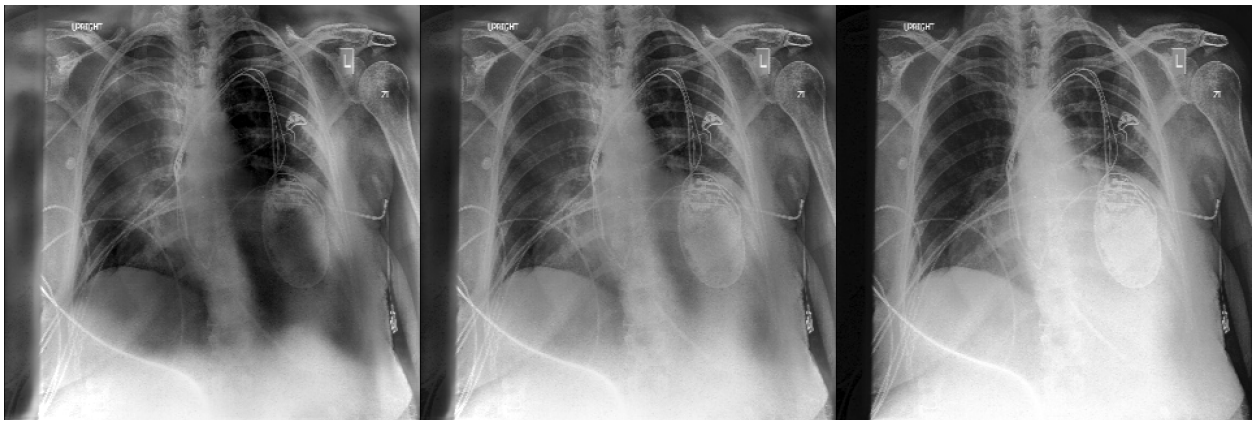

Interpreting Machine Learning Models using Conditional Counterfactual Generation

Master's thesis in Engineering mathematics and computational science

SAMUEL MARTINSSON

DEPARTMENT OF PHYSICS AND ASTRONOMY

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Interpreting Machine Learning Models using Conditional Counterfactual Generation

SAMUEL MARTINSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics and Astronomy
Chalmers University of Technology
Gothenburg, Sweden 2026

Interpreting Machine Learning Models using Conditional Counterfactual Generation
SAMUEL MARTINSSON

Supervisor: Andreas Gillgren, Department of Physics and Astronomy
Examiner: Per Bjerkeli, Department of Physics and Astronomy

Master's Thesis 2026
Department of Physics and Astronomy
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Reconstructions of chest X-ray images from a generative model conditioned on different values corresponding to the presence of the lung disease pleural effusion. See section 4.4.4 for further explanation of figure.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Interpreting Machine Learning Models using Conditional Counterfactual Generation

SAMUEL MARTINSSON
Department of Physics and Astronomy
Chalmers University of Technology

Abstract

With the rapid development and application of complex machine learning models, the need to interpret the internal processes of such models have become increasingly relevant. In this thesis, a novel method for interpreting black box machine learning models is proposed, where an autoencoder is used to generate reconstructions of data to visualize in an interpretable way what patterns a model has learned to detect. The method is first shown to work for a simple constructed problem, being able to interpret a model that has learned to predict the mean of an underlying normal distribution from samples. It is then evaluated for a more complex problem, where a model has learned to classify the existence of disease in images from the CheXpert dataset of X-ray images. It is demonstrated that naively implementing the method to interpret this model leads to the autoencoder generating adversarial patterns to trick the model, instead of showing the an interpretable explanation of what the model has learned. To mitigate this issue, the thesis explores adding an additional model in the latent space of the conditional autoencoder and demonstrates that this can provide a certain degree of interpretability. Because of this, the method shows promise for interpreting black box models and with further research it might become viable for practical use.

Keywords: machine learning, interpretability, autoencoders, counterfactual, chest X-ray images.

Acknowledgements

I want to thank my supervisor Andreas Gillgren for providing the initial idea for the project and for all the support he has provided throughout the duration of this project. His expertise has been invaluable to this project and his supervision has really made this project into an educational experience. I want to thank my examiner Per Bjerkeli for his constructive feedback on the initial project proposal and for his support with the administration. I also want to thank Rachael Harkness for the insightful discussion me and Andreas had with her before the project started concerning her previous research and Andreas initial idea. Lastly, I also want to thank Michail Marios Skyllas for trying to help me with computational resources for the project, even though an alternative solution was chosen in the end.

Samuel Martinsson, Gothenburg, May 2026

Disclosure of use of generative AI

When programming, the models Claude Sonnet 4.6, Claude Sonnet 4.5 and Claude Haiku 4.5 from Anthropic have been used via web interface for debugging code or as a complement to python library documentation. The suggestions from the models have been scrutinized before adjusting the code based on them and the models have not been asked to produce original code not based on code already present in the project. These models have also been used when searching the literature for references for this thesis and to construct bibtex citations for references.

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AE	Autoencoder(s)
ANN	Artificial Neural Network(s)
AP	Anterior Posterior
CAE	Conditional Autoencoder(s)
CNN	Convolutional Neural Network(s)
KDE	Kernel Density Estimation(s)
ML	Machine Learning
VAE	Variational Autoencoder(s)

Nomenclature

Below is the nomenclature of indices, sets, variables, parameters, and functions that have been used throughout this thesis.

Indices

i	Indices for data points

Sets

$\mathcal{X}$	Set of data points x
$\mathcal{Z}$	Latent space, set of latent variables z
$\mathcal{S}$	General set from which counterfactual values can be sampled
$\mathcal{Y}$	Set of possible values for the output y' of the primary task model
$\mathcal{I}$	Set of indices for data points x

Variables

x	Data point
x'	Reconstruction of data point x from conditional autoencoder for conditional value y'
$\tilde{x}$	Reconstruction of data point x from conditional autoencoder for counterfactual conditional value $\tilde{y}$
z	Latent variables of an autoencoder
$\tilde{z}$	Latent variables of a conditional autoencoder trained on the latent variables z of an autoencoder
z'	Reconstruction of latent variables z from a model trained in the latent space of an autoencoder
δz	Residual constructed by a conditional residual model
y	Reconstruction or output from an autoencoder
y'	Output of a primary task model for data point x
y''	Output from a primary task model for reconstruction $\tilde{x}$

$\tilde{y}$	Counterfactual conditional value, sampled uniformly from $\mathcal{Y}$

Parameters

α	Weighting coefficient for weighting components $\mathcal{L}_{\text{recon}}$ and $\mathcal{L}_{\text{cons}}$ of weighted loss function
β	Weighting coefficient for weighting component $\mathcal{L}_{\text{cons}}$ of loss function for conditional autoencoder with perceptual loss
γ	Weighting coefficient for weighting component $\mathcal{L}_{\text{perc}}$ of loss function for conditional autoencoder with perceptual loss

Functions

f	Function learned by encoder of autoencoder
g	Function learned by decoder of autoencoder
U	Uniform sample from a set $\mathcal{S}$
$\mathcal{L}_{\text{recon}}$	Reconstruction loss function
$\mathcal{L}_{\text{cons}}$	Consistency loss function
$\mathcal{L}_{\text{perc}}$	Perceptual loss function

Contents

1

Introduction

As machine learning (ML) models have become more useful and more prevalently used for a range of different applications, it has become increasingly relevant to explain the decisions or predictions of these models. Examples of such applications are medicine, where a physician might require a model to provide an explanation for its advice before relying on it to administer care, or within the legal system, where it needs to be confirmed whether a verdict complies with existing laws and specific evidence. Since the focus of ML often is to produce models which perform the best on a given task with a certain dataset, simplicity of a model or the ability to explain a model's behavior is not always prioritized. Being able to explain the model or keeping it simple might even be directly contrary to the goal of performance. The behavior of many of the state-of-the-art models used today is therefore not easily explainable, where an example are models based on artificial neural network (ANN) architectures. Such models are usually referred to as black boxes.

Because of this need for explanation, some research has pivoted towards developing methods of explaining the outputs from these black box models. One method which has been proposed is using counterfactual explanations or counterfactual examples. Counterfactual examples show how data would need to be different in order for a certain decision to be made. By observing such examples, one might be able to understand how the model is using the data to come to a certain decision compared to another, thus receiving an explanation of the behavior of the model.

This thesis proposes a novel method of using an autoencoder to generate counterfactual examples as explanations for black box ML models. The method distinguishes itself from prior methods in the field, primarily through directly conditioning the generation of counterfactual examples on a conditional variable and through the consistency training objective that it introduces. The method will be described more thoroughly in Section 3.1.

The proposed method is evaluated in two different settings. It is first evaluated on a constructed problem where the dataset is synthetically produced from a known data distribution. This constructed problem is described in more detail in Section 3.2. Secondly, the method will be evaluated in a realistic setting using the CheXpert dataset of chest X-ray images [1]. The dataset as well as the problem for this setting is described further in Section 4.1. The aim is to show whether this proposed method can provide human interpretable explanations in both simple and high-dimensional domains.

2

Background

This chapter will describe interpretability in greater detail, including how one can categorize different methods for achieving interpretability. It will also focus specifically on attribution methods and what limitations these present. The ML model architecture of an autoencoder will then be described, as it will be central to the method proposed in this thesis. Lastly, the idea of using counterfactual examples for interpreting ML models will be explained, as well as how counterfactual examples can address limitations of attribution methods.

2.1 Interpretability

Interpretability can be defined as the degree to which a human could understand the decisions of a model in the given context [2] or one could say that a model is interpretable if a human can efficiently and successfully predict the model's decisions [3]. There can be many reasons for prioritizing interpretability in ML, as briefly discussed in the previous chapter. Interpretability might additionally be useful within a scientific context, where an explanation for a model's decision or prediction in a certain task can help researchers understand the underlying problem better. It can also be useful to highlight bias within a model which one might want to discourage, in order to not discriminate against certain groups or miss important edge cases of a task. For some systems built on ML models small errors can be critical, in which case they are regarded as high-risk systems. A high level of interpretability can then be a crucial safety precaution [4].

Interpretability in ML can generally be split into two overarching classes; using models that are interpretable by design and so called post-hoc methods, applied to a model after training [4]. These classes will be discussed in more detail in the following two sections.

2.1.1 Interpretability by design

To enable interpreting the process by which an ML model comes to a certain decision or prediction, one can choose to train a model that is interpretable by design. A model can be considered interpretable by design if it learns a relatively simple function for making predictions from data. A linear regression model is a good example, since it learns one or more linear relationships. Because of this simplicity, one can therefore look at those simple linear relationships directly to understand the prediction of the model. There are also models which can learn more complex functions, but whose structure lends itself to

explanation. One such model architecture is a decision tree, where one can follow the structure of the tree to understand why a prediction was made for a certain data point.

When an inherently interpretable model is possible to use, it might be the preferred choice for achieving interpretability. However, in some cases one might not be able to use models that are interpretable by design. This can be due to the dataset on uses, which can follow complex relationships that might not be possible to describe using these models. Inherently interpretable models might therefore perform poorly when it comes to predictions regarding these types of datasets. It may then be necessary to apply ML models of more complex structure to achieve satisfactory predictive accuracy.

2.1.2 Post-hoc interpretability methods

Post-hoc interpretability methods interpret the ML models predictions after it has been trained. There can be several problems with using post-hoc methods, including that explanations after the fact must necessarily leave out some details and can thus be misleading [5]. However, for some models it might be the only way of interpreting their internal processes. Many models used today are based on ANN acrhitectures, which are generally considered to not be interpretable by design and therefore post-hoc methods allows for understanding these models better. If the goal is to gain insight regarding the problem that an ML model is trained for, a more complex model that is not interpretable by design can probably capture the phenomenon in great detail. This model can then be analyzed using post-hoc methods to produce valuable insights [6].

Post-hoc interpretability methods can be divided into two categories; some methods study specific parts of a model, in which case they are considered *model-specific*, and some methods study how input data affects the model's output, in which case they are considered *model-agnostic* [4]. Model-specific methods can be used to provide deeper insights about specific models that are actionable, since they are specific to the architecture of the model. A common model-specific method for convolutional neural newtorks (CNN) in image classification tasks is using so called saliency maps or activation maps. Saliency maps were first introduced as a technique by Simonyan et.al. [7], visualizing the pixels of an image which contribute the most to predicting a certain class.

While model-specific methods might be preferred in certain cases, model-agnostic methods have been shown to provide high interpretability and provide flexibility compared to model-specific methods [8]. When using model-agnostic methods, several different models can be compared using the same interpretability method and it is also easier to switch to another model architecture while maintaining interpretability. An example of a model-agnostic method is occlusion, where certain parts or features of input data is masked or removed to measure how important those features are for the prediction [9].

2.1.3 Attribution methods and their limitations

Many commonly used methods, including saliency maps and occlusion, fall in the category of attribution methods. These methods generally aim to attribute which feature of the input data is most important for a prediction [10]. One limitation of such methods is that they do not generally describe how a model uses features of the input data. For

example, an attribution method might highlight which region of an image was of highest importance for an image classifier when predicting which class the image belongs to, but no information is gained about what pattern in that region was key to the prediction. It has also been shown that attribution methods can have issues with lack of sensitivity to data and model, meaning that their explanations do not depend on the specific data or model being used and thus they do not actually explain the model's behavior [11].

2.2 Autoencoders

In the method that will be proposed in this thesis, an autoencoder (AE) will be used. AE are a type of ANN based ML models which are trained with the goal of learning lower dimensional representations of unlabeled data. An AE consists of two ANN, one called the encoder and one called the decoder. We can denote these as functions, f for the encoder and g for the decoder, satisfying

$$\begin{aligned} f(x) = z : \mathcal{X} \to \mathcal{Z}, \\ g(z) = y : \mathcal{Z} \to \mathcal{X}, \end{aligned} \tag{2.1}$$

where $|\mathcal{X}| > |\mathcal{Z}|$. Here, x denotes a data point, z denotes the lower dimensional representation of data called the latent variables or latent space and y denotes the output of the AE. The AE is trained with the objective of minimizing the discrepancy, the so called reconstruction loss, between x and y, so that $x = y$ in the limit [12].

A specific variant of the AE architecture is the variational autoencoder (VAE), which instead of learning a deterministic mapping from data to latent space learns a distribution of the data over the latent space. Since a probability distribution is constructed, a VAE can be used as a generative model, generating new data samples [13].

2.2.1 Conditional autoencoders

One way of affecting the reconstructions of an AE is to condition its latent space on some external information, constructing a so called conditional autoencoder (CAE). The notion of conditioning comes from probability theory, where a probability distribution can be conditioned on already known information, reshaping the distribution based on this information. One example would be conditioning the probability distribution for flipping a fair coin twice on the information that the first flip yielded heads, changing the probability of flipping tails twice from 0.25 to 0.

In the realm of AE, conditioning the latent space of a VAE on data has been explored, showing that it can improve the quality of the output [14]. Specifically conditioning the latent space using class labels has been explored, showing that it can significantly affect the structure of the latent space [15]. In this thesis, a similar approach will be used, but where the latent space of an AE is conditioned on the output of a model rather than true data labels.

2.3 Counterfactual examples

One common interpretability method is using counterfactual examples or counterfactual explanations. This involves generating an example of how the actual input would need to be different in order to produce a desired output. For example, a counterfactual explanation could involve showing how an image would need to differ in order to be classified as a certain category by an ML model. Counterfactual explanations are often attributed as being first proposed in the context of algorithmic decision-making by Wachter et al., who proposed using counterfactual examples to comply with a citizen's right to explanation in relation to GDPR [16].

This idea of counterfactual explanations has been used and implemented in a number of different ways in the ML literature. An approach which has been explored before is using a VAE conditioned on an image classifier to generate counterfactual examples, in order to interpret which features the classifier is using to determine its prediction [17]. The initial method described in this thesis has similarities to this approach, but differs in choice of model (using a simple AE) and how the model is conditioned (conditioning only the decoder as supposed to conditioning both the encoder and decoder).

Advantages of using this type of method is that it is post-hoc and that it is model-agnostic, enabling its use for a large range of ML models. In contrast to attribution methods it can explain not just what features of data are important for a certain prediction, but how the model uses those features, by showing how that data would need to differ in order to yield another prediction.

