-angle is continuously updated according to the last 3 points added. When no neighboring points can be found such that the angle-change does not exceed our chosen limit, the line is complete. The line-creation process is depicted in Figure E.3.

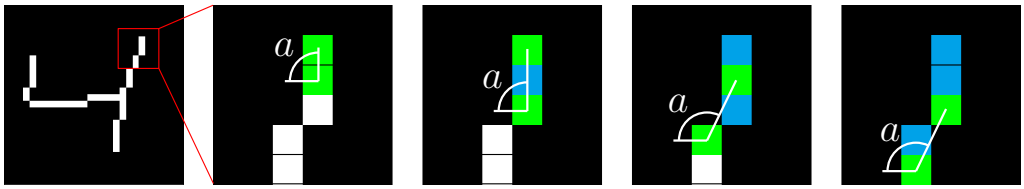
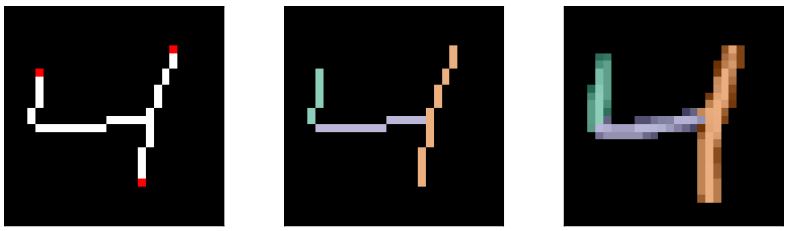


Figure E.3: The line-creation process. We iteratively add neighbors such that angle a changes the least. Green pixels indicate the pixels currently determining the line-angle, whereas blue pixels indicate the remaining pixels added to the line.

This line-creation process is repeated for all starting pixels. When all starting pixels are processed, lines are started from arbitrary points that neighbor already-found lines (given that we have remaining unmapped pixels). Finally, outlier points (pixels with no neighbors) are added to their closest line. As the entire process can be noisy, it is possible that insignificant 'lines' were identified. As a post-processing step, we add these small lines (≤ 3 pixels) to their largest neighboring line (if a line is surrounded by only small lines, we add it to its neighbor-lines' neighbor, and so on). To conclude, we create the line-split of the source digit. Pixels of the (unmodified) datapoint are mapped to lines by assigning them to the line with the minimal Euclidean distance (to any pixel on said line). The result of this line-generation process is depicted in Figure E.4.



(a) The identified starting points.

(b) The identified lines. Each color indicates a different line.

(c) The final line-split. Each color indicates a different line.

Figure E.4: The line-generation step of augmenting MNIST-digits.

