Evaluation of CNN performance in semantically relevant latent spaces

van Doorenmalen, J.

Award date:
2020

EINDHOVEN UNIVERSITY OF TECHNOLOGY

MASTER THESIS

Evaluation of CNN performance in semantically relevant latent spaces

Author:
Jeroen van Doorenmalen

Supervisors:
Dr. Vlado Menkovski
Dr. Sibylle Hess
Dr. Nikolay Yakovets

A thesis submitted in fulfillment of the requirements for the degree Master of Science

in the

Data Mining Research Group
Department of Mathematics and Computer Science

February 18, 2020

Abstract

Deep neural networks are a black box, it is unknown as to what factors contribute to the decision process. This thesis examines deep neural network's behaviour and performance by utilizing a weakly-supervised generative model as a proxy. The weakly-supervised generative model aims to uncover the generative factors underlying the data and separate abstract classes by applying metric learning. The proxy's goal is three-fold: the semantically relevant space will be the base for a linear support vector machine; The model's generative capabilities will be used to generate images that can be probed against the black box in question; the latent space is traversed and sampled from an anchor I to another class k in order to find the minimal important difference that changes both classifier's predictions. The goal of the framework is to be sure of the predictions made by the black box by better understanding the behaviour of the CNN by simulating questions of the form 'Why a and not b?' where a and b are different classes.

We examine deep neural network (DNN) performance and behaviour using contrasting explanations generated from a semantically relevant latent space. The results show that each of the above goals can be achieved and the framework performs as expected. We develop a semantically relevant latent space by training an variational autoencoder (VAE) augmented by a metric learning loss on the latent space. The properties of the VAE provide for a smooth latent space supported by a simple density and the metric learning term organizes the space in a semantically relevant way with respect to the target classes. In this space we can both linearly separate the classes and generate relevant interpolation of contrasting data points across decision boundaries and find the minimal important difference that changes the classifier's predictions. This allows us to examine the DNN model beyond its performance on a test set for potential biases and its sensitivity to perturbations of individual factors in the latent space.

Preface

This thesis was written to fulfill the requirement for obtaining a Master degree in Computer Science & Engineering - specifially, Data Science In Engineering - from TU Eindhoven. I am first of all grateful for the opportunity to research and explore the science of machine- and deep-learning with respect to interpretability at the Eindhoven University of Technology in Eindhoven. I would especially like to thank my university supervisor, Vlado Menkovski, for his guidance and support throughout this graduation project. The great freedom of exploratory and experimental research has been a blessing and curse. I loved the opportunity to experience and experiment new ideas and let my curiosity run free. New results led to new questions and experiments in a never-ending cycle, and whilst this is the essence of science there is only so much time for a master thesis. The master thesis has been a very informative and enlightening experience and made it clear to me that I love the creativity in experimenting and science.

My time at the Eindhoven University of Technology has been a great experience and would like to especially thank Irene, Joep, Markus and Rumjana for all the fun times throughout the years - Thank you for everything. Additionally, I would like to thank Aimée for her love, encouragement and support throughout the whole process. Finally, I would like to sincerely thank my family: Annie, Jos and Paul for their continuous support and encouragement.

Contents

Abstract iii

Preface v

1 Introduction 1

2 Related works 5
2.1 Essentials of Interpretability . . . 5
2.2 State-of-the-art methods . . . 8
2.2.1 Explanations of Deep Network Processing . . . 8
Proxy Models . . . 8
Salience mapping . . . 9
2.2.2 Explanations of Deep Network Representations . . . 9
Role of layers . . . 9
Role of neurons . . . 9
Role of vectors . . . 10
2.2.3 Explanation-Producing Networks . . . 10
Attention Networks . . . 10
Disentangled Representations . . . 10
Generated Explanations . . . 10
Contrastive explanations . . . 11
2.3 Relevance of research . . . 11

3 Background 15
3.1 Machine learning . . . 15
3.1.1 Discriminative models . . . 16
(Linear) Support vector machine (SVM) . . . 16
(deep) feedforward Neural Networks . . . 17
Convolutional Neural Networks . . . 19
3.1.2 Autoencoder . . . 20
3.1.3 Generative models . . . 21
Variational Autoencoder (VAE) . . . 21

4 Method 25
4.0.1 Semantically relevant latent space . . . 26
4.0.2 Decision Validation . . . 27
4.0.3 Generating (contrastive) explanations . . . 28

5 Experiment Design 31
5.1 Datasets . . . 31
5.1.1 Synthetic data set . . . 31
5.1.2 MNIST . . . 32
5.1.3 Fashion MNIST . . . 32
5.1.4 Dsprites . . . 32
5.2 Model Setup . . . 33

5.2.1 Experiment . . . 34

6 Results and Discussion **35**
6.1 Synthetic Data set . . . 35
6.1.1 Latent space . . . 35
6.1.2 Classification . . . 36
6.1.3 Generating explanations . . . 37
6.2 MNIST Data set . . . 39
6.2.1 Latent space . . . 39
6.2.2 Classification . . . 40
6.2.3 Generating Explanations . . . 41
6.3 Fashion-Mnist Data . . . 43
6.3.1 Latent space . . . 43
6.3.2 Classification . . . 44
6.3.3 Generating explanations . . . 45
6.4 dSprites Data set . . . 47
6.4.1 Latent space . . . 47
6.4.2 Classification . . . 48
6.4.3 Generating explanations . . . 49

7 Conclusion and Recommendations **51**
7.1 Conclusion . . . 51
7.2 Recommendations . . . 52

Bibliography **53**

A Models **57**

B Synthetic data - Generated explanations **65**

List of Figures

1.1 The diagnostics approach to validate and understand the behavior of the CNN. 1) Metric learning is applied during training of the VAE in order to create semantically relevant latent spaces. The generative model captures the essential semantics within the data and is used by 2) A linear Support Vector Machine. The linear SVM is trained on top of the latent space to classify input within a semantically relevant latent space rather than the direct mapping from input data X and labels Y. If the SVM and CNN do not agree on a prediction then 3) we traverse the latent space in order to generate and capture semantically relevant synthetic images, tested against the CNN, in order to check what elements have to change in order to change its prediction from a to b, where a and b are different classes. 3

2.1 Saliency map of CNN classifying a cat 9
2.2 Combination of visual and textual evidence 11

3.1 The goal of an SVM is to find a decision boundary such that the margin is maximized, where the margin is the distance from the support vectors to the decision boundary. . 16
3.2 Simple FFN or MLP consisting of four layers, input layer, two hidden layers and an output layer 18
3.3 Example of an convolution step, where a $n * n$ filter/kernel is slid over $n * n$ patches of the input feature map, the response feature map is an image where the pixel values are replaced by the convolution of the kernel with the underlying image patch. image taken from [51] 19
3.4 The above figure shows the structure of an Autoencoder. The autoencoder attempts to best reconstruct an input image x by $\hat{x} = \mathcal{G}_\theta(\mathcal{F}_\phi(x))$ where the encoder = $\mathcal{F}_\phi(x)$, the decoder = $\mathcal{G}_\theta(z)$ and z is a lower dimensional latent space where factors are encoded as efficiently as possible. 20
3.5 A VAE learns stochastic mappings between an observed x-space, whose empirical distribution $\amalg_D(x)$ is typically complicated, and a latent z-space, whose distribution can be relatively simple (such as spherical, as in this figure). The generative model learns a joint distribution $p_\theta(\mathbf{x},\mathbf{z})$ that is often (but not always) factorized as $p_\theta(\mathbf{x},\mathbf{z}) = p_\theta(z)p_\theta(x|z)$, with a prior distribution over latent space $p_\theta(z)$, and a stochastic decoder $p_\theta(x|z)$. The stochastic encoder $q_\phi(z|x)$, also called inference model, approximates the true but intractable posterior $p_\theta(z|x)$ of the generative model. Image taken from [44] 21
3.6 VAE objective is a combination of KL divergence and reconstruction loss. The recognition model creates a meaningful mapping from input data x to a latent space z measuring distance between the prior and the posterior via KL divergence. Additionally, using the posterior and the generative model recreates the input image x where the quality of these images are measured by the reconstruction loss. 23

4.1 The diagnostics approach to validate and understand the behavior of the CNN. 1) Metric learning is applied during training of the VAE in order to create semantically relevant latent spaces. The generative model captures the essential semantics within the data and is used by 2) A linear Support Vector Machine. The linear SVM is trained on top of the latent space to classify input within a semantically relevant latent space rather than the direct mapping from input data X and labels Y. If the SVM and CNN do not agree on a prediction then 3) we traverse the latent space in order to generate and capture semantically relevant synthetic images, tested against the CNN, in order to check what elements have to change in order to change its prediction from a to b, where a and b are different classes. 25
4.2 $\delta_{ap} > \delta_{an}$ 26
4.3 $\delta_{ap} < \delta_{an}$ 26
4.4 $\delta_{ap} = \delta_{an}$ 26
4.5 Example of a Triplet VAE network. Blue blocks represent input data, red blocks represent functions or parts of the VAE neural network, green blocks represent output and yellow blocks represent loss functions. 27
4.6 Given an input image I we check the prediction of the CNN as well as the SVM. If both classifiers predict the same class, we return the predicted class. In contrast, if the classifiers do not predict the same class, we propose to return the top most probable answer(s) as well as an explanation why those classes are the most probable. 28
4.7 Generating (contrastive) explanations consist of several steps: First, given an input image I in question and the top most probable answer. K denotes training data X for class k labeled with $y = k$. We feed both I and K through the encoder $\mathcal{F}(X)$ to receive their respective semantic location in the latent space. We then find the closest training point that belongs to the target class k and find the vector $\vec{v}$; the direction of that point. Afterwards, uniformly sample ϵ data points along this vector $\vec{v}$, where j iterates over $0 \cdots j \cdots \epsilon$ and is denoted as $Z^{\mu}_{\vec{v}}$. $Z^{\mu}_{\vec{v}}$ is then used to check these against the SVM and use them to generate images $X_{Z^{\mu}_{\vec{v}}}$ using the decoder $\mathcal{G}(Z^{\mu}_{\vec{v}})$. The generated images are then fed to the CNN to make a prediction and as the images will semantically change along the vector the prediction will change as well. Afterwards, we can compare the predictions from both the CNN and SVM. Subsequently, we use the first moment where both predictions are equal to target class k, denoted as moment l for generating an explanation - minimal semantic difference necessary to be equal to the target class, ΔU_l. 29

5.1 30 images taken from the synthetic data, row one represents class 0 (minus sign), row two represents class 1 (plus sign) and row three represents class 2 (multiplication sign. The latent factors are linearly altered from left to right. 31
5.2 Samples from MNIST dataset 32
5.3 Samples from Fashion MNIST dataset 32
5.4 Randomly selected images from the dsprites data set 33

6.1 visualization of a 2-dimensional latent space of a vanilla VAE on Synthetic data, red and blue lines indicate the decision boundaries of the CNN and SVM respectively. . . 36
6.2 visualization of a 2-dimensional latent space of a $\mathcal{T}$-VAE on Synthetic, red and blue lines indicate the decision boundaries of the CNN and SVM respectively. 36
6.3 These figures show the selected / closest data points of another class which largely retains the generative factors that are not related to changing the class. 38
6.4 visualization of a two-dimensional latent space of a vanilla VAE on MNIST, red and blue lines indicate the decision boundaries of the CNN and SVM respectively. 39
6.5 visualization of a two-dimensional latent space of a $\mathcal{T}$-VAE on MNIST, red and blue lines indicate the decision boundaries of the CNN and SVM respectively. 39

6.6 In both figures the red lines indicate the decision boundaries of the CNN whereas the blue lines indicate the decision boundaries of the SVM. It can clearly be seen that the Triplet-VAE has impact on the boundaries and keeps the space well-separated. 39
6.7 Per top k probable answers we traverse and sample the latent space to generate images that can be used to test the behaviour of the CNN. The red line indicates the moment where both the SVM and the CNN predict the target class 41
6.8 Once the SVM and the CNN both predict the target class we capture the minimal changes that are necessary to change their predictions 42
6.9 visualization of a two-dimensional latent space of a vanilla VAE on Fashion-MNIST, red and blue lines indicate the decision boundaries of the CNN and SVM respectively. 43
6.10 visualization of a two-dimensional latent space of a $\mathcal{T}$-VAE on Fashion-MNIST, red and blue lines indicate the decision boundaries of the CNN and SVM respectively . . 43
6.11 The visualization of generated images from uniformly sampling vector v and decoding them using the decoder. 45
6.12 These figures show the selected / closest data points of another class which largely retains the generative factors that are not related to changing the class. 45
6.13 visualization of a 2-dimensional latent space of a vanilla VAE on dsprites 47
6.14 visualization of a 2-dimensional latent space of a $\mathcal{T}$-VAE on dpsrites 47
6.15 The visualization of generated images from uniformly sampling vector v and decoding them using the decoder. 49
6.16 These figures show the selected / closest data points of another class which largely retains the generative factors that are not related to changing the class. 50

A.1 CNN Architecture 58
A.2 Vanilla VAE architecture 59
A.3 Vanilla Convolutional encoder architecture 60
A.4 Vanilla Convolutional decoder architecture 61
A.5 Triplet VAE architecture 62
A.6 Triplet Convolutional VAE - encoder architecture 63
A.7 Triplet Convolutional VAE - decoder architecture 64

List of Tables

5.1 Distribution of MNIST datset 33

6.1 Comparison of Vanilla VAE and triplet-VAE loss on the Synthetic data set 35
6.2 CNN and SVM performance on (reconstructed) training and (reconstructed) test data. 37
6.3 This table shows the percentages of agreement with respect to all possible cases. . . . 37
6.4 Comparison of Vanilla VAE and triplet-VAE loss on the MNIST data set 40
6.5 CNN and SVM performance on (reconstructed) training and (reconstructed) test data 40
6.6 This table shows the percentages of agreement with respect to all possible cases. . . . 41
6.7 Comparison of Vanilla VAE and triplet-VAE loss on the Fashion-MNIST data set . . . 43
6.8 CNN and SVM performance on (reconstructed) training and (reconstructed) test data 44
6.9 This table shows the percentages of agreement with respect to all possible cases. . . . 44
6.10 Comparison of Vanilla VAE and triplet-VAE loss on the dsprites data set 47
6.11 This table shows the percentages of agreement with respect to all possible cases. . . . 48
6.12 Linear support vector machine accuracies per VAE type and latent dimension. It can clearly be seen that in lower dimensions the space cannot be well separated with respect to classes. However, given higher latent dimensions the triplet-VAE clearly creates a more semanticallh meaningful latent space than the vanilla-VAE. 48
6.13 CNN and SVM performance on (reconstructed) training and (reconstructed) test data 49

7.1 linear SVM accuracies with a 32 latent dimension VAE 51

Chapter 1

Introduction

Advances in machine learning and deep learning have had a profound impact on many tasks involving high dimensional data such as object recognition and behavior monitoring. The domain of Computer Vision especially has been witnessing a great growth in bridging the gap between the capabilities of humans and machines. This field tries to enable machines to view the world as humans do, perceive it similar and even use the knowledge for a multitude of tasks such as Image & Video Recognition, Image Analysis and Classification, Media Recreation, recommender systems, etc. And, has since been implemented in high-level domains like COMPAS [1], healthcare [2] and politics [3]. However, as black-box models inner workings are still hardly understood, can lead to undesirable situations [2], such as racial bias [1], gender inequality [4].

This directly or indirectly lead to two important decisions regarding AI, first, The European Union General Data Protection Regulations is extended with a "Right to Explanation" [5], [6] and, second, a "Trustworthy AI framework" is devised by the High-Level Expert Group on AI (AI HLEG) [7]. It is important machine learning practitioners and researchers in the field of interpretability and explainability to take into account the essence of explanations studied by the psychology and social sciences research areas to be able to accurately explain its behavior.

Social- and psychology studies by Miller [8] state that an explanation about a prediction or an interpretation of a example I is usually - in human-to-human conversation - presented in a social, selective and constrastive manner. i.e. we explain a prediction on anchor I by presenting a similar (selected) example J that is similar to I and explain why J is not anchor I as a contrastive example. For instance, take the example of explaining a zebra; Person A believes that it is a zebra and person B does not agree then person A would try to explain that it is definitely a zebra and not something else. And easy way to then explain it is to state that it is a zebra and not a horse because of the addition of stripes, or similarly, it is a zebra and not a donkey because of the addition of stripes. It would not make sense to explain anchor I with an example that is too far-fetched as predicting it as such would be illogical. It would not make sense for Person A's first choice for classification to be a zebra and second choice a orange because they are completely dissimilar.

CNNs are neural networks that are typically used as a discriminative model used for classification, given input data x in the form of an image, it will predict a class y. The direct mapping from x to y is done through several convolutional layers as to capture spatial and temporal aspects within the data, i.e. extract features. The CNN then uses all these aspects embedded in a latent space to make a classification. The decisions are based on the aspects of the embedded latent space and the parameters of the CNN. It does not capture, encode and structure the underlying generative factors meaningfully, it doesn't have to and is not trained to do so. The embedded latent space is therefore hard to understand and even intractable given highly complex architectures.

In contrast to CNN's, generative models aim to capture and understand the underlying generative processes and aim to encode them meaningfully in a latent space. We propose to utilize such a latent space to make a substantiated proposal for a contrastive example. In this thesis we use a Variational Auto encoder (VAE) combined with metric learning to further structure the underlying generative processes in semantically relevant sub-groups with respect to abstract classes defined by humans. The VAE has as extra benefit the ability to generate images by sampling the latent space. This is done to build a semantically relevant latent space that is smooth and that corresponds well

with the generating factors of the data (i.e. regions well-supported by the associated density should correspond to realistic data points) and with a distance metric that conveys semantic information about the target task. The vanilla VAE without any extra constraints is insufficient as is does not necessarily deliver smooth areas or well separated sub-groups that corresponds to the semantics of the target class assignment (in our task). Our target is to develop meaningful decision boundaries in the latent space, which we can use to examine our target classification model and generate explanations.

Additionally, we train a linear SVM on top of the semantically relevant well-separated latent space to evaluate and compare against the target classification model with respect to generative factors underlying the data. This, combined with the generative capacity of the VAE creates a common ground between generative factors, classes, SVM and the trained classification model (CNN). We then use the VAE, SVM, and CNN to generate explanations for a specified anchor I; the anchor is an input image we wish to understand and be 100% sure about. The explanation put forth is similar to the human-to-human explanation explained previously. We propose and select a closely related target class that is not the initial prediction and present the difference or change that would change the initial prediction to the target class. The selective contrastive explanation is generated by finding the minimal important difference of generative factors that are related to changing the class and explain what changes the CNN's and SVM's prediction. By Occam's razor, in general, we prefer the linear interpretable SVM above the highly complex CNN's behavior and see it as ground-truth. However, we can use the latent space to investigate both predictions. The framework we propose for these tasks is more specifically explained using Figure 1.1.

In this paper, the key contributions are: 1) an approach that can be used in order to validate and check predictions made by a CNN by utilizing a weakly-supervised generative model, a VAE combined with metric learning, that is trained to create semantically relevant latent spaces. 2) The semantically relevant latent spaces are then used in order to train a linear support vector machine to capture decision rules that define a class assignment. The SVM is then used to check predictions based on semantics rather than the direct mapping of the CNN. 3) if there is an oddity in the predictions (i.e. the CNN and SVM do not agree, or we wish to investigate a prediction) then we generate a selective contrastive explanation towards a to be questioned class. This is done by traversing the latent space in order to present the minimal important difference that is required to change the input I to the target class and check the behavior of the CNN by generating images along the traversed path.

To conclude, This thesis posits a method that can be used to understand the behavior of an SVM and CNN by generating selected contrastive explanations. This allows for the validation and understanding of both models as to see if they make understandable predictions. This is especially important when both classifiers do not agree on a given input image as this would mean that two classifiers trained for the same goals with the same input disagree with each other. The explanations are provided qualitatively to an expert within the field. This explanation encompasses the original image, reconstructed images and the path towards its most probable answers. Additionally, it shows the minimal difference that makes the classifiers change its prediction to one of the most probable answers. The expert can then check these results to make a quick assessment to which class the image actually belongs to. Additionally, the framework provides the ability to further investigate the model mathematically using the linear classifier as a proxy model.

The main question this paper aims to answer is as follows:

Is it possible to use the properties within the latent space of a generative model in order to generate explanations: contrastive, selective and social, to explain the behaviour of a convolutional neural network(CNN)?

In order to answer the question above this study aims to answer the following sub-questions:

- How can we utilize a generative model in order to generate an explanation to interpret a CNN?
  - What is an interpretation/explanation?
  - How can we measure behaviour of the CNN when comparing it against the generative model?

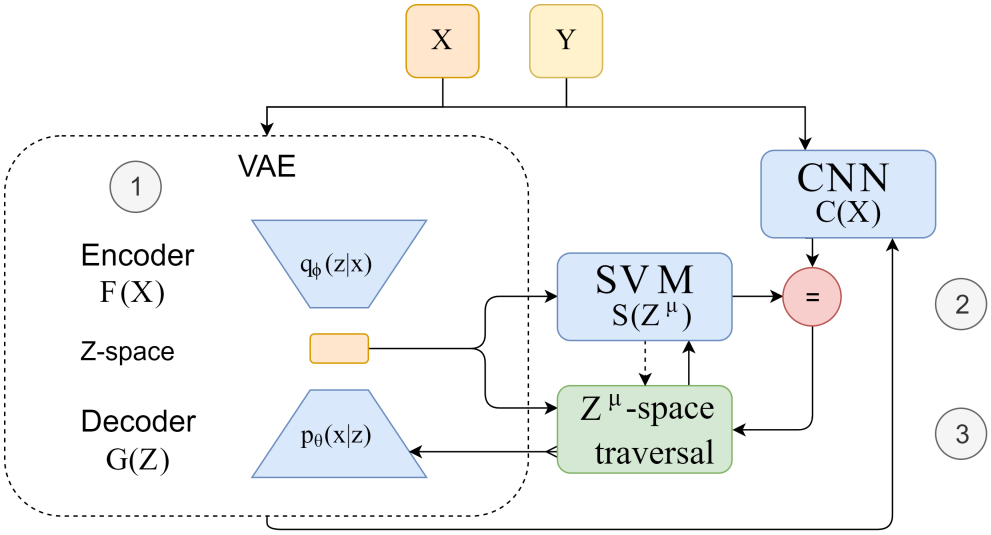


FIGURE 1.1: The diagnostics approach to validate and understand the behavior of the CNN. 1) Metric learning is applied during training of the VAE in order to create semantically relevant latent spaces. The generative model captures the essential semantics within the data and is used by 2) A linear Support Vector Machine. The linear SVM is trained on top of the latent space to classify input within a semantically relevant latent space rather than the direct mapping from input data X and labels Y. If the SVM and CNN do not agree on a prediction then 3) we traverse the latent space in order to generate and capture semantically relevant synthetic images, tested against the CNN, in order to check what elements have to change in order to change its prediction from a to b, where a and b are different classes.

- How can we find semantically relevant areas or paths within the z-space of the generative model?
  - Can we apply different types of loss to alter and better structurize the z-space of the generative model?
  - How can we meaningfully select relevant (contrastive) explanations to interpret the behaviour of the CNN?

This study starts by with a review of related works, followed by current state-of-the-art methods for machine learning in the chapter 2 Related works. Thereafter, chapter 3 Background describes the essential elements of this framework and explains why certain parts are needed. Followed by a complete methodology of the framework, how it works and what additions it brings in chapter 4 Method. Then, in chapter 5 Experiment Design I describe the experimental set up and what data sets are used in order to validate the approach. Subsequently, chapter 6 Results and Discussion reports the scores and findings of the approach and how it can be utilized as well as examples of contrastive examples that are generated using the network. And, finally in chapter 7 Conclusion and Recommendations, I state the conclusions and recommendations for this framework.

Chapter 2

Related works

Interest in interpretability and explainability studies has significantly grown since the inception of "Right to Explanation" [5] and ethicality studies into the behaviour of machine learning models [1], [3], [2], [4]. As a result, developers of AI are promoted and required, amongst others, to create algorithms that are transparent, non-discriminatory, robust and safe. Interpretability is most commonly used as an umbrella term and stands for providing insight into the behaviour and thought processes behind machine-learning algorithms and many other terms are used for this phenomenon, such as, Interpretable AI, Explainable machine learning, causality, safe AI, computational social science, etc. [9]. we posit our research as an interpretability study, but it does not necessarily mean that other interpretability studies are directly closely related to this work.

There have been many different approaches that all work towards the goal of understanding black-box models: Linear Proxy Models: Lime [10] are approaches that locally approximate complex models using linear fits., Decision trees and Rule extraction methods, such as deepred [11] are also considered highly explainable, but quickly become intractable as complexity increases and salience mapping [12] that provide visual information as to which part of an image is most likely used in its prediction, however, it has been demonstrated to be unreliable if not strongly conditioned [13]. Additionally, another approach to interpretability is explaining the role of each part within a black-box models such as the role of a layer or individual neurons [14] or representation vectors within the activation space[15]. All these technique focus on improving interpretability that, while admittedly non-representative of the underlying decision processes, provide some degree of justification for emitted choices that may be used as response to demands for explanation in order to build human trust in the systems accuracy and reasonableness. These systems emulate the processing of the data to draw connections between the inputs and outputs of the system.

Most of the approaches stated above assume that there has to be a trade-off between model performance and explainability. Additionally, as the current interpretable methods for black-box models are still insufficient and approximated can cause more harm than good when communicated as a method that solves all problems. A lot of the interpretability methods do not take into account the actual needs that stakeholders require[16]. Or, fail to take into account the vast research into explanations or interpretability of the field of psychology[17] and social sciences[8]. The "Explanation in Artificial Intelligence" study by Miller [8] describes the current state of interpretable and explainable algorithms, how most of the techniques currently fail to capture the essence of an explanation and how to improve: an interpretability or explainability method should at least include, but is not limited to, a non-disputable textual- and/or mathematical- and/or visual explanation that is selective, social and depending on the proof, contrastive.

2.1 Essentials of Interpretability

Tim Miller [8] states that interpretability research in deep-learning typically does not cite or build on frameworks of explanation from social science. This means that at this moment it is fair to say that most of the work in interpretability uses only the researchers' intuition of what constitutes a 'good' explanation. As such, it is important to first know the essentials of explanations in order to transfer

the correct contextual information. This will help users in understanding the deep-learning models and increase (dis)trust. So, what is interpretability and what are the essentials?