3

Method and constructed problem

In this chapter, the central interpretability method proposed in this thesis will be described. This method involves using a CAE to generate counterfactual examples, providing post-hoc interpretability of an ML model without access to the internals of that model. It will then describe the constructed problem that will be used as a first application of the method, as well as show the results of the method for the constructed problem.

3.1 Task conditioned autoencoder

Suppose there is an ML model trained for some specific task which one wants to interpret. This model to be studied will be referred to as the primary task model throughout this thesis. To study this model, one can construct a CAE with a number of latent variables denoted as z (see Figure 3.1 for an illustration of the CAE). This CAE is conditioned on the output y' of the primary task model, meaning that the decoder is fed both z and y'. This CAE is then trained to reconstruct data x as closely as possible, minimizing the so called reconstruction loss $\mathcal{L}_{\text{recon}}(x, x')$, where x' is the reconstruction of x. It is worth noting that the primary task model is frozen during this training, meaning that it is not updated during the training of the AE. See Figure 3.1 for an illustration of the described CAE.

As illustrated by Figure 3.2, the idea is to show the effect of y' on the reconstruction x' through counterfactual explanation, by traversing different values of y' for the same x and comparing the corresponding reconstructions. Since this method is only dependent

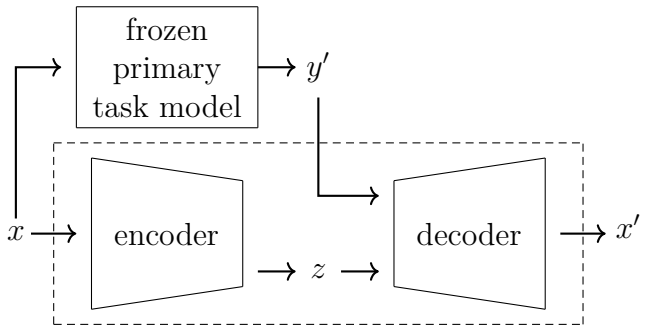

Figure 3.1: Illustration of a CAE trained for reconstruction. x denotes data, x' denotes the reconstruction of x, z denotes the latent variables of the CAE and y' denotes the output of the frozen primary task model.

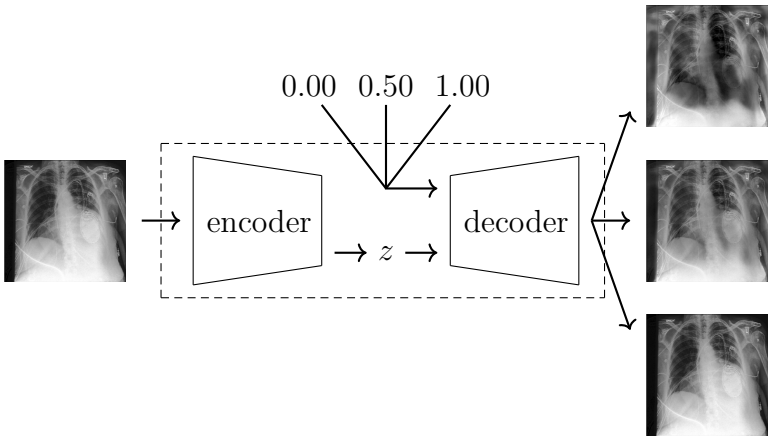

Figure 3.2: Illustration of generating counterfactual examples using a CAE, exemplified using X-ray images.

on the output from the primary task model, this should permit interpretability without explicit knowledge of the primary task model's architecture or weights.

3.1.1 Consistency training

A possible problem with only training the CAE based on reconstruction quality is that z captures information present in the output y' of the primary task model. The decoder can then reconstruct the data without using y', meaning that the reconstructions x' will be equivalent for different values of y'. This would defeat the purpose of using the CAE for interpretability. To handle this problem, consistency training is proposed as a possible solution. Figure 3.3 shows an illustration of a CAE trained for consistency. Instead of only training the CAE with the actual output of the primary task model as conditioning variable, we also train on counterfactual values, where a value $\tilde{y} \neq y'$ is sampled uniformly from the set of possible outputs $\mathcal{Y}$ of the primary task model. When training on such examples, the objective instead becomes to minimize consistency loss $\mathcal{L}_{\text{cons}}(\tilde{y}, y'')$, where y'' is the output of the frozen primary task model when fed the reconstruction $\tilde{x}$ of the AE. This type of complementary training should encourage the decoder to rely on the information in the output of the frozen model, instead of possibly using task-relevant information from z to replace the conditional variable.

In practice, consistency training is performed by first producing counterfactual values, sampling $\tilde{y}_i \sim U(\mathcal{Y})$, $i \in \mathcal{I}$, where $\sim U(\mathcal{S})$ denotes sampling uniformly from a set $\mathcal{S}$ and $\mathcal{I}$ is the index set of the data. When training, the procedure switches between the reconstruction objective for the actual model outputs y'_i and the consistency objective for the counterfactual values $\tilde{y}_i$. The total loss

$$\sum_{i \in \mathcal{I}} \left(\mathcal{L}_{\text{recon}}(x_i, x'_i) + \mathcal{L}_{\text{cons}}(\tilde{y}_i, y''_i) \right) \tag{3.1}$$

is calculated and used for updating the model.

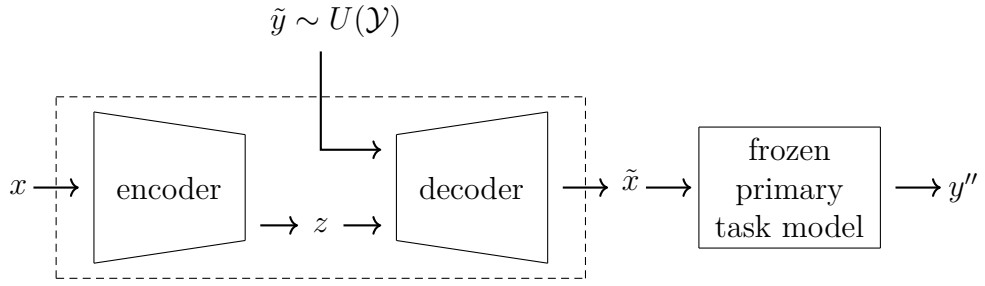

Figure 3.3: Illustration of a CAE trained for consistency. x denotes data, x' denotes the reconstruction of x, z denotes the latent variables of the autoencoder, $\tilde{y}$ denotes a counterfactual value being samples uniformly from the set $\mathcal{Y}$ of possible primary task model outputs and y'' denotes the output of the frozen primary task model when fed the reconstruction x'.

3.1.2 Disentanglement

As mentioned above, it is probable that the information captured by y' is also captured by z. To study if this is the case, a useful concept is disentanglement. Disentanglement can be defined as the degree to which a set of inferred latent variables are independent and interpretable [18]. By this definition, it is not enough for the latent variables to be independent of each other, but they should also be interpretable, which is achieved if each latent variable corresponds to a single generative factor. It is worth mentioning that what constitutes a single generative factor can be subjective and has to be defined for the underlying data distribution being studied.

There are multiple metrics that have been proposed for measuring the disentanglement of a representation. Three notable proposals are the mutual information gap [19], separated attribute predictability [20] and the three metric proposal of disentanglement, completeness, and informativeness [21].

3.1.3 Functional disentanglement

In this thesis, full disentanglement of the latent space is not necessarily desirable in itself. (For an exploration of whether the proposed CAE with consistency training achieves disentanglement for the constructed problem, see Appendix B.2.) Instead, we define a related notion called *functional disentanglement*, which we define as the degree to which a conditional variable is independent and interpretable with respect to the latent space. More precisely, the aim is to see whether changing the value of a conditional variable leads to an interpretable effect on the reconstruction of data and if this effect is independent of the latent space. Here, functional disentanglement will mostly be measured qualitatively by inspecting reconstructions from the CAE and no specific quantitative measure will be presented. However, measures of how well a CAE performs regarding consistency can be said to partially measure functional disentanglement by proxy, since a model with lower consistency loss necessarily needs to use the information given by the conditional variable to a high degree.

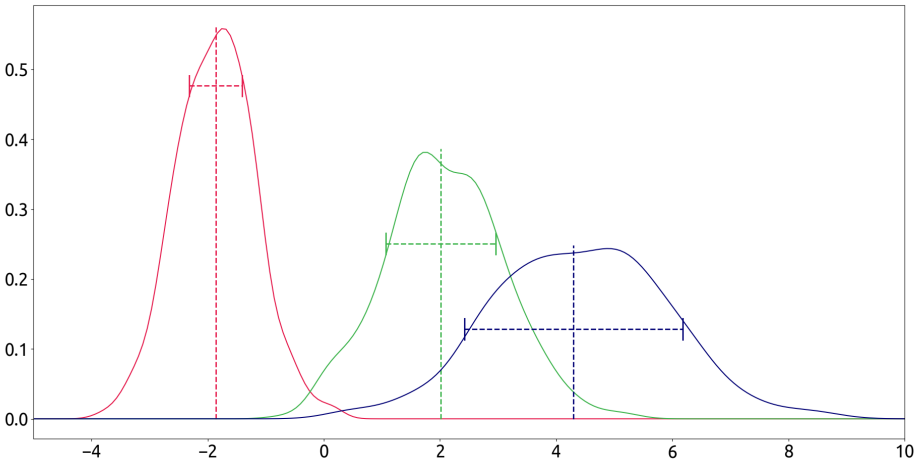

Figure 3.4: Three examples of a sample from a normal distribution, presented as KDE. The mean of each sample is denoted by a dashed vertical line and the variance illustrated by a dashed horizontal line with solid lines at the ends.

3.2 Application to constructed problem

To evaluate the interpretability achieved by the proposed task conditioned AE, a simple problem has been constructed. Each data point is constructed by sampling means and variances of a normal distribution uniformly, and then sampling a number of values from each normal distribution. This is illustrated in Figure 3.4, where three samples are shown as kernel density estimations (KDE) of a sample. KDE estimates the density function based on a sample, by constructing the density as a sum of non-negative functions centered on each data point of the sample [22]. In the figure, the mean of each sample is denoted by a dashed vertical line and the variance of each sample is denoted by a dashed horizontal line with solid lines on the ends, showing the scale of one variance on each side of the mean line.

To construct an entire dataset, 3000 normal distributions are constructed by sampling means from $[-5, 5]$ and variances from $[0, 2]$. For each normal distribution, 400 values are sampled, meaning the data has dimensions $(3000, 400)$. The sampled values of each data point are ordered in ascending order, since it might otherwise be difficult for a simpler ML model to infer a pattern from the data.

The primary task is defined as predicting the mean of the normal distribution which has generated a data point. The target is therefore the mean of a normal distribution and the input data is the 400 values sampled from this distribution. A multilayer perceptron is trained as primary task model, since this is a simple model architecture that through empirical tests proved capable of predicting the mean with a high precision. For more details regarding the model architecture, see Appendix A.1.1.

A CAE is then trained conditioned on the output of this primary task model. The hope is that traversing the conditional value (which should represent the mean of a normal distribution) after training the CAE should show clearly how the mean of the reconstructed distribution is shifted.

3.2.1 Comparison

In order to benchmark how well the CAE performs regarding reconstruction quality and consistency, a comparison is made. A normal AE without the conditional variable is trained, along with two CAE, where the first is trained without and the second is trained with counterfactual examples. All AE have the same structure, except for the conditional variable being added to the task conditioned models.

3.2.2 Metrics

A set of metrics are computed on a test partition of the dataset in order to compare the models. This set of metrics will be used throughout the report.

Reconstruction loss is computed for all models, in order to yield a relative measure of how faithful reconstructions from the models are to original data. This is the only metric computed for the non-conditional AE. *Consistency loss*, being the loss function between $\tilde{y}$ and y'', is also calculated for all CAE. This should give a relative measure of how consistent the reconstructions of the CAE are to the conditional value of the conditional variable. This is computed for a pre-determined set of counterfactual values to be comparable between CAE.

Three metrics based on computing coefficients of determination R^2 are presented. R^2 can be informally described as measuring how well a model's predictions replicate actual values present in data. It generally ranges between 0 and 1, where 1 indicates a perfect fit to the data, but negative values can arise when the predictions from a model fit the data worse than the mean of the data.

R^2 is computed between $\tilde{y}$ and y'', which in this thesis is referred to as *Consistency* R^2, in order to give a more absolute measure of consistency. R^2 between each of the latent variables of a CAE and the conditional variable is also calculated, as a measure of the correlation between the latent space and the conditional variable. The average of these values are presented as *Latent-conditional correlation*.

A so called *Conditional accuracy* is also calculated for all CAE. This metric should show how easy it is to predict the conditional variable from the latent space, indicating how much information about the conditional variable the latent space contains. A lower value should therefore indicate a higher degree of disentanglement between the conditional and latent variables. To compute this metric, a simple classifier is first trained to predict the value of the conditional variable given the values of the latent variables of the CAE, using the training partition of the dataset. The accuracy, as R^2, is then computed for this classifier on the test partition.

Apart from these metrics, some additional metrics regarding disentanglement and identifiability have been calculated. These metrics, along with further details regarding disentanglement and identifiability, can be found in Appendix B.1 and B.2 respectively.

3.2.3 Results

The latent dimension of the AE and CAE was chosen to 5, even though 2 dimensions for the AE and 1 dimension for the CAE should be sufficient for this problem, since there are only two underlying generative variables (the mean and the variance of a normal distribution). This was mostly done to see if the method would work even with superfluous ability to store information in the latent space. For more details regarding the model architectures and dimensions for each layer, see Appendix A.1.2-A.1.3.

In Table 3.1, average metrics after training a normal AE, a CAE without consistency training and a CAE with consistency training on 20 different datasets are presented. Metrics denoted with "N/A" are not calculated for the normal AE, since they are not applicable without a conditional variable.

Table 3.1: Average metrics after training instances of an AE, a CAE with and without consistency training on 20 different constructed datasets.

Metric	**AE**	**CAE no consistency**	**CAE consistency**
Reconstruction loss (10^{-3})	4.50 ± 3.74	2.22 ± 0.53	2.16 ± 0.28
Consistency loss (10^{-1})	N/A	2.9651 ± 6.2088	0.0008 ± 0.0010
Consistency R^2	N/A	0.96465 ± 0.07200	0.99999 ± 0.00001
Latent-conditional correlation (R^2)	N/A	0.53991 ± 0.35826	0.02648 ± 0.04882
Conditional accuracy (R^2)	-	0.98636 ± 0.00777	0.14018 ± 0.17923

One can observe that the reconstruction loss is on average lower for the CAE than the normal AE, which might be expected, since the CAE don't need to learn the information captured by the conditional variable and thus converge quicker in training. Considering that the CAE trained with consistency is being trained on two distinct objectives in practice, one could expect that this would lead to lower reconstruction accuracy compared to the CAE with a single objective. However, from the table one can see that the reconstruction loss does not seem to be significantly higher with consistency compared to without and is in fact on average lower.

Regarding the metrics consistency loss and consistency R^2, the CAE with consistency training seems to perform better than the one without, which should be expected since it is trained with consistency as objective. One can also observe that the latent space of the CAE with consistency has a very low correlation with the conditional variable, while the CAE without consistency has a significantly higher correlation between latent space and conditional variable. The conditional accuracy is also almost perfect for the CAE without consistency, compared with a low accuracy for with consistency. This implies that training without consistency training generally leads to information about the conditional variable being encoded in the latent space, while introducing counterfactual values in training forces the model to use the information from the conditional variable and thus it encodes that information in the latent space to a lower degree. The consistency training therefore seems to disentangle the conditional variable from the latent space of the CAE, which was the aim.