Appendix F

MNIST Change-Pairs

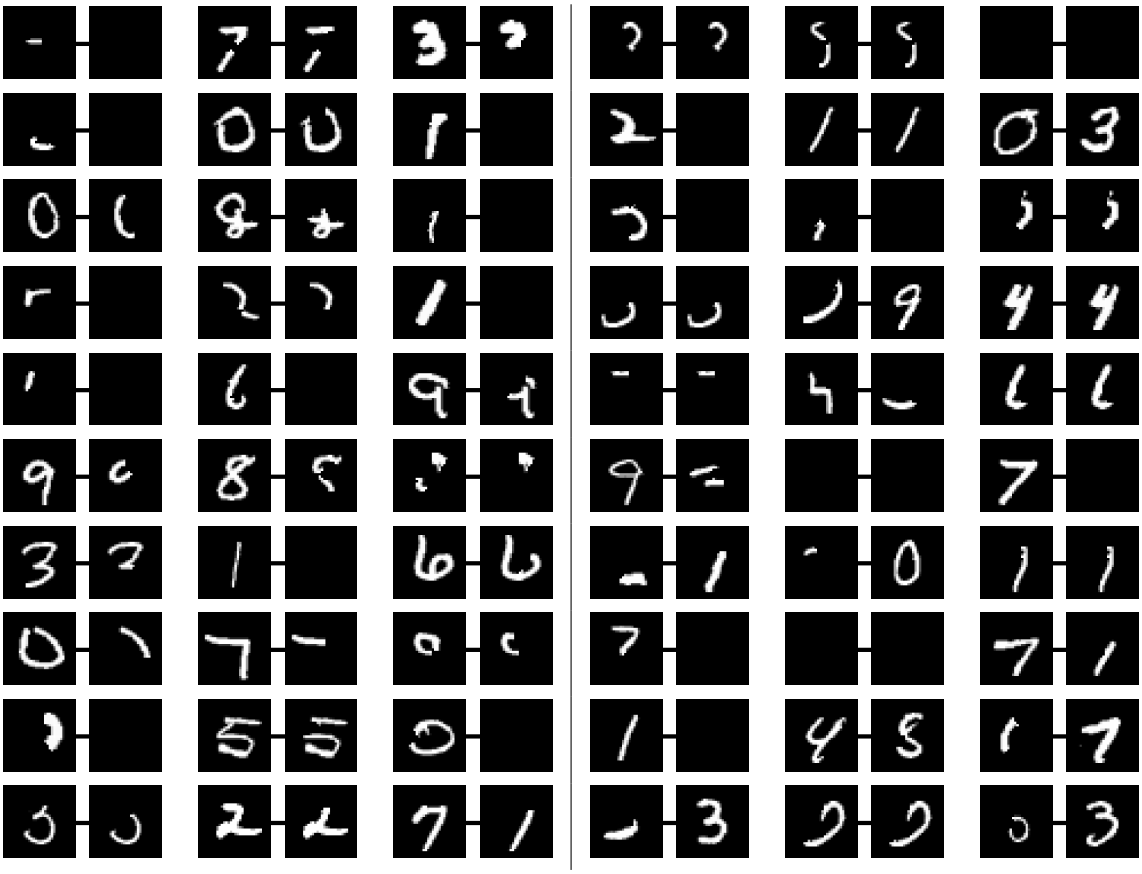


Figure F.1: Sixty MNIST change-pairs. Positive pairs are depicted in the three leftmost columns, whereas negative pairs are depicted in the three rightmost columns. While the positive pairs exhibit changes in line-occurrence, they are rather noisy.

Appendix G

Method Implementation

In this appendix, we describe the implementation of all methods we consider in the experimental evaluation. We provide the model architectures, the hyperparameter selection process, and minor implementation details.

G.1 Architecture

This section denotes all model architectures. The shared components are present in DVAE, LVAE, GVAE, ADA-GVAE, and VAE-CE. Method-specific components are denoted as such. The number of latent dimensions for all model types is set to 8 per subspace ($|z_y| = |z_x| = 8$). For CD, the number of latent dimensions per subspace is set to 16. All convolutional layers use 'same' padding, keeping the activation-map resolution consistent.

G.1.1 Shared Components

	Layer	**Activation**	**Output**
x	Input ($32 \times 32 \times 1$)		($32 \times 32 \times 1$)
1	4×4 Convolution, stride 2, 32 filters	LeakyReLU(0.1)	($16 \times 16 \times 32$)
2	Batch Normalization		($16 \times 16 \times 32$)
3	4×4 Convolution, stride 1, 64 filters	LeakyReLU(0.1)	($16 \times 16 \times 64$)
4	Batch Normalization		($16 \times 16 \times 64$)
5	4×4 Convolution, stride 1, 128 filters	LeakyReLU(0.1)	($16 \times 16 \times 128$)
6	Batch Normalization		($16 \times 16 \times 128$)
μ_y/μ_x	Fully connected from layer 6, 8 dimensions		(8)
σ_y/σ_x	Fully connected from layer 6, 8 dimensions		(8)

Table G.1: The encoders' architecture (ENC_y and ENC_x).

	Layer	**Activation**	**Output**
z	Input (16)		(16)
1	Fully connected, 32768 dimensions	LeakyReLU(0.1)	($16 \times 16 \times 128$)
2	Batch Normalization		($16 \times 16 \times 128$)
3	4×4 Transposed convolution, stride 1, 64 filters	LeakyReLU(0.1)	($16 \times 16 \times 64$)
4	Batch Normalization		($16 \times 16 \times 64$)
5	4×4 Transposed convolution, stride 1, 32 filters	LeakyReLU(0.1)	($16 \times 16 \times 32$)
6	Batch Normalization		($16 \times 16 \times 32$)
$\widetilde{x}$	4×4 Transposed convolution, stride 2, 1 filter	Sigmoid	($32 \times 32 \times 1$)

Table G.2: The decoder's architecture (DEC).

	Layer	Activation	Output
z_y	Input (8)		(8)
$\widetilde{y}$	Fully connected, 10 dimensions	Softmax	(10)

Table G.3: The label-classifier's architecture (CY).