One definition of interpretability is: "Interpretability is the degree to which a human can understand the cause of a decision." Or, the degree to which a human understands a cause is based on the amount of evidence that is available as an explanation for the decision. In general, a complete conviction takes the form of a combination of non-disputable textual- or mathematical- and visual explanation as evidence. In turn, humans use several essentials to make explanations efficient and effective to converse and convince others, the most important ones are: Constrastive explanations, Selected explanations, Social explanations, probability and causality in explanations.[8]

Constrastive Explanations, Explanations are *constrative* - they are sought in response to particular counterfactual cases. That is, people do not ask why event P happened, but rather why event P happened instead of some event Q. which means that most why-questions are constrastive.

Selected Explanations, Explanations are *selected* (in a biased manner) - people rarely, if ever, expect an explanation that consists of an actual and complete cause of an event. Humans are adept at selecting one or two causes from a sometimes infinite number of causes to be the explanation. However, this selection is influenced by certain cognitive biases.

Social Explanations, Explanations are *social* they are a transfer of knowledge, presented as part of a conversation or interaction, and are thus presented relative to the explainers beliefs about the explainees beliefs.

probability and causality in explanations Probabilities probably dont matter while truth and likelihood are important in explanation and probabilities really do matter, referring to probabilities or statistical relationships in explanation is not as effective as referring to causes. The most likely explanation is not always the best explanation for a person, and importantly, using statistical generalizations to explain why events occur is unsatisfying, unless accompanied by an underlying causal explanation for the generalization itself.

The above types of explanations inform us about the way we humans interact. In a day-to-day conversation we try to convince others by providing an explanation of the decision or prediction. We do this by selecting information that is most relevant to the transfer of knowledge we wish to achieve, and prepare contrast for questions that could be asked. This information is then combined in an explanation, and as such, if we want to explain machine- and deep-learning and convince users about the decision we should prepare explanations similarly. How this can be explained is explained a little later in this section, first it is important to note what types of interpretability areas there are.

What are the essentials to interpretability or explanations?; what do we absolutely need to know for research? In interpretability, it is, first, important to know the kind of explanations that are most beneficial in answering the questions we might have of the neural network. section 2.1 Essentials of Interpretability states that most human explanations are *constrastive*, *selected* and *social*. It is, therefore, important to take into account that if explanations are used in an interpretability method then in order to be most effective it should incorporate: constrastive examples that are selected to be most important and are socially picked depending on the public requesting explanations.

Recent years have seen a significant increase in interpretability research. The term interpretability is used as an umbrella term for providing insight into the behavior and thought processes behind machine-learning algorithms and many other terms are used for this phenomenon, such as, Interpretable AI, Explainable machine learning, causality, safe AI, computational social science, etc.[18]. For this thesis I will use the term interpretability. But, what does interpretability really mean and stand for? The main contributing papers I used for this literature analysis are: T. Miller [8], LH Gilpin et al. [18], R. Guidotti et al. [19], F. Doshi-Velez and K. Been [20], A. Adadi and M. Berrada [21] and C. Molnar [22]. Each of these papers explain the notion of interpretability (within machine-learning), or a part of it.

Finally, At the moment of writing there are several papers explaining several intuitive notions of interpretability methods. Explaining how a machine- and deep-learning method works can be subdivided into two main areas. First, the algorithm is interpretable and transparent by design and can be easily interpreted mathematically. And, second, The algorithm is complex and difficult to understand mathematically and requires more elaborate methods to interpret how a prediction is

decided. This we refer to explaining the black box and explaining the black box can be done in several ways. An explanation can be given to the general (global) inner workings of the black box. Accordingly, meaning that we are able to understand the whole logic of the model and follow the entire reasoning leading to all the different outcomes. Additionally, an explanation can be given to a specific (local) instance, providing insight in the reasoning behind the outcome of a specific input example. Or, find a way to inspect the model when we wish to understand how internally the black box behaves changing the input.

Each of these categories; *transparent*, *global*, *local* and *inspection* make use of a variation of several methods that achieve a certain goal towards one or more categories. There are several methods that are regarded as transparent machine- and deep-learning methods are Linear regression, logistic regression, Decision trees, Naive Bayes. For global, local and inspection, each category doesn't necessarily have a specific method or approach, one method can and might be used for multiple categories.

The methods that apply to the categories global, local and inspection can be further categorized as follows: **Deep Network Processing**, **Deep Network Representations** and **Explanation-producing systems**.

1. **Explanations of Deep Network Processing** Some papers propose explanations that, while admittedly non-representative of the underlying decision processes, provide some degree of justification for emitted choices that may be used as response to demands for explanation in order to build human trust in the systems accuracy and reasonableness. These systems emulate the processing of the data to draw connections between the inputs and outputs of the system.

   (a) **Proxy Model**

      i. A **linear proxy model** explains a black box model by probing the behavior on perturbations on the input, and then that data is used to construct a local linear model that serves as a simplified proxy for the full model in the neighborhood of the input. This can be used in order to identify regions of the input that are most influential for a decision across a variety of types and problem domains. One such example is LIME [10].

      ii. **Decision Trees and rule extraction** are a form of proxy models that decomposes a neural network or a model into a set of decisions rules. These decision rules aid in showing which decisions lead to which classifications. The decision trees for neural networks typically involves extracting rules to mimic the behavior by means of decomposing individual units(neurons).

   (b) **Salience mapping** Salience mapping is the technique of perturbing pixels of the image to find the neurons that maximally activate and create a heat map to show the areas of the image that have the most influence for a prediction.

2. **Explanations of Deep Network Representations** Another purpose of an explanation is to explain the representation of data inside the network. These provide insight about the internal operation of the network and can be used to facilitate explanations or interpretations of activation data within the network. This is comparative to explaining the internal data structures of the program, to start to gain insights about why certain intermediate representations provide information that enables specific choices.

   (a) **Role of Layers** Role of layers with respect to the representations that are learned within a layer. Each layer can be tested by their ability to solve different problems than what the network was initially trained on. If it is understood what a certain layer is trained for representation/concept-wise, these can be used for other problems without retraining a new network.

   (b) **Role of Individual neurons** The information that is encapsulated within a layer can be further subdivided into individual neurons or convolutional filters. Each individual neuron

can then be tested on what their role is by visualizing the pattern the neuron maximally responds to. One way to do this is by applying gradient ascent.

(c) **Role of (representation) Vectors** Another way to approach this is to find a way to learn representations by vectors. It is possible to find representation vectors within the activation space such that each vector represents a certain concept. This technique is relatively new but seems promising as it allows us to find concepts that are closely related to the concepts humans use.

3. **Explanation-producing networks** These networks are specifically built to explain themselves, and they are designed to simplify the interpretation of an opaque subsystem. They are steps towards improving the transparency of these subsystems; where processing, representations, or other parts are justified and easier to understand.

   (a) **Attention Networks** Attention-based networks learn functions that provide a weighting over inputs or internal features to steer the information visible to other parts of a network. In essence, attention mechanisms in neural networks serve to orient perception as well as memory access. Attention mechanisms filters the perceptions twice; first it filters perceptions to store them in memory and a second time when they are to be retrieved from memory. It is mainly used for natural language processing tasks and machine translation. While this is not directly created for interpretability, it provides a means of extract extra information from the network that could be used as an explanation.

   (b) **Disentangled Representations** Another way to maximize interpretability is to make use of disentangled representations; these have individual dimensions that describe meaningful and independent factors of variation.

   (c) **Generated Explanations** It is also possible to design networks in such a way that they generate their own human-understandable explanations as part of the explicit training of the network.

Interpretability goals can be subdivided into sub-goals of explanation: *global* explanation, *local* explanation, *inspection* of non-transparant models or the use of *transparant* models. There are currently several methods and categories that are being researched: Explanations of the deep network processing; which explains the inner workings of a neural network by means of a proxy, decision tree, rule extraction or salience mapping. Explanations of deep network representations; Which explains the representations that are used in the network, by means of explaining the role of layers, neurons or vectors. And, finally. Explanation-producing systems, which are systems that are designed in such a way that explanations are integrated within the network. This can be in the form of disentangled representations, generated explanations, or attention networks.

2.2 State-of-the-art methods

This section describes the current state-of-the-art interpretability techniques and algorithms related to each category stated in section 2.1 Essentials of Interpretability

2.2.1 Explanations of Deep Network Processing

Proxy Models

linear proxy model **Lime**[10] is an example of a linear proxy model. In essence, LIME explains a neural network model by probing behavior on perturbations of an input, and then that data is used to construct a local linear model that serves as a simplified proxy for the full model in the neighborhood of the input. This allows the method to identify regions of the input that are most influential for decisions.

Decision Trees and Rule Extraction There are three techniques that stand out when it comes to rule extraction, **DeepRED**[11], **Anchors**[23] and **LORE**[24] are techniques try to automatically extract rules out of a neural network(DeepRED, or any model(Anchors and LORE). First, DeepRED extends work from the 1990s in order to generalize the method for deep neural networks and not just shallow networks. This is done by a decomposition algorithm that extracts intermediate rules for each layer of a DNN. These rules are then merged to describe the DNN's behavior by means of its input. Second, the Anchors' algorithm is an extension of LIME and randomly constructs anchors with the highest coverage and respecting a user-specified precision threshold. An anchor explanation is a decision rule that sufficiently ties a prediction locally such that the rest of the values do not matter. That is, similar instances covered by the same anchor have the same predictive outcome. Third, LORE learns a local interpretable predictor on a synthetically generated neighborhood through a genetic algorithm. It then uses its logic to derive decision rules in the form of a decision tree, explaining reasons for a decision and also providing counterfactual rules that would change the outcome.

Salience mapping

There are a lot of Saliency mappings currently available, all with a slight alteration that tries to improve results. A few examples of current state of the art saliency mapping methods are: **GRAD-CAM**[12]. **SmoothGrad**[25], **Integrated gradients**[26]. All of these methods involve the network being tested with portions of the input occluded to create a map showing which parts of the data actually influence on the network data. an example of a saliency map can be seen in 2.1.

FIGURE 2.1: Saliency map of CNN classifying a cat

2.2.2 Explanations of Deep Network Representations

Role of layers

There are several ways to explain and visualize the role of layers. The papers about **feature visualization**[27], **building blocks of Interpretability**[28] and the **Activation Atlas**[14] by the community of Distill show a great way to visualize and probe the inner workings of a neural network. There are several methods available to allow for the visualization of the layers of a network or by more specifically targeting a single neuron for information. These methods thus also apply for section 2.2.2 Role of neurons

Role of neurons

Next to the methods in section 2.2.2 Role of layers, there are also the following: **dissection**[29], **maximize activation sampling**[30] and **generative networks** [31]. First, **Network Dissection** works by measuring the ability of individual units to solve a segmentation problem over a broad set of labeled visual concepts. The accuracy of these units will for a certain task can then be used to characterize the kind of information that a unit activates to. Second, maximize activation sampling makes use of sampling in order to find to what kind of images the neuron activates the most. The subset of images that is sampled can then be used to give an indication of what that specific neuron is looking for and

is activated by. Next, the method described by Nguyen et al. makes use of the Activation Maximization method in combination with a Generative Adversarial Network in order to generate images that relate to what the neuron activates as well as an image that looks real and therefore interpretable by humans.

Role of vectors

Achieving interpretability by means of vectors within neural networks is relatively new and has not been investigated fully. **TCAV**[15] is a global perturbation method which perturbs data to human-relatable concepts in order to generate an explanation of the model. The method described in the paper consists of multiple steps, first, define human-relatable concepts in the form of a vector space. This can be done by training a binary linear classifier on a set of positive concept example inputs and a negative set of example inputs. Then, by finding the vector orthogonal to the decision boundary in the direction of the positive examples a concept activation vector(CAV) is found.

This method results in a human-relatable linear interpretation of the internal state of a deep learning model. This means that this technique can be used as an interpretation method to link human-relatable concepts to a model and analyze if this concept is used or not.

2.2.3 Explanation-Producing Networks

Attention Networks

One of the first papers where attention was introduced into natural language processing is "Attention is all you need" [32] by Ashish Vaswani et al. It describes a neural network that uses a attention mechanism in order to translate sentences from one language to another. This is done by directing the attention of the network such that it disregards noise and focuses on what is relevant. Another paper [33] by Tianjun Xiao et al. focuses on applying attention mechanisms in deep convolutional neural networks for fine-grained image classification. It uses three-types of attention: first, the bottom-up attention that proposes candidate patches, the object-level top-down attention that selects relevant patches to a certain object, and the part-level top-down attention that localizes discriminative parts. As such, it finds areas of interest within the images that could be used for interpretability.

Disentangled Representations

The problem of disentangling latent factors has been around for quite a while, and a has previously been tackled by **Principal Component Analysis**[34], **Independent Component Analysis** [35]. More recently the **Variational Autoencoder(VAE)**[36] and Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets (**InfoGAN**)[37] have shown promise in disentangling latent factors.

Generated Explanations

The following paper "**Grounding visual explanations**" by [38] is a continuation of an earlier work [39], [40] and [40]. This relatively new paper from August 2018 states the current state-of-the-art interpretability approach for textual and visual interpretability.

The technique combines the LCRN and the ground-truth bounding-box of the papers stated above and adds as additional element a counter-factual explanation why it is not another class. A deep-learning algorithm combined with these techniques produces results that not only provide a prediction of the class but also a textual- and visual explanation where the textual explanation is class- and image relevant and provides information on why it is not another class or prediction. See figure 2.2 as an example of textual and visual explanation.

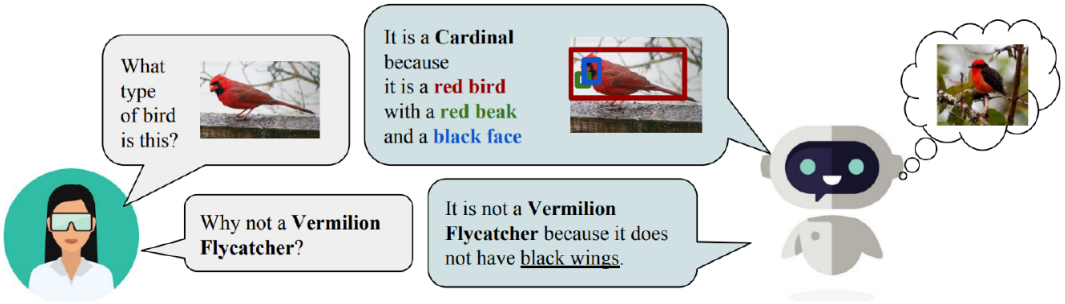


FIGURE 2.2: Combination of visual and textual evidence

Contrastive explanations

The following papers: a (rejected) pre-print of **CDeepEx**[41] by Amir Feghahati et al. and **Xgems** [42] by Shalmali Joshi et al. I came across while preparing this paper. These works both focus on generating explanations using an implicit generative models, primarily using generative adversarial networks (GANs). The xGems paper presents an approach to characterize and explaining black-box supervised models via examples. An unsupervised implicit generative model is used to approximate the data manifold and subsequently used to guide the generation of increasingly confounding examples given a starting point. These examples are used to probe the target black-box in several ways. They try to utilize manifold guided examples in order to detect bias in black-box learning. The CDeepEx paper mainly focuses on answering the question "why a and not b?" with GANs, where a is the class of an input example I and b is a chosen class to which to capture the differences. Next, it uses the captured differences in order to generate contrastive examples. These papers are very similar to the approach I am taking and both try to provide an explanation using a trained generative model. The approach I am taking is different in the way of manifold learning and trying to further structure the generative model further to understand the implicit generative model better and to allow new techniques.

2.3 Relevance of research

The interpretable machine-learning research community has devised various methods in order to try to convince and secure trust with high-level domains as shown in section 2.2 State-of-the-art methods. However, Most explanations fail to take into account the essentials of an explanation, 2.1. And, Also, the existing approaches have at least one of the following downsides:

- A proxy model will never explain the exact behavior of the to be validated model but gives an indication of its decision behavior.
- The need of extensive computational time and intractability of a single explanation by means of a decision tree. It will be able to explain all reasoning, however, computing a complete tree for every parameter in a deep neural network will require some kind of approximation and loss of expressiveness to make it readable.
- The use of approximations during back-propagation to generate explanations as is the case in salience maps makes it unreliable and is not invariant to random simple changes.
- Explanations of single neurons or layers within a (deep) neural network give some indication of what concepts a black box model uses but are hard to interpret and is intractable, and does not scale for deep neural networks as it requires the need to check all the units to see which one (or multiple) unit(s) correspond to a certain concept as shown by Morcos et al.[43]. And, additionally, these concepts does not have to relate to concepts humans use, making the interpretations subjective.

- The TCAV method shows promise by being able to test for certain concepts within black box model. However, it does not provide a method for checking which concepts are used for a single prediction but rather tests models to see if specific concepts are used one way or another by the model.
- The need for a set images or concepts with complex annotations and information which may not be available or cannot be easily obtained in real life situations
- The generated explanations created by Hendricks et al.[38] are similar to explaining the why of a input image. The explanation network is trained to match human explanations and therefore create explanations that humans would like to see, but, do not show how the to be validated network makes its decisions

For these reasons, our approach focuses on providing selective (contrastive) explanations that combines visual aspects as well as the ability to further investigate the model mathematically using a proxy model that does not impact the CNN directly. Usually, generative models such as the Variational Autoencoders(VAE)[44] and Generative Adversarial Networks (GAN)s are unsupervised and used in order to sample and generate images from a latent space, provided by training the generative network. However, we posit to use a weakly-supervised generative network in order to impose (discriminative) structure by means of metric learning [45], [46] in addition to variational inference to the latent space. As such we will create a semantically relevant latent space that can be used for various tasks. If we can meaningfully traverse and sample the latent space, we can use the generated synthetic data to probe the behavior of the CNN. As such we can probe questions related to bias or why a certain prediction was made; 'why was class a predicted over class b?'.

This approach and method is therefore most related to the interpretability area of sub-sampling proxy generative models to answer questions about a discriminative black box model. The two closest studies that attempt similar research is a preprint of CDeepEx[41] by Amir Feghahati et al. and xGEMs [42] by Shalmali Joshi et al. Both cDeepEx and xGEMS propose the use of a proxy generative model in order to explain the behavior of a black-box model, primarily using generative adversarial networks (GANs). The xGems paper presents a framework to characterize and explaining binary classification models by generating manifold guided examples using a generative model. The behavior of the black box model is summarized by quantitatively perturbing data samples along the manifold. Additionally, xGEMS detects and quantifies bias during model training to understand how bias affects black box models. The xGEMS approach is similar to our approach as in using a generative model in order to explain a black box model. Similarly, the cDeepEx paper posits their work as generating contrastive explanations using a proxy generative model. The generated explanations focus on answering the question "why a and not b?" with GANs, where a is the class of an input example I and b is a chosen class to which to capture the differences.

However, both of these papers do not state that in a multi-class (discriminative) classification problem if the generative models' latent space is not smooth, well seperated and semantically relevant then unexpected behavior can happen. For instance, when traversing the latent space it is possible to pass from class a through any number of classes before reaching class b because the space is not well seperated and smooth. This will create ineffective explanations, as depending on how they generate explanations will give information on 'why class a and not b using properties of c'. An exact geodesic path along the manifold would require great effort, especially in high dimensions. Also, our approach is different in the fact that we utilize a weakly-supervised generative model as well as an extra linear classifier on top of the latent space to provide us with extra information on the data and the latent space. Some approaches we take, however, are very similar, such as using a generative model as a proxy to explain a black-box model as well as sub-sampling the latent space to probe the behavior of a black-box model and generate explanations using the predictions.

All in all, I would like to propose an interpretability method that does not require changes or impacts of a to be validated model in any way. I'd like to focus more on probing, criticizing and explaining the decisions made by a model by applying the expert knowledge available in the field. In short, a model-agnostic method that provides meaningful information to a given prediction as to

help experts in the field to understand and (dis)trust the reasoning behind a decision. The technique I'd like to propose does not suffer from the aforementioned downsides except for the fact that it utilizes another black box to explain another black box by generating synthetic images and encoding the data into a simpler representation. It will then use the synthetic images to probe the decision process of the to be validated model and difference between images with different predictions. A more elaborate description of the proposed method can be found in chapter 4 Method.

Chapter 3

Background

3.1 Machine learning

Machine learning is the science of getting computers to act without being explicitly programmed [47]. The machine learning algorithms try to create a statistical model to achieve a certain task such as regression, classification, feature selection, clustering, etc. The algorithms are trained to optimize its parameters in order to lower the loss function and define decision rules related to its task. For example, take the task of automating tagging of vacation pictures, you could present a machine-learning system with many examples of pictures already tagged by humans, and the machine learning algorithm will learn statistical rules for associating specific pictures to specific tags by optimizing its parameters as to lower its loss function.

Machine learning models can be subdivided into black and white-box models. Black models such as neural networks, whilst having a high accuracy in tasks, can be generalized as models that are too complex and hard to explain. Besides, they are not trained to explain their decision but rather be the best at their specific task. Additionally, the black-box models do not provide an estimate of the importance of each feature on the model predictions, nor is it easy to understand how the different features interact. On the other hand, White models such as linear regression and decision trees, whilst weaker in accuracy, are generally easier to understand, explain and interpret but are not always able to capture the inherent complexity of the data. In this thesis we mainly focus on black-box models such as neural networks as we wish to create a framework which would allow the black-box models to be interpreted or explained.

Black-box neural networks can be sub-categorized into two categories; Discriminative and generative models. Discriminative models are usually supervised and of the form $P(y \mid x)$ i.e. a direct mapping from the input x to a class label y. These types of models are trained to make future predictions y for any given x. Whereas generative models focus on studying the joint probabilities P(X,Y). A generative model tries to study the underlying generative factors to understand how the data is generated in the real world but also make stronger assumptions than discriminative models. For this reason, if the task is primarily discriminative it might be better to just use a discriminative model instead of a generative model As often when the assumptions on the data are wrong this often leads to a higher asymptotic bias [48]. However, they can still be used in order to understand the underlying process and try to understand how well the discriminative model adheres to the generative model.

The information from trying to understand the joint distributions over all variables has as extra benefit that it can act as auxiliary truth. This truth can then be used to understand how the abstract classes by humans are defined and adhere to the generative factors and if bias is introduced. Similarly, we can use the generative factors to see if the CNN is trained in the way as we expect it to. This is what I am going to try to do, use a generative model (VAE) to understand the underlying processes within the data and use this information to understand the decision process of a CNN. Another application would be to use the capability of generating images to counteract bias within a CNN.

Most machine learning systems require the ability to explain to stakeholders why certain predictions are made. When choosing a suitable machine learning model, we often think in terms of the accuracy vs. interpretability trade-off. This means choosing for black-box algorithm(s) that

are highly accurate but complex to interpret such as neural networks, gradient boosting models or complicated ensembles. Or, conversely, white-box algorithm(s) that are slightly worse in terms of accuracy but more interpretable such as linear regression, linear classification models and decision trees. These models provide less predictive capacity and are not always capable of modelling the inherent complexity of the data but are significantly easier to explain and interpret. Next section specifically focuses on the models that are used within this approach; linear Support vector machine which is highly interpretable but is not always capable of modelling the inherent complexity of the data, Convolutional neural networks have been proven to be highly accurate but hard to interpret and a Variational Autoencoder, which focuses on trying to uncover the generative factors underlying the data and encode it in a latent space.

3.1.1 Discriminative models

(Linear) Support vector machine (SVM)

In machine learning, Support-vector Networks by Vapnik, Vladimir N. [49] are supervised learning models with associated learning algorithms that analyze data used for classification, regression analysis and pattern recognition tasks. The idea is to find a hyper-plane that best divides a data set into two classes. A hyper-plane can be written as: $\vec{w} \cdot \vec{x} = b$ where at least one of the w's is non-zero and b is an arbitrary constant. The classifier works by drawing a hyper-plane between two classes; All the points on one side of the line will be labeled as class 1 and all the points on the other side as class 2. We aim to fit the following function: $\hat{y} = \vec{w} \cdot \vec{x} + b \geq 0$ or $\hat{y} = w_0 * x_0 + w_1 * x_1 + \cdot + w_p * x_p + b \geq 0$. When $\hat{y} < 0$, predict -1, otherwise if $\hat{y} \geq 0$ predict class + 1. The distance from the closest samples to the decision boundary is the *margin* where the closest samples to the decision boundary are the *support vectors*. The optimization objective of the SVMs is to maximize the margin.

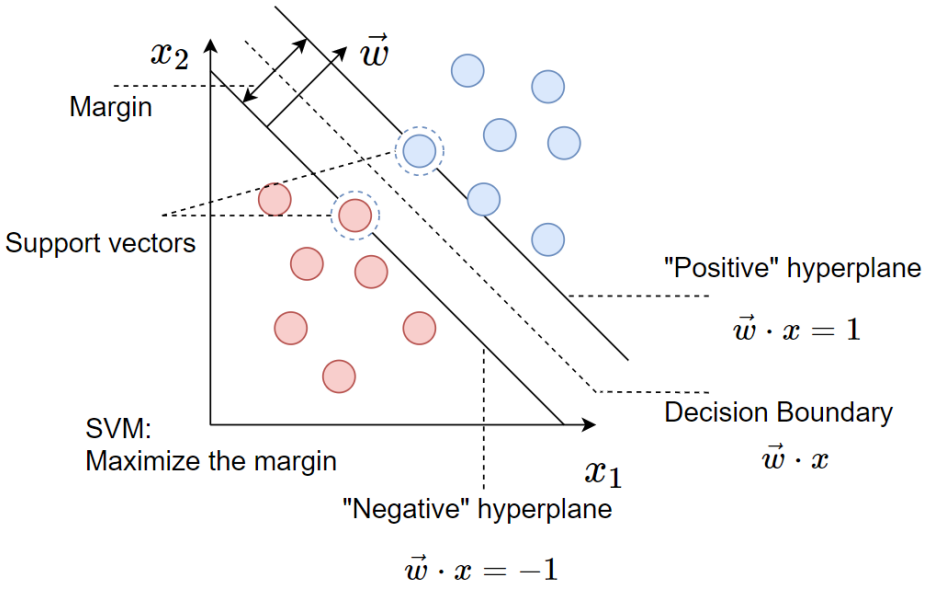


FIGURE 3.1: The goal of an SVM is to find a decision boundary such that the margin is maximized, where the margin is the distance from the support vectors to the decision boundary.