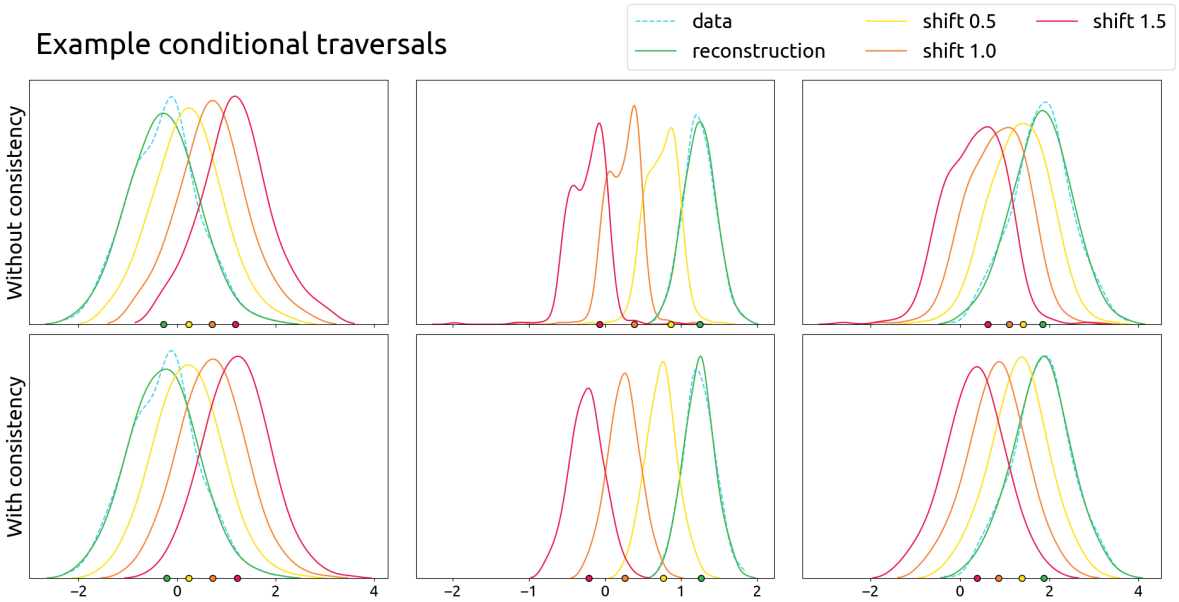

Figure 3.5: Example KDE of counterfactual examples generated with different shifts to the factual conditional value, for CAE trained with and without consistency training. The dots signify the maximum points of the corresponding KDE.

In Figure 3.5 we can see examples of conditional traversals, reconstructing based on different conditional values, for CAE trained with and without consistency training on the same dataset. In the figures, the conditional value is shifted by multiples of 0.5. The value is either shifted up or down, based on where the actual value from the primary task model lies, to keep the means within the range $[-5, 5]$. The reconstructed samples are presented as KDE of the reconstructed data series, since this visualizes the data series in an intuitive way. In the figure, the maximum points of the KDE for each reconstruction, which can be thought of as the inferred mean of that distribution, is also marked by circular markers in the color corresponding to the curve.

From the KDE, it can clearly be seen that the primary task model has learned to predict the mean of an underlying normal distribution, since the CAE shift the mean proportionally to changes in the conditional variable. One noticeable difference between the conditional traversals from the two CAE is that the CAE with consistency seems to keep the shape of the distribution to a larger extent when shifting the mean compared to the CAE without consistency. This can probably be attributed to the latent space of the CAE with consistency being disentangled to a higher degree from the conditional variable than the latent space of the CAE without consistency, meaning that reconstructions are less affected by values of the latent space not shifting along with the conditional variable.

Overall, one can conclude that training a CAE with consistency training for this constructed problem produces a model which has high consistency with the primary task model, without sacrificing reconstruction quality compared to not training for consistency or training a normal AE. The CAE can be used to produce counterfactual examples that clearly show what the primary task model has learned, leading to interpretability of the model.

3.3 Relation to previous work

Since counterfactual explanations were first suggested in the context of ML by Wachter et.al. [16], there have been multiple methods proposed for interpreting ML models through counterfactual explanation [23]. Many methods use gradient-based optimization to generate counterfactual examples close to the original data point while giving a different output. This necessitates access to the internals of the model or requires querying the model several times when constructing a counterfactual example, which means the method is either slow or model-dependent. The proposed method generates counterfactuals in one forward pass by changing the conditional variable, which means it is model-agnostic and can save computational time when generating counterfactual examples.

The proposed method is based on using a CAE. The AE architecture was chosen since it is simple and since it was thought to suffice for the counterfactual generation needed. Since a deterministic mapping from is learned by the CAE, it will also always give the same counterfactual reconstruction for a certain data point and conditional value, making the results reproducible. A lot of the literature instead uses VAE for similar methods. An advantage of using a VAE could be that it models the data distribution explicitly, facilitating sampling reconstructed data close to but not entirely identical to training data [13]. Using a VAE as part of the proposed method would however introduce stochasticity and since the aim is to generate counterfactual examples in a controlled way by only changing the conditional variable, an AE is a more natural choice.

In regards to achieving disentanglement, several approaches have been proposed. It has been shown that, by choosing the form of distribution learned by a VAE carefully, a high level of disentanglement can be achieved [24]. Another approach is to use a so called conditional subspace VAE, where the latent variables are split into two sets: one set of variables correlating to the label of the data and one that contains the additional information needed for reconstruction [25]. This approach thus tries to achieve disentanglement within the latent space. In the proposed method of this thesis, the main mechanism for achieving disentanglement is as described earlier consistency training. This is a structurally different approach, since instead of disentangling variables within the latent space of the AE, the method aims to move the label-specific information outside of this latent space through the conditional variable. To the author's knowledge, this is a novel method that has not been used before and is one of the main distinguishing features of the proposed interpretability method.

4

Application to chest X-ray images

In order to see how interpretability can be achieved through conditioning on a primary task model for a real world task, a classification task is defined for the so called CheXpert dataset. This chapter describes this dataset and the primary task defined for this dataset. It will then describe the results of applying the method to this problem.

4.1 CheXpert dataset

The CheXpert dataset is a large dataset of chest X-ray images first presented in 2019 [1]. It contains labels for different lung diseases, as well as whether a medical support device is present or not. This dataset has been used in an earlier study investigating how VAE with a certain prior distribution can be used to achieve interpretability in image classification [24]. In that study, achieving interpretability for a certain classifier architecture was studied, while in this thesis a model-agnostic approach is explored.

4.1.1 Primary task

In this thesis we have chosen to define a single label binary classification task, namely to classify whether an image shows a patient suffering from the lung disease *pleural effusion* or not. Pleural effusion refers to a build up of fluid between the pleural layers lining the outside of the lungs [26]. Figure 4.1 shows example images from the CheXpert dataset, showing both patients suffering from and not suffering from pleural effusion. As can be seen from the images, pleural effusion seems to be signified by a lighter convex area present at the bottom of the lungs, continuing along the spine.

As with the constructed dataset, a primary task model will be trained on this classification task, but instead of a multilayer perceptron a CNN is now used. This choice is made due to a CNN being able to use the spatial dependencies of an image and therefore generally performing better on image data. A CAE is then trained on the output of this model and traversing the conditional value (which in this case should represent whether the image shows a patient suffering from pleural effusion or not) after training this CAE should show what features of the image the primary task model associates with pleural effusion. As described earlier, images showing pleural effusion should show a lighter region in the bottom of the lungs and along the spine, while images not showing pleural effusion should not contain this lighter area.

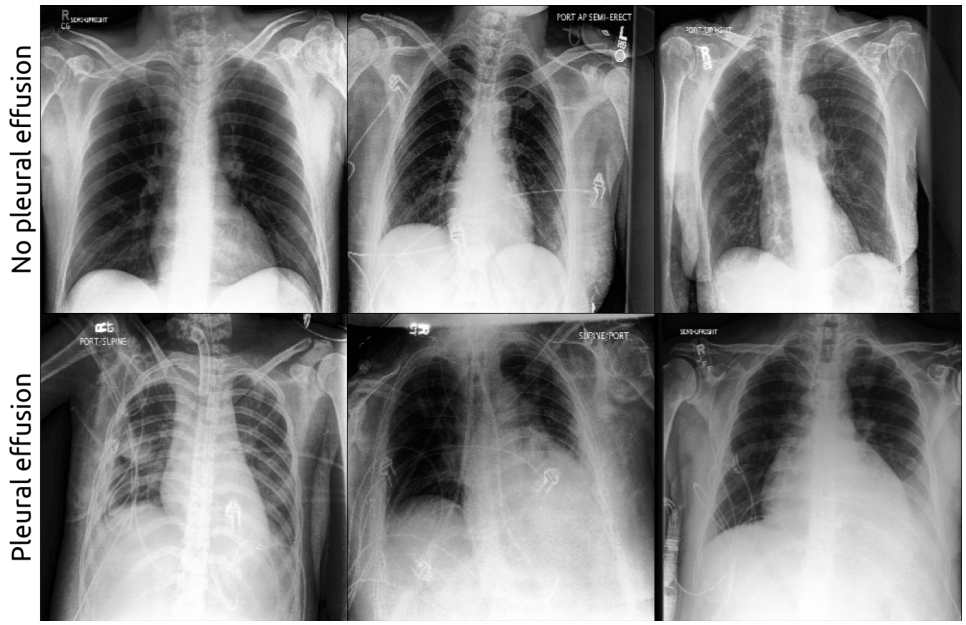

Figure 4.1: Examples of images from the CheXpert dataset showing patients either suffering from pleural effusion (upper row) or not suffering from pleural effusion (lower row).

4.1.2 Data pre-processing

In this thesis, a down-sampled version of the dataset is used, where the images have been compressed to a lower resolution. This choice is made mainly so that fewer computational resources and less time will be needed to produce results. Since models trained on this smaller dataset will need fewer parameters, it will also be easier to evaluate and analyze them.

Some pre-processing of the data is performed to create a dataset that is easier to train the primary task model on and to yield a more homogeneous dataset. This should lead to the conditional value traversals from a trained CAE being easier to analyze. Firstly, only images from a frontal view are included, excluding images from a lateral view. This choice was made because images differ significantly between these views, such that excluding one lowers the difficulty of the primary task. This reduces the dataset by 14.50%.

Secondly, only images where the patient has been imaged in a so called anterior posterior (AP) position are included. Since the pleural fluid that signifies pleural effusion will move based on the position of the patient, such that images in different positions showing signs of pleural effusion can differ significantly [26]. Since AP is the majority patient position present in the dataset, it was chosen. This further reduces the dataset by 15.41%, leaving 72.33% of the original dataset.

After applying these filters, approximately 52.11% of the images are labeled with pleural effusion and 47.89% of the images are not, meaning that the resulting dataset is balanced with regards to this label. It is interesting to note that many other labels in the dataset coincide with pleural effusion. Of images labeled with pleural effusion, 99.45% are also labeled with lung opacity and 99.42% are labeled with atelectasis. Every label connected

to a disease or the presence of "support devices" has a prevalence of more than 50% among these images, which implies that most images showing signs of pleural effusion shows signs of at least one other disease or support devices. Of images not labeled with pleural effusion, only the label support devices has a prevalence of over 50% (60.35%).

Since the images are of varying resolution, all images are downsized to a common resolution of $(320, 320)$ using bilinear interpolation. All images are also normalized along each channel.

4.2 Comparison using initial method

As for the constructed problem, both a normal AE and two CAE, one trained with and one trained without consistency training, will be trained to reconstruct the images. The metrics described in Section 3.2.2, that were computed for the constructed problem, will be computed for this case as well.

4.2.1 Results

The latent dimension of the AE and CAE was chosen to be 256. This was a rather arbitrary value, but the hope was that it would be sufficient for storing the necessary amount of information needed for good quality reconstructions, but still compressed enough to avoid too much duplicate information between latent variables. With more computational resources and time, this latent dimension could be optimized further. For more details regarding the model architectures and dimensions for each layer, see Appendix A.2.1-A.2.3.

Figure 4.2 shows examples of images with labels, along with the corresponding reconstructions from the convolutional AE. The reconstructions capture the overall appearance of an image, but the reconstructions generally miss details and sharper features. This might be expected, since the model is trained to reconstruct images in a large dataset and therefore it cannot capture all details without overfitting to the training data.

Table 4.1: Metrics after training instances of an AE, a CAE with and without consistency training on the CheXpert dataset.

Metric	**AE**	**CAE no consistency**	**CAE consistency**
Reconstruction loss	1.689582	1.730698	3.843712
Consistency loss	N/A	101.356552	0.296790
Consistency R^2	N/A	-1.406845	0.992952
Latent-conditional correlation (R^2)	N/A	0.006005	0.016362
Conditional accuracy (R^2)	0.464565	0.480310	0.514306

Table 4.1 shows metrics for each of the trained AE. One can notice that the normal AE has the best reconstruction loss of all models, with CAE without consistency being fairly close and CAE with consistency having more than double the loss of the other models. This contradicts the results from the experiments on the toy problem, where the order was

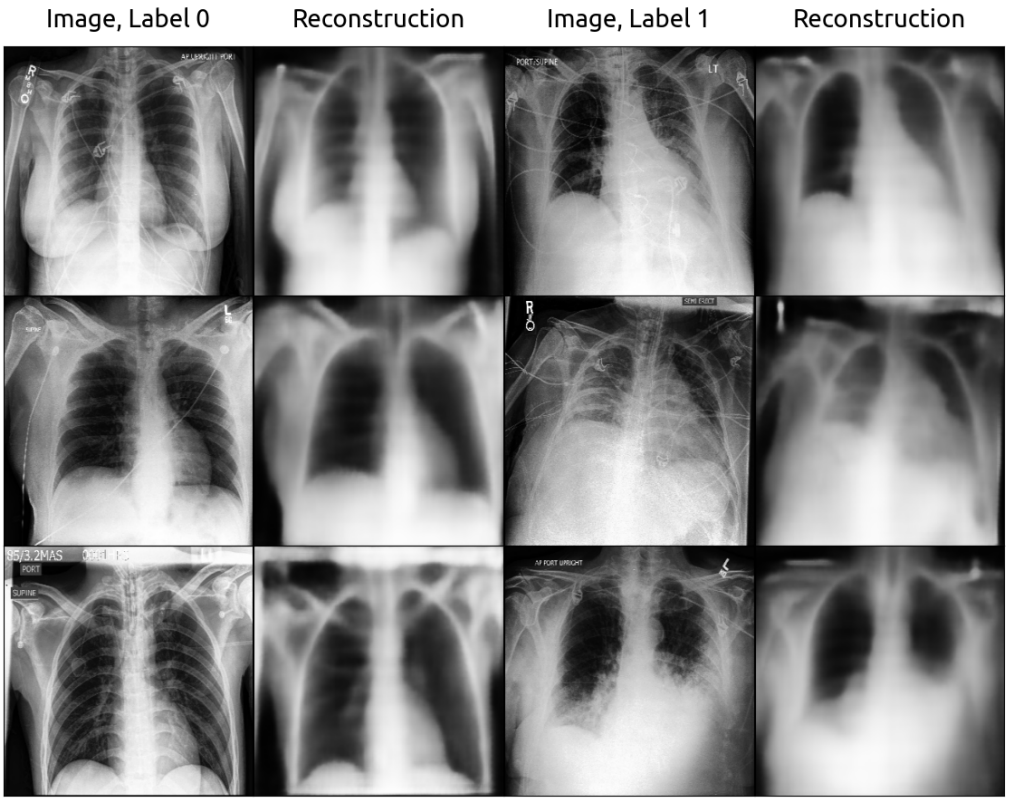

Figure 4.2: Examples of images and the corresponding reconstructions from the convolutional AE

reversed. As mentioned for the toy problem, the dual-objective training might be expected to lead to the CAE with consistency performing suboptimally regarding reconstruction, which seems to be the case here. There might be too high complexity in this dataset, compared to the constructed one, to achieve similar performance.

Regarding the consistency metrics, one can observe that the CAE with consistency training performs significantly better than the CAE without it. With this dataset, the CAE without consistency training does not seem to be able to achieve high consistency without explicitly training for it, which it did for the constructed dataset. This could be explained by the higher complexity of this dataset compared to the constructed one leading to reconstruction performance not generalizing to consistency performance.

The latent-conditional correlation is low for both CAE, so consistency training does not seem to have a disentangling effect like it did for the constructed problem. Conditional accuracy is at approximately the same level for all models, which for the CAE differs significantly from the values in the constructed problem. Since the values lie close to that of the AE, the conditioning does not seem to alter the information overlapping with the conditional variable present in the latent space significantly. Interestingly, the latent-conditional correlation and conditional accuracy seems to be higher for the CAE with consistency training than without, contradicting the results for the constructed problem.