	Layer	Activation	Output
z_x	Input (8)		(8)
1	Fully connected, 50 dimensions	LeakyReLU(0.1)	(50)
2	Batch Normalization		(50)
$\widetilde{y}$	Fully connected, 10 dimensions	Softmax	(10)

Table G.4: The adverse label-classifier's architecture (CX).

G.1.2 LVAE

	Layer	Activation	Output
z_{y_i}	Input (1)		(1)
$\widetilde{f_i}$	Fully connected, 2 dimensions	Softmax	(2)

Table G.5: The dimension classifiers' architecture ($class_i$).

	Layer	Activation	Output
$z_{y_{ic}}$	Input (7)		(7)
1	Fully connected, 50 dimensions	LeakyReLU(0.1)	(50)
2	Batch Normalization		(50)
$\widetilde{f_i}$	Fully connected, 2 dimensions	Softmax	(2)

Table G.6: The adverse complementary-dimension classifiers' architecture ($class_{ic}$).

G.1.3 VAE-CE

	Layer	Activation	Output
x	Input ($32 \times 32 \times 1$)		($32 \times 32 \times 1$)
1	4×4 Convolution, stride 2, 32 filters	LeakyReLU(0.1)	($16 \times 16 \times 32$)
2	Batch Normalization		($16 \times 16 \times 32$)
3	Dropout(0.3)		($16 \times 16 \times 32$)
4	4×4 Convolution, stride 1, 64 filters	LeakyReLU(0.1)	($16 \times 16 \times 64$)
5	Batch Normalization		($16 \times 16 \times 64$)
6	Dropout(0.3)		($16 \times 16 \times 64$)
7	4×4 Convolution, stride 1, 128 filters	LeakyReLU(0.1)	($16 \times 16 \times 128$)
8	Batch Normalization		($16 \times 16 \times 128$)
9	Dropout(0.3)		($16 \times 16 \times 128$)
real	Fully connected, 2 dimensions	Softmax	(2)

Table G.7: The realism-discriminator's architecture (D).

G.1.4 CD

	Layer	Activation	Output
x	Input ($32 \times 32 \times 1$)		($32 \times 32 \times 1$)
1	4×4 Convolution, stride 2, 32 filters	LeakyReLU(0.1)	($16 \times 16 \times 32$)
2	Batch Normalization		($16 \times 16 \times 32$)
3	4×4 Convolution, stride 1, 128 filters	LeakyReLU(0.1)	($16 \times 16 \times 128$)
4	Batch Normalization		($16 \times 16 \times 128$)
μ_y/μ_x	Fully connected from layer 4, 16 dimensions		(16)
σ_y/σ_x	Fully connected from layer 4, 16 dimensions		(16)

Table G.8: The CD-encoders' architecture ($CD\text{-}ENC_y$ and $CD\text{-}ENC_x$).

	Layer	Activation	Output
z	Input (32)		(32)
1	Fully connected, 32768 dimensions	LeakyReLU(0.1)	($16 \times 16 \times 128$)
2	Batch Normalization		($16 \times 16 \times 128$)
3	4×4 Transposed convolution, stride 1, 32 filters	LeakyReLU(0.1)	($16 \times 16 \times 32$)
4	Batch Normalization		($16 \times 16 \times 32$)
$\widetilde{x}$	4×4 Transposed convolution, stride 2, 1 filter	Sigmoid	($32 \times 32 \times 1$)

Table G.9: The CD-decoder's architecture ($CD\text{-}DEC$).

	Layer	Activation	Output
z_y/z_x	Input (16)		(16)
1	Fully connected, 50 dimensions	LeakyReLU(0.1)	(50)
2	Batch Normalization		(50)
$\widetilde{y}$	Fully connected, 10 dimensions	Softmax	(10)

Table G.10: The label-disentanglement classifiers' architecture ($CD\text{-}CY$ and $CD\text{-}CX$).

	Layer	Activation	Output
z_y	Input (16)		(16)
1	Fully connected, 50 dimensions	LeakyReLU(0.1)	(50)
2	Batch Normalization		(50)
2	Dropout(0.3)		(50)
change	Fully connected, 2 dimensions	Softmax	(2)

Table G.11: The latent-difference change-discriminator's architecture ($DISC$).