In particular, we define a positive hyper-plane as $\vec{w} \cdot \vec{x_+} + b \geq 1$ with the x_+ as the positive support vectors. Similarly, the negative hyperplane is defined as $\vec{w} \cdot \vec{x_-} + b \leq -1$. Now, for mathematical convenience we introduce an extra variable y_i such that $y_i = +1$ for positive samples and $y_i = -1$ for negative samples. Now, this can be used to simplify the expressions above to

$y_i(\vec{w} \cdot x_i + b) \geq 1$ or $y_i(\vec{w} \cdot x_i + b) - 1 \geq 0$ for all i, i.e. all negative samples should be in the negative hyper-plane and all positive samples should be in the positive hyper-plane. If we subtract these expressions we yield $\vec{w}(x_+ - x_-) = 2$ and then normalize by the length of $||w|| = \sqrt{\sum_{j=1}^{m}(w_j^2)}$ we get $\frac{\vec{w}(x_+ - x_-)}{||w||} = \frac{(1-b+1+b)}{||w||} = \frac{2}{||\vec{w}||}$ Which is in fact the distance between the positive and the negative hyper-plane, or, the *margin* which we wish to maximize. Now, we wish to maximize $\frac{2}{||w||}$ which is the same as saying we minimize $\frac{||w||^2}{2}$. This is now a quadratic objective with linear constraints and can thus be solved by the Lagrangian multiplier method.

The primal formulation of the lagrangian objective function is:

$$minL_p = \frac{||w||^2}{2} - \sum_{i=1}^{n}[a_i y_i(x_i \cdot \vec{w} + b)] + \sum_{i=1}^{n}[a_i]$$

subject to $a_i \geq 0$, $\frac{dL}{d\vec{w}} = \vec{w} - \sum_{i=1}^{n}[a_i y_i x_i] = 0 \implies \vec{w} = \sum_{i=1}^{n}[a_i y_i x_i]$ and $\frac{dL}{db} = -\sum_{i=1}^{n}[a_i y_i] = 0 \implies \sum_{i=1}^{n}[a_i y_i] = 0$. Where n is the number of training examples and a is the dual variable, which will act as a weight for each training example. First, the optimal set of a's are found and then afterwards we can easily compute w. Substituting for $\vec{w}$ gives us the dual formulation:

$$minL_d = \frac{1}{2}(\sum_{i=1}^{n}[a_i y_i x_i]) \cdot (\sum_{j=1}^{n}[a_j y_j x_j]) - (\sum_{i=1}^{n}[a_i y_i x_i]) \cdot (\sum_{j=1}^{n}[a_j y_j x_j]) - \sum_{i=1}^{n}[a_i y_i] + \sum_{i=1}^{n}[a_i] \tag{3.1}$$

$$= \frac{1}{2}(\sum_{i=1}^{n}[a_i y_i x_i]) \cdot (\sum_{j=1}^{n}[a_j y_j x_j]) - (\sum_{i=1}^{n}[a_i y_i x_i]) \cdot (\sum_{j=1}^{n}[a_j y_j x_j]) - 0 + \sum_{i=1}^{n}[a_i] \tag{3.2}$$

$$= \sum_{i=1}^{n}[a_i] - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}[a_i a_j y_i y_j (x_i \cdot x_j)] \tag{3.3}$$

This, in essence, tells us that the optimization is only dependent on $x_i \cdot x_j$ or pairs of samples. The training samples for which a_i is not 0 are the *support vectors* and the SVM is therefore defined by the support vectors and the coefficients. Also, in a similar sense, if we know the coefficients a_i we can classify a new sample u with the decision function $\vec{w} \cdot \vec{u} + b$ and if we substitute $\vec{w} = \sum_{i=1}^{n}[a_i y_i x_i]$ in the decision function. We see that $(\sum_i^n = 1[a_i y_i \vec{x}_i \cdot \vec{x}] + b) \geq 0$. To conclude, with this method we can find a decision boundary which maximizes the margin between a couple of the closest points to the decision boundary. In this way we can classify new unknown points by checking on which side of the decision boundary the new unknown points lie. However, this (linear) method only works when the data is linearly separable. If the data is not linearly separable tricks then can be applied in order to still find a suitable decision boundary. This can be done by kernel tricks and/or a slack variable C.

A linear SVM is of interest for this study as we know that we have the following information: input data X, labels Y, a Convolutional Neural Network(CNN) and a Variational AutoEncoder(VAE). We can utilize the VAE to construct a chosen z lower dimensional representation z-space of the input data x via the encoder of the VAE. Afterwards, this lower dimensional space be used in order to generate new images that look like the real images. As linear models work well in high dimensional spaces we can try to find a well-performing linear classifier by adjusting the dimensionality of the z-space of the VAE. And, compare the linear classifier of the lower dimensional space against the CNN, which is trained against the higher dimensional data x. Also, the loss function of the VAE can be altered to impose more structure in the data, allowing for a better separation of the data.

(deep) feedforward Neural Networks

(Deep) feed forward neural networks, often also called multi-layer perceptrons(MLP) are the most common deep learning models and typically of the form as shown in 3.2. And, is typically a mathematical construct consisting of an input layer, one or more hidden layers and an output layer. Its goal is to approximate some function f^*. For example, for a classifier, $y = f^*(x)$ the function maps an

input x to a category y. A feed-forward network defines a mapping $y = f^*(x;\theta)$ and learns the value of the parameters θ that results in the best function approximation. [50] Each layer of the neural network consists of a number of perceptrons and could be seen as a filter of the data where each layer tries to extract new representations of the data. Each perceptron processes the data with an assigned weight and bias; in the simplest form : $y = \sum_i x_i w_i + b$. However, in order to learn a non-linear model, each hidden node and the output node has to output a non-linear activation function f on the weighed sum of the inputs $h(x) = f(w_1 x + b)$. Hidden nodes most commonly use RELU and tanh.

RELU

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Tanh

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

sigmoid

$$f(x) = \frac{1}{1+e^{-}x}$$

Whereas the output node use softmax or sigmoid, which transforms the input into a probability for each specific output or class which is very similar to logistic regression. This happens for each and every layer except for the input layer and, thus, get functions of the form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. These chain constructions are he most commonly used structures of neural networks.

Ultimately, the goal of training a neural network is to minimize the error of the class prediction. For classification it is common to use a loss function like cross entropy. If we have M classes and $y_{i,m} = \begin{cases} 1 & \text{if } x_i \text{ belongs to class } m \\ 0 & \text{Otherwise} \end{cases}$ Then the loss function of the classifier can be formulated with cross entropy as:

$$L(Y, Y^{'}) = -\log P(Y, Y^{'}) \tag{3.4}$$

$$= -\frac{1}{n}\sum_{i}^{n}\sum_{m}^{M} y_{i,m} \log y^{'}_{i,m} \tag{3.5}$$

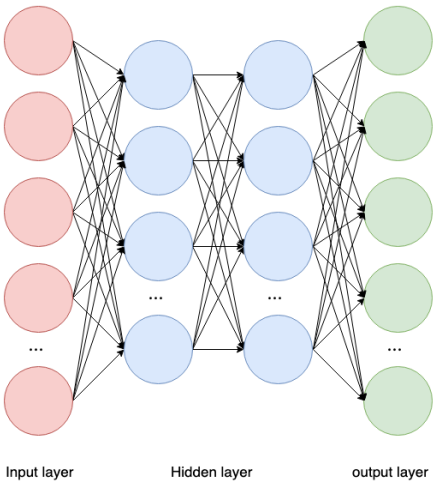


FIGURE 3.2: Simple FFN or MLP consisting of four layers, input layer, two hidden layers and an output layer

where $y^{'}_{i,m} \in Y^{'}$ is (softmax) output of the output layer in the form of predicted probability estimate that x_i belongs to class m. MLP's are used almost everywhere in machine- and deep- learning research and form the basis for various (un)supervised discriminative and genertive models.

Convolutional Neural Networks

On November 1998, Yann Lecun et al. presented Convolutional Neural Networks(CNNs) as an extension on Multilayer Perceptron networks[52]. Convolutional neural networks are designed to process data that come in the form of multiple arrays, for example a color image composed of three 2D arrays containing pixel intensities in the three color channels. His work involves adding a feature extractor to the typical MLPs in order to retain spatial and temporal aspects within the data. In other words, we wish to discover local patterns (between nearby pixels) such as edges lines or structures. This is done through convolutional layer - which is where the model gets its name from - and can be seen as mathematical operation on two functions (f and g) that produce a third function expressing the shape of how one function is modified by another, See figure 3.3 for an example.

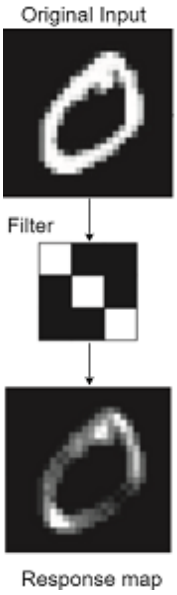


FIGURE 3.3: Example of an convolution step, where a $n * n$ filter/kernel is slid over $n * n$ patches of the input feature map, the response feature map is an image where the pixel values are replaced by the convolution of the kernel with the underlying image patch. image taken from [51]

The convolutional layer is an automated feature extractor and can be seen as a preprocess step as previously convolutions on the data were computed manually before passing it to the model. This is done by initializing a set of d filters/kernels randomly and wish to optimize these in order to learn the optimal values for classification. These d filters are slid across the input image in parallel producing a $(1 * 1 * d)$ output per batch which can then be reassembled into the final feature map with d channels. Additionally, convolutional neural networks have since been developed further to include other layers such as sub- or under-sampling, max-pooling, average-pooling or dropout to further increase efficiency and effectiveness.

The current state of the art convolutional neural network architectures are typically structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. Detection of local conjunctions of features is done by the convolutional layer, whereas the role of the pooling layer is to merge semantically similar features into one.[51] Neurons in a convolutional layer are organized in feature maps, where each neuron is connected to local patches in the feature maps of the previous layer through a set of weights. Furthermore, The last few layers are usually fully connected layers in order to process all the extracted features. Since the inception of CNNs there have been a lot of experiments with great success, using a combination of convolutional layers and typical Fully connected layers in order to reach great levels of accuracy in image classification, such as AlexNet in 2012 [53] with accuracies of 39.7% top-1 and 18.9%top-5.

3.1.2 Autoencoder

Autoencoders are simple learning circuits which aim to transform inputs into outputs with the least possible amount of distortion. Additionally, autoencoders achieve this by learning a dense representation that contains the most important features of data[54]. This means that autoencoders have to be very efficient and selective when deciding which factors help best in reconstructing the original image and are therefore usually used for dimensionality reduction, denoise images and finding data embeddings[55]. Autoencoders are comprised of two connected components, the encoder and decoder, as shown in figure 3.4. These two are mirror image's of each other; the encoder encodes the input data x into a lower dimensional latent space z, meaning that the number of neurons decrease with each layer until the z layer. Whereas the decoder uses the latent space z in order to reconstruct x where each layer increase in the number of neurons until it has the same dimensions as the input. The autoencoder is trained in order to best reconstruct the input image and thus needs to be efficient with the encoding in order to best reconstruct. Given input data $X : x_1 \cdots x_n$ where each x_i is an input vector of the form $x_i = [0,1]^d \in X$. The input data is mapped to a lower dimensional representation $z_i = [0,1]^{d'} \in Z$ through a deterministic mapping $z_i = \mathcal{F}_\theta(x_i) = s(Wx_i + b)$, where W is a $d * d'$ weight matrix, b is a bias vector and is parametrized by $\theta = \{W, b\}$. The resulting latent representation z_i is then mapped back to a "reconstructed" vector $\hat{x}_i \in [0,1]^d$ to the input dimensions $\hat{x}_i = \mathcal{G}_\phi(z_i) = s(W'z_i + b')$ where W' is a $d * d'$ weight matrix, b' is a bias vector and is parametrized by $\phi = \{W', b'\}$. Each training sample x_i is thus mapped to a latent z_i and then reconstructed $\hat{x}_i$ [56], [57]. The parameters θ and ϕ are optimized in order to minimize the reconstruction error and is typically done with squared error loss or cross-entropy loss and can be defined as follows:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \frac{1}{n} \sum_{i=1}^{n} [L(x_i, \hat{x}_i)] \tag{3.6}$$

$$= \arg\min_{\theta,\phi} \frac{1}{n} \sum_{i=1}^{n} [L(x_i, \mathcal{G}_\phi(\mathcal{F}_\theta(x_i)))] \tag{3.7}$$

Where L is the squared error $L(x, \hat{x}) = ||x - \hat{x}||^2$ or the cross entropy loss $L_H(x, \hat{x}) = \mathcal{H}(\mathcal{B}_x || \mathcal{B}_{\hat{x}}) = -\sum_{j=1}^{m} [x_j \log \hat{x}_j + (1 - x_j) \log(1 - \hat{x}_j)]$. An example of a typical autoencoder architecture is shown in 3.4

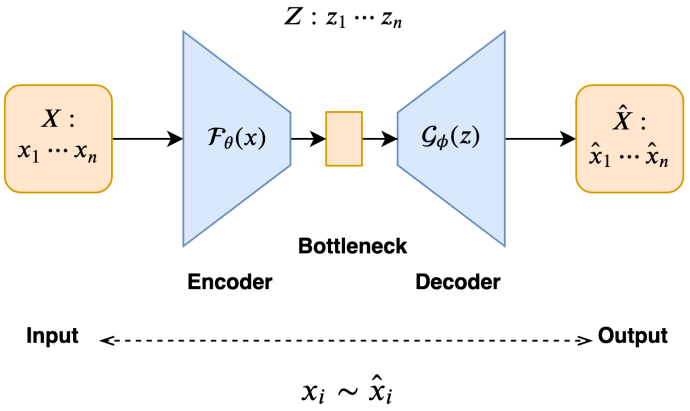


FIGURE 3.4: The above figure shows the structure of an Autoencoder. The autoencoder attempts to best reconstruct an input image x by $\hat{x} = \mathcal{G}_\theta(\mathcal{F}_\phi(x))$ where the encoder = $\mathcal{F}_\phi(x)$, the decoder = $\mathcal{G}_\theta(z)$ and z is a lower dimensional latent space where factors are encoded as efficiently as possible.

3.1.3 Generative models

Variational Autoencoder (VAE)

The Variational AutoEncoder, described by Kingma et al. in [44],[58], is a principled framework for learning deep latent-variable models i.e. a latent variable model $p_\theta(\mathbf{x},\mathbf{z})$ whose distributions are parameterized by neural networks. The VAE framework consists of two coupled, but independently parameterized models: A recognition model, a generative model. The framework proposed in this thesis will utilize both in order to validate or test the behaviour of a CNN. The recognition model creates a mapping from the input space to a latent variable space. The latent variable space is roughly distributed according to its (assumed) prior. This latent variable space can then be sampled and used in combination with the generative model to reconstruct an image that is close to the input images, see figure 3.5. These objectives are combined in a single objective for a neural network, specifically minimizing the minus of the evidence lower bound (ELBO) where the ELBO is the loss that encompasses inference between the true distribution and the assumed prior and the reconstruction error of true images and reconstructed images. This requires two steps: first, we have to measure the difference between the true distribution, the approximated distribution and secondly, the difference between the true image and the reconstructed image. However, construction of the true distribution is typically intractable, as it requires computing the true posterior density $p_\theta(\mathbf{z} \mid \mathbf{x})$.

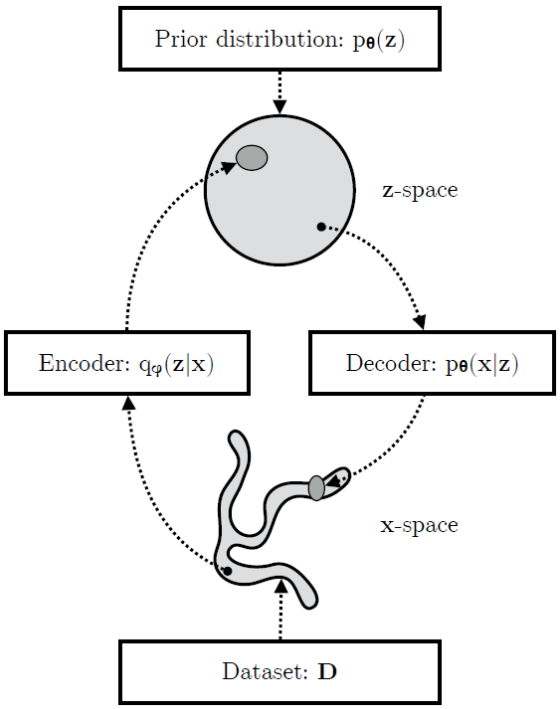


FIGURE 3.5: A VAE learns stochastic mappings between an observed x-space, whose empirical distribution $\amalg_D(x)$ is typically complicated, and a latent z-space, whose distribution can be relatively simple (such as spherical, as in this figure). The generative model learns a joint distribution $p_\theta(\mathbf{x},\mathbf{z})$ that is often (but not always) factorized as $p_\theta(\mathbf{x},\mathbf{z}) = p_\theta(z)p_\theta(x|z)$, with a prior distribution over latent space $p_\theta(z)$, and a stochastic decoder $p_\theta(x|z)$. The stochastic encoder $q_\phi(z|x)$, also called inference model, approximates the true but intractable posterior $p_\theta(z|x)$ of the generative model. Image taken from [44]

This can be done by applying bayes' formula. Let x be the observation (i.e. the input) which is a discrete random variable and z be a latent continuous random variable (i.e. the encoding), where $\mathbf{z}$ is

generated from some prior distribution $p_\theta(z)$ and $\mathbf{x}$ is generated from some conditional distribution $p_\theta(x|z)$, with both distributions parametrized over θ. Usually constructing an encoding requires computing the true posterior density $p_\theta(\mathbf{z}\,|\,\mathbf{x})$ which is intractable as

$$p_\theta(\mathbf{z}\,|\,\mathbf{x}) = \frac{p_\theta(z|x)p_\theta(z)}{p_\theta(x)} = \frac{p_\theta(z|x)p_\theta(z)}{\int p_\theta(x|z)p_\theta(z)dz}$$

where the integral of the marginal likelihood in the equation is intractable and $p_\theta(\mathbf{z}\,|\,\mathbf{x})$ is therefore also intractable.

The posterior density is therefore approximated by applying variational bayes and the recognition model delivers to the generative model the approximation to its posterior over latent random variables, which the neural network updates every iteration. Reversely, the generative model is a base for the recognition model as to learn meaningful representations of the data. Like in other variational methods the optimization objective is the Evidence lower bound (ELBO). For any choice of recognition model $q_\phi(z|x)$, including the choice of variational parameters ϕ, we have:

$$\begin{aligned} log(p_\theta(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}\,|\,\mathbf{x})}[log(p_\theta(\mathbf{x})] && (3.8)\\ &= \mathbb{E}_{q_\phi(\mathbf{z}\,|\,\mathbf{x})}[log(\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}\,|\,\mathbf{x})})] && (3.9)\\ &= \mathbb{E}_{q_\phi(\mathbf{z}\,|\,\mathbf{x})}[log(\frac{p_\theta(\mathbf{x},\mathbf{z}}{q_\phi(\mathbf{z}\,|\,\mathbf{x})}\frac{q_\phi(\mathbf{z}\,|\,\mathbf{x})}{p_\theta(\mathbf{z}\,|\,\mathbf{x})})] && (3.10)\\ &= \mathbb{E}_{q_\phi(\mathbf{z}\,|\,\mathbf{x})}[log(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})}] + \mathbb{E}_{q_\phi(\mathbf{z}\,|\,\mathbf{x})}[log(\frac{q_\phi(\mathbf{z}\,|\,\mathbf{x})}{p_\theta(\mathbf{z}\,|\,\mathbf{x})})] && (3.11) \end{aligned}$$

The second term in the equation is the Kullback-Leibler (KL) divergence between $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ and $p_\theta(\mathbf{z}\,|\,\mathbf{x})$, which is non-negative as $D_{KL}(q_\phi(\mathbf{z}\,|\,\mathbf{x})||p_\theta(\mathbf{z}\,|\,\mathbf{x})) \geq 0$. and zero if, and only if, $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ equals the true posterior distribution. The first term in the equation is the *variational lower bound*, also called the *evidence lower bound* or ELBO: $\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(\mathbf{z}\,|\,\mathbf{x})}[log(p_\theta(\mathbf{x},\mathbf{z}) - log(q_\phi(\mathbf{z}\,|\,\mathbf{x}))]$. Due to the non-negativity of the KL divergence, the ELBO is a lower bound on the log-likelihood of the data.

$$\begin{aligned} \mathcal{L}_{\theta,\phi}(x) &= log(p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}\,|\,\mathbf{x})||p_\theta(\mathbf{z}\,|\,\mathbf{x})) && (3.12)\\ &\leq log(p_\theta(\mathbf{x}) && (3.13) \end{aligned}$$

So the KL divergence $D_{KL}(q_\phi(\mathbf{z}\,|\,\mathbf{x})||p_\theta(\mathbf{z}\,|\,\mathbf{x}))$ determines two 'distances:

1. By definition, the KL divergence of the approximate posterior from the true posterior;
2. The gap between the ELBO $\mathcal{L}_{\theta,\phi}(x)$ and the marginal likelihood $log(p_\theta(\mathbf{x}))$; this is also called the *tightness* of the bound. The better $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ approximates the true (posterior) distribution $p_\theta(\mathbf{x}\,|\,\mathbf{z})$, in terms of the KL divergence, the smaller the gap

Similarly, It can be understood that ELBO $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ with respect to the parameters θ and ϕ, will concurrently optimize the two things we wish to achieve:

- It will approximately maximize the marginal likelihood $p_\theta(\mathbf{x})$, meaning that the generative model will become better.
- It will minimize the KL divergence of the approximation $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ from the true posterior $p_\theta(\mathbf{z}\,|\,\mathbf{x})$, so $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ becomes better.

For the neural network to be able to optimize over parameters θ and ϕ using stochastic gradient descent it is required to apply a reparameterization trick in order to create a differentiable inference model and be able to back-propogate through the network. This trick requires assuming the true posterior takes on an approximate of a specified random variable. In this case, we express the random

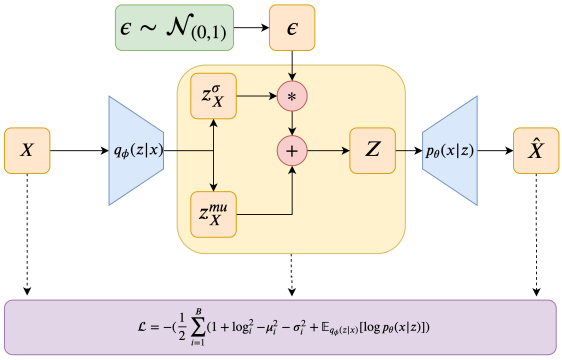


FIGURE 3.6: VAE objective is a combination of KL divergence and reconstruction loss. The recognition model creates a meaningful mapping from input data x to a latent space z measuring distance between the prior and the posterior via KL divergence. Additionally, using the posterior and the generative model recreates the input image x where the quality of these images are measured by the reconstruction loss.

variable $z \sim q_\phi(\mathbf{z} \mid \mathbf{x})$ as some differentiable (and invertible) transformation of another random variable $\epsilon \sim \mathcal{N}(0, I)$. The approximate posterior then becomes $q_\phi(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(z|\mu, diag(\sigma^2) * I)$ where mean μ and variance σ^2 are encoded by the recognition model using data point x and variational parameters ϕ: $z = g(\epsilon, \phi, x) = \mu + \sigma \odot \epsilon$. We can then sample from $z \sim q_\phi(\mathbf{z} \mid \mathbf{x})$ given recognition model $q_\phi(\mathbf{z} \mid \mathbf{x})$, data points x and ϵ as shown in figure 3.6

As a result, the loss function for the VAE becomes a combination of the **1)** reconstruction loss i.e. negative expected log likelihood or binary cross-entropy loss if $p_\theta(\mathbf{z} \mid \mathbf{x})$ is assumed to consist of independent Bernoulli distributions such as data in the form of normalized RGB channels with values between 0 and 1. **2)** KL-divergence between the recognition model $q_\phi(\mathbf{z} \mid \mathbf{x})$ from the assumed prior $p\theta(z)$ which makes the the loss function as follows:

$$\mathcal{L} = -(\frac{1}{2}\sum_{i=1}^{B}(1 + \log_i^2 - \mu_i^2 - \sigma_i^2) + \mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})}[\log p_\theta(x|z)])$$

Chapter 4

Method

This thesis posits its methodology as a way to explain and validate decisions made by a CNN. The predictions made by the CNN are validated and explained utilizing the properties of a weakly-supervised proxy generative model, more specifically, a triplet-vae. There are three main factors that contribute to the validation and explanation of the CNN. First, a triplet-vae is trained in order to provide a semantically relevant and well separated latent space. Second, this latent space is then used to train an interpretable linear support vector machine and is used to validate decisions by the CNN by comparison. Third, when a CNN decision is misaligned with the decision boundaries in the latent space, we provide a warning and generate explanations through stating the most probable answer as well as provide a qualitative explanation to validate the most probable answer. Each of these factors respectively refer to the number stated in figure 4.1 as well as link to each section: 1) triplet-vae section 4.0.1, 2) CNN Decision Validation, section 4.0.2, 3) Generating (contrastive) Explanations, section 4.0.3.

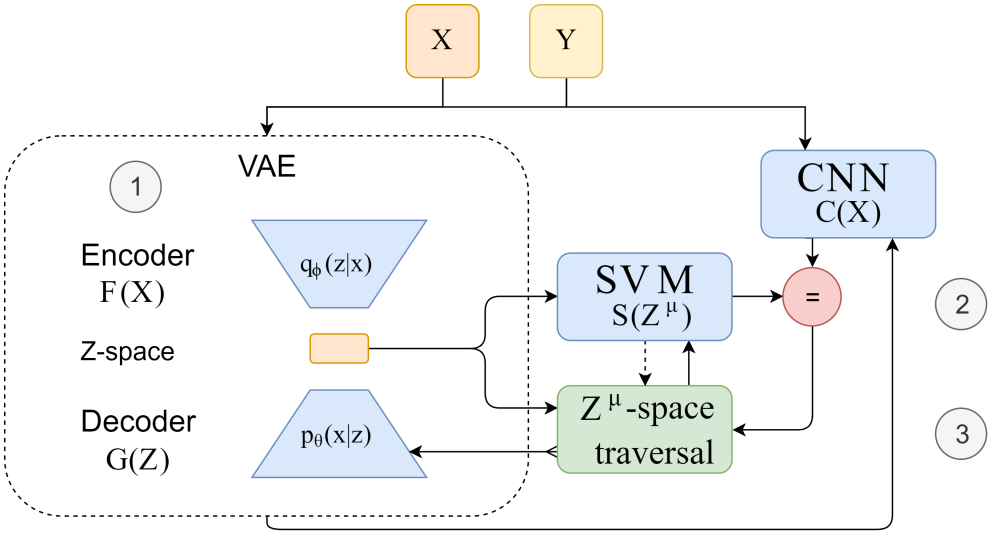


FIGURE 4.1: The diagnostics approach to validate and understand the behavior of the CNN. 1) Metric learning is applied during training of the VAE in order to create semantically relevant latent spaces. The generative model captures the essential semantics within the data and is used by 2) A linear Support Vector Machine. The linear SVM is trained on top of the latent space to classify input within a semantically relevant latent space rather than the direct mapping from input data X and labels Y. If the SVM and CNN do not agree on a prediction then 3) we traverse the latent space in order to generate and capture semantically relevant synthetic images, tested against the CNN, in order to check what elements have to change in order to change its prediction from a to b, where a and b are different classes.