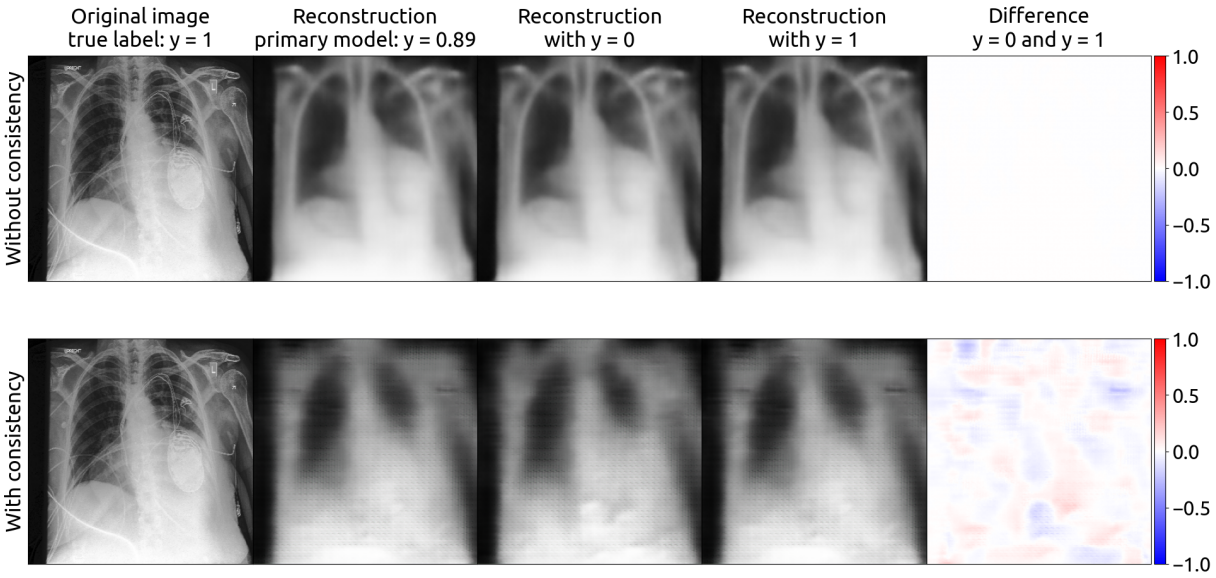

Figure 4.3: Example of an image and reconstructions from CAE trained with or without consistency, along with visualizations of the difference between the reconstructions for conditional values 1 and 0.

This would imply that consistency training leads to a lower degree of disentanglement. Because of time constraints, the CAE have not been trained until validation loss plateaus or increases (see Appendix A.2.3). It is therefore difficult to draw strong conclusions, since the latent space might change significantly with further training.

Figure 4.3 shows an example of an image along with reconstructions with the true predicted label from the primary task model, and reconstructions where the conditional value was set to either 0 or 1, for both the CAE without and with consistency training. Difference plots are also included to show the difference between the reconstruction with conditional value set to 0 and 1. One can observe that the reconstructions from the CAE without consistency for both a conditional value of 0 and 1 are identical. Figure 4.4 shows difference visualizations averaged over 1000 images for both CAE with and without consistency. From this figure one can see clear evidence that the previous observation is a general pattern; the CAE without consistency training outputs on average the same reconstructions for both conditional extreme values. Not training for consistency thus seems to lead to the CAE not using the information from the conditional variable when reconstructing the image, which was the potential issue that we wanted to address with consistency training.

When studying the reconstructions from the CAE with consistency, it becomes visually obvious why its reconstruction loss is higher. The reconstructions include patterns which are not present in the original image. By looking at the difference visualization, one can see that some patterns are present when the conditional variable is set to 0 and some others are present when the variable is set to 1. Referring to Figure 4.4, it seems evident that the same patterns are used for all reconstructions. It therefore seems as though the CAE has learned to steer the prediction of the primary task model during consistency

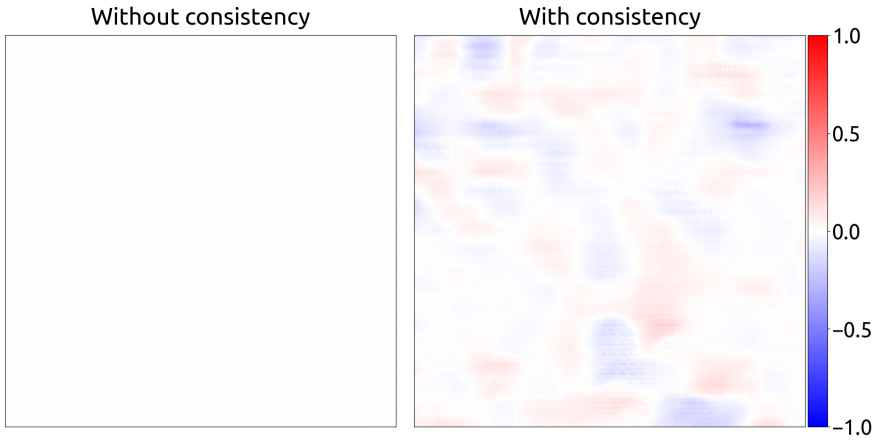

Figure 4.4: Difference between reconstructions with conditional value 1 and 0, averaged over 1000 images, for CAE with or without consistency training.

training using these patterns, since these patterns are not what we expect the primary task model to learn in order to classify pleural effusion that were described in Section 4.1.1.

Small perturbations of input data that can change the prediction of a classification model are known in the literature as adversarial examples or adversarial perturbations [27]. Such perturbations have been found to be easy to construct for image classification models and they can be indistinguishable to the human eye [28]. Adversarial examples correspond to local optima in the input space of a model and how robust a model is to adversarial attacks is strongly correlated to the geometry of this space [29]. That the CAE produces adversarial patterns to trick the primary task model can also be seen as an example of the phenomenon of specification gaming, where a model learns to perform a task in a way that scores highly on its given objective without achieving the intended behavior [30].

It is worth noting that the method showing these adversarial patterns can be seen as an interpretable result in a way, since it shows us that these patterns can be used to affect the classification of the primary task model. They do however not represent a meaningful and understandable explanation for a human observer of how the model classifies pleural effusion. The question is therefore if a higher level of interpretability can be achieved and these adversarial patterns be addressed by modifying the method.

4.3 Adjusting the loss function

The training of an ML model is principally guided by the loss function used for the optimization. An idea for yielding better interpretability is therefore to adjust the loss function used in some way. Two different adjustments to the loss function are therefore be suggested.

4.3.1 Weighted loss function

One way of trying to guide the optimization in a certain direction is to weigh one component of the loss function higher than the other. The loss function that is described by (3.1) can be modified by adding a factor dependent on the scalar α in front of both components:

$$\sum_{i \in \mathcal{I}} \left((1-\alpha)\mathcal{L}_{\text{recon}}(x_i, x_i') + \alpha \mathcal{L}_{\text{cons}}(\tilde{y}_i, y_i'') \right), \ \alpha \in [0, 1]. \tag{4.1}$$

The parameter α can then be adjusted to affect the relative scales of the values from both loss functions. When $\alpha = 0.5$, the loss functions have the same relationship as in (3.1). By introducing these factors, the total loss function is lowered in magnitude, which might affect the speed of convergence for the training since an optimization step will be smaller in magnitude in general.

4.3.2 Perceptual loss

To guide an image generation model to generate more realistic images, so called perceptual loss has been proposed. When calculating the perceptual loss, one uses a pre-trained image classifier and calculates values for high-level features of the network. This could be features that recognize edges or specific patterns of an image. One can then compare images using these high-level features instead of the per-pixel values, which has been shown to be effective in measuring meaningful differences perceptual to the human eye [31]. The hope is that introducing such a loss function will penalize the model for diverging from the original image regarding features which are visually obvious, which might disincentivize learning of adversarial patterns, as well as lead to reconstructions that look more similar to an observer.

One way of implementing perceptual loss is through Learned Perceptual Image Patch Similarity (LPIPS). In LPIPS, a model that has been pre-trained for image classification is used. Features are extracted from different layers of the model and a weighted sum is then calculated over all these features, where the weights have been optimized to correspond to how important a feature is for a human to perceive a semantic difference in an image [32].

In this work, LPIPS values are calculated using AlexNet, since it is faster to calculate than the commonly used VGG model and since empirical tests showed that the values were fairly similar between models. Adding a perceptual loss function, as well as weights to be able to adjust the scales of the different components, yields the following loss function:

$$\sum_{i \in \mathcal{I}} \left(\mathcal{L}_{\text{recon}}(x_i, x_i') + \beta \mathcal{L}_{\text{cons}}(\tilde{y}_i, y_i'') + \gamma \mathcal{L}_{\text{perc}}(x_i, x_i') \right), \ \beta, \gamma \in \mathbb{R}, \tag{4.2}$$

where $\mathcal{L}_{\text{perc}}(x_i, x_i')$ denotes the perceptual loss between data x_i and the corresponding reconstruction x_i'.

4.3.3 Comparison

In this comparison, two models with weighted loss functions will be trained: one with $\alpha = 0.67$ and one with $\alpha = 0.33$. Tests showed that $\mathcal{L}_{\text{recon}}$ and $\mathcal{L}_{\text{cons}}$ are of the same magnitude initially, so therefore these values for α should imply that the initial magnitude of one of the components should be approximately double the magnitude of the other after applying the weighting.

A model with perceptual loss added to the loss function will also be trained. Initial tests showed that the perceptual loss is generally approximately one order of magnitude larger than the other loss functions. The values for the weights have therefore been chosen to $\beta = 1.0$ and $\gamma = 0.1$, to set the components of the loss function at equal initial magnitudes after weighting.

The same test metrics as for the previous comparison, described in detail in Section 3.2.2, will be computed for all three models.

4.3.4 Results

Table 4.2 shows test metrics for both CAE with weighted loss functions and $\alpha = 0.67$ or $\alpha = 0.33$, as well as for a CAE with perceptual loss. One can observe that the CAE with $\alpha = 0.67$ performs better than the CAE with $\alpha = 0.33$ on the metrics concerning consistency, which is expected since its loss function emphasizes the consistency component while the other model's loss function emphasizes the reconstruction component. By comparing with Table 4.1, one can see that a higher value of α (if we think of the original CAE with consistency training as having $\alpha = 0.50$) correlates to higher performance regarding consistency, and vice versa for reconstruction quality, which one should expect. The same relationship can be seen for correlation between the latent space and conditional variable, although the differences between models are not particularly large.

Table 4.2: Metrics after training a CAE with weighted loss function and $\alpha = 0.67$ or 0.33, as well as a CAE with perceptual loss.

Metric	$\alpha = 0.67$	$\alpha = 0.33$	**CAE w. perceptual loss**
Reconstruction loss	4.230025	2.712388	3.680705
Consistency loss	0.128662	0.295385	0.421494
Consistency R^2	0.996945	0.992986	0.989991
Latent-conditional correlation (R^2)	0.013083	0.008946	0.010207
Conditional accuracy (R^2)	0.528321	0.508238	0.553875

Observing the column for the CAE with perceptual loss, it seems as though the model performs similarly to the original CAE with consistency regarding reconstruction. Regardring the consistency metrics however, the CAE with perceptual loss seems to perform significantly worse than all other CAE with consistency. This might be a sign of the optimization pressures from the consistency and perceptual objectives pointing are conflicting, so that improving in regards to one decreases performance in regards to the other. If the primary task model observes a different set of features than the LPIPS tests for when calculating perceptual loss, then this is likely the case.

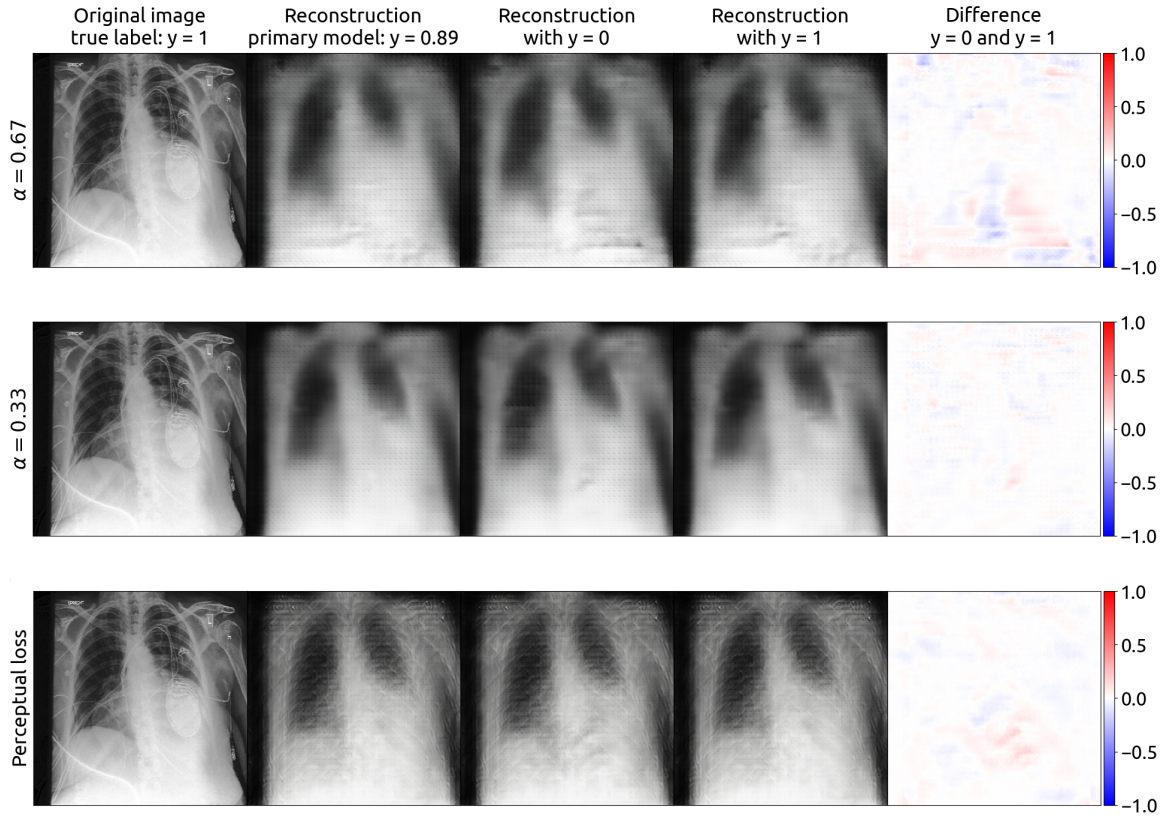

Figure 4.5: Example of an image and reconstructions from CAE with weighted loss functions, for $\alpha = 0.67$ and $\alpha = 0.33$, as well as with perceptual loss, along with visualizations of the difference between the reconstructions for conditional values 1 and 0.

Figure 4.5 shows an example of an image from the test partition of the data set, along with reconstructions from each of the three models for the true conditional value, 0 and 1, along with a visualization of the difference between the reconstructions for values 1 and 0. Both weighted loss CAEs show signs of adversarial patterns like for the original CAE with consistency training, but the CAE with lower α seems to have a more subtle pattern. This observation seems to align with consistency training being the cause of adversarial patterns in the CAE. Since the training of these CAEs have not been stopped through early stopping (see appendix A for more details), it might still be the case that the CAE with $\alpha = 0.33$ will exhibit a more similar pattern to the other CAEs when trained further. The CAE with perceptual loss also seems to exhibit adversarial patterns in its reconstructions, but the reconstructions also look visibly different compared to the reconstructions from the other models. The perceptual loss seems to introduce patterns in how the image is reconstructed, which probably are related to keeping the features that LPIPS observe similar. The reconstructions also look more clear than the reconstructions from the other CAEs.

Figure 4.6 shows difference visualizations averaged over 1000 images for the three models with adjusted loss function. These averages also strengthen the conclusion of the models having learned adversarial patterns, since they look almost identical to the differences for the example image.