G.2 Hyperparameter Selection

The settings shared between all model-training procedures are depicted in Table G.12. All hyperparameters are depicted in Table G.13. To tune these hyperparameters, we adhere to the following procedure. First, we identify a non-degenerate configuration by hand (for each model type). According to these configurations, we define a range of parameters and explore all configurations in this range. All considered hyperparameter values are denoted in Table G.14. The metric we use for model selection is the explanation alignment cost (*eac*). We generate a smooth interpolation and a dimension-wise interpolation and take minimum *eac* out of those two. For VAE-CE, we also consider its graph-based explanation. As computing the *eac* requires access to the ground-truth generating process, we search for hyperparameters using the synthetic data. The identified optimal parameters are also used for the MNIST models.

Each model is trained for approximately 2 000 000 steps: 20 epochs on the synthetic dataset and 33 epochs on MNIST. We found that training models with less data generally resulted in worse *eac*-values, whereas training for significantly longer did not improve (and sometimes regressed) the *eac*-score. As we cannot reasonably evaluate the model quality on every epoch (as the evaluation process is somewhat expensive), we settled on training on $\approx 2\,000\,000$ datapoints per model.

For each model type, we consider the Cartesian product of the hyperparameter values denoted in Table G.13. We train each configuration four times, giving us a total of 176 models. For each configuration, we compute the mean *eac* and select the hyperparameters corresponding to the lowest cost. These hyperparameter-values are depicted in Table G.15. An overview of all model runs is provided in Figure G.1.

CD is pre-trained; its hyperparameters are not tuned in the aforementioned search. For simplicity, we used a single configuration that gave a satisfactory test-set change-pair accuracy (96.4% on the synthetic data and 87.1% on [noisy] MNIST-pairs). This configuration is depicted in Table G.16. *CD* was trained for 5 000 000 steps, 50 epochs of 100 000 datapoints (for both datasets).

Setting	**Value**
Batch size	128
Optimizer	Adam[69]
Adam: learning rate	0.001
Adam: β_1	0.9
Adam: β_2	0.999
Adam: ϵ	0.0001

Table G.12: Shared settings.

Parameter	**Meaning**	**Model**
β_0	z_y KL divergence weight	*all*
β_1	z_x KL divergence weight	*all*
α	z_y label-classification weight	*all*
γ	z_x adverse label-classification weight	*all*
α_f	z_y per-dimension classification weight	LVAE
γ_f	z_y per-dimension adverse classification weight	LVAE
α_r	Pair-based conditioning realism weight	VAE-CE
α_p	Pair-based conditioning change-quality weight	VAE-CE

Table G.13: All hyperparameters. If not specified, the value is set to 1.

Model	Parameter	Values
DVAE	β_0	{2, 4}
	$\alpha = \gamma$	{5, 10, 15}
LVAE	β_0	{1, 2}
	$\alpha = \gamma$	{5, 7}
	$\alpha_f = \gamma_f$	{20, 25, 30}
GVAE	β_0	{1, 2, 4}
	$\alpha = \gamma$	{2, 4, 6}
ADA-GVAE	β_0	{1, 2, 4}
	$\alpha = \gamma$	{1, 2, 4}
VAE-CE	β_0	{2, 4}
	$\alpha = \gamma$	{5, 7}
	α_p	{3, 5}