4.0.1 Semantically relevant latent space

Deep metric learning is the notion of applying similarity or distance metrics in order to learn an embedding specifically applied to neural networks. Specifically, we wish to understand the effects of applying metric learning to a variational autoencoder. A triplet network consists of three instances of a neural network (with shared parameters). Moreover, using these three instances, give each instance a different type of input: An anchor, a positive- and negative sample. Afterwards we use the results from the instances in order to compute a (distance or similarity) metric between both the anchor and positive sample and the anchor and negative sample

Typically, a triplet network consists of three instances of a neural network that share parameters. These three instances are separately fed different types of input: anchor(s), positive sample(s) and negative sample(s). These are then used to learn useful representations by distance comparisons. Triplet networks have previously been explored by Theofanis Karaletsos et al. [46] but generally applied for different purposes than interpretability or explanations. We propose to incorporate metric learning by means of a triplet network to semantically structure and well-separate the latent space of the VAE using the available input and labels. A triplet-VAE consists of three instances of the encoder with shared parameters that are each fed pre computed triplets: an anchor, positive sample and negative sample; x_a, x_p and x_n. The anchor x_a and positive sample x_p are of the same class but not the same image, whereas negative sample x_n is from a different class. In each iteration of training, the input triplet is fed to the encoder network to get their mean latent embedding: $\mathcal{F}(x_a)^\mu = z_a^\mu$, $\mathcal{F}(x_p)^\mu = z_p^\mu$ and $\mathcal{F}(x_n)^\mu = z_n^\mu$. These are then used to compute a similarity loss function as to induce loss when a negative sample z_n^μ is closer to anchor z_a^μ than the positive sample z_p^μ distance-wise. i.e. $\delta_{ap}(z_a^\mu, z_p^\mu) = ||z_a^\mu - z_p^\mu||$ and $\delta_{an}(z_a^\mu, z_n^\mu) = ||z_a^\mu - z_n^\mu||$. Subsequently, provides us with three possible situations: $\delta_{ap} > \delta_{an}$, $\delta_{ap} < \delta_{an}$ or $\delta_{ap} = \delta_{an}$, See 4.2, 4.3 and 4.4, images inspired by [45].

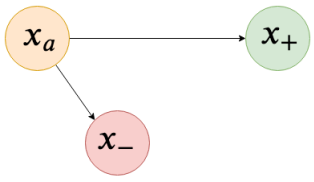


FIGURE 4.2: $\delta_{ap} > \delta_{an}$

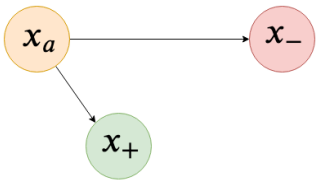


FIGURE 4.3: $\delta_{ap} < \delta_{an}$

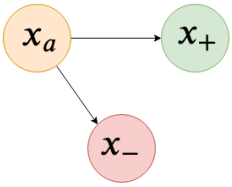


FIGURE 4.4: $\delta_{ap} = \delta_{an}$

We wish to find an embedding where samples of a certain class lie close to each other in the latent space of the VAE. For this reason, we wish to add loss the algorithm when we arrive in the situation where $\delta_{ap} > \delta_{an}$. In other words, we wish to push x_n further away, such that we ultimately arrive in the situation where $\delta_{ap} < \delta_{an}$ or $\delta_{ap} = \delta_{an}$ with some margin ϕ. As such we arrive at the triplet loss function that we'll use in addition to the KL divergence and reconstruction loss within the VAE:

$$L(z_a^\mu, z_p^\mu, z_n^\mu) = \alpha * \text{argmax}\{||z_a^\mu - z_p^\mu|| - ||z_a^\mu - z_n^\mu|| + \phi\,, 0\}$$

Where ϕ will provide leeway when $\delta_{ap} = \delta_{an}$ and push the negative sample away even when the distances are equal and α acts as a scalar.

We use the triplet-VAE to create a latent space that can be explored to validate and probe the behavior of the already present CNN. The already present CNN which we would like to validate is trained by input data $X : x_i ... x_n$ and labels $Y : y_i ... y_n$ where each y_i states the true class of x_i. We then use the same X and Y to train the triplet-VAE. 1). First, we compute triplets of the form $x_a, x_p x_n$ from the input data X and labels Y which are then used to train the triplet VAE. A typical VAE consists of an $\mathcal{F}(x) = Encoder(x) \sim q(z|x)$ which compresses the data into a latent space Z, a $\mathcal{G}(z) = Decoder(z) \sim p(x|z)$ which reconstructs the data given the latent space Z and a prior $p(z)$, in our case a gaussian $\mathcal{N}(0, 1)$, imposed on the model. In order for the VAE to train a latent space similar to its prior and be able to reconstruct images it is trained by minimizing the Evidence Lower Bound(ELBO). $ELBO = -\mathbb{E}_{z\sim\mathcal{Q}(z|X)}[\log P(x|z)] + \mathcal{KL}[\mathcal{Q}(z|X)||P(z)]$ This can

be explained as the reconstruction loss or expected negative log likelihood: $-\mathbb{E}_{z\sim\mathcal{Q}(z|X)}[\log P(x|z)]$ and the KL divergence loss $\mathcal{KL}[\mathcal{Q}(z|X)||P(z)]$, to which we add the triplet loss: $\mathcal{L}(z_a^\mu, z_p^\mu, z_n^\mu) = \alpha * \text{argmax}\{||z_a^\mu - z_p^\mu|| - ||z_a^\mu - z_n^\mu|| + \phi\ , 0\}$ This compound loss semi-forces the latent space of the VAE to be well seperated due to the triplet loss, disentangled due to the KL divergence loss combined with β scalar, and provides a means of (reasonably) reconstructing images by the reconstruction loss. And, thus results in the following loss function for training the VAE:

$$loss = -\mathbb{E}_{z\sim\mathcal{Q}(z|X)}[\log P(x|z)] + \beta * \mathcal{KL}[\mathcal{Q}(z|X)||P(z)] + \mathcal{L}(z_a^\mu, z_p^\mu, z_n^\mu)$$

.

As a result, a triplet VAE network can be constructed as follows, see figure 4.5. Where the input batch X with m samples contains $1/3$ anchor samples, $1/3$ positive samples and $1/3$ negative samples. We then use this semantically relevant latent space to generate explanations with respect to questions of the form why a and not b?

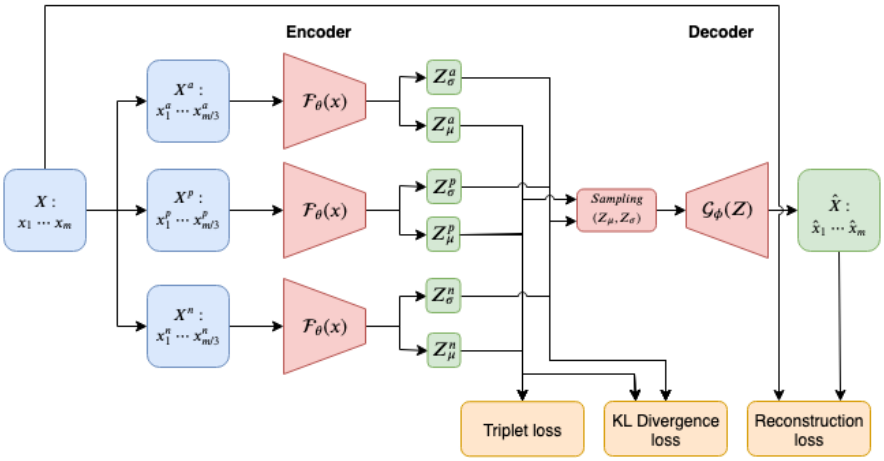


FIGURE 4.5: Example of a Triplet VAE network. Blue blocks represent input data, red blocks represent functions or parts of the VAE neural network, green blocks represent output and yellow blocks represent loss functions.

4.0.2 Decision Validation

The semantically relevant latent space we can use it for step two and three as indicated in figure 1.1. 2). Second step - CNN Decision Validation, we train an additional classifier on top of the triplet-VAE latent space, specifically z^μ. We train the linear Support Vector Machine using Z^μs as input data and Y as labels where $[Z^\mu, Z^\sigma] = \mathcal{F}(\mathcal{X})$. The goal of the linear support vector machine is two-fold. It provides a means of validating each prediction made by the CNN by using the encoder and the linear classifier. i.e. given an input example I, we have $\mathcal{C}(I) = \hat{y}_{\mathcal{C}(I)}$ and $\mathcal{S}(\mathcal{F}(I)^\mu) = \hat{y}_{\mathcal{S}(I)}$, and compare them against each other $\hat{y}_{\mathcal{C}(I)} = \hat{y}_{\mathcal{S}(I)}$. The linear support vector machine has a simple intuitive goal, it tries to fit a hyper-plane to the data as to separate classes. Whereas the convolutional neural network take on almost any shape or form depending on the complexity and degrees of freedom provided to the neural network, in a sense the linear SVM is a lot simpler and conservative than the CNN. The CNN is, because of its complexity and degrees of freedom, far more prone to over-fit on the data.

We use the triplet-vae's latent space as a common space to track classifications for both classifiers as a validation step for understanding the behavior of the models. All data points in the latent space can be tested/validated against the support vector machine as well as the convolutional neural network. The data point in the latent space can be directly fed to the SVM whereas for the CNN we

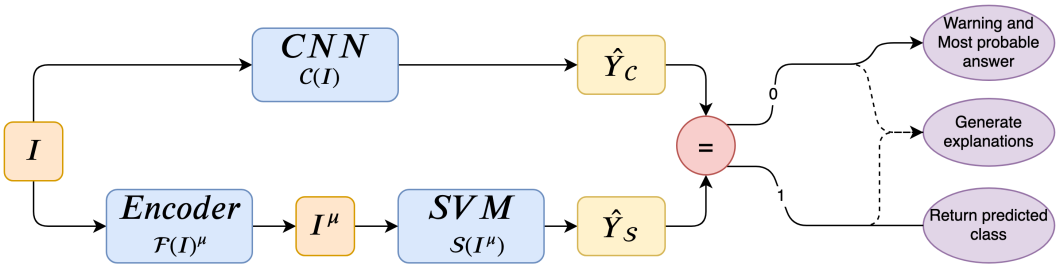


FIGURE 4.6: Given an input image I we check the prediction of the CNN as well as the SVM. If both classifiers predict the same class, we return the predicted class. In contrast, if the classifiers do not predict the same class, we propose to return the top most probable answer(s) as well as an explanation why those classes are the most probable.

generate an image for that specific location/data point and then feed the reconstructed image to the CNN. We can therefore compare both classifiers; the simple linear SVM model and complex CNN model. Metaphorically speaking we compare the knowledge of a student against that of the professor. The student knows the subject but not its intricacies whereas the professor is an expert in the field. Here, the student resembles the SVM whereas the professor is the CNN. If a certain topic is discussed between both parties, two cases can occur; both parties agree, or disagree. Either way, in both cases, a short explanation about why it is (in)correct can be discussed by giving providing a closely related comparison and stating the differences. In this thesis, we try to simulate such behavior by having the SVM and CNN classify the common space into the direction of closely related other classes. First, it is important to know what situation/case occurs when validating a data point, then appropriately we can state and sketch the situation to an actual expert in the field overseeing the classifications.

$$\text{Comparison(I)} = \begin{cases} \text{Positive} & \text{if } (\hat{y}_{\mathcal{C}(I)} = \hat{y}_{\mathcal{S}(I)}) \\ \text{Negative} & \text{if } (\hat{y}_{\mathcal{C}(I)} \neq \hat{y}_{\mathcal{S}(I)}) \end{cases} \tag{4.1}$$

First, If both classifiers agree then we arrive at an optimal state, meaning that the prediction is based on semantics and the direct mapping found by the CNN. In this way, we can say with high confidence that the prediction is correct. In this case, the answer is unambiguous according to the CNN and SVM and no warning is necessary. In the second case, if the classifiers do not agree, the answer is ambiguous towards the CNN and SVM and a warning is appropriate. Furthermore, three cases can occur: the SVM is correct and the CNN is incorrect, the SVM is incorrect and the CNN is correct, or both the SVM and the CNN is incorrect. In each of these cases the framework suggest a most probable answer or k most probable answers. In both cases if there is doubt about a prediction we can select and generate explanations to answer questions that one might have: why is the anchor prediction as class a and not b? This algorithm is indicated as step three of the framework and explained in figure 4.6.

4.0.3 Generating (contrastive) explanations

An explanation consists of 1) the most probable answer(s) and 2) a qualitative investigation of latent traversal towards the most probable answer(s). The most probable answer is presented by the averaged sum rule [59] over the predicted probabilities per class for both the CNN and SVM and selecting the top answer(s). By default, an SVM does not return a probabilistic answer, however, applying Platts [60] method we apply an additional sigmoid function to map the SVM outputs into probabilities. The top answer is then used in order to present and generate selected contrastive explanations.

The top prediction or class will be used in order to traverse and sub-sample the latent space from the initial representation or Z_I^μ location towards another class. We can find a path by finding the closest point within the latent space such that the decision boundary is crossed and the SVM predicts the target class. Alternatively we could use the closest data point in the latent space that adheres to the training set; argmin $\mathcal{F}(x_i)^\mu - Z_I^\mu$ for every $x_i \in X$. We can then traverse and sub-sample the latent space and change the semantics minimally as to change the class prediction. We capture the minimal change needed in order to change both the SVM and CNN prediction to the target class. This information is then presented to the domain expert for verification and answers the following question: The most probable answer is a because the input image I is semantically close to a with respect to the generative factors underlying the data, where the features are presented qualitatively. The explanations are generated as follows: see figure 4.7

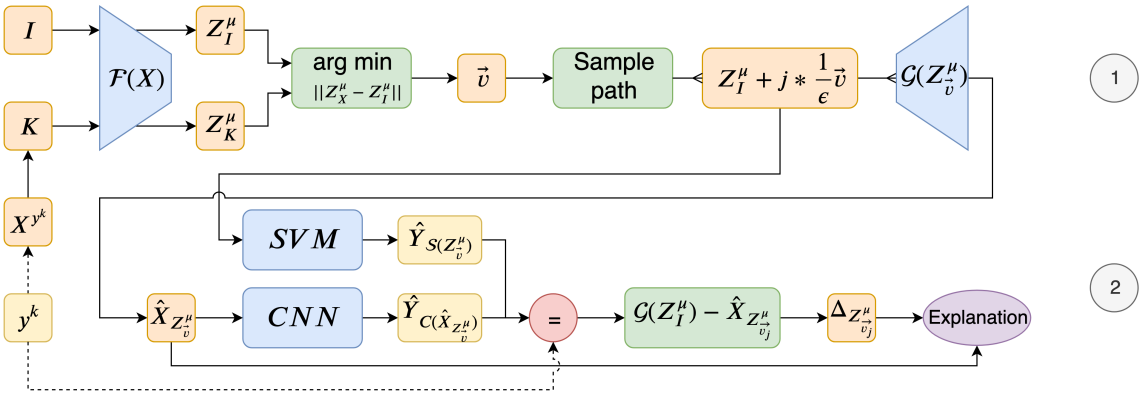


FIGURE 4.7: Generating (contrastive) explanations consist of several steps: First, given an input image I in question and the top most probable answer. K denotes training data X for class k labeled with $y = k$. We feed both I and K through the encoder $\mathcal{F}(X)$ to receive their respective semantic location in the latent space. We then find the closest training point that belongs to the target class k and find the vector $\vec{v}$; the direction of that point. Afterwards, uniformly sample ϵ data points along this vector $\vec{v}$, where j iterates over $0 \cdots j \cdots \epsilon$ and is denoted as $Z_{\vec{v}}^\mu$. $Z_{\vec{v}}^\mu$ is then used to check these against the SVM and use them to generate images $X_{Z_{\vec{v}}^\mu}$ using the decoder $\mathcal{G}(Z_{\vec{v}}^\mu)$. The generated images are then fed to the CNN to make a prediction and as the images will semantically change along the vector the prediction will change as well. Afterwards, we can compare the predictions from both the CNN and SVM. Subsequently, we use the first moment where both predictions are equal to target class k, denoted as moment l for generating an explanation - minimal semantic difference necessary to be equal to the target class, ΔU_l.

The decision boundaries around the clusters within the latent space are created by the SVM and can be used to answer questions of the form 'why a and not b?'. If $\hat{y}_{\mathcal{C}(I)}$ and $\hat{y}_{\mathcal{S}(I)}$ do not predict the same class, then, we assume that $\hat{y}_{\mathcal{S}(I)}$ is correct. We then use the find a path, indicated by $\vec{v}$ from $\hat{y}_{\mathcal{S}(I)}$ to $\hat{y}_{\mathcal{C}(I)}$, Z_I^μ to the target class. This can be done by calculating a vector orthogonal to the hyper-plane fitted by the SVM towards the target class. Alternatively, we can find the closest $z^\mu \in Z^\mu$ that satisfies $\hat{y}_{S(z^\mu)} = \hat{y}_{\mathcal{C}(z^\mu)}$ that are not the same as the initial prediction $\hat{y}_{\mathcal{C}(I)}$. This means that $\vec{v}$ is the vector from I to the closest data point of the target class, with respect to Euclidean distance.

We then uniformly sample points along vector $\vec{v}$ and check them against the SVM as well as the CNN. The sampled points can directly be fed to the SVM to get a prediction $\hat{y}(\int(v_i)$ for every $v_i \in V$. Similarly, we can get predictions of the CNN by transforming the images using the decoder $\mathcal{D}$. The images are then fed to the CNN to get a prediction $\hat{y}(\mathcal{C}(\mathcal{D}(v_i))$ for every $v_i \in V$. The predictions of both classifiers will change as the images start looking more and more like the target class as generative factors change along the vector. If we capture the changes that make the change happen, we can show the minimal difference required in order to change the prediction of the CNN. In this

way we can generate contrastive examples: For the top 'close' class that is not $\hat{y}_I$ we answer the question: 'why $\hat{y}_I$ and not the other semantically close class?'. Hence, we find the answer to the question "why *a* and not *b*?", as the answer is the shortest approximate changes between the two classes that make the CNN change its prediction. As a result, we have found a way to validate the inner workings of the CNN. If there are doubts about a prediction it can be investigated and checked.

Algorithm 1 Generate Explanations towards class k

Require: Anchor I, target class k, training data X, training labels Y, steps ϵ, encoder $\mathcal{F}$, decoder $\mathcal{G}$

function GEN_EXPL($I, k, X, Y, \mathcal{F}, \mathcal{G}$)

Filter training data points such that we only have training data with labels that is equal to target class k

$y^k_{indices} \leftarrow (Y = k)$
$K \leftarrow X[y^k_{index}]$
$Z^{\mu}_I \leftarrow \mathcal{F}(I)$
$Z^{\mu}_K \leftarrow \mathcal{F}(K)$

Calculate closest point of target class k

for $i \leftarrow 0$ to n **do**
  $\delta \leftarrow$ Euclidean_Distance($Z^{\mu_K}_i$, Z^{μ}_I)
  if $\delta <$ min_δ **then**
    min_$\delta \leftarrow \delta$
    min_$\delta_{index} \leftarrow i$
$\vec{v} \leftarrow Z^{\mu_K}_{\text{min\_}\delta_{index}} - Z^{\mu}_I$

Sample path; vector $\vec{v}$

for $j \leftarrow 0$ to ϵ **do**
  $Z^{\mu}_{\vec{v}}$.append($Z^{\mu}_I + j * \frac{1}{\epsilon} * \vec{v}$)

feed $Z_{\vec{v}}$ to SVM and generate images that can be fed to CNN

$\hat{y}_S \leftarrow \mathcal{S}(Z^{\mu}_{\vec{v}})$
$\hat{X}_{Z_{\vec{v}}} \leftarrow \mathcal{G}(Z^{\mu}_{\vec{v}})$
$\hat{y}_C \leftarrow \mathcal{C}(\hat{X}_{Z^{\mu}_{\vec{v}}})$

Compare classifications and find minimal important difference

for $j \leftarrow 0$ to ϵ **do**
  if $\hat{y}_S == \hat{y}_C == K$ **then**
    $\Delta_{\hat{X}_{Z_{\vec{v}}}} \leftarrow \mathcal{G}(Z^{\mu}_I) - \hat{X}_{Z^{\mu}_{\vec{v}}}[j]$

An explanation then consists of the generated images from $\vec{v}$ as well as the minimal important difference $\Delta_{\hat{X}_{Z_{\vec{v}}}}$ that were found when traversing the latent space towards k.

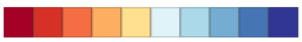
Chapter 5

Experiment Design

5.1 Datasets

5.1.1 Synthetic data set

The synthetic data set consists of three mathematical symbols: minus, plus and multiplication signs. The data set is generated with four additional latent generative factors, namely: color, rotation and the length and width of lines. Furthermore, the data set contains an 60000 (3*10*10*20*10) training images and 60000 testing images, where each image is a 64x64 RGB image, represented by three color channels containing values in the range of $[0, 255]$ which represent one channel for each color: red, green, blue. The images are divided equally into three different classes: 0) minus sign, 1) plus sign, 2) multiplication sign with linearly spaced latent factors to test how well the model(s) work with unseen data the test data has a different color palette. Examples from the data set can be seen in figure 5.1, the latent factors that are present in the data are:

- three classes: 0) - (minus sign), 1) + (plus sign), 2) x (multiplication sign)
- train data color: 10 diverging colors; 
- test data color: 10 diverging colors; 
- thickness, 10 values linearly spaced over $[4, 13]$ pixels.
- skewness: 20 values linearly spaced over $(-\frac{\pi}{8}, ..., \frac{\pi}{8})$
- length: 10 values linearly spaced over $[20, 60]$ pixels

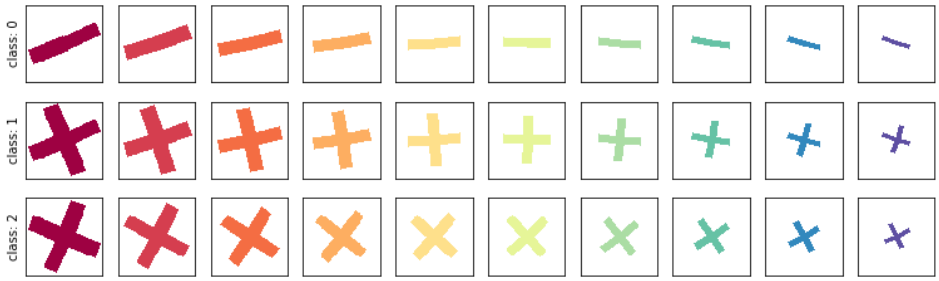


FIGURE 5.1: 30 images taken from the synthetic data, row one represents class 0 (minus sign), row two represents class 1 (plus sign) and row three represents class 2 (multiplication sign. The latent factors are linearly altered from left to right.

5.1.2 MNIST

MNIST is an open dataset comprised of handwritten digits written by high school students and employees of the United States Census Bureau.[61] It contains 60,000 training images and 10,000 testing images, where each image is a 28Œ28 grayscale image, represented by 1 color channel containing values in the range of $[0, 255]$, where the value represents shades from black to white. The images are divided equally into 10 different classes; digits: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$. Before feeding the data to the network the data values are normalized to be within the range of $[0, 1]$. See Figure 5.2 Samples from MNIST dataset for examples images and Table **??** Fashion MNIST for the distribution of classes.

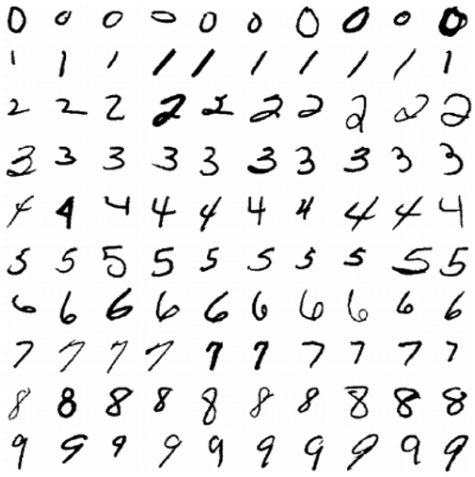


FIGURE 5.2: Samples from MNIST dataset

5.1.3 Fashion MNIST

Fashion-MNIST is an open dataset comprised of images of fashion items sold by the e-commerce website Zalando [62]. Similarly to MNIST, it contains 60,000 training images and 10,000 testing images, where each image is a 28x28 grayscale image, represented by 1 color channel containing values in the range of $[0, 255]$ Which represent the shades of black and white. The images are divided equally into 10 different classes: 0) T-shirt/Top, 1) Trouser, 2) Pullover, 3) Dress, 4) Coat, 5) Sandal, 6) Shirt, 7) Sneaker, 8) Bag, and 9) Ankle boots. Before feeding the data to the network the data values are normalized to be within the range of $[0, 1]$. Fashion MNIST is uniformly distributed for all classes in training images and test images; each class has 6000 images in training images, and 1000 in test images. See Figure 5.3 Samples from Fashion MNIST dataset for example images.

FIGURE 5.3: Samples from Fashion MNIST dataset

5.1.4 Dsprites

Dsprites dataset is a dataset of 2D shapes procedural generated from 6 ground truth independent latent factors, these factors are: color, shape, scale, rotation, x and y positions of a sprite. All possible combinations of these latent factors are present exactly once, generating N = 737280 images. The synthetic dataset Dsprites is chosen in order to be able to test the framework and be able to vary different class settings and is shown in 5.4. As such, We'd like to uncover semantics of the latent factors within the VAE or find semantically relevant directions within the latent space of the VAE.