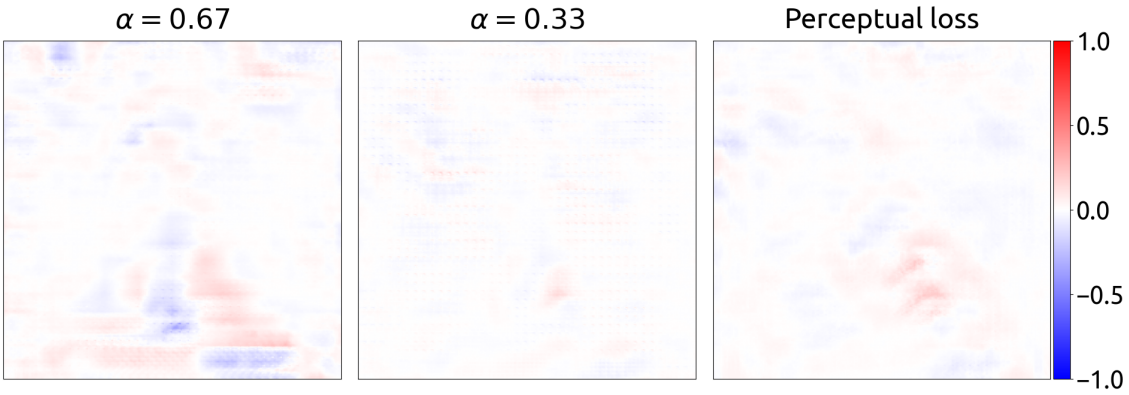

Figure 4.6: Difference between reconstructions with conditional value 1 and 0, averaged over 1000 images, for CAE with weighted loss functions and $\alpha = 0.67$ or 0.33, as well as with perceptual loss.

4.4 Working in latent space of autoencoder

Instead of training a CAE directly on the original images, the model can be trained on latent representations of the images from an AE trained only for reconstruction. The idea is that training a model in this lower dimensional space will affect the geometry of the optimization problem, which as stated previously strongly correlates to the presence of adversarial perturbations [29].

In this new framing, the primary task is altered slightly, so that the primary task becomes classifying presence of pleural effusion from latent representations of the images, instead of from the original images directly. Figure 4.7 illustrates how this model is trained. Since the variables of latent representations won't have spatial dependencies in the same way that images do and multilayer perceptrons are easier to train, such a model is used as primary task model.

To clarify how training a CAE in latent space of the AE is performed, this is described in the following section. A residual model is also introduced and evaluated in the latent space as a possible substitution for the CAE. This model is therefore also described below.

4.4.1 Task conditioned autoencoder

The first method is similar to the method that was implemented previously for both the toy problem and the binary classification task, but within this latent space framing. Figure 4.8 tries to illustrate this setup. A CAE conditioned on a primary task model is trained on latent representations of the original images. The reconstruction of an original image can be obtained by feeding the reconstructed latent representation from this CAE to the decoder of the AE which produced the latent representations. Important to note

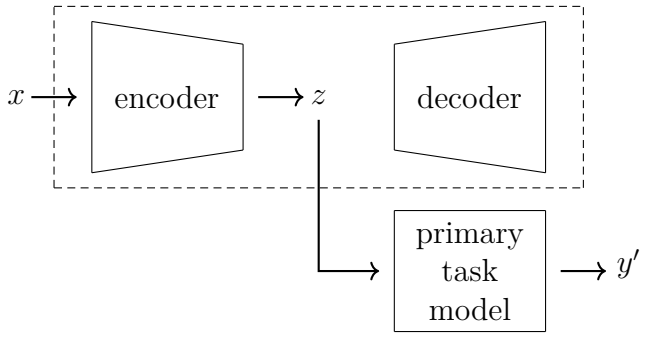

Figure 4.7: Setup for training the primary task model in latent space. The encoder and decoder are part of an AE trained on data x. z denotes the latent representation of data x from the decoder and y' denotes the corresponding output of the primary task model.

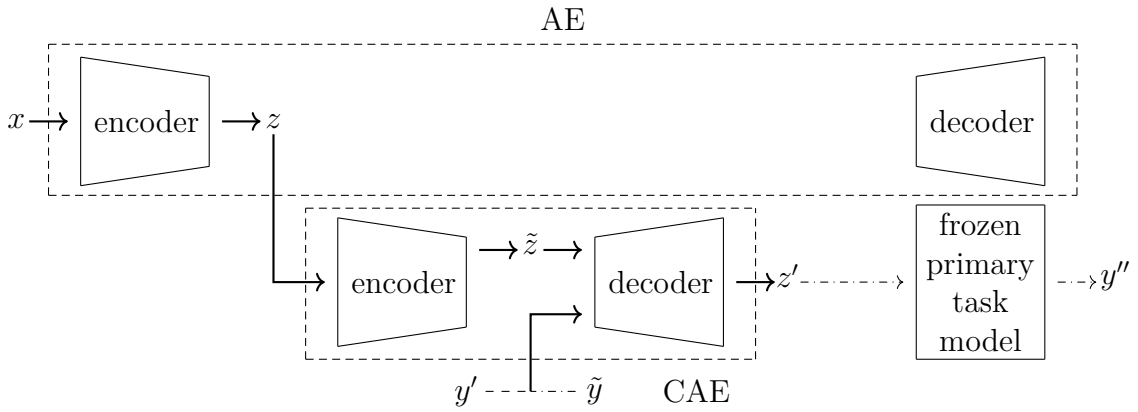

Figure 4.8: Illustration of training CAE in latent space of AE trained on original data. x denotes original data, z denotes latent representation of x from encoder of AE, $\tilde{z}$ denotes latent variables of CAE and z' denotes the output of the CAE. y' denotes the output from the frozen primary task model for input z and $\tilde{y}$ denotes a sampled value $\tilde{y}$. y'' denotes the output of the frozen primary task model for input z'. The dashed arrows show reconstruction training and the dashed and dotted arrows show consistency training.

is that the CAE is trained on the reconstruction and consistency loss with respect to the latent space, meaning $\mathcal{L}_{\text{recon}}(z, z')$ and $\mathcal{L}_{\text{cons}}(\tilde{y}, y'')$, not with respect to reconstructed images. As before, the hope is that varying the conditional variable will show what the primary task model has learned in the reconstruction.

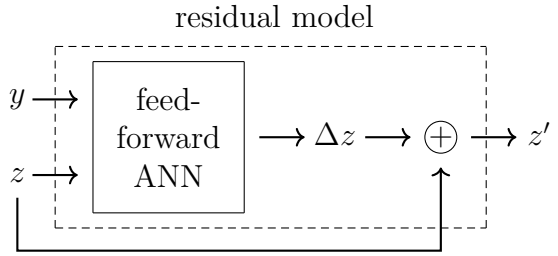

Figure 4.9: Illustration of a residual model in latent space, where z denotes latent representations of data from an AE, y denotes the conditional variable, Δz denotes the residual computed by the feedforward ANN and z' denotes the final output of the residual model. The circled plus sign denotes an addition operator performing elementwise addition of vectors and the arrow from z to the addition operator illustrates the skip connection.

Since the latent representations from the AE on the original images already are compressed representations, important information may be lost by compressing the latent space of the CAE trained on these representations even further. The CAE will therefore have the same number of latent dimensions as the latent space of the AE. The hidden layers of the encoder and decoder of the CAE are also allowed to have a higher dimension than the latent dimension of the AE and CAE to allow flexibility in learning the data distribution, which should simplify the learning process. In practice, this CAE trained in the latent space can be seen as a multilayer perceptron with a conditional variable added in the middle.

4.4.2 Residual model

Since the CAE as described above is not directly used for producing the reconstructed images, one can propose other architectures for the conditional model. One such model is a residual model. Figure 4.9 illustrates how the residual model is structured. In the residual model model, layers are constructed to learn the difference or residual between a goal function and the input. This is done by adding a so called skip connection, where the input to a set of layers is added to the output of that set. This has been shown to increase performance compared to a normal feedforward network on image recognition tasks [33].

In this implementation, a skip connection is added between the input latent representation and the output of the last feedforward layer. Thus, the model is trained to learn the residual between the latent representation and the desired reconstructed latent representation. To condition the model on the prediction from the primary task model, the residual model is fed the conditional variable in its first layer, instead of in the middle where the CAE is conditioned. The hope is that the residual model will learn to output a residual close to the zero vector when it is fed the actual prediction from the primary task model, but add some sort of non-zero residual when a counterfactual value is used to adjust the latent representation to this value.

Figure 4.10 shows how the residual model is trained in the latent space of an AE. The conjecture is that constructing the residual might be an easier task for a model to learn than latent reconstruction. The residual model might thus perform better than the CAE.

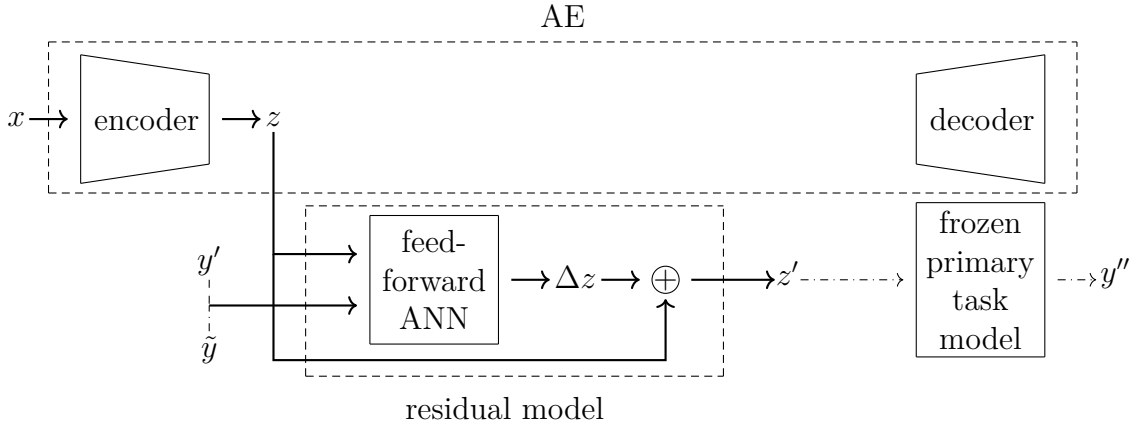

Figure 4.10: Illustration of training residual model in latent space of AE trained on original data. x denotes original data, z denotes latent representation of x from encoder of AE, Δz denotes residual computed by the residual model and z' denotes the output of the residual model. y' denotes the output from the frozen primary task model for input z and $\tilde{y}$ denotes a sampled value $\tilde{y}$. y'' denotes the output of the frozen primary task model for input z'. The dashed arrows show reconstruction training and the dashed and dotted arrows show consistency training.

4.4.3 Comparison

For the comparison, latent representations of all images are constructed. The AE used as baseline in Chapter 4.2 will be used for this purpose. Both a CAE and a residual model will then be trained using reconstruction and consistency training on these latent representations.

The same metrics as were computed for the previous comparison are computed for these models. *Reconstruction loss* still measures the reconstruction loss in the image space, meaning it is computed on the images reconstructed from the latent representations that the models trained in the latent space produce, which should make the metric comparable to the values in Table 4.1. The new metric *latent reconstruction loss* instead measures the reconstruction loss between the input latent representations and the representations that the models output. Since the residual model has no clear latent space of its own, the two metrics concerning the latent space of a model are not applicable to it.

4.4.4 Results

Table 4.3 shows metrics after training single instances of the CAE and residual model on latent representations of the AE trained on original images. One can observe that the reconstruction loss for both models is an order of magnitude larger than for the models trained on original images, shown in Table 4.1. This is to be expected, since the models in latent space are trained on the objective of reconstructing the latents as faithfully as possible and not necessarily on constructing latents which, when reconstructed into images by the AE, are faithful to the original images. If they were to be trained only for reconstruction accuracy, then these objectives should probably be parallel, but the optimization pressure from the consistency training probably perturbs the latent representations from

Table 4.3: Metrics after training single instances of a CAE and a residual model on the latent representations of the CheXpert dataset, produced by an AE trained on original images.

Metric	**CAE**	**Residual model**
Reconstruction loss	71.726469	61.425832
Latent reconstruction loss	0.011473	0.001475
Consistency loss	0.013188	0.016759
Consistency R^2	0.841844	0.799030
Latent-conditional correlation (R^2)	0.018854	N/A
Conditional accuracy (R^2)	0.580810	N/A

being optimal for image reconstruction.

The residual model has both a lower reconstruction loss and lower latent reconstruction loss than the CAE, implying that it performs better on the reconstruction objective. However, since the reconstruction loss does not differ greatly between the models, this difference is probably not that visible when viewing reconstructed images.

Comparing the metrics concerning consistency, the models are fairly close with the CAE performing slightly better (slightly lower consistency loss and slightly higher consistency R^2). The values are however not significantly different and are good for both models, implying that the models perform well on consistency.

The latent-conditional correlation for the CAE is fairly low but on a comparable level to the CAE trained with consistency training on original images. Scrutinizing the conditional accuracy, one can observe that the CAE in latent space has a higher value than the CAE in image space, implying a lower level of disentanglement between its latent space and the primary task model than the CAE in image space had with its corresponding primary model.

In Figure 4.11 shows an example of an image along with reconstructions with the true predicted label from the primary task model and reconstructions where the conditional value was set to either 0 or 1, for both the CAE and residual model. From the reconstructions, one can can clearly see how both models remove the white areas from the lungs that is present in the original image and reconstruction with the factual conditional variable. This shows that the primary task model has learned to predict based on if there are white areas present in the lungs (indicating fluid) or not and that the CAE and residual model have learned to show this in their reconstructed latents. The models seem to have very similar differences between their reconstructions for conditional values of 1 and 0. This observation is strengthened when observing Figure 4.12, which shows a visualization of the average difference between reconstructions with conditional value 1 and 0 for 1000 images. These are almost visually indistinguishable from each other, with only slight differences, showing that the models have learned the same pattern from the primary task model.

Since the author is not a medical professional, it is difficult to draw conclusions regarding if this is a reasonable pattern or not, but it seems to align with the description of the

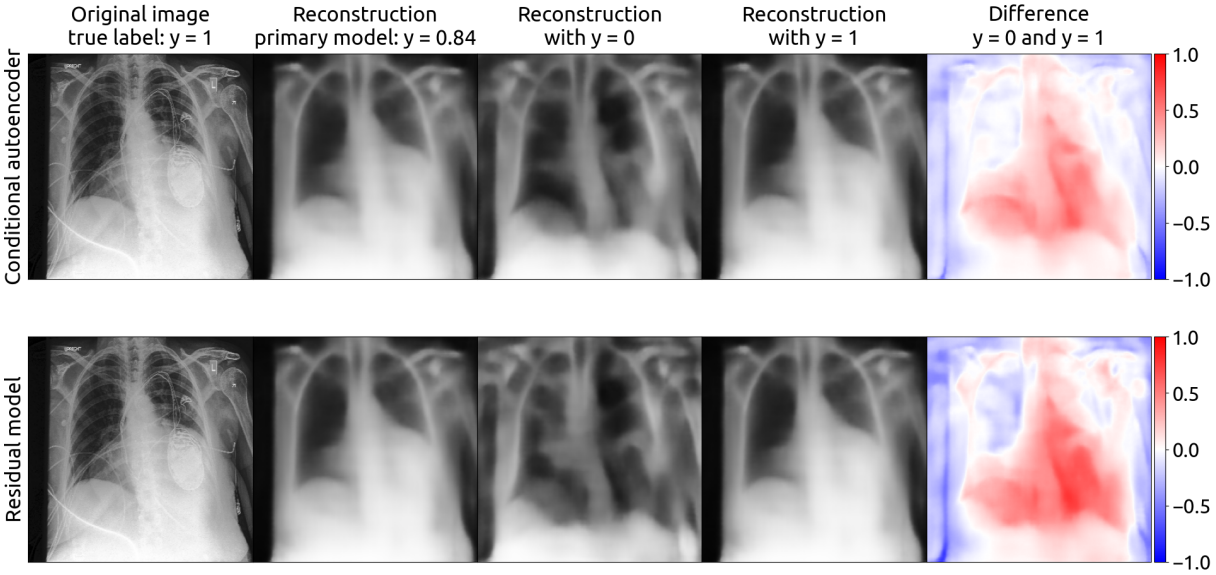

Figure 4.11: Example of an image and reconstructions from both a CAE and residual model trained in latent space, along with visualizations of the difference between the reconstructions for conditional values 1 and 0.