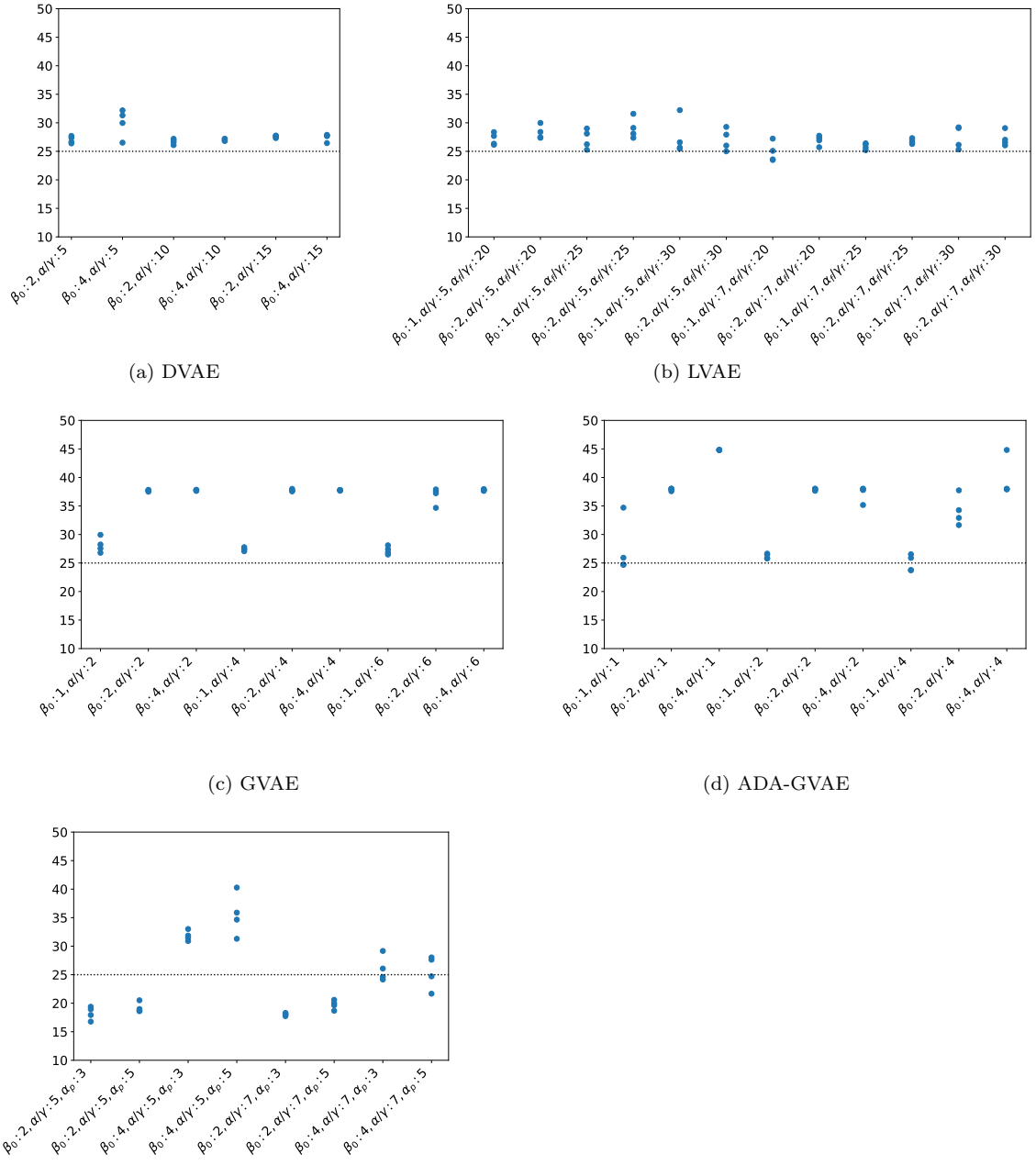
Table G.14: All explored hyperparameter values. Parameters not mentioned are set to 1. The Cartesian product of the per-parameter values denotes all configurations we consider for a model.

Model	Parameter	Value
DVAE	β_0	2
	$\alpha = \gamma$	10
LVAE	β_0	1
	$\alpha = \gamma$	7
	$\alpha_f = \gamma_f$	20
GVAE	β_0	1
	$\alpha = \gamma$	6
ADA-GVAE	β_0	1
	$\alpha = \gamma$	4
VAE-CE	β_0	2
	$\alpha = \gamma$	7
	α_p	3

Table G.15: The selected hyperparameter values.

Parameter	Meaning	Value
β_0	z_y KL divergence weight	1
β_1	z_x KL divergence weight	0.5
α	z_y label-classification weight	16
γ	z_x adverse label-classification weight	16
α_c	Latent-difference discriminator weight	50

Table G.16: CD hyperparameters.



(a) DVAE

(b) LVAE

(c) GVAE

(d) ADA-GVAE

(e) VAE-CE

Figure G.1: The minimum *eac* of each model-run (on the validation data). The x-axis depicts the different configurations, whereas the y-axis depicts the *eac* (lower is better).

G.3 Remaining Implementation Details

The remaining implementation details are as follows. First, we introduced two approaches to using an adverse classifier in the case of label-based disentanglement. For our experiments, we made use of a negated loss (Equation 4.10), implemented using Gradient Reversal Layers[16]. As this approach resulted in effective disentanglement, we opted to reduce our model search-space and did not consider other approaches. Finally, two slight deviations in the implementation are as follows. First, as can be inferred from Appendix G, we implement binary labels as a two-class multi-class classification problem, i.e., we create a 2-dimensional one-hot vector and compute the likelihood using categorical cross-entropy. This implementation is functionally equivalent to using a single output with binary cross-entropy, as described in this work. Second, for models considering multiple training passes with uneven numbers of datapoints (e.g., a pair of images in one pass and a single image in the other pass), we scale the respective losses by this ratio in order to balance the different passes.