Class	Training Samples	Test samples
0	5923	980
1	6742	1135
2	5958	1032
3	6131	1010
4	5842	982
5	5421	892
6	5918	958
7	6265	1028
8	5851	974
9	5949	1009

TABLE 5.1: Distribution of MNIST datset

The latent factor values that are present in the data are:

- Color: white
- Shape: square, ellipse, heart
- Scale: 6 values linearly spaced in [0.5, 1]
- training data Orientation: 1 values at an angle of 0
- test data Orientation: 1 values at an angle of $\frac{2\pi}{40}$
- Position X: 32 values in [0,1]
- Position Y: 32 values in [0,1]

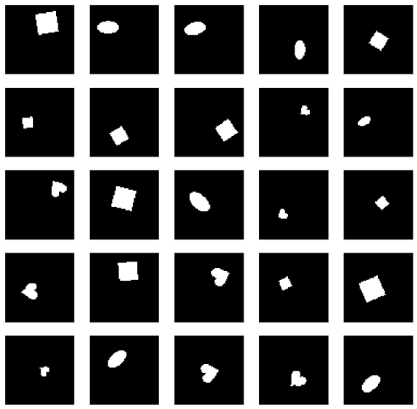


FIGURE 5.4: Randomly selected images from the dsprites data set

5.2 Model Setup

In this thesis we present the added benefit of creating a semantically relevant latent space in combination with a linear classifier on top of the latent space. The goal is to generate explanations to provide extra insights on the decision process of the CNN and SVM. The models used in this thesis are set up using the keras library with the Tensorflow backend for Python 3.5. The training was done locally, on google colab and on a high performance computer equipped with nVidia GPUs. Weights of the models are updated using the adam optimizer with the same parameters as described in the works of Kingma and Ba [63]. The CNNs are trained for several epochs until the validation loss stabilizes. The model and parameters from the best performing epoch (i.e. has the lowest validation loss) are selected as the final model and evaluated. The VAEs are trained for 1000 epochs to best structurize the latent space according to the loss functions applied. The 1000 epochs ensure a stable balance between the loss functions of the triplet-vae.

We investigate the behavior of the latent space by applying metric learning to the vanilla VAE; triplet-vae. The semantically relevant latent space of the $triplet - VAE$ is then used to generate images to investigate the CNN. We therefore investigate two types of VAE's, namely: Vanilla VAE and $triplet - VAE$. Due to the differences in the number of channels and the size of the images between the datasets MNIST, Fashion-Mnist and Dsprites and the synthetic data. The setup for the synthetic data is slightly altered to adapt to three channels instead of one channel. This is done by altering the encoder and decoder with two convolutional neural network layers with padding, ReLU activation and increasing number of filters per layer. For every type of VAE we specify 2-dimensional and 32-dimensional latent variables μ, σ and z in order to get a feel for the latent space. All model architectures can be found in Appendix A

The trained VAE models were evaluated by having the models reconstruct input images that the model has not seen before, validating for a semantically relevant latent space that is well-separated. The accuracy of the linear support vector machine as well as the normalized mutual information measure will tell us how well the latent space is separated with respect the defined classes. On the other hand the CNN and the SVM are evaluated by having them predict the classification of images that the models have never seen before during training, i.e. the test set. The predictive quality of these models are compared by accuracies.

Furthermore, comparing the classification clusters with each other is done by computing the normalized mutual information between both classifiers. That is, it quantifies the "amount of information"obtained about one random variable through observing the other random variable and can be thought of as the reduction in uncertainty about one random variable given knowledge of another. High mutual information indicates a large reduction in uncertainty; low mutual information indicates a small reduction; and zero mutual information between two random variables means the variables are independent. The calculations of these metrics were done using keras and the scikit-learn libraries.

Additionally, in order to find the Minimal Important Difference (MID) for generating explanations we use the anchor-based approach. We have an initial anchor with a specific class prediction and traverse and generate images from the latent space in order to find the applied minimal important difference that if applied to the anchor changes the predictions made by the CNN and the SVM.

5.2.1 Experiment

The experiment is set up as follows: We wish to create a semantically relevant latent space that is well seperated. First, we investigate and experiment with various parameter settings of the vanilla VAE and triplet-vae and select the parameter settings where both the accuracies of the linear SVM on top of the VAE and NMI is the high and where the reconstruction loss is relatively low. Further investigation of the space is then done by checking the accuracies of the support vector machine and by visual inspection of the latent space by encoding training data and visualizing it via tSNE or if the latent space is 2-dimensional directly by direct visualization of the training data in the latent space. The next step of the experiment starts when we have found the desired effect on the latent space which involves finding and selecting meaningful samples of the latent space to generate (contrastive) explanations/images. These generated samples are then fed to the CNN in order to test the behavior of the CNN.

The selection of latent space samples is simplified by the semantically relevant latent space; the VAE structures the latent space based on similarity of input images by the reconstruction loss and separates classes by applying metric learning. One assumption that we deal with is that because of both of these loss functions the triplet-vae will separate classes but still align the data point of said classes so that when we appoint an anchor a data point with class a, the closest data point of class b is closely related to the anchor with respect to the generative factors not specific to class a. Selecting the closest data point of another class than the anchor in the latent space therefore presents a sample with Minimal Important Difference which is similar to how humans select an explanation. Furthermore, we can then contrast this sample against the anchor as to show which factors affect the class difference. The samples are then presented to the CNN as to see the behavior of the CNN and if it classifies images correctly, this is especially important when we have an anchor that is ambiguous towards class a and b.

The experiment is set up to answer questions about the predictions made by the CNN as to be able to investigate specific anchors and select, or specify a class to which you think the anchor belongs to. The generated explanations then provide the minimal important difference that has to change in order to change the CNNs prediction.

Chapter 6

Results and Discussion

This thesis presents the results on the synthetic data set, MNIST, fashion-MNIST and Dsprites per data set and VAE type (Vanilla-VAE (**V-VAE**) and Triplet-VAE(**T-VAE**)). The results per data set consist of three parts, semantically relevant latent spaces, classification and generating explanations. The latent space is verified by visualization, the linear support vector machine and the loss metrics applied to the VAE. Classification states the accuracies and the effect of comparing classifiers and lastly explanations are shown and explained.

6.1 Synthetic Data set

6.1.1 Latent space

First, we present the advantage of applying metric learning to the VAE, as shown in figures 6.1, 6.2. There is a distinct difference between the Vanilla-VAE (left) and Triplet-VAE (right), metric learning creates a semantically relevant latent space where each class is well-separated. In the Vanilla-VAE's latent space the classes are entangled and entwined, whereas in the Triplet-VAE latent space there is a clear separation between the classes. The red and blue lines indicate the decision boundaries of the CNN and the SVM respectively. In the Vanilla-VAE the SVM is not able to find clear linear boundaries to best separate data but in the Triplet-VAE it is able to find clear linear boundaries. This effect can also be seen in 6.2, in two-dimensions the SVM's accuracy for the VAE's respectively are 0,7374 and 0,9330 for training data and 0,6853 and 0,8522 for test data.

The Vanilla-VAE mainly focuses on minimizing the reconstruction and Kullback-Leibler divergence loss, whereas the Triplet-VAE has the extra metric learning triplet loss constraint and as a result has slightly higher reconstruction and KL loss than the Vanilla-VAE, as shown in table 6.1. None of the losses are perfect and can be explained as the model not being able to completely capture the generative factors and processes underlying the data. This results in the reconstructed images not being perfectly reconstructed and therefore slightly different from the real images.

	Latent dimension	**Recon loss**	**KL loss**	**Triplet Loss**	**Total loss**
Vanilla VAE	2	123.106	57	0	123.163,4195
	16	100.674	85	0	100.759,8938
	32	100.497	120	0	100.617,4213
Triplet VAE	2	140.014	29	144	140.186,3631
	16	118.472	90	111	118.672,9606
	32	118.082	125	106	118.312,4256

TABLE 6.1: Comparison of Vanilla VAE and triplet-VAE loss on the Synthetic data set

One of the reasons that the input images cannot be perfectly reconstructed is the fact that the assumption of a Gaussian prior might not fit the exact distributions underlying the data causing a disconnect between the reconstruction and KL-loss. This effect can be seen in table 6.2; i.e. insufficient latent complexity or dimensions results in a discrepancy between the accuracy of real images

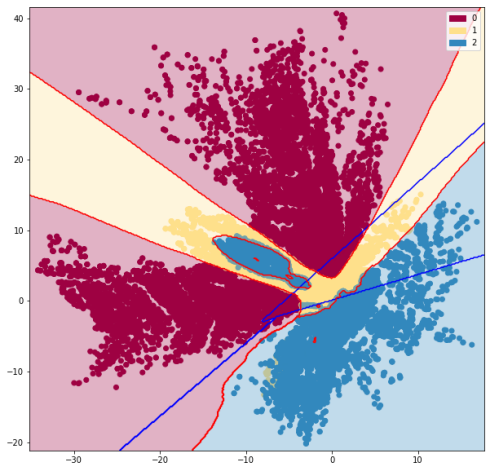


FIGURE 6.1: visualization of a 2-dimensional latent space of a vanilla VAE on Synthetic data, red and blue lines indicate the decision boundaries of the CNN and SVM respectively.

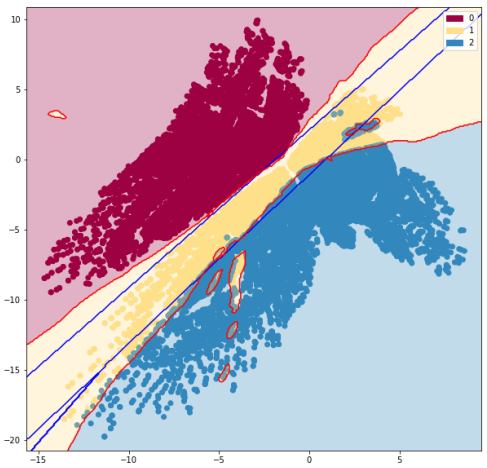


FIGURE 6.2: visualization of a 2-dimensional latent space of a $\mathcal{T}$-VAE on Synthetic, red and blue lines indicate the decision boundaries of the CNN and SVM respectively.

and reconstructed images of those real images. However, with advances in the generative models and flexible priors in the future this might be solved.

6.1.2 Classification

One of the parts of the framework compares the classifiers to see if they agree, as such three classifiers are trained in order to see how they perform. As a baseline we have a standard linear SVM trained on the data directly. Next, we have the linear SVM trained on top of the latent space per VAE type and dimensions. Lastly we have the CNN, which accuracies are shown on both the real data and reconstructed images, as shown in 6.2. The decision boundaries of the two-dimensional latent spaces can be visualized and shown in 6.1 and 6.2. It can be seen that the CNN is far more complex and takes on any shape or form to fit the data, which causes great accuracy but also prone to over-fitting. The No-VAE SVM's accuracy drops quite significantly when presented unknown data. This is similar for all the classifiers, however, the SVM trained on top of the VAE is the most robust of all the trained classifiers and only drops 0,0034 in accuracy whereas the CNN drops 0,0090. This shows that the SVM on top of the VAE generalizes better than the CNN.

Also, interestingly, the CNN performs better on reconstructed test data than on the actual test data. This effect can be explained by the encoder of the VAE linking/relating the test data as best it can to earlier seen training data. This means that the test image encoded in the latent space is the location where the difference in generative factors between the test image and location is minimized. This can be subdivided into the test data point being related to class specific features as well as generative factors that are similar. The reconstructed images are therefore more like the actual training data than the test data, improving the accuracy. For example, if the VAE and the CNN are both trained on red and blue images and the to be tested image is orange the VAE will predict red as reconstructed image. On the other hand, CNNs behavior will be unpredictable as it is prone to over-fitting. This makes the SVM more predictable and stable than the CNN.

	V-VAE		T-VAE		No VAE
	2d	32d	2d	32d	
CNN true train accuracy	-	-	-	-	***0,9975***
CNN rec train accuracy	0,9769	***0,9975***	0,9872	***0,9975***	-
SVM train accuracy	0,7374	0,9962	0,9330	***0,9966***	0.9962
CNN true test accuracy	-	-	-	-	***0,9885***
CNN rec test accuracy	0,8875	***0,9931***	0,8940	***0,9899***	-
SVM test accuracy	0,6853	***0,9946***	0,8522	***0,9932***	0.93955

TABLE 6.2: CNN and SVM performance on (reconstructed) training and (reconstructed) test data.

6.1.3 Generating explanations

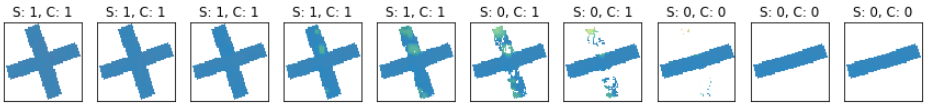
Explanations are generated in order to answer several questions: Uncover why class a was predicted instead of class b, next, what the decision rule difference is between the complex CNN model and the simpler SVM model. And, what changes have to be applied in order to change the prediction of both the CNN and the SVM to class b.

Case	Percentage
1) $\hat{Y_S} = \hat{Y_C} = Y$	0.9841
2) $\hat{Y_S} = \hat{Y_C} \neq Y$	0.000783
3) $(\hat{Y_S} = Y) \neq \hat{Y_C}$	0.0099
4) $\hat{Y_S} \neq (\hat{Y_C} = Y)$	0.00522
5) $\hat{Y_S} \neq \hat{Y_C} \neq Y$	0.0016

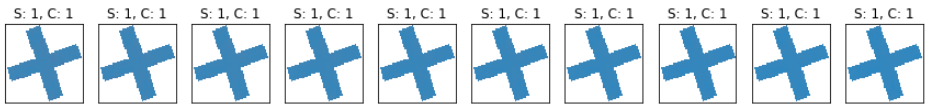
TABLE 6.3: This table shows the percentages of agreement with respect to all possible cases.

It is important to state that the loss function of the triplet-vae imposes two main important traits upon the latent space of the VAE. First, the reconstruction loss places similar images close to each other. Second, classes are separated with respect to distance because of the metric learning triplet loss. As such, we get areas for a specific class, and within the generative factors are contained within that area as to improve reconstruction loss. Similarly, having different areas with the same pattern of generative factors are generally aligned in such a way as to improve reconstruction loss, meaning that generative factors that are closely related are closer than other generative factors. Therefore, if we find a minimal distance path from class a to class b, then the end point within class b will have similar generative factors as the starting point except for the factors related to class assignment.

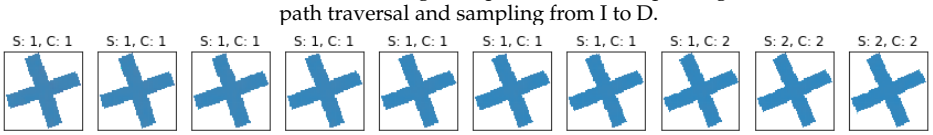
This minimal important difference is captured by traversing the latent space and capturing the change that happens when traversing the path from a to b. In this way we select closely related examples to which we can contrast against, providing us with a way to meaningfully find contrastive examples that make sense to explain a question of the form 'why a and not b?'



(A) Selection of closest training data point of class 0 - minus sign - with path traversal and sampling from I to D.



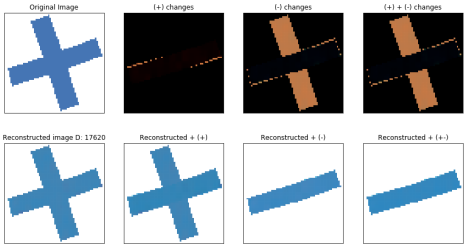
(B) Selection of closest training data point of class 1 - plus sign - with path traversal and sampling from I to D.



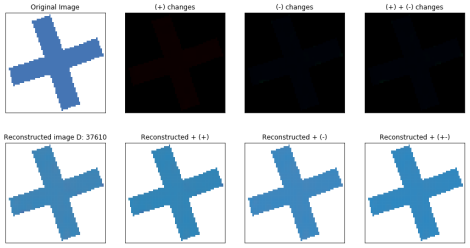
(C) Selection of closest training data point of class 2 - multiplication sign - with path traversal and sampling from I to D.

Figures A, B and C, above, show the selection of training data points that are closely related to I but of another class. Additionally, the images represent vector v from I to the selected training data point. The vector is then uniformly sampled and images are generated from these samples using the VAE's decoder. Additionally, each location and sampled image is tested against the SVM and CNN to evaluate the behavior of both as indicated by the S and C respectively on top of the images.

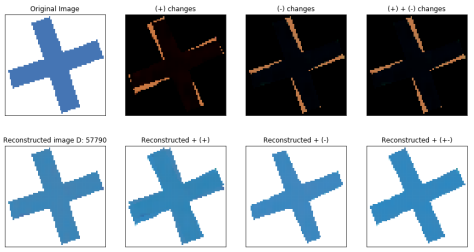
Furthermore, The minimal important difference to change the classes from a to b are then captured and shown in order to answer the question: 'why a and not b?". I.e. the predicted class is of class a because of the indicated difference that is (not) present in anchor I. e.g. as shown in 6.3a, 6.3b and 6.3c, respectively to class 0 (minus), 1 (plus), 2 (multiplication). The Minimal Important Difference is the extra vertical bar that is present in anchor I and states that I is not classified as a minus bar as long as the vertical bar is present. Next, anchor I is classified as a plus sign as there is no change necessary to make it a plus sign. Finally, anchor I is not classified as a multiplication sign as the minimal important difference to classify it as a multiplication sign requires it to be rotated slightly.



(A) Minimal important difference between anchor I class 0 - minus sign -



(B) Minimal important difference between anchor I class 1 - plus sign -



(C) Minimal important difference between anchor I class 2 - multiplication sign -

FIGURE 6.3: These figures show the selected / closest data points of another class which largely retains the generative factors that are not related to changing the class.

This provides us with the means to answer several questions that we may have about the predictions made by the CNN and SVM. 1) By capturing the minimal important difference experts can make a well-informed decision to classify I as either class a or as the proposed class b depending on the selected contrastive example shown. 2) The images show the behavior of the CNN with respect to images that are in between two classes. For instance, 6.3a, the CNN only classifies the image as a minus sign when the vertical bar is almost completely gone whereas the SVM classifies it as a minus if it is even slightly disrupted. In this way we can help experts understand the behavior of the CNN and counteract undesirable behavior by using the generated data as new training data to unlearn specific behavior. More results can be find in Appendix B

6.2 MNIST Data set

6.2.1 Latent space

First, we present the advantages of applying metric learning to the VAE, as shown in figures 6.4 and 6.5. There is a distinct difference between the latent space of the Vanilla-VAE(left) and Triplet-VAE(right), applying Triplet loss to the VAE creates a semantically relevant latent space where each class is well separated. If we compare the Vanilla-VAE and the Triplet-VAE, it can be seen that the classes in the Vanilla-VAE's latent space are entangled whereas the classes are clearly disentangled in the latent space of the Triplet-VAE. This results in the linear SVM not being able to find clear separating decision boundaries in the latent space of the Vanilla-VAE. The red and blue lines indicate the decision boundaries of the CNN and SVM respectively. This effect can also be seen in 6.5, with a two-dimensional latent space the SVM's accuracy for the VAE's respectively are 0,7577 and 0,93305 on training data and 0,7527 and 0,9088 on test data. As a result, in the center, classes nine (dark blue) and four (yellow) are sub-divided into separate patches with other classes in between. This results in unexpected behavior when traversing the latent space from one class to another. Additionally, this also affects the linear SVM severely as it cannot find a clear decision boundary to separate two classes. The Triplet-VAE however has a clear separation of classes and makes validation easy and understandable.

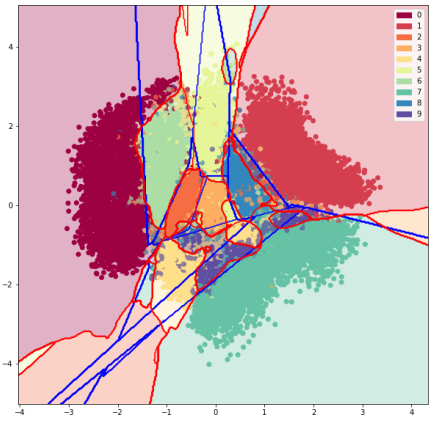


FIGURE 6.4: visualization of a two-dimensional latent space of a vanilla VAE on MNIST, red and blue lines indicate the decision boundaries of the CNN and SVM respectively.

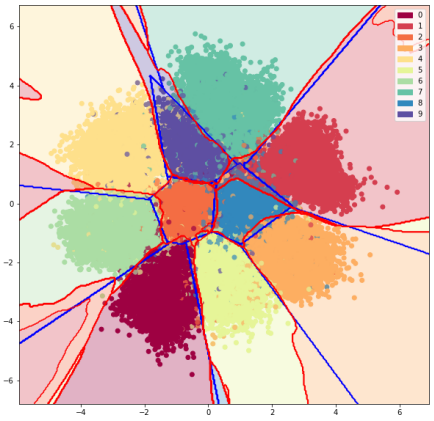


FIGURE 6.5: visualization of a two-dimensional latent space of a $\mathcal{T}$-VAE on MNIST, red and blue lines indicate the decision boundaries of the CNN and SVM respectively.

FIGURE 6.6: In both figures the red lines indicate the decision boundaries of the CNN whereas the blue lines indicate the decision boundaries of the SVM. It can clearly be seen that the Triplet-VAE has impact on the boundaries and keeps the space well-separated.

The Vanilla-VAE mainly focuses on minimizing the reconstruction and Kullback-Leibler divergence loss. A low reconstruction loss leads to the VAE being able to find a good representation or combination of latent factors that explains an input image. The input images are encoded into the latent space with a specific location. If we decode this latent space location we get a reconstructed image that closely resembles the input image, if trained correctly. The extra metric learning loss that is applied in the Triplet-VAE causes, in overall, a slight increase in reconstruction and KL loss, as

shown in 6.4. A high reconstruction loss means that the model is not complex enough to provide the reconstructive expressiveness it needs in order to perfectly reconstruct the inputted image or that the assumed prior does not exactly fit the underlying distribution of the data. The addition of the triplet loss term to the VAE causes a slight increase in reconstruction and KL loss, this can be explained by the triplet-loss term trying to push classes that are not equal away from each other even though the image might be similar. The loss per VAE and latent dimension can be found in 6.4

	Latent dimension	**Recon loss**	**KL loss**	**Triplet Loss**	**Total loss**
Vanilla VAE	2	128,2465	6,9846	n/a	135,2311
	16	70,0926	15,3466	n/a	85,4392
	32	76,2236	23,2645	n/a	99,4881
Triplet VAE	2	131,8879	9,7440	0,4566	142,0885
	16	73,8670	17,4683	0,3564	91,6917
	32	78,6606	25,2827	0,2570	104,2003

TABLE 6.4: Comparison of Vanilla VAE and triplet-VAE loss on the MNIST data set

Similar to the results with the synthetic data set, higher reconstruction loss leads to the VAE not being able to capture all the generative features and processes within the data in the assumed Gaussian distribution. As a result, the images that are reconstructed might not be 100 percent equal to the real images and will result in a discrepancy between real images and reconstructed real images. The reconstructed real images however generally still maintain the structure but are more generalized and less specific, meaning that two thin lines close to each other are blurred together to make one line. An increase in expressive power of the model results in a lower reconstruction loss. Thus, for future research it would be interesting to increase model complexity to better the solution,

6.2.2 Classification

The second part of the model poses to compare the classifications made by the linear SVM (on top of the VAE) and the CNN. First, if we compare the accuracies of the SVM with No-VAE, Vanilla-VAE and Triplet-VAE we can see that for the MNIST data set the accuracies of the linear SVM on top of the VAE are increased when comparing it against the Vanilla-VAE, especially in a low dimensional latent space. The SVM with VAE clearly performs better than without the VAE; 0,9798 against 0,943 for the training data and 0,9642 against 0,9446 for the test data. Whilst the original CNN is still better in terms of accuracy, this is not unexpected as the CNN is far more complex with respect to the shape of the decision boundaries than the SVM. This can be seen in 6.4 and 6.5. Unfortunately, for the CNN the reconstructed images do not fair as well as the CNN in terms of accuracy but this does affect the framework too much when we traverse the latent space from one class to another. All classification accuracies of the MNIST data set can be found in table 6.5.

	V-VAE		**T-VAE**		**No VAE**
	2d	**32d**	**2d**	**32d**	
CNN true train accuracy	-	-	-	-	***0.9943***
CNN rec train accuracy	0.8030	0.9550	0.9229	0.9581	-
SVM train accuracy	0.7577	0.9493	0.93305	***0.9798***	0.943
CNN true test accuracy	-	-	-	-	***0.987***
CNN rec test accuracy	0.7947	0.9548	0.9008	0.9516	-
SVM test accuracy	0.7527	0.9511	0.9088	***0.9642***	0.9446

TABLE 6.5: CNN and SVM performance on (reconstructed) training and (reconstructed) test data

The second part of the method consists of utilizing the fact that we have two classifiers that aim to achieve the same result from a different stand point. One focuses on uncovering the generative factors of the data whereas the other finds a direct mapping from the input data to the labels. If we compare the classifications made by both classifiers we can end up in a couple of different scenarios or cases, as shown in 6.6.
We take advantage of this fact by providing a warning when the classifiers do not agree as in this case it is most likely that the predictions made are ambiguous.

First, if we take case one and two, where the classifiers agree then we can see that for a large portion of the test data the classifiers agree, and they are correct, in 95,86 percent, in only 0,44 percent they are not correct. This means that we can give an indication that when they agree, we assume they are correct, if they differ than we investigate the images further as there is some kind of behavior that makes the classifier not agree. These cases are case three, four and five.

Case	Percentage
1) $\hat{Y_S} = \hat{Y_C} = Y$	0.9586
2) $\hat{Y_S} = \hat{Y_C} \neq Y$	0.0044
3) $(\hat{Y_S} = Y) \neq \hat{Y_C}$	0.0056
4) $\hat{Y_S} \neq (\hat{Y_C} = Y)$	0.0284
5) $\hat{Y_S} \neq \hat{Y_C} \neq Y$	0.003

TABLE 6.6: This table shows the percentages of agreement with respect to all possible cases.