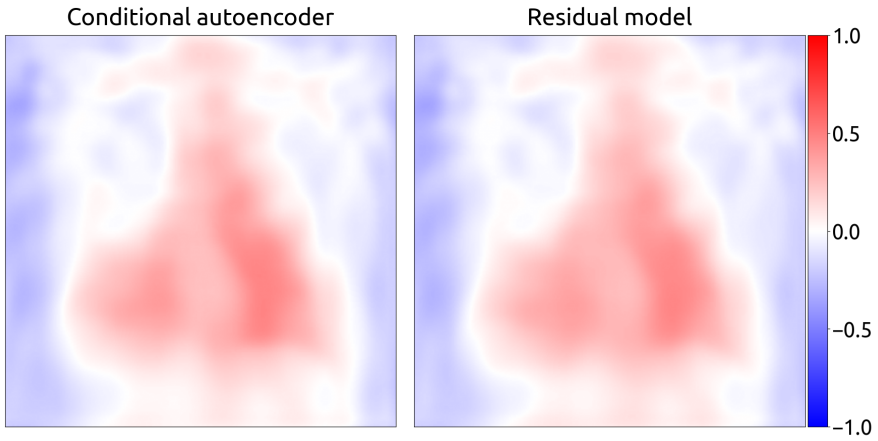

Figure 4.12: Difference between reconstructions with conditional value 1 and 0, averaged over 1000 images, for CAE and residual model trained in latent space.

appearance of pleural effusion outlined in 4.1.1. As described in 4.1.2, when an image is labeled with pleural effusion it is likely also labeled with other diseases. This might lead to the primary task model learning to pick up on patterns that are not indicative of pleural effusion but of other illnesses. To be able to separate the features in the images indicative of pleural effusion from other categories in the dataset, the primary task model would need to be trained to classify these as well.

Comparing the reconstructions, one can see that there is a difference between the area outside of the lungs and not just inside of the lungs as one might expect. The reconstructions have a lighter color outside of the lungs when pleural effusion should be absent and a darker color when pleural effusion should be present. This implies that the model does not only study the image inside of the lungs, but seems to account for the contrast between the areas inside and outside of the lungs as well. This is an interesting insight gained through observing the counterfactual examples.

To show the difference between reconstructions in a more illustrative way, one can compute the difference between the reconstructed image with the true conditional value and either of the images with counterfactual values, and then plot the sum of the original image and this difference. Figure 4.13 shows such a visualization for an example image reconstructed by the CAE and residual model trained in latent space. One can also create similar figures with a larger number of counterfactual conditional values in the range $[0, 1]$, showing a sort of gradient for the effect of the conditional variable. Figure 4.14 shows such plots for the CAE and residual model trained in latent space for conditional values of $\{0.00, 0.25, 0.50, 0.75, 1.00\}$. (The cover image also shows such plots for the residual model trained in latent space with conditional values of $\{0.00, 0.20, 1.00\}$.)

An interesting observation from Figure 4.14 is that there seems to be a larger visual difference between the reconstructions for values 0.00 and 0.25 than between the reconstructions for subsequent consecutive values. This might point to the primary task model generally classifying images that do not show pleural effusion with higher certainty than images classified as showing pleural effusion. Since images labeled with pleural effusion are also to a high degree labeled with other diseases or support devices, a majority of these images probably show signs of several illnesses, which might lead to them being clearly distinguishable from images containing no signs. A more thorough analysis of the distribution of predictions from the primary task model would however be needed to motivate such conclusions.

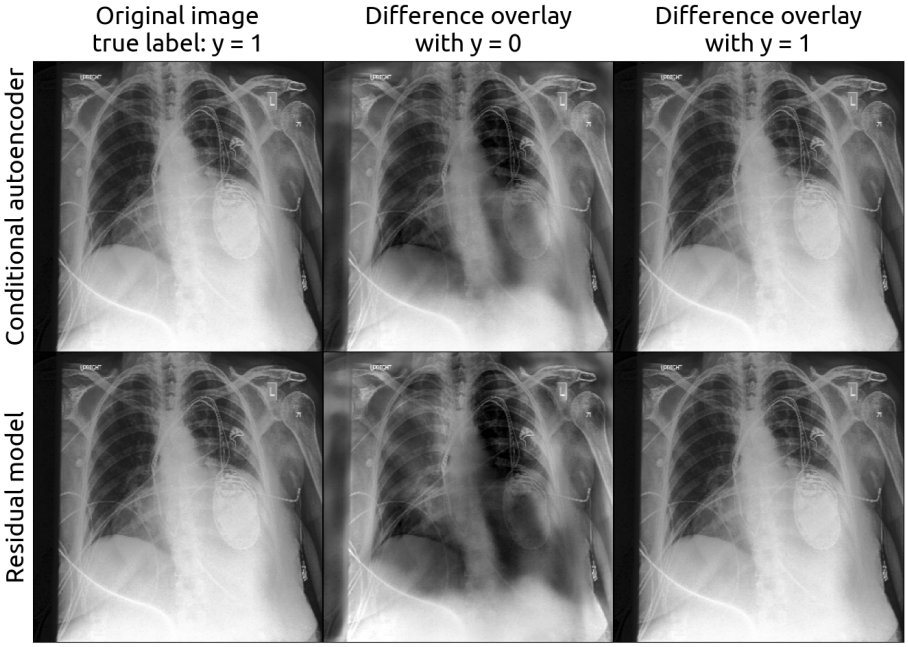

Figure 4.13: Example of an image, as well as differences between the reconstruction from the true conditional value and conditional values of 0 or 1 added to the original image, for a CAE and residual model trained in latent space.

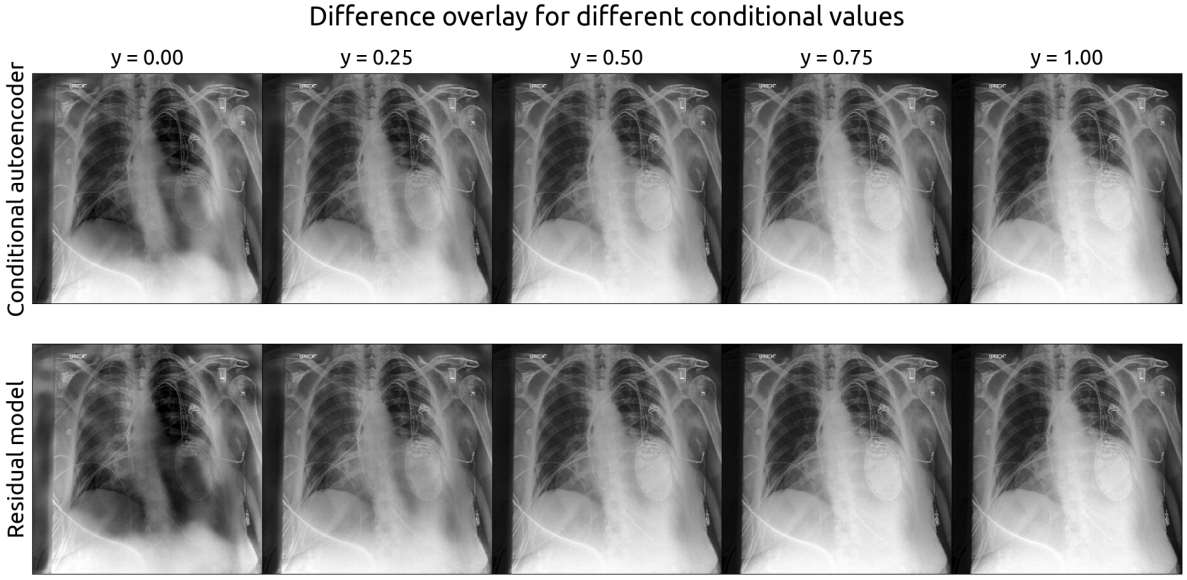

Figure 4.14: Differences between the reconstruction from the true conditional value and a range of conditional values between 0 and 1 added to an original image, produced from a CAE and residual model trained in latent space.

5

Further exploration

A problem with the approaches discussed in the previous chapter, regarding training models in the latent space of an AE, is that they are impractical compared to the approaches trained directly in the space of the data. One first has to train an AE on the data, train an alternative primary task model on the latent representations of data from the AE and then also train a CAE or residual model on the latent representations. It is therefore interesting to see whether this approach can be improved with respect to practicality and for this purpose two frameworks are suggested.

In both of these approaches, training is performed by reconstructing the latent representations as images before calculating the loss function. Figure 5.1 shows the setup for both of these cases. Compared to the previous models trained in latent space, the primary task model trained in latent space is now removed and the conditional model is conditioned directly on the original primary task model. This eliminates the need to train a primary task model on latent representations and one could also argue that this gives a more direct interpretability of the original primary task model, which is the one that the method actually aims to study.

5.1 Training autoencoder and residual model simultaneously

One might suggest that, instead of having to train multiple models in sequence, it would be more practical to train them in parallel. A possible framework is therefore to train an AE on the data and a conditional residual model on its data simultaneously. So instead of training the AE beforehand, freezing its weights and training the residual model on its latent representations, the AE is updated alongside the residual model.

5.2 Training with respect to image space

Another suggested framework entails still training an AE beforehand, but then training the residual model with a loss function with respect to the reconstructions in image space, instead of with respect to the latent space. This framework also avoids training a primary task model in latent space and only uses the original primary task model.

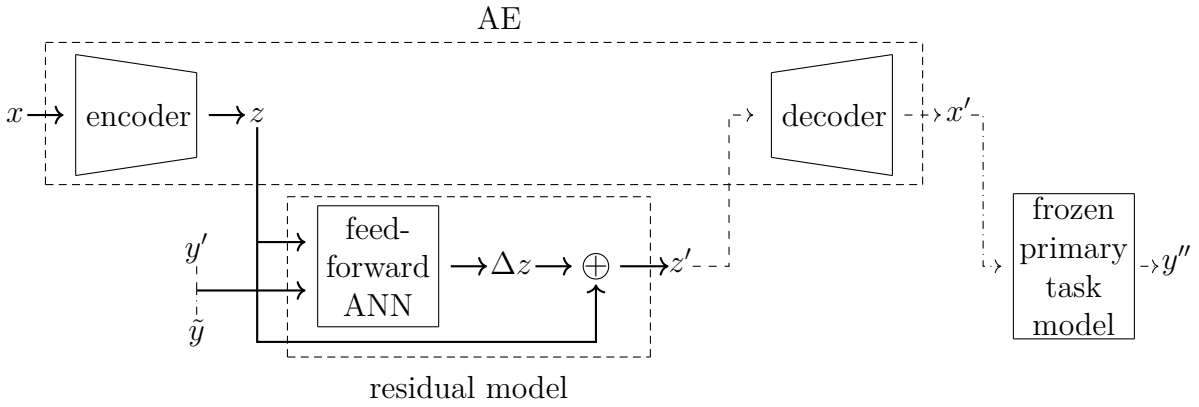

Figure 5.1: Illustration of training residual model in latent space of AE trained on original data, with respect to the reconstructed data. x denotes original data, z denotes latent representation of x from encoder of AE, x' denotes the reconstruction corresponding to x from the decoder of the AE, Δz denotes residual computed by the residual model and z' denotes the output of the residual model. y' denotes the output from the frozen primary task model for input z and $\tilde{y}$ denotes a sampled value $\tilde{y}$. y'' denotes the output of the frozen primary task model for input x'. The dashed arrows show reconstruction training and the dashed and dotted arrows show consistency training.

5.3 Results

Table 5.1 shows test metrics for an AE and residual model, trained in its latent space, trained simultaneously, as well as for a residual model trained in the latent space of an AE with loss function calculated with respect to image space. The reconstruction loss for simultaneous training is worse than, but still in the same magnitude as, the CAE trained earlier. The consistency metrics also seem too follow this pattern, being slightly worse than the CAE but still close in value. The model seems to have a higher latent-conditional correlation than the CAE however, which is contrary to the pattern earlier where consistency seemed to correlate with latent-conditional correlation. This phenomenon is likely caused by the dynamics of altering both the AE and residual model during training, but describing these dynamics accurately would necessitate a more thorough investigation of this architecture. Worth noting is that the latent reconstruction loss is several orders of magnitude larger than when training on an objective in latent space, which is to be expected since the residual model is no longer trained to reconstruct latent representations accurately, but instead to construct latent representations that lead to an accurate image reconstruction. And since the AE also changes each iteration, it is possible that the decoder of the AE is not necessarily close to the inverse of the encoder and instead implementing a function that needs some alteration of a latent representation through the residual model to reconstruct the corresponding image accurately.

Observing the metrics for the residual model trained w.r.t. image space, one can see that the reconstruction loss is approximately an order of magnitude higher than for the simultaneous training. It is closer in value to the reconstruction loss for the models trained w.r.t. latent space shown in Table 4.3. It is interesting to observe that training a model just in latent space seems to be negative for the reconstruction quality, but that

Table 5.1: Metrics after training an AE and residual model in its latent space simultaneously, as well as training a residual model on the latent representations from a pre-trained AE, but with a loss function relative to the original images.

Metric	**Train together**	**Residual w.r.t. image space**
Reconstruction loss	4.579532	44.301884
Latent reconstruction loss	140747×10^4	325.251
Consistency loss	0.429636	49.659882
Consistency R^2	0.989798	-0.179255
Latent-conditional correlation (R^2)	0.024349	0.005588
Conditional accuracy (R^2)	0.517102	N/A

changing the objective from latent space to image space improves performance. The latent reconstruction loss is better than for the simultaneous training, but still significantly worse than training w.r.t latent space. The model seems to perform catastrophically regarding consistency, since consistency loss is very high compared to other models and the consistency R^2 is negative, implying that the model is worse at consistency than a model that would just output the mean of all images.

Figure 5.2 shows an example of an image along with reconstructions with the true predicted label from the primary task model and reconstructions where the conditional value was set to either 0 or 1, as well as the difference between the reconstructions for conditional value 0 and 1, for both models. Simultaneous training seems to lead to reconstructions which are not dependent on the conditional variable, as was seen for the CAE trained without consistency training. This is strengthened by Figure 5.3, which shows the differences between the reconstructions for conditional values 1 and 0 averaged over 1000 images. This seems rather counterintuitive when observing that the consistency R^2 is rather high, since the aim of consistency training is for the trained model to adjust based on the conditional value. The reconstructions look fairly dissimilar from the original image, since the shape of the skeleton in the reconstructions does not mirror the shape in the original image that well and the colors are visibly different. Training both an AE and residual model in its latent space simultaneously therefore does not seem to be a viable way of interpreting the predictions of the primary task model.

For the residual model w.r.t. image space, in contrast to simultaneous training, the reconstructions seem to depend on the conditional variable. The difference between the reconstructions for conditional value 0 and 1 also seem reasonable, "filling in" the lungs when pleural effusion is present and "emptying" the lungs when it is not. The pattern is however different from the pattern learned by the models trained in 4.4, filling in the whole lung when pleural effusion should be present instead of only the lower part. The reconstructions look fairly blurry and they are not that close to the original image visually, which explains the high reconstruction loss. This residual model was trained for a number of epochs that is significantly lower than the residual model in latent space was trained for (see appendix A for details), which might mean that it has been trained too little to perform close to optimally. It would therefore be interesting to explore if training the model further would lead to better reconstruction quality and consistency, since the current results show promise for achieving interpretability.

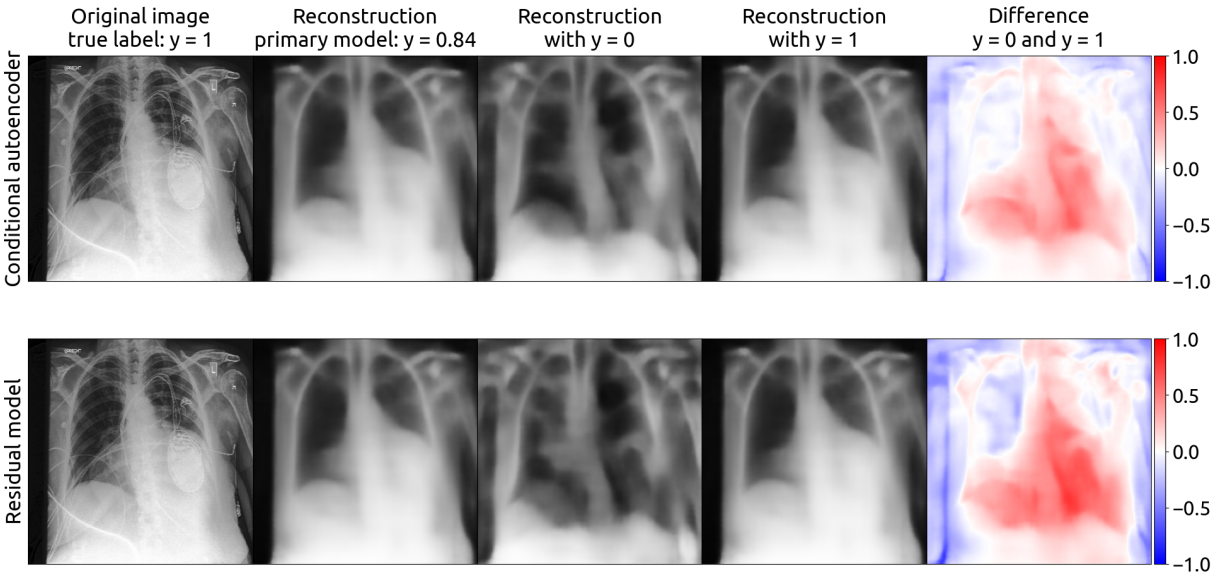

Figure 5.2: Example of an image and reconstructions from both an AE and residual model trained simultaneously, and a residual model trained w.r.t image space, along with visualizations of the difference between the reconstructions for conditional values 1 and 0.

Figure 5.3: Difference between reconstructions with conditional value 1 and 0, averaged over 1000 images, for an AE and residual model trained simultaneously, as well for a resdiual model trained w.r.t image space.

6

Discussion and conclusion

6.1 General discussion

Overall, the approach outlined in this thesis using conditional counterfactual generation to interpret an ML model trained on a primary task seems promising. The results for the constructed problem show that the method can yield interpretability of the primary task model, showing that the primary task model has learned the true underlying structure of the data to a great extent. The method therefore show that the method can be promising for interpreting ML models applied to problems with simple one-dimensional dataseries. The results for the constructed problem also show that training for consistency using counterfactual examples is necessary for the model to fit the underlying distribution accurately.

However, when applying the method to a more realistic problem with data of higher complexity, we see that the naive approach is not sufficient. The CAE that is directly generalized from the constructed problem seems to learn to use adversarial patterns to trick the primary task model, instead of learning to represent what the primary task model actually observes when classifying images. It is worth noting that this result still yields some interpretability of the model, since we see that the model is susceptible to adversarial patterns in its classification. However, this is not the explanation that is expected for this problem and it is not a useful explanation if one wants to learn what patterns in the image signify pleural effusion

By adjusting the loss function used during training, it is possible to produce clearer images and a less prominent adversarial pattern, but the CAE still seems to use adversarial patterns and does not succeed in providing interpretability of the primary task model.

By introducing models in the latent space of an AE trained on the image data, it seems possible to show interpretable results for what the primary task model has learned that align with what a layman expects to see. However, this is of course based on the author's interpretation and limited medical knowledge. To guarantee that the patterns observed align with actual visual signs of pleural effusion, one might suggest consulting a radiologist in a future study.

The approach of training in latent space introduces a new primary task model trained in the latent space and one can thus question whether the counterfactual explanations actually mirror what the original primary task model has learned. Since the latent representations of images constitute a compression of the original images, patterns in the

image data regarding pleural effusion should be present in the latent representations as well. But since the space of latent representations is dependent on the encoder of the AE, the relative positions of two images in image space might be different from their relative positions in latent space. This could lead to a primary task model learning different patterns depending on if it is trained w.r.t. image space or latent space. Comparing the average differences between reconstructions with conditional value 1 and 0 from Figure 4.12 and Figure 5.3, it seems like the patterns learned by the primary task model trained in latent space and the original primary task model trained om images are fairly different, with some similarities. Interpreting one primary task model therefore does not seem to be equivalent to interpreting the other.

It can also be seen as rather impractical to have to train this latent primary task model, as well as an AE on data and then train a model in its latent space afterwards. Training both the AE and a residual model in its latent space simultaneously did not seem to yield interpretable results, but training a residual model in latent space but with a loss function based on reconstructions in image space looks promising. This approach avoids the need for training a second primary task model and works only with the primary task model that is the actual object of study. However, further research would be needed to actually see if the approach can produce as interpretable reconstruction differences as the residual model trained w.r.t. latent space.

6.1.1 Disentanglement

The relationship between the level of disentanglement and the reconstruction quality or consistency seems complicated. For the constructed problem, both the reconstruction quality and consistency improves when consistency training is used, along with the level of disentanglement improving. So for the constructed problem, the metrics seem to correlate. When using the initial approach for the CheXpert dataset however, lower reconstruction loss still seems to correlate to higher disentanglement, but consistency seems to have the opposite relationship to disentanglement. When introducing adjustments to the method, comparing pairs of models sometimes seems to show the reconstruction quality correlating to disentanglement as well. Since the relationship changes based on problem, there might not be a direct relationship between disentanglement and the performance metrics. However, since most trained on the CheXpert dataset have not been trained until validation loss starts to plateau or increase (see Appendix A for details), it is likely that these models could be improved with further training and it is therefore difficult to draw any clear conclusions from the apparent lack of clear relationships. Optimally trained models might follow a similar pattern to the constructed problem, where higher levels of disentanglement leads to increased reconstruction quality and consistency.

6.2 Limitations

There are several limitations and disadvantages with the method in its current state. A minor limitation is that the method is only able to study differentiable models in its current state. This is because gradients need to be able to propagate through the model when updating the generative model based on consistency. So even though a large family

of models used in ML today are differentiable, the method cannot be used for all currently existing model architectures.

Another limitation is that we have only tested this method for a model with a one-dimensional output, meaning a one-dimensional conditional variable for the generative model. This can be quite limiting, since many tasks for an ML model might necessitate multi-label classification or prediction of more than a scalar variable. For the method to be truly practical, it would need to be able to handle these cases as well. This would require further testing, which might present complications. Introducing more than one value as conditional variable might increase the training time of the generative model used in the method or introduce difficulties for the generative model in distinguishing which of the conditional values should affect the reconstructions.

One more inherent limitation is that the method requires training of one or more models. This can mean that the method is more resource intensive and time consuming to use than other alternative methods. One way of mitigating this slightly is to not train an AE from scratch when implementing the method, but instead use an AE that has been trained extensively on the type of data used and then train it on the specific dataset used. By doing this, the time that it takes for an AE to learn the general structure of the data type can be avoided and the interpretability might even improve if the AE has learned from more general data. There is however still the need for training a model in latent space if one wants to use the variant of the method that seemed to perform the best.

6.3 Future research

The method has ample room for improvement and there are multiple avenues for further research. As mentioned above, it would be interesting to see if the method can be generalized to a primary task model with higher dimensional output. It would also be interesting to see if the model can generalize to other types of primary tasks. In this thesis, apart from the constructed problem, only application to an image classification model is explored. It would be interesting to explore if the method can be applied to other types of model tasks and data types, to gauge how versatile the method is.

It would also be interesting to apply the method to an already trained, often called pre-trained, model. In this thesis, both for the constructed problem and the more realistic image problem, a primary task model is trained specifically for the purpose of being studied. However, when applying this method in practice the model will probably already have been trained already and one might not be completely sure what to expect from the counterfactual explanations generated by the method. To simulate this case, it would be interesting to apply the method to an already trained model. Either through applying the method to an ML model that is freely available online or through partnering with an organization that can provide a model to study.

Something that was considered for this thesis, but which was not done due to a lack of time, is to compare the method to other interpretability methods. Comparing to contemporary methods would allow for a deeper evaluation of the interpretability that the method provides and would be necessary to clearly motivate using this method over others.

As mentioned in the general discussion, training the residual model in latent space w.r.t. image space further than what has been done in this thesis would also be of interest. It shows promising results concerning interpretability, but the reconstructions would need to be of higher quality to actually be useful and the consistency metrics would also need to improve to be able to claim that the reconstructions represent what the primary task model has learned. By training for longer time, the approach might perform better in these areas.

It might also be interesting to explore adjustments to the loss function further. The exploration done in this thesis is quite surface level and there might be other modifications that can be made to increase reconstruction quality or disincentivize adversarial patterns.

The use of a CAE is central to the method being proposed, but could be a limiting factor in some respects. When the aim is to teach a model a data distribution and generate samples from this distribution, using a VAE might be more advantageous. Since VAE model a data distribution in the latent space, they enable generation of samples from a continuous distribution and not only samples that are near equivalent to samples present in the data [13]. It would be interesting to try and adapt the proposed method to the use of a conditional VAE. As mentioned previously, such approaches have been proposed before [17], so future explorations need to guarantee some degree of novelty.

6.4 Conclusion

In conclusion, it seems to be possible to use models conditioned on a primary task model to interpret what that model has learned through generating counterfactual examples. For a simple constructed problem with one-dimensional data series, the method gives expected interpretable explanations of what the primary task model has learned. It therefore shows promise for being used as an interpretability method for ML models with similar tasks. However, for a primary task defined for X-ray image data, the method needs to be altered to yield an understandable explanation of what the primary task model has learned. It can be argued that the most promising alteration, where a conditional model is trained in the latent space of an AE, does not lead to direct interpretability of the studied model. The method overall has potential for being improved. The use of an AE that is pre-trained for the type of data used can for example probably lead to better reconstruction quality and thus clearer counterfactual explanations. It would also be interesting to explore if the method can generalize beyond current limitations, such as only being usable for models with one-dimensional output and whether a pre-trained AE can be used instead of training one from the ground up. If improved, the method presents a promising way of interpreting black box models post-hoc, without having any access to the internals of the model itself.

Bibliography

[1] J. Irvin et al., *Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison*, 2019. arXiv: 1901.07031 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1901.07031.

[2] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019, ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2018.07.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370218305988.

[3] B. Kim, R. Khanna, and O. O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf.

[4] C. Molnar, *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*, 3rd ed. 2025, ISBN: 978-3-911578-03-5. [Online]. Available: https://christophm.github.io/interpretable-ml-book.

[5] C. Rudin, *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*, 2019. arXiv: 1811.10154 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1811.10154.

[6] T. Freiesleben, G. König, C. Molnar, and Á. Tejero-Cantero, "Scientific inference with interpretable machine learning: Analyzing models to learn about real-world phenomena," *Minds and Machines*, vol. 34, no. 3, Jul. 2024, ISSN: 1572-8641. DOI: 10.1007/s11023-024-09691-z. [Online]. Available: http://dx.doi.org/10.1007/s11023-024-09691-z.

[7] K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep inside convolutional networks: Visualising image classification models and saliency maps*, 2014. arXiv: 1312.6034 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1312.6034.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, *Model-agnostic interpretability of machine learning*, 2016. arXiv: 1606.05386 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1606.05386.

[9] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013. arXiv: 1311.2901. [Online]. Available: http://arxiv.org/abs/1311.2901.

[10] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," *CoRR*, vol. abs/1703.01365, 2017. arXiv: 1703.01365. [Online]. Available: http://arxiv.org/abs/1703.01365.

[11] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf.

[12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006. DOI: 10.1126/science.1127647.

[13] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *CoRR*, vol. abs/1906.02691, 2019. arXiv: 1906.02691. [Online]. Available: http://arxiv.org/abs/1906.02691.

[14] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

[15] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," *CoRR*, vol. abs/1406.5298, 2014. arXiv: 1406.5298. [Online]. Available: http://arxiv.org/abs/1406.5298.

[16] S. Wachter, B. D. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *CoRR*, vol. abs/1711.00399, 2017. arXiv: 1711.00399. [Online]. Available: http://arxiv.org/abs/1711.00399.

[17] N. Vercheval and A. Pizurica, "Hierarchical variational autoencoder for visual counterfactuals," *CoRR*, vol. abs/2102.00854, 2021. arXiv: 2102.00854. [Online]. Available: https://arxiv.org/abs/2102.00854.

[18] I. Higgins et al., "Beta-vae: Learning basic visual concepts with a constrained variational framework.," in *ICLR (Poster)*, OpenReview.net, 2017. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2017.html#HigginsMPBGBML17.

[19] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud, *Isolating sources of disentanglement in variational autoencoders*, 2019. arXiv: 1802.04942 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1802.04942.

[20] A. Kumar, P. Sattigeri, and A. Balakrishnan, *Variational inference of disentangled latent concepts from unlabeled observations*, 2018. arXiv: 1711.00848 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1711.00848.

[21] C. Eastwood and C. K. I. Williams, "A framework for the quantitative evaluation of disentangled representations," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:19571619.

[22] E. Parzen, "On Estimation of a Probability Density Function and Mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962. DOI: 10.1214/aoms/1177704472. [Online]. Available: https://doi.org/10.1214/aoms/1177704472.

[23] S. Verma, J. P. Dickerson, and K. Hines, "Counterfactual explanations for machine learning: A review," *CoRR*, vol. abs/2010.10596, 2020. arXiv: 2010.10596. [Online]. Available: https://arxiv.org/abs/2010.10596.

[24] R. Harkness, A. F. Frangi, K. Zucker, and N. Ravikumar, *Learning disentangled representations for explainable chest x-ray classification using dirichlet vaes*, 2023. arXiv: 2302.02979 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2302.02979.

[25] M. Downs, J. Chu, Y. Yacoby, F. Doshi-Velez, and P. WeiWei, "Cruds: Counterfactual recourse using disentangled subspaces," eng, *ICML Workshop on Human Interpretability in Machine Learning*, pp. 1–23, 2020.

[26] B. Raasch, E. Carsky, E. Lane, J. O'Callaghan, and E. Heitzman, "Pleural effusion: Explanation of some typical appearances," *American Journal of Roentgenology*, vol. 139, no. 5, pp. 899–904, 1982, PMID: 6981972. DOI: 10.2214/ajr.139.5.899. eprint: https://doi.org/10.2214/ajr.139.5.899. [Online]. Available: https://doi.org/10.2214/ajr.139.5.899.

[27] C. Szegedy et al., *Intriguing properties of neural networks*, 2014. arXiv: 1312.6199 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1312.6199.

[28] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, 2015. arXiv: 1412.6572 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1412.6572.

[29] F. Yu, C. Liu, Y. Wang, L. Zhao, and X. Chen, *Interpreting adversarial robustness: A view from decision surface in input space*, 2018. arXiv: 1810.00144 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1810.00144.

[30] V. Krakovna et al., *Specification gaming: The flip side of AI ingenuity*, Google DeepMind Blog, Apr. 2020. [Online]. Available: https://deepmind.google/blog/specification-gaming-the-flip-side-of-ai-ingenuity/.

[31] J. Johnson, A. Alahi, and L. Fei-Fei, *Perceptual losses for real-time style transfer and super-resolution*, 2016. arXiv: 1603.08155 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1603.08155.

[32] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, *The unreasonable effectiveness of deep features as a perceptual metric*, 2018. arXiv: 1801.03924 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1801.03924.

[33] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1512.03385.

[34] T. J. Rothenberg, "Identification in parametric models," *Econometrica*, vol. 39, no. 3, pp. 577–591, 1971, ISSN: 00129682, 14680262. Accessed: May 7, 2026. [Online]. Available: http://www.jstor.org/stable/1913267.

[35] G. Roeder, L. Metz, and D. P. Kingma, *On linear identifiability of learned representations*, 2020. arXiv: 2007.00810 [stat.ML]. [Online]. Available: https://arxiv.org/abs/2007.00810.

[36] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, no. 3/4, pp. 321–377, 1936, ISSN: 00063444. Accessed: May 7, 2026. [Online]. Available: http://www.jstor.org/stable/2333955.

A

Methodological details

This chapter contains details regarding the model architectures and hyperparameters used, as well as the duration of training.

All models were implemented in Python using the commonly used machine learning library PyTorch. The loss function used for reconstruction and consistency loss was mean squared error.

A.1 Constructed problem

For the constructed problem, a data split of 70/15/15 was used, meaning 70% of the data was used for training, 15% was used for validation and 15% was used for testing or evaluating a model.

A.1.1 Primary task model

The primary task model was implemented as a multi-layer perceptron, each layer consisting of a linear feed-forward layer along with a ReLU activation function, except for the last layer which had no ReLU activation function. After testing, it was decided that each model has three layers of dimensions [400, 200], [200, 100] and [100, 1] respectively.