6.2.3 Generating Explanations

It is especially important to investigate predictions when the classifiers do not agree as then the image might be a little strange. Is it then also for these cases that explanations are essential and important. One such example is test data point number 6783, this data point is of case 5 and can be seen in 6.7. In this case we take the decision boundaries of the SVM as ground-truth and move from anchor I the initial position of the test point in the latent space to another class. The most probable answers for this test point, by averaging the probabilities of both classifiers are: 6, 8 and then 1 with averaged probabilities 0,512332, 0.3382, 0.1150, for this example the true label is 1. The selection and explanation of the vector traversal from I to said classes can be seen in 6.7

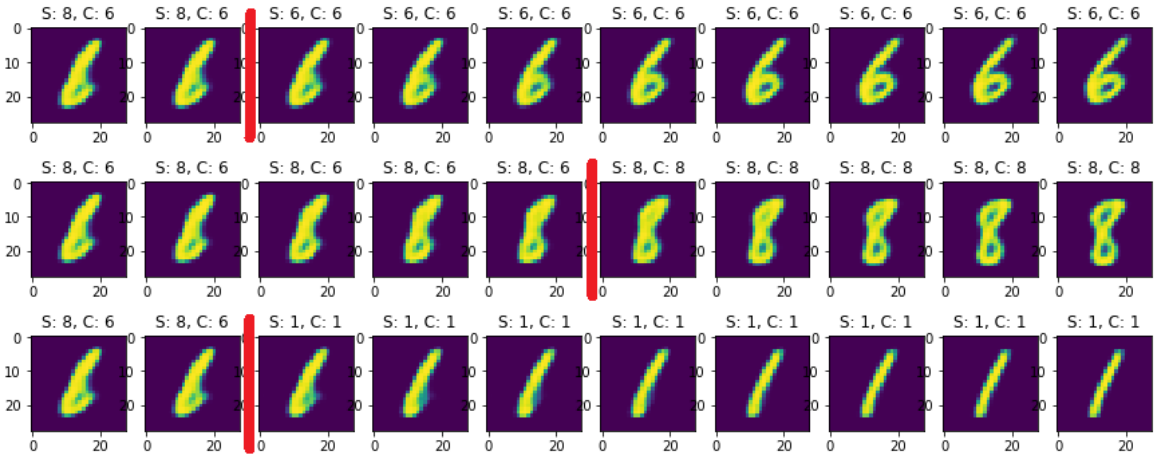


FIGURE 6.7: Per top k probable answers we traverse and sample the latent space to generate images that can be used to test the behaviour of the CNN. The red line indicates the moment where both the SVM and the CNN predict the target class

The image shows the images that are generated from uniformly sampling vector v and decoding them using the vector v. The selected end-points are training data points of the targeted classes where the change in generative factors are minimized, but the class specific factors are affect maximally. The figures show the change that happens when traversing the latent space in the direction of those

classes. The images change over the length of the vector and start to look more and more like the target class. We can then capture and see the minimal important difference that changes the classifier's predictions, indicated by the red line. This minimal important differences states the positive or negative change that has to happen in order to get the classifiers to agree that it is the target class. This minimal important difference is then visualized as an extra explanation as to answer questions of the form 'why a and not b?', as shown in figure 6.8

For the traversal from Z_I^μ to class 6 it can be seen that rather quickly both classifiers agree and only minimal changes are required to change the predictions. Third, for such an occurrence we can further zoom in on what is happening and what really makes that the most probable answer. Figure 6.8 shows these minimal changes required to change its prediction as well as the transformed image on which the classifiers agree. The first row shows the original image, positive changes, negative changes and the changes combined. The second row shows the reconstructed image and the reconstructed images with the positive changes, negative changes and positive and negative changes respectively. In this way, for each probable answer it shows its closest representative and the changes required to be part of that class.

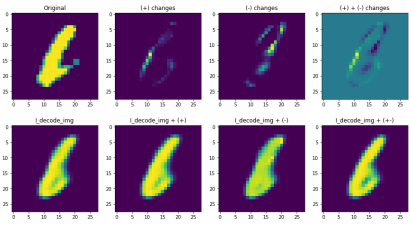


FIGURE 6.8: Once the SVM and the CNN both predict the target class we capture the minimal changes that are necessary to change their predictions

6.3 Fashion-Mnist Data

6.3.1 Latent space

Figures 6.9, 6.10 show the effects of applying metric learning to the vanilla VAE providing us with a semantically relevant latent space on the Fashion-MNIST dataset. It can clearly be seen that the metric learning loss separates the space based on classes whereas the latent space of the vanilla VAE is more entangled and is not well-separated with respect to the 10 classes present in the data set.

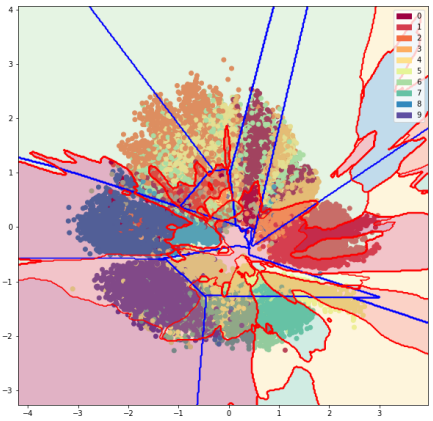


FIGURE 6.9: visualization of a two-dimensional latent space of a vanilla VAE on Fashion-MNIST, red and blue lines indicate the decision boundaries of the CNN and SVM respectively.

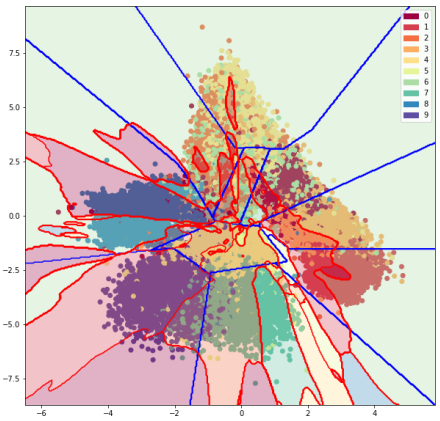


FIGURE 6.10: visualization of a two-dimensional latent space of a $\mathcal{T}$-VAE on Fashion-MNIST, red and blue lines indicate the decision boundaries of the CNN and SVM respectively

The (Triplet/Vanilla)-VAE however is not able to accurately reconstruct input images given a 2-dimensional latent space, this observation can be seen by the decision boundaries of the CNN. for example, on the right-hand side of figure 6.10 it can be seen that the reconstructed images of the classes zero - t-shirt/top, one - trouser, three -dress and a part of four - coat - are seen as the same class by the CNN. Similarly, when comparing the accuracy of the CNN on the reconstructed images, see 6.8, it can be seen that there is a discrepancy between the true and reconstructed images. This is an indication that the expressiveness of the model is just not powerful enough given a two-dimensional latent space. This discrepancy becomes smaller given higher-dimensional latent spaces.

Table 6.7 shows that providing a higher dimensional latent space lowers the reconstruction loss and triplet loss whereas the KL loss increases. The main advantage of having low KL loss is that there

	Latent dimension	Recon loss	KL loss	Triplet Loss	Total loss
Vanilla VAE	2	245,1566	7,0797	n/a	252,2363
	16	221,4129	15,3742	n/a	236,7871
	32	221,2025	15,3227	n/a	236,5252
Triplet VAE	2	251,1768	10,0958	1,7267	262,9993
	16	224,7624	16,2814	0,3075	241,3513
	32	223,9622	16,9686	0,2884	241,2192

TABLE 6.7: Comparison of Vanilla VAE and triplet-VAE loss on the Fashion-MNIST data set

	V-VAE		T-VAE		No VAE
	2d	32d	2d	32d	
CNN true train accuracy	-	-	-	-	**0,9126**
CNN rec train accuracy	0,6256	0,7558	0,6255	0,7532	
SVM train accuracy	0,6922	0,8243	0,6922	**0,8885**	0,8623
CNN true test accuracy	-	-	-	-	**0,8808**
CNN rec test accuracy	0,6242	0,7440	0,6236	0,7352	
SVM test accuracy	0,6855	0,8123	0,6855	**0,8526**	0,8461

TABLE 6.8: CNN and SVM performance on (reconstructed) training and (reconstructed) test data

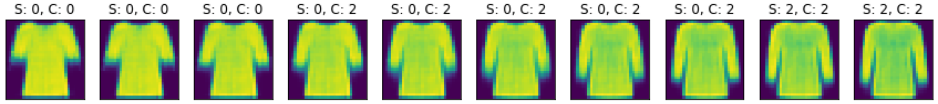
is a steady and stable minimum and maximum that can be sampled from within the latent space - a Gaussian distribution. High KL loss however does not influence or obstruct the goals of a semantically relevant latent space and being able to meaningfully generate explanations. However, the model not being able to perfectly reconstruct images can lead to undesirable results. The generated explanations may be not specific enough to closely resemble the actual data, in this way the explanation might be hard to interpret. In my opinion, the Fashion-MNIST data given the resolution of the images and the lack of color makes it hard to differentiate between several classes, for instance, a pullover, shirt and coat. It is therefore no surprise that the classifiers are not able to perfectly classify these images.

6.3.2 Classification

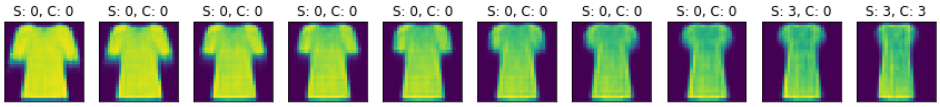
Table 6.8 shows the scores the accuracies scores per model and latent dimension complexity. The Fashion-MNIST dataset has similar results to that of the MNIST data set. The CNN is still the best option when classifying images with an accuracy of 0.9126 on training data and 0.8808 on test data. The classical linear classifier is slightly worse with an accuracy of 0.8623 on training data and 0.8461 on test data. It can more easily deal with images it has not seen before. The accuracies of the SVM on top of the latent space is best with the 32 dimensional latent space triplet-vae with an accuracy of 0.8885 on training data and 0.8526 on test data. As a result of lower reconstruction loss the bigger the discrepancy between the CNN and the SVM accuracies and thus also a bigger gap between the various cases that can occur as shown in table 6.9
It can be seen that the accuracy actually drops, in case one in 80 percent of the cases the SVM and the CNN agree whilst they both have 85 percent plus when they are not combined. Whilst both the SVM and CNN have relatively good accuracies, when combining and comparing them both. This is not necessarily bad, but it just means that when using this framework in more cases a warning would occur that would indicate that the input image might be a little strange. In this way, you might say that more false positives / false negatives might occur, where false positives and negatives mean getting a warning when the image is not strange or the other way around. Therefore, higher reconstruction loss within VAE's relate to higher false positives / false negatives with respect to giving a warning. However, if we consider case two, if the SVM and the CNN's predictions are equal then only two percent of the time we will not get a warning, which is pretty significant. If we'd only have a CNN predicting Fashion-MNIST then in 88 percent of the time it would be correct but you can never be sure when it would not be (in)correct. With this framework at least you can capture and find a larger percentage of predictions where the predictions are just not 100% certain. Additionally, when this latter happens

Case	Percentage
1) $\hat{Y_S} = \hat{Y_C} = Y$	0.8048
2) $\hat{Y_S} = \hat{Y_C} \neq Y$	0.021
3) $(\hat{Y_S} = Y) \neq \hat{Y_C}$	0.0688
4) $\hat{Y_S} \neq (\hat{Y_C} = Y)$	0.097
5) $\hat{Y_S} \neq \hat{Y_C} \neq Y$	0.0504

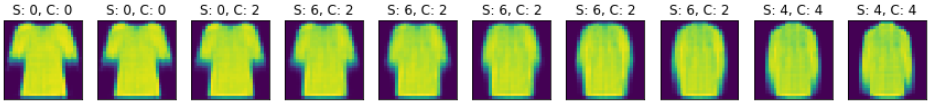
TABLE 6.9: This table shows the percentages of agreement with respect to all possible cases.



(A) Selection of closest training data point of class 2 - pullover - with path traversal and sampling from I to k.



(B) Selection of closest training data point of class 3 - Dress - with path traversal and sampling from I to k.



(C) Selection of closest training data point of class 4 - Coat - with path traversal and sampling from I to k.

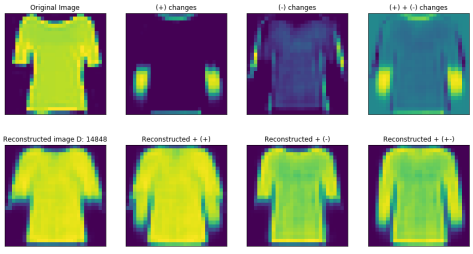
FIGURE 6.11: The visualization of generated images from uniformly sampling vector v and decoding them using the decoder.

- SVM and CNN do not agree - then we can provide extra information and insights towards what might be the appropriate class according to the CNN and the SVM.

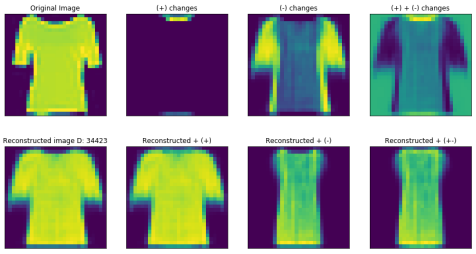
6.3.3 Generating explanations

The importance of generating explanations is to understand the behavior of the predictions made by the SVM and the CNN, especially in cases two, three, four and five. These cases occur because there is a discrepancy between what the SVM and CNN decision rules they use to predict the classes. For the CNN it might be the case that one specific pixel having a high value means class a whereas the SVM only predicts it as class a when there are multiple pixels with a higher value in a region. The explanations are generated to understand this behavior to have an appropriate expectation of the trained classifiers. One such example can be found when generating explanations for test sample 9668, which has a label of 0; it is a t-shirt/top.

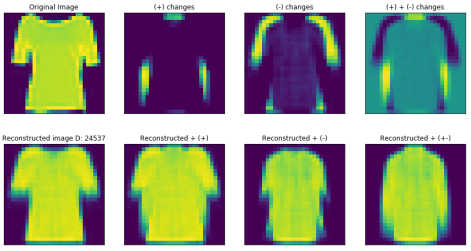
First, three training points are selected that are semantically close with respect to generative factors, as shown in 6.11a, 6.11b, 6.11c. These are selected as to answer the questions why was 0 - a t-shirt/top - predicted above a pullover, dress or coat. Secondly, The generated explanation shows that especially for the class pullover the classifiers behave differently. The CNN quickly predicts the item as a pullover when even the smallest addition of sleeve is applied, whereas the Minimal important different for the SVM requires a lot more of the sleeves to be applied in order to change its prediction, as indicated by images 6.11a and 6.12a. additionally, can also be seen that in 6.12b I is not classified as a dress because of the addition of sleeves. Lastly, I is not classified as



(A) Minimal important difference between anchor I and class 2 - pullover -



(B) Minimal important difference between anchor I and class 3 - Dress -



(C) Minimal important difference between anchor I and class 4 - Coat -

FIGURE 6.12: These figures show the selected / closest data points of another class which largely retains the generative factors that are not related to changing the class.

a coat because it is generally bigger (difference in scale), as shown in 6.12c. As a result, the method provides us with extra information with respect to several notions. First, a stable and relatively confident way to ensure that when both classifiers agree the prediction is correct. Second, when they are unsure and do not agree - then a most probable answer is returned with a way to check the decision behavior when traversing from the most probable answer to other similar possible classes. This provides a way to understand and be confident about the way the classifiers predict classes.

6.4 dSprites Data set

6.4.1 Latent space

The dsprites data set consists of several latent factors that are harder to distinguish given a low dimensional latent space. It is also for this reason that when training the Triplet-VAE for a semantically relevant latent space two-dimensions is insufficient. This effect can be seen in figures 6.13 and 6.14 and the height of the loss functions as well as the linear SVM accuracies, see 6.12.

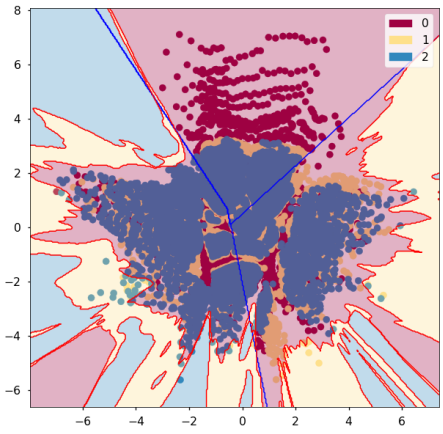


FIGURE 6.13: visualization of a 2-dimensional latent space of a vanilla VAE on dsprites

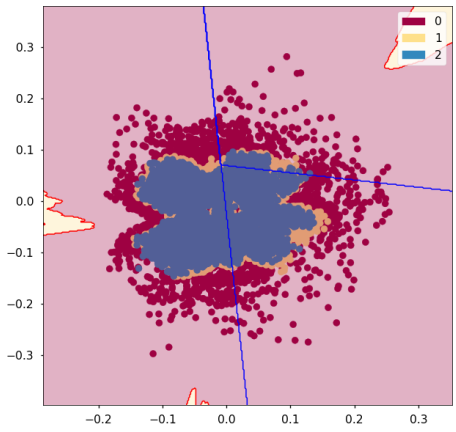


FIGURE 6.14: visualization of a 2-dimensional latent space of a $\mathcal{T}$-VAE on dpsrites

	Latent dimension	Recon loss	KL loss	Triplet Loss	Total loss
Vanilla VAE	2	105,9252	9,9567	n/a	115,8819
	16	6,7491	26,6928	n/a	33,4419
	32	7,0766	26,6928	n/a	33,7694
Triplet VAE	2	124,2106	13,4157	4,0000	141,6263
	16	9,4593	27,8814	2,0000	39,3407
	32	9,0635	29,2904	1,9560	40,3099

TABLE 6.10: Comparison of Vanilla VAE and triplet-VAE loss on the dsprites data set

Similar to the results of the other data sets, the reconstruction loss of the triplet vae is slightly worse than that of the vanilla VAE. However, the increase of well-separated classes within the latent space is significantly increased, as shown in 6.10 and 6.12.

6.4.2 Classification

The results of the classification part of the Dsprites data set is similar to the synthetic data set but different from the (Fashion-)MNIST data set. The objects in the test set are slightly off-set with an angle of $\frac{2/pi}{40}$ as to see how both the CNN and the SVM deal with unknown data. Very interestingly, even though the CNN performs perfectly on the training data set it performs much worse on the test data set. Whereas the SVM trained on top of the VAE latent space has a lower initial training accuracy but is able to generalize much better than the CNN and, also, the linear SVM without a VAE. This can be explained by the fact that the VAE encoded the unknown test data sets according to the training data points that are closely related and it has seen before, as such the CNN also performs better on the reconstructed test images than the actual test images. Additionally, given the complexity of the data the linear SVM without the VAE is only able to accurately predict 50% of the test data, whereas the SVM on top of the Triplet-VAE is able to accurately predict 87.6 percent of the test data. Furthermore, there is a big gap between the accuracies of the linear SVM on the Vanilla-VAE and the Triplet-VAE showing that the triplet-VAE is definitely necessary for finding a semantically relevant and well-separated latent space, as can be seen in 6.12 and 6.13. It is a great advantage if the model is able to generalize well, especially in environments where data is scarce and not all possible cases according to the generative factors are captured within the training data. Next, if we compare the case percentages for the comparison of the two classifiers, it can be observed that in 70 percent of the cases they agree and are correct and only in 0.01 percent they are incorrect. In this way we can generally be sure that if the classifiers agree then it is unlikely that you have to check those unknown data points. In the other cases however, it is still best to check to which class the actual images belong to with the help of an explanation.

Case	**Percentage**
1) $\hat{Y_S} = \hat{Y_C} = Y$	0.68267
2) $\hat{Y_S} = \hat{Y_C} \neq Y$	0.00010
3) $(\hat{Y_S} = Y) \neq \hat{Y_C}$	0.19384
4) $\hat{Y_S} \neq (\hat{Y_C} = Y)$	0.10297
5) $\hat{Y_S} \neq \hat{Y_C} \neq Y$	0.020399

TABLE 6.11: This table shows the percentages of agreement with respect to all possible cases.

Latent	**linear SVM accuracies**	
dimensions	**vanilla-VAE**	**Triplet-VAE**
2	0.3505	0.34939
3	0.4119	0.3708
4	0.45133	0.8660
5	0.4478	0.8409
6	0.46153	0.83584
7	0.46112	0.8829
8	0.4734	0.83912
9	0.4493	0.81277
10	0.4488	0.8594

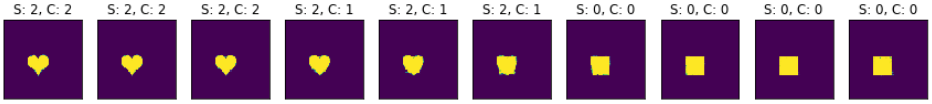
TABLE 6.12: Linear support vector machine accuracies per VAE type and latent dimension. It can clearly be seen that in lower dimensions the space cannot be well separated with respect to classes. However, given higher latent dimensions the triplet-VAE clearly creates a more semanticallh meaningful latent space than the vanilla-VAE.

	V-VAE		T-VAE		No VAE
	2d	32d	2d	32d	
CNN true train accuracy	-	-	-	-	1,0000
CNN rec train accuracy	0,3344	0,9924	0,3333	0,9671	-
SVM train accuracy	0,3528	0,5737	0,3582	0,8907	0,5403
CNN true test accuracy	-	-	-	-	0,7856
CNN rec test accuracy	0,3341	0,9080	0,3333	0,8144	-
SVM test accuracy	0,3541	0,5690	0,3549	**0,8765**	0,4964

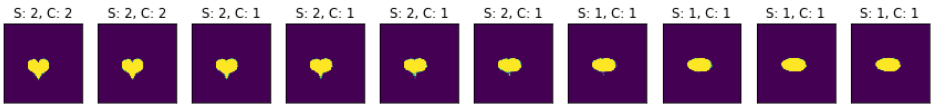
TABLE 6.13: CNN and SVM performance on (reconstructed) training and (reconstructed) test data

6.4.3 Generating explanations

Similar to the results from earlier datasets, selecting and generating explanations behave and act as expected. This means that when selecting closely related training data points from the semantically relevant latent space, the selected points are only different from the anchor in terms of class specific factors and not general generative factors such as scale, x location and y location. This behavior can be seen in figures 6.15a and 6.15b. We start with an anchor I which is a shape in the form of a heart and state that we wish to see the minimal important difference that changes the class from a heart to a square and an oval shape. That is, we are asking: 'Why is this classified as a heart and not a square?' or 'why is this classified as a heart and not an oval?'. The path traversal from I to that specific class shows the behavior of both the SVM and the CNN when classifying the images that change along its path. We can then specifically zoom in to the moment where the predictions change from the initial class - heart - to the target class - square and oval -. For instance, with 6.15b, the CNN is far quicker in changing its prediction than the linear SVM. Even when the image still looks like a heart. This means that even the slightest difference or change of pixel values changes the CNN's prediction. This can clearly be seen in figures 6.16a and 6.16b, which shows the minimal important difference for the CNN and SVM separately. The SVM is a lot more conservative in changing its decision than the CNN.

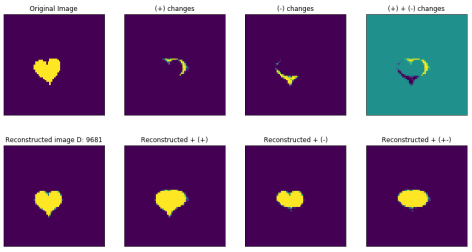


(A) Selection of closest training data point of class 0 - Square - with path traversal and sampling from I to k.

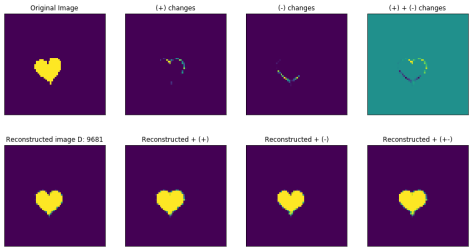


(B) Selection of closest training data point of class 1 - Ellipse - with path traversal and sampling from I to k.

FIGURE 6.15: The visualization of generated images from uniformly sampling vector v and decoding them using the decoder.



(A) Minimal important difference between anchor I and class 2 - pullover -



(B) Minimal important difference between anchor I and class 3 - Dress -

FIGURE 6.16: These figures show the selected / closest data points of another class which largely retains the generative factors that are not related to changing the class.

Chapter 7

Conclusion and Recommendations

7.1 Conclusion

Deep neural networks are a black box, it is unknown as to what factors contribute to the decision process. This thesis examines deep neural network's behavior and performance by utilizing a weakly-supervised generative model as a proxy. The weakly-supervised generative model aims to uncover the generative factors underlying the data and separate abstract classes by applying metric learning. The proxy's goal is three-fold: the semantically relevant space will be the base for a linear support vector machine; The model's generative capabilities will be used to generate images that can be probed against the black box in question; the latent space is traversed and sampled from an anchor I to another class k in order to find the minimal important difference that changes both classifier's predictions. The goal of the framework is to be sure of the predictions made by the black box by better understanding the behavior of the CNN by simulating questions of the form 'Why a and not b?' where a and b are different classes.

The results show that each of the above goals can be achieved and the framework performs as expected. First, a semantically relevant latent space is found by applying metric learning to a VAE. If we compare the latent space of the Vanilla-VAE and the Triplet-VAE we see a great improvement with respect to linear separation, especially in lower dimensions, as shown by comparing the linear support vector machine's accuracies on test data, as shown in 6.2, 6.5, 6.8 and 6.13, summarized in 7.1.

	No VAE	**V-VAE**	**T-VAE**
Synthetic data	0.93955	**0.9946**	**0.9932***
Mnist	0.9446	0.9511	**0.9642***
Fashion-Mnist	0.8461	0.8123	**0.8526***
dsprites	0,4964	0,5690	**0,8765***

TABLE 7.1: linear SVM accuracies with a 32 latent dimension VAE

Secondly, the latent space is (still) able to generate images that closely resembles the real images. The reconstruction loss that states how well the model is able to reconstruct images only slightly increases when adding the Triplet loss term. Similarly, the Kullback-Leibler divergence slightly increases. Next, when we combine and compare both classifiers - the SVM and the CNN -, if they agree, then there is only a tiny percentage in which we do not provide a warning, 0.0783 percent, 0.44 percent, 2.1 percent and 0.01 percent for each data set as presented in this thesis, see 6.3, 6.6, 6.9 and 6.11. In all other cases, especially, where the SVM and the CNN do not agree we generate an explanation as to understand the behavior of both the SVM and CNN. The SVM's decision boundaries are used as ground truth when selecting training data that are closely related to the anchor image I. The explanations consist of two parts, first, anchor I is used as a base for selecting a representative from another class. The selection process aims to find a training data point where the change in generative factors other than class related factors are minimized i.e. color, thickness, rotation, position, scale etc. The first part of an explanation then consists of visualizing the vector/line between I and the selected training data point by utilizing the decoder of the VAE. Class related factors change along the vector meaning that the images generated gradually change anchor I to the selected training data point. This shift is probed against both the SVM and CNN to find the minimal important difference that changes the predictions. The second

part of the explanation then consists of visualizing the minimal important difference per classifier or for both classifiers as to understand the behavior. The extent of these explanations as to answer questions of the form 'why a and not b?' can be found in the results for each data set respectively.

As a result, we posit a framework that is an able to answer all the research questions that were stated in the introduction. The framework is a model-agnostic method, that can be trained in parallel of the CNN and only influences the CNN by having it predict generated data points. We trained a Triplet-VAE to structure the latent space such that is semantically relevant with respect to classes and found a way to utilize this latent space to select relevant (contrastive) explanations. An explanation consists of selecting a training data point of another closely related class where the change in generative factors are minimized. We then contrast an anchor against this selected training point to measure the behavior of the CNN. The extent of these explanations can still be improved as described in 7.2

7.2 Recommendations

The first and foremost recommendation is to improve the extent to which the explanations measure and state behavior. The explanations, at this point, consist of visual inspection of the change in pixel values when traversing from anchor I to the selected training data point of the targeted class. It should be investigated to see if it is possible to link changes of pixel values can be expressed as actual real life concepts. One approach to achieve this goal is to try to add an extra loss function that disentangles generative factors per feature in the latent space such as [64], [65] and [66]. This loss function can be added in addition to the loss functions currently active within the Triplet-VAE. The latent space of the disentangled Triplet-VAE should be carefully observed and loss functions should be balanced. I expect that the addition of this loss function might cause some strange behavior to the latent space. My intuition is that the Triplet loss and disentangling loss pull at each other and create a latent space that is a compromise between both loss functions. The Triplet loss will disentangle the abstract classes and the disentangling loss will disentangle each generative factors successfully but the Triplet loss might entangle factors related to the class. This effect of this combination of loss functions on the latent space should be carefully observed. Additionally, a measure should be put in place to capture and label each factor of the latent space as to which generative factor it adheres to. Afterwards, if we then generate explanations we can explain the vector from I to the closely related training data point more concretely, I.e. there is a c amount of change per labelled factor.

Secondly, If the factors are disentangled in the latent space then it would also make it possible to test the black-box model on bias. Similar to the approach stated in this thesis we could take an anchor I and specifically change singular factors such as for example, height, width, (hair) color. This gradual change can then be visualized by uniformly sampling along the change vector and using the decoder to generate images. These generated images can then be used to probe and measure the behavior of the black-box with respect to the abstract classes defined by humans.

Thirdly, The images that are generated by the decoder when we traverse the latent space are currently direct interpolations between the anchor and the target. The interpolated images might not actually represent images that exist in a real situation. An approach by David Berthelot et al.[67] describes a method of adding an adversarial regularizer in autoencoders as to make the interpolated outputs appear more realistic. In this way, if there are gaps in between classes in the latent space of the AE then the adversarial regularizer will encourage the images in this gap to be a realistic interpolation. This method could be added to the framework to make the output images when traversing the latent space more realistic. As a result, the explanations might make more sense than the direct interpolations between the two classes.

Lastly, whilst not originally the goal for the framework it could also be used in order to annotate and label (and generate) new data. If we have two classifiers then if they agree we can be confident to a certain degree that the prediction is correct. If this is the case, then we can test and annotate a large part of a data set. On the other hand, if they do not agree we will not annotate the images and mark them as data that needs more investigation and if verified by experts can then be added to the training data of the classifiers as to then annotate and label a new part of the data set.

Bibliography

[1] Surya Mattu Julia Angwin Jeff Larson and ProPublica Lauren Kirchner. *Machine Bias*. 2016, 05. URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[2] Robert Challen et al. "Artificial intelligence, bias and clinical safety". In: *BMJ Quality & Safety* 28.3 (2019), pp. 231–237. ISSN: 2044-5415. DOI: 10.1136/bmjqs-2018-008370. eprint: https://qualitysafety.bmj.com/content/28/3/231.full.pdf. URL: https://qualitysafety.bmj.com/content/28/3/231.

[3] Abishur Prakash. "AI-Politicians: A Revolution In Politics". In: (2018). URL: https://medium.com/politics-ai/ai-politicians-a-revolution-in-politics-11a7e4ce90b0.

[4] Joy Buolamwini and Timnit Gebru. "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification". In: *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. Ed. by Sorelle A. Friedler and Christo Wilson. Vol. 81. Proceedings of Machine Learning Research. New York, NY, USA: PMLR, 2018, pp. 77–91. URL: http://proceedings.mlr.press/v81/buolamwini18a.html.

[5] European Union. *Official Journal of the European Union: Regulations*. 2016. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679\&from=EN.

[6] Bryce Goodman and Seth Flaxman. "EU regulations on algorithmic decision-making and a "right to explanation"". In: (2016). URL: http://arxiv.org/abs/1606.08813.

[7] High Level Expert Group. *Ethics Guidelines Trustworthy AI*. Sept. 2019. URL: https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai.

[8] Tim Miller. "Explanation in Artificial Intelligence: Insights from the Social Sciences". In: *CoRR* abs/1706.07269 (2017). arXiv: 1706.07269. URL: http://arxiv.org/abs/1706.07269.

[9] Leilani H. Gilpin et al. "Explaining Explanations: An Approach to Evaluating Interpretability of Machine Learning". In: *CoRR* abs/1806.00069 (2018). arXiv: 1806.00069. URL: http://arxiv.org/abs/1806.00069.

[10] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *CoRR* abs/1602.04938 (2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[11] Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. "DeepRED – Rule Extraction from Deep Neural Networks". In: *Discovery Science*. Ed. by Toon Calders, Michelangelo Ceci, and Donato Malerba. Cham: Springer International Publishing, 2016, pp. 457–473. ISBN: 978-3-319-46307-0.

[12] Ramprasaath R. Selvaraju et al. "Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization". In: *CoRR* abs/1610.02391 (2016). arXiv: 1610.02391. URL: http://arxiv.org/abs/1610.02391.

[13] Pieter-Jan Kindermans et al. "The (Un)reliability of saliency methods." In: *CoRR* abs/1711.00867 (2017). URL: http://dblp.uni-trier.de/db/journals/corr/corr1711.html\#abs-1711-00867.

[14] Shan Carter et al. "Activation Atlas". In: *Distill* (2019). https://distill.pub/2019/activation-atlas. DOI: 10.23915/distill.00015.

[15] Been Kim et al. "Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 2668–2677. URL: http://proceedings.mlr.press/v80/kim18d.html.

[16] Zachary C Lipton. "The Doctor Just Won't Accept That!" In: *NIPS Proceedings 2017* 24 (Nov. 2017), pp. 1–3. URL: https://arxiv.org/pdf/1711.08037.pdf.

[17] Tania Lombrozo. "Explanation and Abductive Inference". In: *Oxford Handbook of Thinking and Reasonin* (Jan. 2012), pp. 260–276. DOI: 10.1093/oxfordhb/9780199734689.013.0014.

[18] Leilani H. Gilpin et al. "Explaining Explanations: An Approach to Evaluating Interpretability of Machine Learning". In: *CoRR* abs/1806.00069 (2018). arXiv: 1806.00069. URL: http://arxiv.org/abs/1806.00069.

[19] Riccardo Guidotti et al. "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* 51.5 (Aug. 2018), 93:1–93:42. ISSN: 0360-0300. DOI: 10.1145/3236009. URL: http://doi.acm.org/10.1145/3236009.

[20] Finale Doshi-Velez and Been Kim. "Towards A Rigorous Science of Interpretable Machine Learning". In: *arXiv* (2017). URL: https://arxiv.org/abs/1702.08608.

[21] A. Adadi and M. Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2870052.

[22] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/. 2019.

[23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-Precision Model-Agnostic Explanations". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

[24] Riccardo Guidotti et al. "Local Rule-Based Explanations of Black Box Decision Systems". In: *CoRR* abs/1805.10820 (2018). arXiv: 1805.10820. URL: http://arxiv.org/abs/1805.10820.

[25] Daniel Smilkov et al. "SmoothGrad: removing noise by adding noise". In: *CoRR* abs/1706.03825 (2017). arXiv: 1706.03825. URL: http://arxiv.org/abs/1706.03825.

[26] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *CoRR* abs/1703.01365 (2017). arXiv: 1703.01365. URL: http://arxiv.org/abs/1703.01365.

[27] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: *Distill* (2017). https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

[28] Chris Olah et al. "The Building Blocks of Interpretability". In: *Distill* (2018). https://distill.pub/2018/building-blocks. DOI: 10.23915/distill.00010.

[29] David Bau et al. "Network Dissection: Quantifying Interpretability of Deep Visual Representations". In: *CoRR* abs/1704.05796 (2017). arXiv: 1704.05796. URL: http://arxiv.org/abs/1704.05796.

[30] Bolei Zhou et al. "Object Detectors Emerge in Deep Scene CNNs". In: *International Conference on Learning Representations (ICLR)*. 2015. URL: https://arxiv.org/abs/1412.6856.

[31] Anh Mai Nguyen et al. "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks". In: *CoRR* abs/1605.09304 (2016). arXiv: 1605.09304. URL: http://arxiv.org/abs/1605.09304.

[32] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[33] Tianjun Xiao et al. "The Application of Two-level Attention Models in Deep Convolutional Neural Network for Fine-grained Image Classification". In: *CoRR* abs/1411.6447 (2014). arXiv: 1411.6447. URL: http://arxiv.org/abs/1411.6447.

[34] Svante Wold, Kim Esbensen, and Paul Geladi. "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[35] A. Hyvärinen and E. Oja. "Independent Component Analysis: Algorithms and Applications". In: *Neural Netw.* 13.4-5 (May 2000), pp. 411–430. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(00)00026-5. URL: http://dx.doi.org/10.1016/S0893-6080(00)00026-5.

[36] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014. URL: http://arxiv.org/abs/1312.6114.

[37] Xi Chen et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *CoRR* abs/1606.03657 (2016). arXiv: 1606.03657. URL: http://arxiv.org/abs/1606.03657.

[38] L. A. Hendricks et al. "Grounding Visual Explanations". In: *ArXiv e-prints* (July 2018). arXiv: 1807.09685 [cs.CV].

[39] Lisa Anne Hendricks et al. "Generating Visual Explanations". In: *CoRR* abs/1603.08507 (2016). arXiv: 1603.08507. URL: http://arxiv.org/abs/1603.08507.

[40] Ruth Fong and Andrea Vedaldi. "Interpretable Explanations of Black Boxes by Meaningful Perturbation". In: *CoRR* abs/1704.03296 (2017). arXiv: 1704.03296. URL: http://arxiv.org/abs/1704.03296.

[41] Amir Feghahati et al. *CDeepEx: Contrastive Deep Explanations*. 2019. URL: https://openreview.net/forum?id=HyNmRiCqtm.

[42] Shalmali Joshi et al. "xGEMs: Generating Examplars to Explain Black-Box Models". In: *CoRR* abs/1806.08867 (2018). arXiv: 1806.08867. URL: http://arxiv.org/abs/1806.08867.

[43] Ari S. Morcos et al. "On the importance of single directions for generalization". In: *arXiv e-prints*, arXiv:1803.06959 (2018), arXiv:1803.06959. arXiv: 1803.06959 [stat.ML].

[44] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *CoRR* abs/1906.02691 (2019). arXiv: 1906.02691. URL: http://arxiv.org/abs/1906.02691.

[45] Haque Ishfaq, Assaf Hoogi, and Daniel Rubin. *TVAE: Triplet-Based Variational Autoencoder using Metric Learning*. 2018. arXiv: 1802.04403 [stat.ML].

[46] Theofanis Karaletsos, Serge Belongie, and Gunnar Rätsch. *Bayesian representation learning with oracle constraints*. 2015. arXiv: 1506.05011 [stat.ML].

[47] .

[48] Arindam Banerjee. "An Analysis of Logistic Models: Exponential Family Connections and Online Performance". In: Apr. 2007. DOI: 10.1137/1.9781611972771.19.

[49] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[50] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[51] Yann LeCun, Y. Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539.

[52] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[53] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[54] Pierre Baldi. "Autoencoders, Unsupervised Learning and Deep Architectures". In: *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*. UTLW'11. Washington, USA: JMLR.org, 2011, pp. 37–50. URL: http://dl.acm.org/citation.cfm?id=3045796.3045801.

[55] Junyuan Xie, Linli Xu, and Enhong Chen. "Image Denoising and Inpainting with Deep Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 341–349. URL: http://papers.nips.cc/paper/4686-image-denoising-and-inpainting-with-deep-neural-networks.pdf.

[56] Yoshua Bengio et al. "Greedy Layer-wise Training of Deep Networks". In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. NIPS'06. Canada: MIT Press, 2006, pp. 153–160. URL: http://dl.acm.org/citation.cfm?id=2976456.2976476.

[57] Pascal Vincent et al. "Extracting and Composing Robust Features with Denoising Autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, 2008, pp. 1096–1103. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390294. URL: http://doi.acm.org/10.1145/1390156.1390294.

[58] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. cite arxiv:1312.6114. 2013. URL: http://arxiv.org/abs/1312.6114.

[59] J. Kittler et al. "On combining classifiers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.3 (1998), pp. 226–239. ISSN: 1939-3539. DOI: 10.1109/34.667881.

[60] John C. Platt. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999, pp. 61–74.

[61] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. ISSN: 0018-9219. DOI: 10.1109/5.726791.

[62] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *CoRR* abs/1708.07747 (2017). arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

[63] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 2014. URL: http://arxiv.org/abs/1412.6980.

[64] Hyunjik Kim and Andriy Mnih. *Disentangling by Factorising*. 2018. arXiv: 1802.05983 [stat.ML].

[65] Tian Qi Chen et al. "Isolating Sources of Disentanglement in Variational Autoencoders". In: *CoRR* abs/1802.04942 (2018). arXiv: 1802.04942. URL: http://arxiv.org/abs/1802.04942.

[66] Christopher P. Burgess et al. *Understanding disentangling in -VAE*. 2018. arXiv: 1804.03599 [stat.ML].

[67] David Berthelot et al. "Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer". In: *CoRR* abs/1807.07543 (2018). arXiv: 1807.07543. URL: http://arxiv.org/abs/1807.07543.

Appendix A

Models

Below the models used in this thesis:

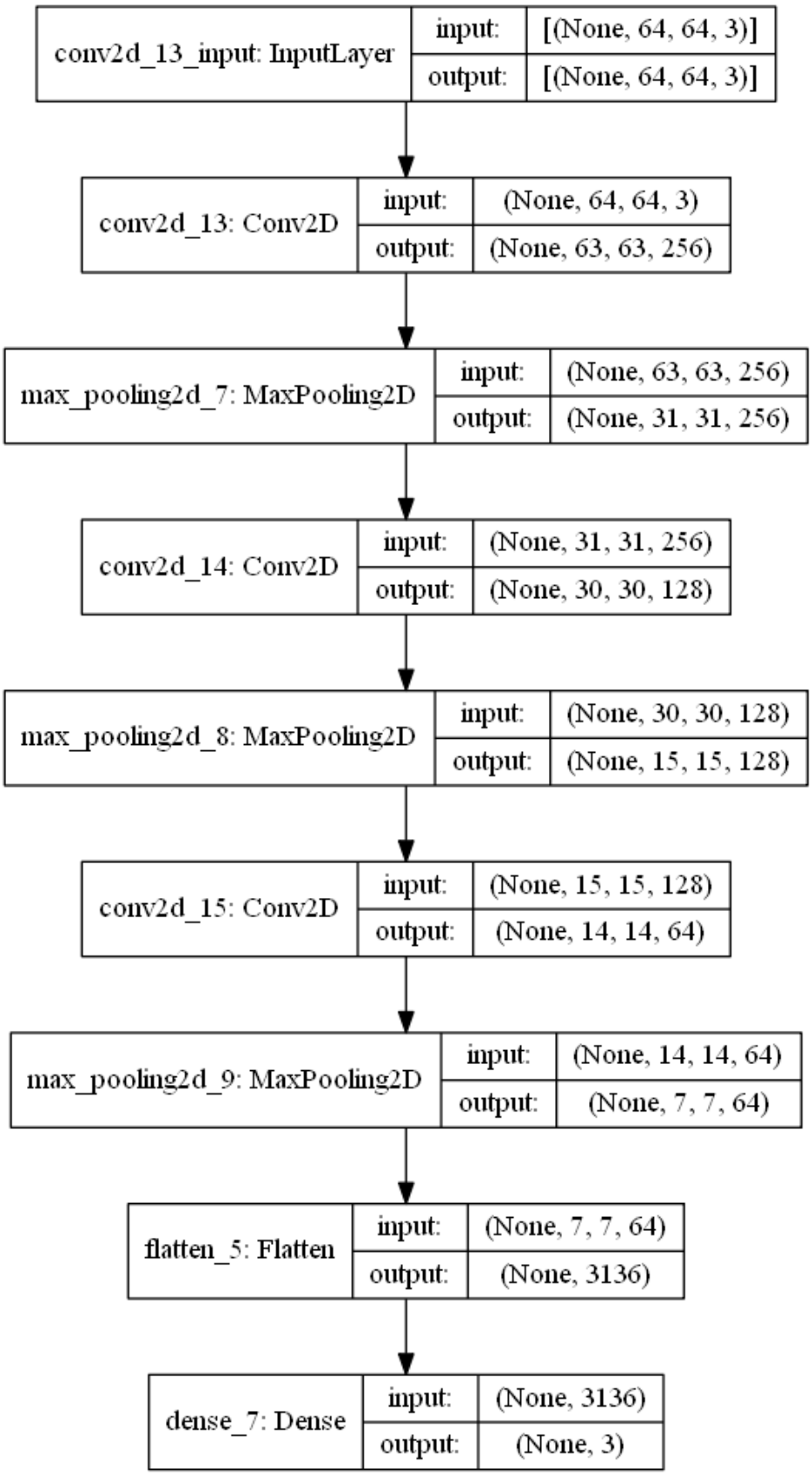


FIGURE A.1: CNN Architecture

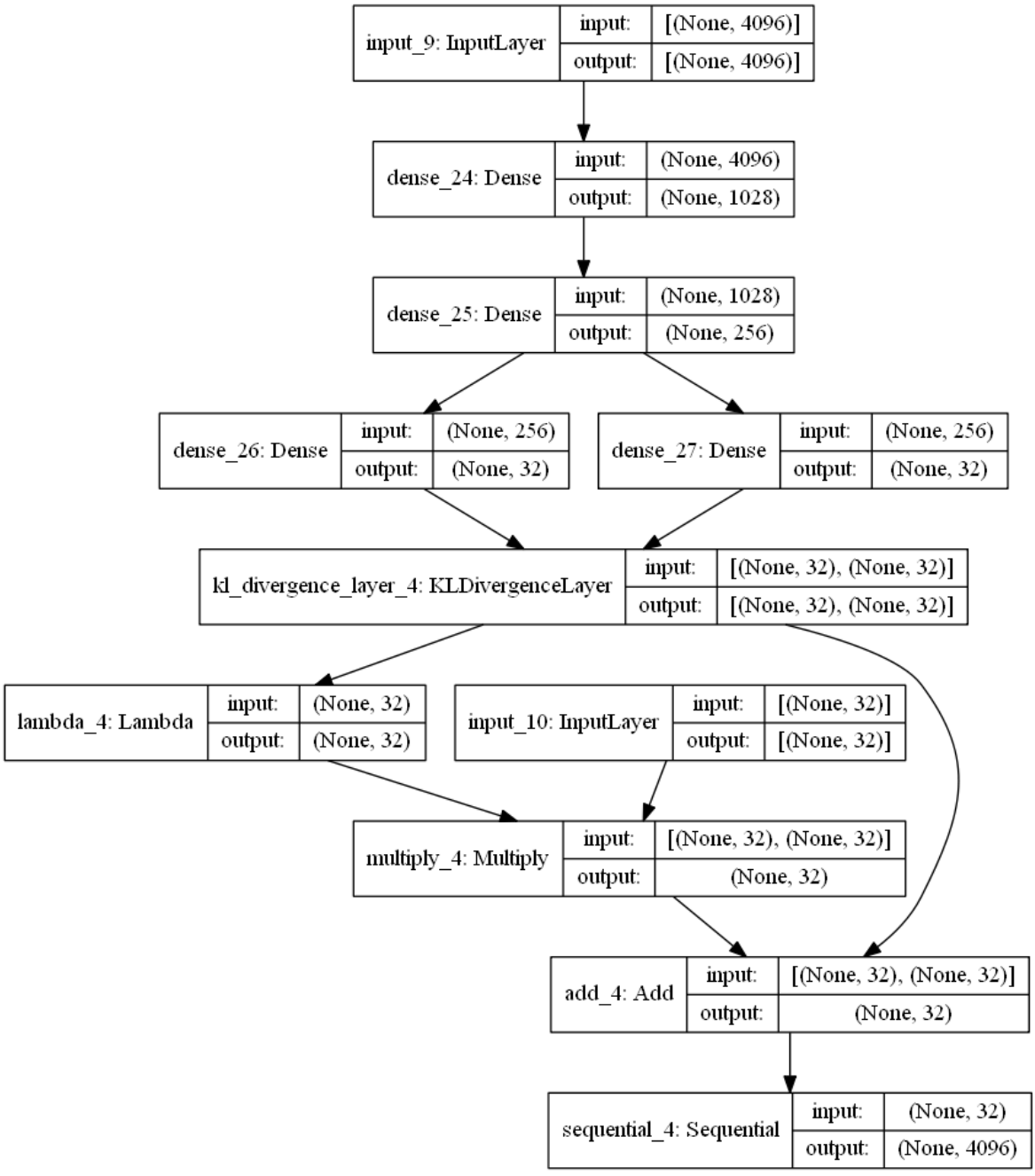


FIGURE A.2: Vanilla VAE architecture

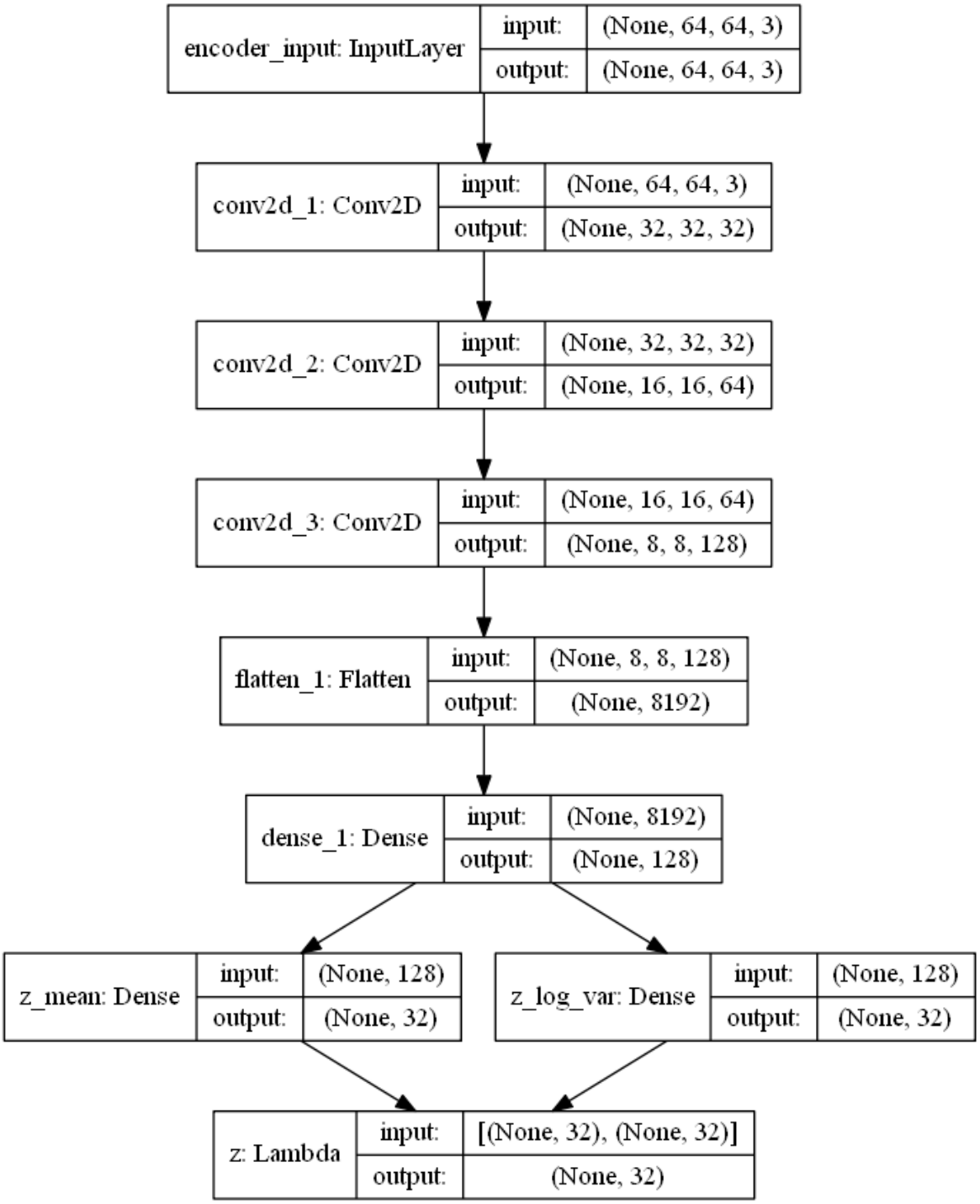


FIGURE A.3: Vanilla Convolutional encoder architecture

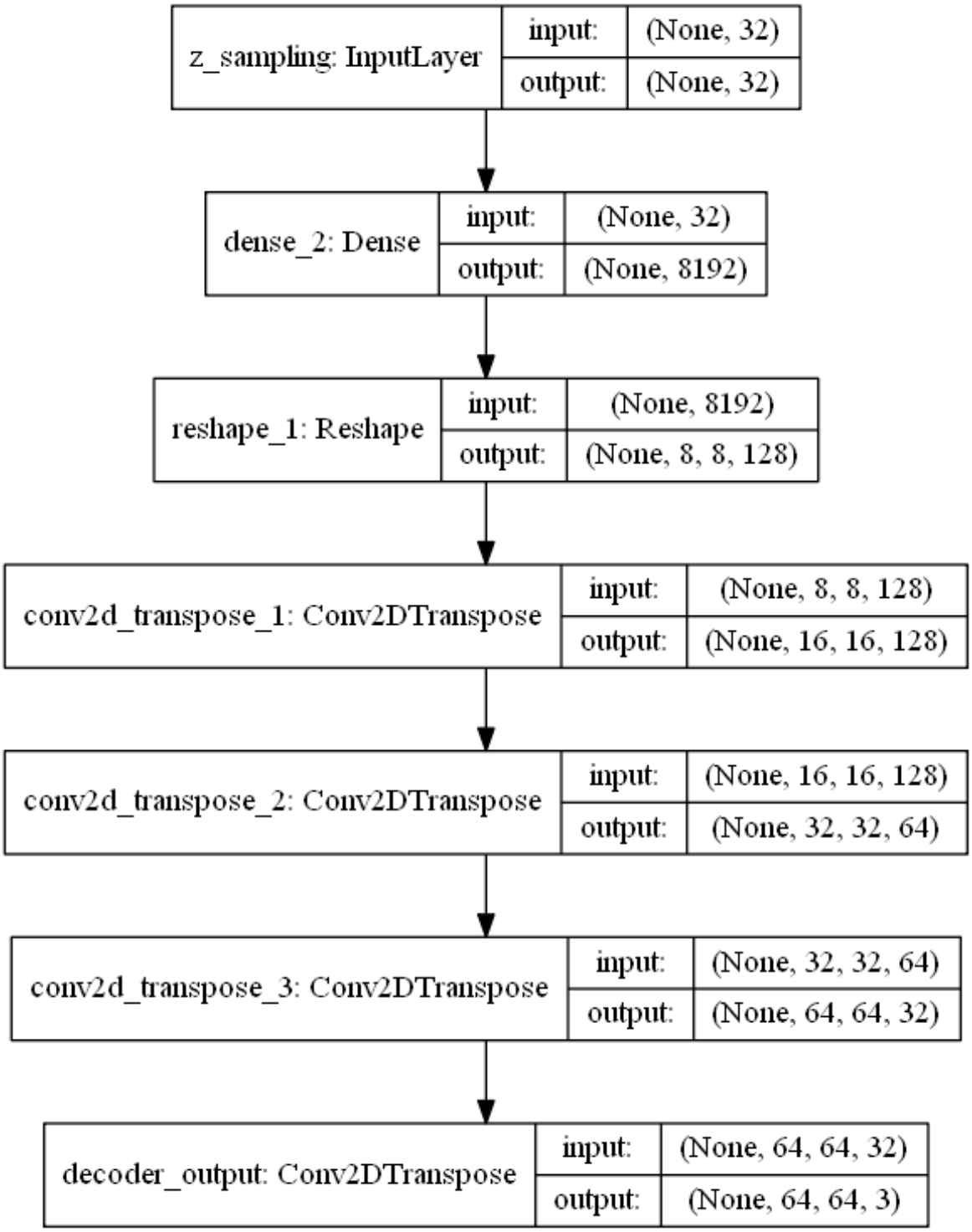


FIGURE A.4: Vanilla Convolutional decoder architecture

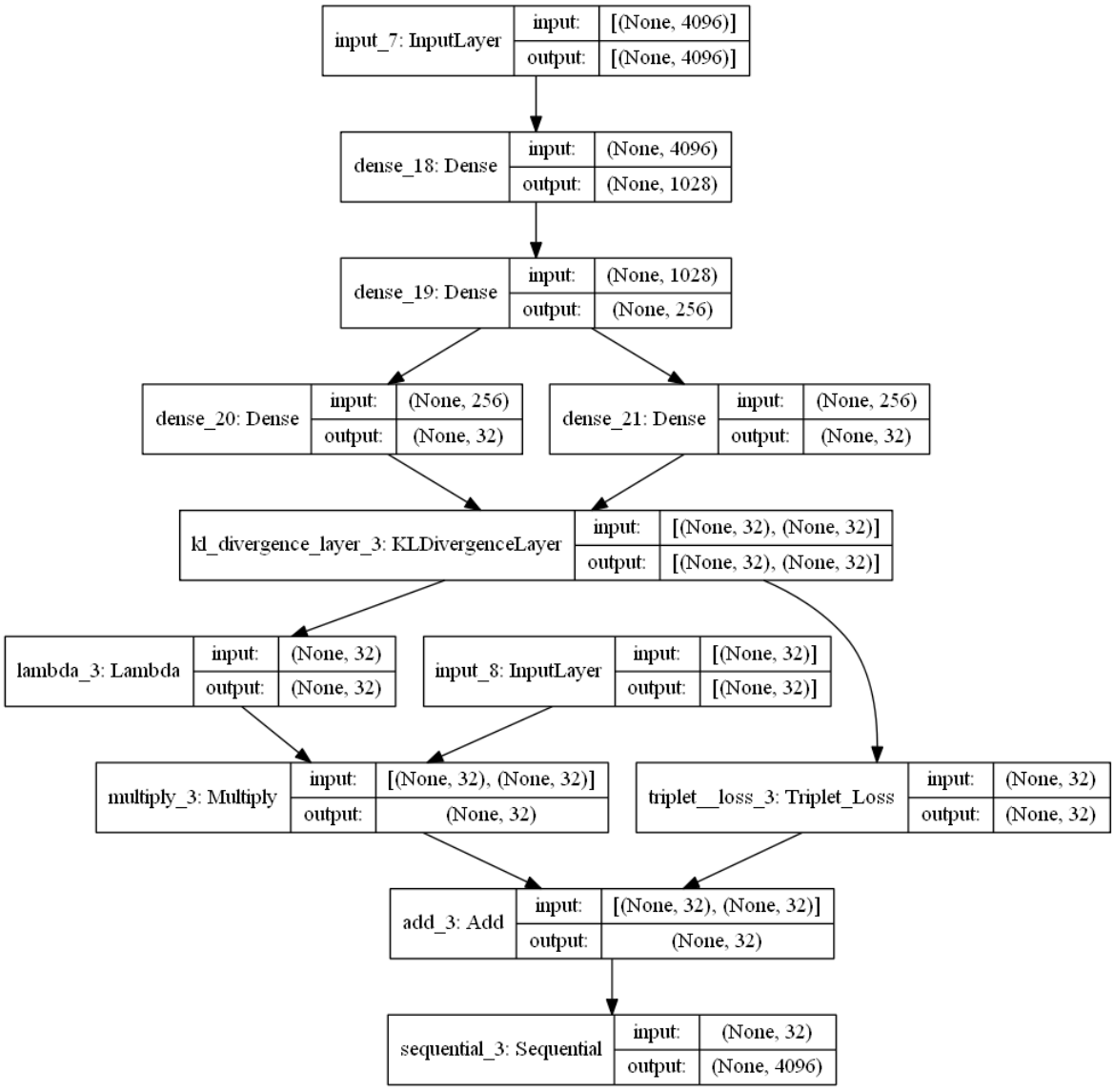


FIGURE A.5: Triplet VAE architecture

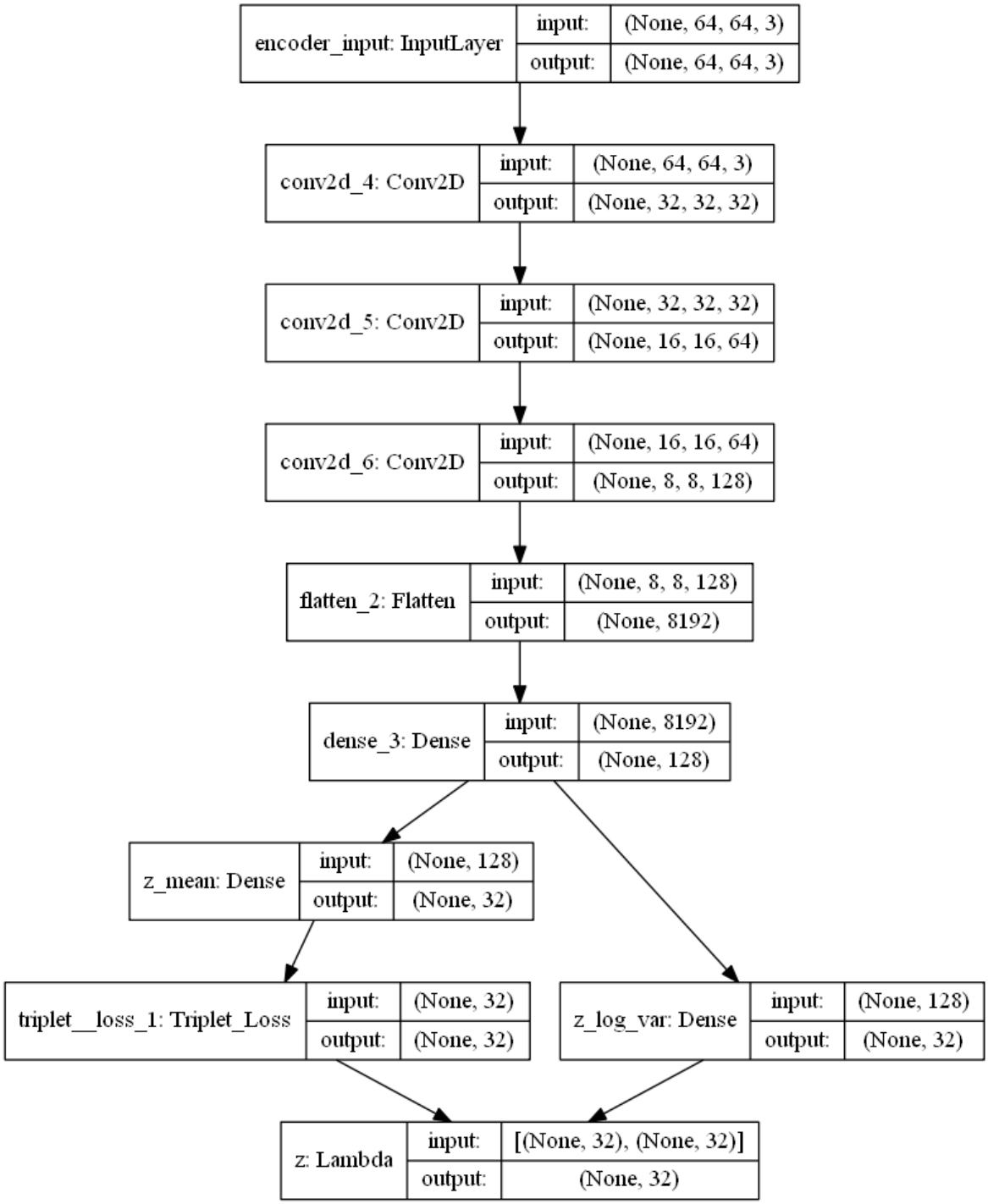


FIGURE A.6: Triplet Convolutional VAE - encoder architecture

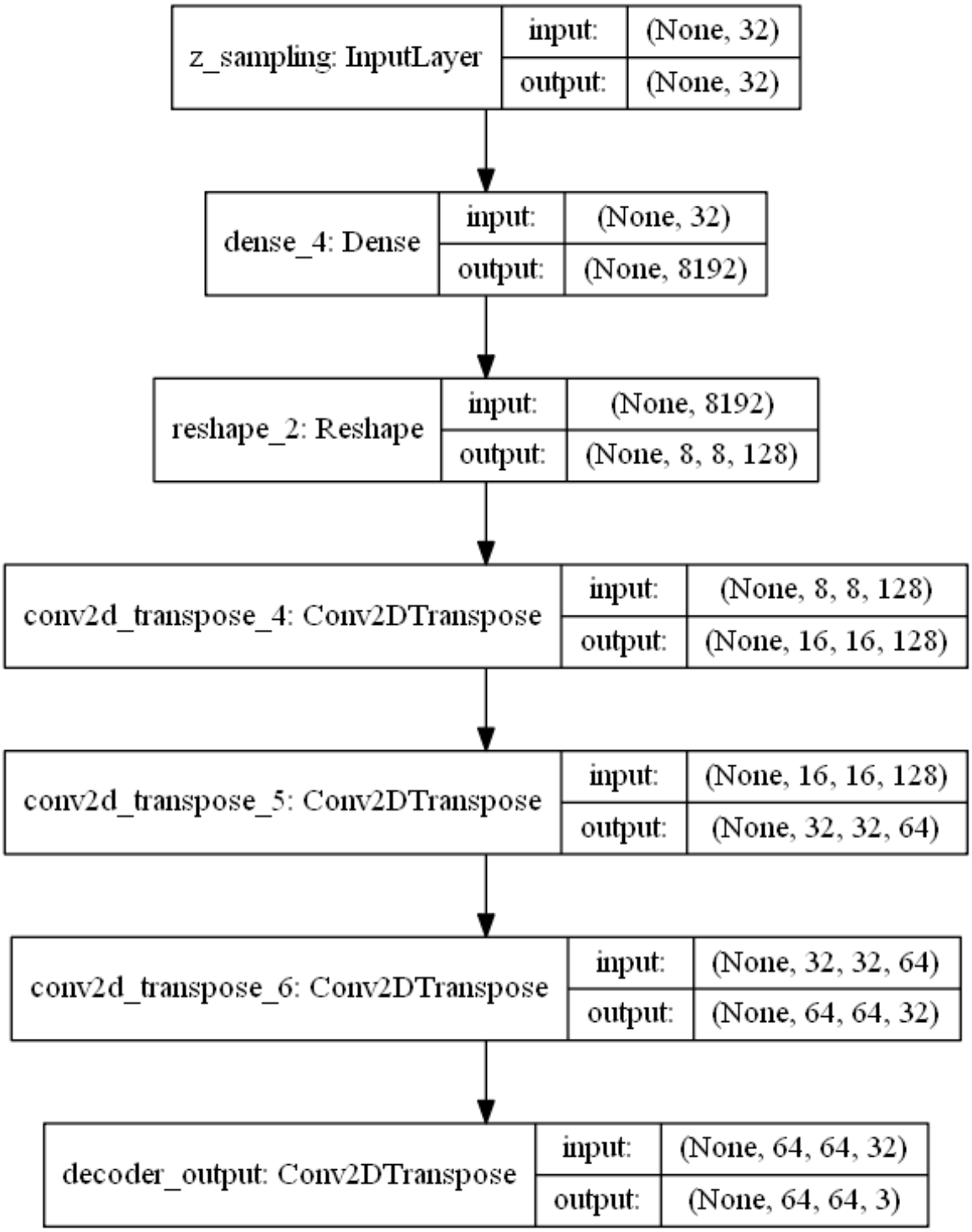
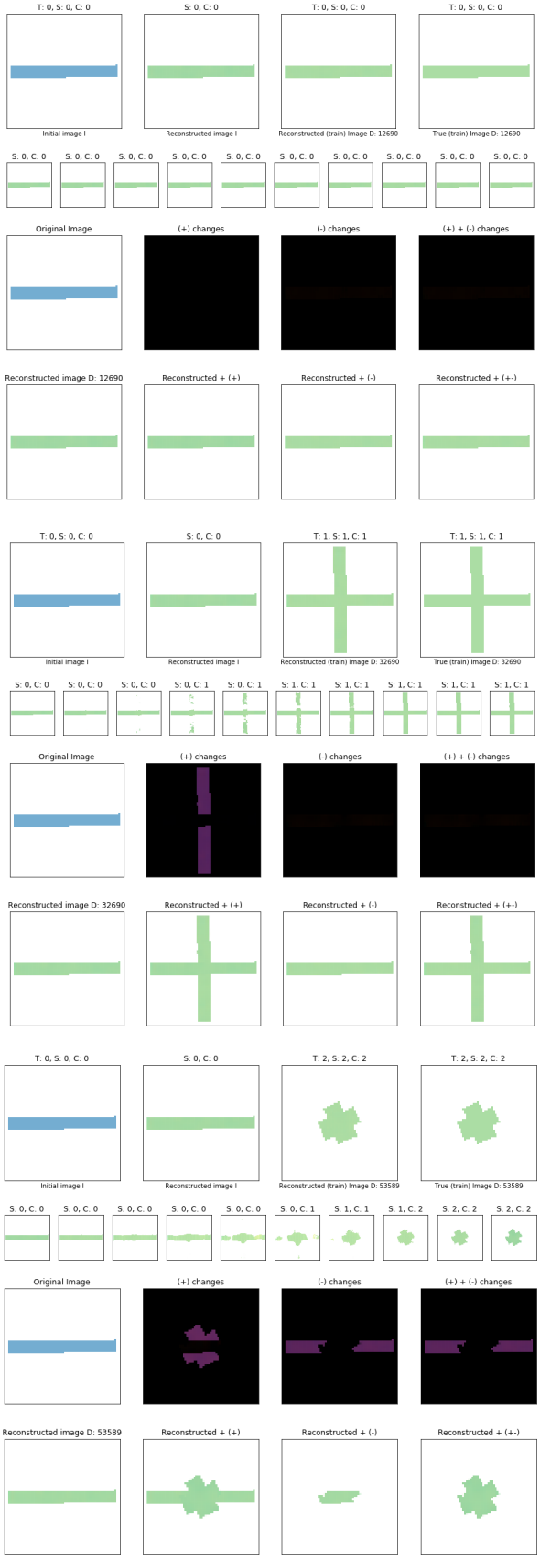
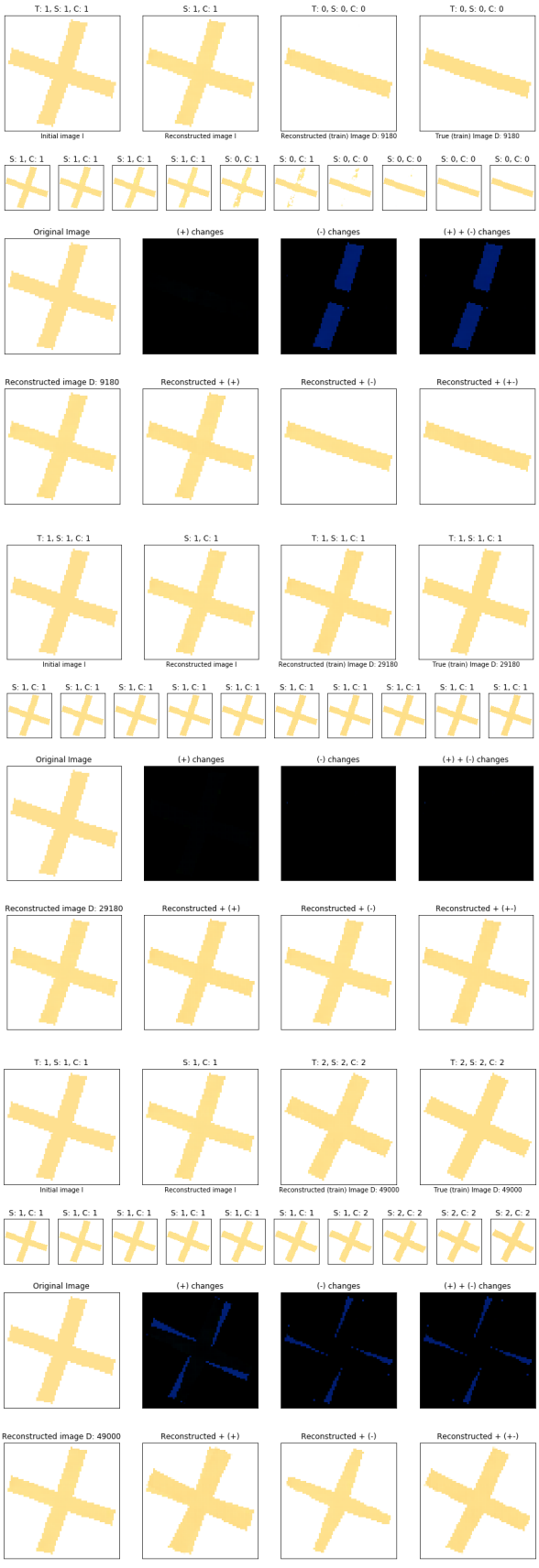


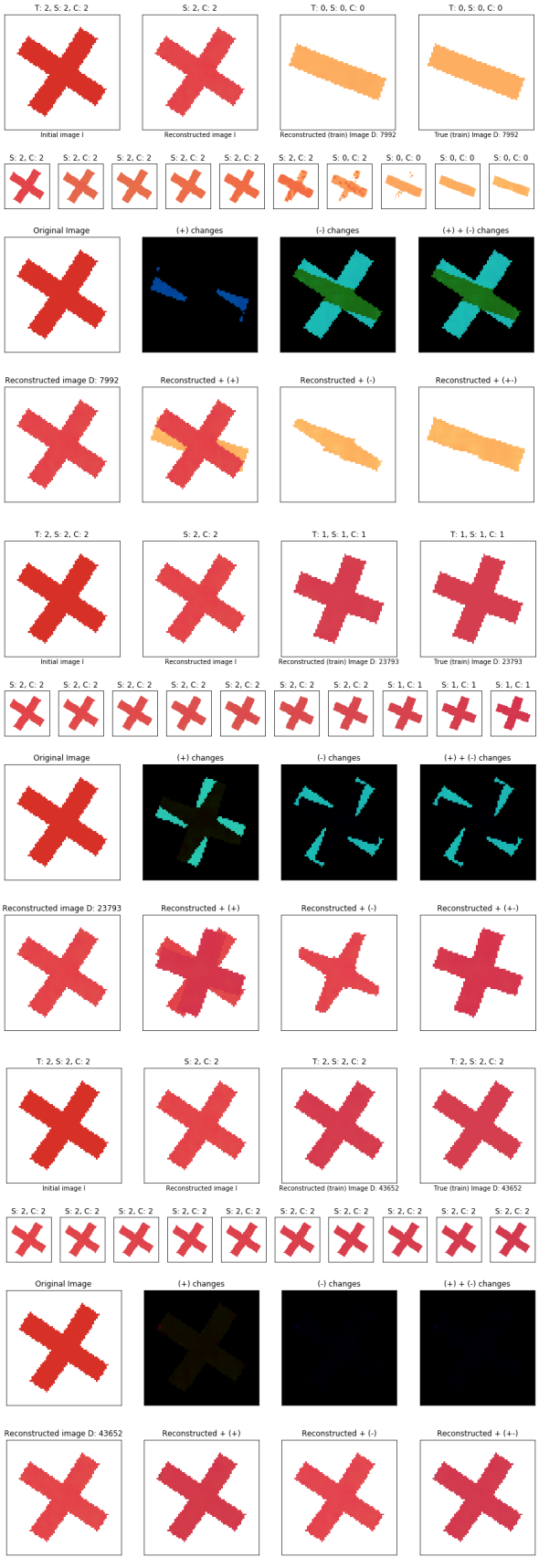
FIGURE A.7: Triplet Convolutional VAE - decoder architecture

Appendix B

Synthetic data - Generated explanations


T: 0, S: 0, C: 0
S: 0, C: 0
T: 0, S: 0, C: 0
T: 0, S: 0, C: 0
Initial image I
Reconstructed image I
Reconstructed (train) Image D: 12690
True (train) Image D: 12690
S: 0, C: 0
Original Image
(+) changes
(-) changes
(+) + (-) changes
Reconstructed image D: 12690
Reconstructed + (+)
Reconstructed + (-)
Reconstructed + (+-)
T: 0, S: 0, C: 0
S: 0, C: 0
T: 1, S: 1, C: 1
T: 1, S: 1, C: 1
Initial image I
Reconstructed image I
Reconstructed (train) Image D: 32690
True (train) Image D: 32690
S: 0, C: 0
S: 0, C: 1
S: 1, C: 1
Original Image
(+) changes
(-) changes
(+) + (-) changes
Reconstructed image D: 32690
Reconstructed + (+)
Reconstructed + (-)
Reconstructed + (+-)
T: 0, S: 0, C: 0
S: 0, C: 0
T: 2, S: 2, C: 2
T: 2, S: 2, C: 2
Initial image I
Reconstructed image I
Reconstructed (train) Image D: 53589
True (train) Image D: 53589
S: 0, C: 0
S: 0, C: 1
S: 1, C: 1
S: 1, C: 2
S: 2, C: 2
Original Image
(+) changes
(-) changes
(+) + (-) changes
Reconstructed image D: 53589
Reconstructed + (+)
Reconstructed + (-)
Reconstructed + (+-)


T: 1, S: 1, C: 1
S: 1, C: 1
T: 0, S: 0, C: 0
T: 0, S: 0, C: 0
Initial image I
Reconstructed image I
Reconstructed (train) Image D: 9180
True (train) Image D: 9180
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 0, C: 1
S: 0, C: 1
S: 0, C: 0
S: 0, C: 0
S: 0, C: 0
S: 0, C: 0
Original Image
(+) changes
(-) changes
(+) + (-) changes
Reconstructed image D: 9180
Reconstructed + (+)
Reconstructed + (-)
Reconstructed + (+-)
T: 1, S: 1, C: 1
S: 1, C: 1
T: 1, S: 1, C: 1
T: 1, S: 1, C: 1
Initial image I
Reconstructed image I
Reconstructed (train) Image D: 29180
True (train) Image D: 29180
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
Original Image
(+) changes
(-) changes
(+) + (-) changes
Reconstructed image D: 29180
Reconstructed + (+)
Reconstructed + (-)
Reconstructed + (+-)
T: 1, S: 1, C: 1
S: 1, C: 1
T: 2, S: 2, C: 2
T: 2, S: 2, C: 2
Initial image I
Reconstructed image I
Reconstructed (train) Image D: 49000
True (train) Image D: 49000
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
S: 1, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
Original Image
(+) changes
(-) changes
(+) + (-) changes
Reconstructed image D: 49000
Reconstructed + (+)
Reconstructed + (-)
Reconstructed + (+-)


T: 2, S: 2, C: 2
S: 2, C: 2
T: 0, S: 0, C: 0
T: 0, S: 0, C: 0
Initial image I
Reconstructed image I
Reconstructed (train) Image D: 7992
True (train) Image D: 7992
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 0, C: 2
S: 0, C: 0
S: 0, C: 0
S: 0, C: 0
Original Image
(+) changes
(-) changes
(+) + (-) changes
Reconstructed image D: 7992
Reconstructed + (+)
Reconstructed + (-)
Reconstructed + (+-)
T: 2, S: 2, C: 2
S: 2, C: 2
T: 1, S: 1, C: 1
T: 1, S: 1, C: 1
Initial image I
Reconstructed image I
Reconstructed (train) Image D: 23793
True (train) Image D: 23793
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 1, C: 1
S: 1, C: 1
S: 1, C: 1
Original Image
(+) changes
(-) changes
(+) + (-) changes
Reconstructed image D: 23793
Reconstructed + (+)
Reconstructed + (-)
Reconstructed + (+-)
T: 2, S: 2, C: 2
S: 2, C: 2
T: 2, S: 2, C: 2
T: 2, S: 2, C: 2
Initial image I
Reconstructed image I
Reconstructed (train) Image D: 43652
True (train) Image D: 43652
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
S: 2, C: 2
Original Image
(+) changes
(-) changes
(+) + (-) changes
Reconstructed image D: 43652
Reconstructed + (+)
Reconstructed + (-)
Reconstructed + (+-)