Each model was trained for a maximum of 200 epochs. Early stopping was used with a patience of 50 epochs, meaning that if the loss function calculated for the validation data did not improve for 50 epochs, then training was stopped before the maximum number of epochs was reached.

A.1.2 Autoencoder

The autoencoder was implemented such that each layer of both the encoder and decoder consisted of a linear feed-forward layer along with a ReLU activation function, except for the last layer which had no ReLU activation function. The encoder and decoder were constructed symmetrically, meaning that the dimensions of the layers in the decoder were the same as those for the encoder, just in reverse order. After testing, it was decided that the encoder should have three layers of dimensions [400, 200], [200, 100] and [100, 5], and thus for the decoder to have three layers of dimensions [5, 100], [100, 200] and [200, 400].

Each AE was trained for a maximum of 1000 epochs, using early stopping with a patience of 100. The reason for choosing the maximum number of epochs and patience to be higher than for the primary task model is that the model architecture is more complex and the problem of reconstruction was thought to be more difficult to solve, meriting longer training time. Empirical tests also showed that longer training time was needed to achieve good results.

A.1.3 Conditional autoencoder

The CAE was implemented equivalently to the normal AE, with the addition of the conditional variable connecting to the decoder. The encoder therefore had three layers of dimensions [400, 200], [200, 100] and [100, 5], and the decoder had three layers of dimensions [6, 100], [100, 200] and [200, 400].

Each CAE was trained for a maximum of 1000 epochs, using early stopping with a patience of 100.

A.2 CheXpert dataset

For the CheXpert dataset, a data split of 80/10/10 was used.

A.2.1 Primary task model

The primary task model was implemented such that the main layers consisted of four parts:

- a two-dimensional convolutional layer with kernel size 3, stride 1 and padding 1,
- a batch norm layer,
- a ReLU activation function,
- a max pooling layer, with kernel size 2 and stride 2.

The final layer consisted of a linear feed-forward layer with a sigmoid activation function, to transform the output to the range [0, 1]. After testing, it was decided that the primary task model should have four main layers, where the number of in- and out-channels (per channel in the images, which was 3) for the convolutional layers were chosen to be [1, 2], [2, 4], [4, 8] and [8, 16]. This means that the number of channels is doubled each main layer, while the spatial dimension of the image is halved. The last linear layer therefore has dimensions [19200, 1].

The model was trained for a maximum of 10 epochs, using early stopping with a patience of 2 epochs. The training stopped early after 9 epochs, meaning that the final model was trained for 7 epochs. After training, the model achieved an AUC of 0.839265 on the test set.

A.2.2 Autoencoder

The encoder of the AE was implemented according to the structure of the primary task model, with the exception of the last sigmoid activation function. The decoder was instead implemented with the following structure for the main layers:

- a two-dimensional transposed convolutional layer with kernel size 2, stride 2 and padding 0,
- a two-dimensional convolutional layer with kernel size 3, stride 1 and padding 1,
- a batch norm layer,
- a ReLU activation function (except for in the last main layer).

The idea was that this structure should mirror the reverse dimensionality transformations of the encoder as well as possible (decreasing the number of channels while increasing the spatial dimension). The first layer, before the main layers, consisted of a linear feed-forward layer, mirroring the last layer of the encoder. After all other layers, a sigmoid function is added to guarantee that the output values were in the range [0, 1].

It was decided that the encoder should have four main layers, where the number of in- and out-channels (per channel in the images, which was 3) for the convolutional layers were chosen to be [1, 2], [2, 4], [4, 8] and [8, 16]. The latent dimension was chosen to be 256, and thus the linear layer has dimensions [19200, 256]. The decoder instead has a linear layer with dimensions [256, 19200] and number of channels (per image channel) for the convolutional layers of [16, 8], [8, 4], [4, 2] and [2, 1].

The AE was trained for a maximum of 10 epochs, using early stopping with a patience of 2 epochs. The training stopped early after 8 epochs, meaning that the final AE was trained for 6 epochs.

A.2.3 Conditional autoencoders

The CAE were implemented equivalently to the normal AE, with the addition of the conditional variable connecting to the decoder. The only difference in dimensions of the layers was therefore that the linear layer of the decoder had dimensions [257, 19200].

The CAE were trained for a maximum of 6 epochs in order to save time and computational resources, since this was the amount of epochs the AE was trained, still with a patience of 2 epochs. Both models were trained for the full 6 epochs without stopping.

A.2.4 Conditional autoencoders with weighted loss function

The CAE with weighted loss function were implemented with the same architecture as the CAE described above. The only difference was in the calculation of the loss function, as described in section 4.3.1, adding a factor of $(1 - \alpha)$ to the reconstruction loss term and α to the consistency loss term.

Both CAE were trained for a maximum of 6 epochs with a patience of 2 epochs, and both models were trained for the full 6 epochs without stopping.

A.2.5 Conditional autoencoder with perceptual loss

The CAE with perceptual loss were implemented with the same architecture as the CAE described above. The only difference was in the calculation of the loss function during training, where a component representing perceptual loss was added to the loss function. This was calculated through LPIPS as described in section 4.3.2, using a pre-trained model based on the AlexNet architecture, imported from the TorchMetrics library. The AlexNet was therefore loaded before training and for every batch of data, both the original data and the reconstructed data was fed through the network and compared to yield a loss, through the library's LPIPS implementation.

The CAE was trained for a maximum of 6 epochs with a patience of 2 epochs, and the model was trained for the full 6 epochs without stopping.

A.2.6 Primary task model in latent space

The primary task model trained in latent space was implemented with the same architecture as the primary task model for the constructed problem, with linear feed-forward layers and ReLU activation functions. The dimensions of the model's layers were [256, 100], [100, 50] and [50, 1] respectively.

The model was trained for a maximum of 100 epochs with a patience of 10 epochs. The training stopped early after 13 epochs, meaning that the final model was trained for 3 epochs. After training, the model achieved an AUC of 0.776352 on the test set.

A.2.7 Conditional autoencoder in latent space

The CAE trained in latent space was implemented with the same architecture as the CAE for the constructed problem. The encoder had two layers with dimensions [256, 1024] and [1024, 256] and the decoder also had two layers with dimensions [257, 1024] (because of the conditional variable) and [1024, 256].

The CAE was trained for a maximum of 100 epochs with a patience of 10 epochs. The training stopped early after 53 epochs, meaning that the final model was trained for 43 epochs.

A.2.8 Residual model in latent space

The residual model was implemented with the same architecture as the primary task model in latent space, with the addition of a skip connection adding the input of the model to the output of the feed-forward layers. The model had three feed-forward layers of dimensions [257, 1024] (because of the conditional variable being added to the input latent representation), [1024, 1024] and [1024, 256].

The residual model was trained for a maximum of 100 epochs with a patience of 10 epochs. The training stopped early after 37 epochs, meaning that the final model was trained for 27 epochs.

A.2.9 Autoencoder and residual model trained simultaneously

The joint model of an AE and residual model in its latent space being trained simultaneously was implemented with the same architecture as for the normal AE and residual model previously described. However, for the training of the residual model with a pre-trained AE the latent representations fed to the residual model were pre-computed to speed up training, while in this case the AE is being trained as well, meaning that it has to be evaluated every training batch as well. Each epoch therefore took considerably longer than when only the residual model was trained.

The joint model was trained for a maximum of 6 epochs with a patience of 2 epochs, and the model was trained for the full 6 epochs without stopping.

A.2.10 Residual model trained w.r.t. image space

The residual model trained w.r.t. image space was implemented with the same architecture as the residual model in latent space. The difference in the implementation was with regards to training, where the latent representations output by the residual model were fed into the decoder of the AE to be able to compute the reconstruction loss. The reconstructions in image space for counterfactual conditional values also had to be fed through the original primary task model in image space to compute the consistency loss, where only the primary task model in latent space was used previously.

The joint model was trained for a maximum of 6 epochs with a patience of 2 epochs, and the model was trained for 6 epochs before stopping early, meaning that the final model was trained for 4 epochs. Initially, the author thought that it would be possible to train the model for longer, closer to the number of epochs of the residual model trained w.r.t. latent space. However, for the same reason as described for the AE and residual model trained simultaneously, the epochs take a considerably longer time.

B

Supplementary results

B.1 Identifiability

A problem which has been of interest in the development of VAE is the problem of identifiability, whether it is possible to infer the exact parameters of a model given (possibly infinite) observations from the model. More formally, we specify a statistical model by a parameter space Θ, a family of distributions $\mathcal{P}$ and a function $\pi(\theta) : \Theta \to \mathcal{P}$. A statistical model is said to be identifiable if π is injective, meaning it is one-to-one [34]. Since this is a strong definition which might be difficult to fulfill in practice, one can introduce relaxed notions of identifiability. A statistical model is said to be identifiable up to an equivalence relation $\sim$ defined on Θ if $\pi(\theta) = \pi(\tilde{\theta}) \implies \theta \sim \tilde{\theta}$.

Here, the notion of linear identifiability [35] will be discussed. In the context of an AE, the encoder function f of an AE is linearly identifiable if it is identifiable up to the equivalence relation $\sim^L$, defined as

$$\theta \sim^L \tilde{\theta} \iff f_\theta(x) = A f_{\tilde{\theta}}(x), \ \forall x, \tag{B.1}$$

where $A \in \mathbb{R}^{d \times d}$ is invertible and d is the dimension of the latent space of the AE.

Lack of identifiability can be problematic, since it can lead to the latent space differing significantly between models trained on slightly different datasets, where one dataset could constitute a slightly perturbed version of the first. Since this issue was not considered central to the thesis and since identifiability analysis was only considered feasible (regarding time and computational resources) for the constructed problem, this was included here in the appendix.

B.1.1 Identifiability for the constructed problem

Since the theoretical results regarding identifiability of the latent representations of a model in the literature do not extend to the AE architecture, since they mostly concern VAE, assessment will need to be based on empirical observations. Based on the empirical methodology presented in previous work [35], an approach to assess linear identifiability based on canonical-correlation analysis (CCA) will be used.

The goal of CCA is, given two vectors $X = (x_1, x_2, ..., x_n)^T$ and $Y = (y_1, y_2, ..., y_m)^T$ of stochastic variables, to find sequences of vectors a_k and b_k, such that the correlation

Table B.1: Mean canonical correlation computed for the latent representations of all models of the same type produced in the comparison for the constructed datasets.

Metric	AE	CAE no counter	CAE counter
Mean canonical correlation	0.820 ±0.075	0.824 ±0.066	0.937 ±0.059

$\text{corr}(a_k^T X, b_k^T Y)$ is maximized [36]. The idea is that calculating this value between the latent representations produced by two models yields a metric of how similar the latent representations are after a linear transformation, and thus gives an empirical metric for the linear identifiability of the latent representations and by proxy the models themselves.

For each type of model in the previous comparison, latent representations from all 20 instances that were trained are produced for a new constructed dataset, not used in any of the model's training. The canonical correlations have then been calculated for all possible pairs of latent representations (in this case the number of possible pairs are $\binom{20}{2} = \frac{20!}{2!\cdot 18!} = 190$) and then averaged over each dimension. Table B.1 shows the mean and standard deviation of these mean canonical correlations for each model type.

The values for all models are relatively close to 1, meaning there is a high correlation between the latent representations after applying linear transformations. The linear identifiability is therefore empirically high for these models, which might be expected with a constructed problem that has a known and simple underlying distribution. One can see that the CAE with consistency training has a significantly higher mean value than the AE and CAE without consistency training, which perform quite similarly. Figure B.1 shows histograms of the mean canonical correlations for each model. From the figure one can see that the distributions for the AE and CAE without consistency training are similar and that the CAE with consistency training has a distribution that peaks just below 1.

These results imply that one can achieve higher linear identifiability of a trained CAE by using consistency training and that only conditioning the AE does not seem to yield significantly higher linear identifiability. Since the constructed problem is relatively simple however, this might not generalize to a problem with a more complex underlying data distribution. One might expect the linear identifiability to be significantly lower overall for a more complex problem.

B.2 Disentanglement metrics

As discussed in section 3.1.2, there are multiple proposed metrics for measuring disentanglement. In this thesis, it was ultimately decided that the metrics of *Latent-conditional correlation* and *Conditional accuracy* were sufficient for capturing correlation and disentanglement between the conditional variable and latent space of a CAE, since this was not a main concern of the work. However, at an early stage, the metrics of disentanglement and completeness as described by Eastwood and Williams [21] were computed for the CAE trained for the constructed problem.

Described shortly, the metric of disentanglement captures to what degree a latent factor encodes a small number of or a single ground-truth factor, and the metric of completeness

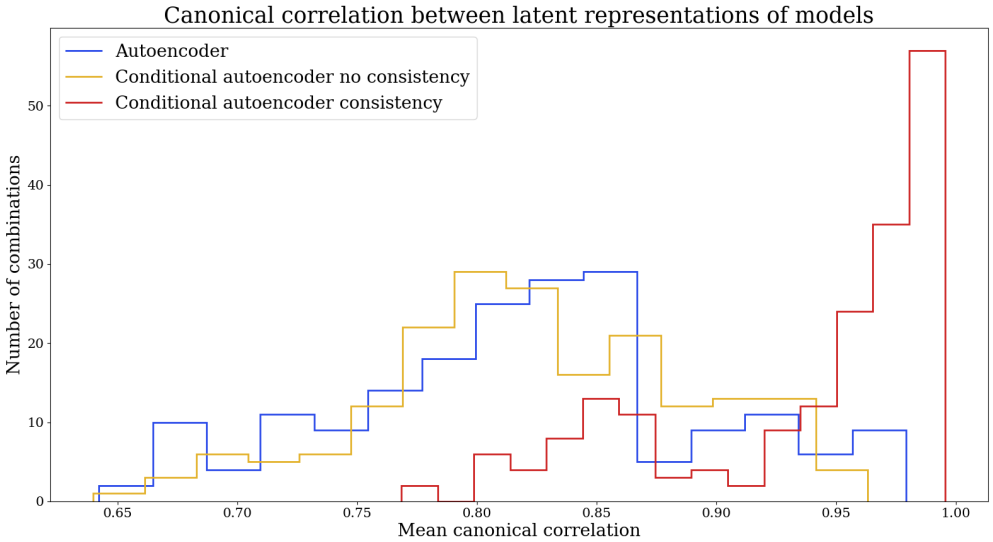

Figure B.1: Histograms of the mean canonical correlation values for the latent representations of all models of the same type produced in the comparison for the constructed datasets.

captures whether a ground-truth factor is primarily encoded by a small number of or a single latent factor. The values range from 0 to 1, where 1 would mean perfect disentanglement or completeness. For the CAE trained to interpret a primary task model for the constructed problem, the latent factors were chosen to be the conditional variable y' and the other latent dimensions z_{other} of the CAE. The ground-truth factors were the mean and variance of the underlying distribution producing a specific sample.

Table B.2 show the metrics of disentanglement and completeness computed for the above described latent and ground-truth factors, averaged over all 20 constructed datasets. The values are generally high, implying that the conditional variable and other latent dimensions are highly disentangled. The CAE with consistency training has a slightly lower disentanglement for y' and completeness for variance, which one could argue shows that consistency training does not necessarily contribute to disentanglement between the conditional and the other latent dimensions. It is also interesting to note that these values are lower than the corresponding values for z_{other} and the mean. This might imply that the conditional variable still contains some information about the variance, even though the primary task model is trained to predict the mean, but further analysis would probably be needed to draw such a conclusion. Overall, the numbers are fairly similar for both models and there does not seem to be a large difference in the disentanglement of the latent spaces.

Table B.2: Metrics for disentanglement and completeness computed for the latent and ground-truth factors of the constructed problem, averaged over all 20 constructed datasets.

Metric	**CAE no counter**	**CAE counter**
Disentanglement y'	0.96284 ± 0.02385	0.95356 ± 0.03841
Disentanglement z_{other}	0.99815 ± 0.00021	0.99815 ± 0.00009
Completeness mean	0.99816 ± 0.00021	0.99817 ± 0.00009
Completeness variance	0.96254 ± 0.02446	0.95296 ± 0.03938

DEPARTMENT OF PHYSICS AND ASTRONOMY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY