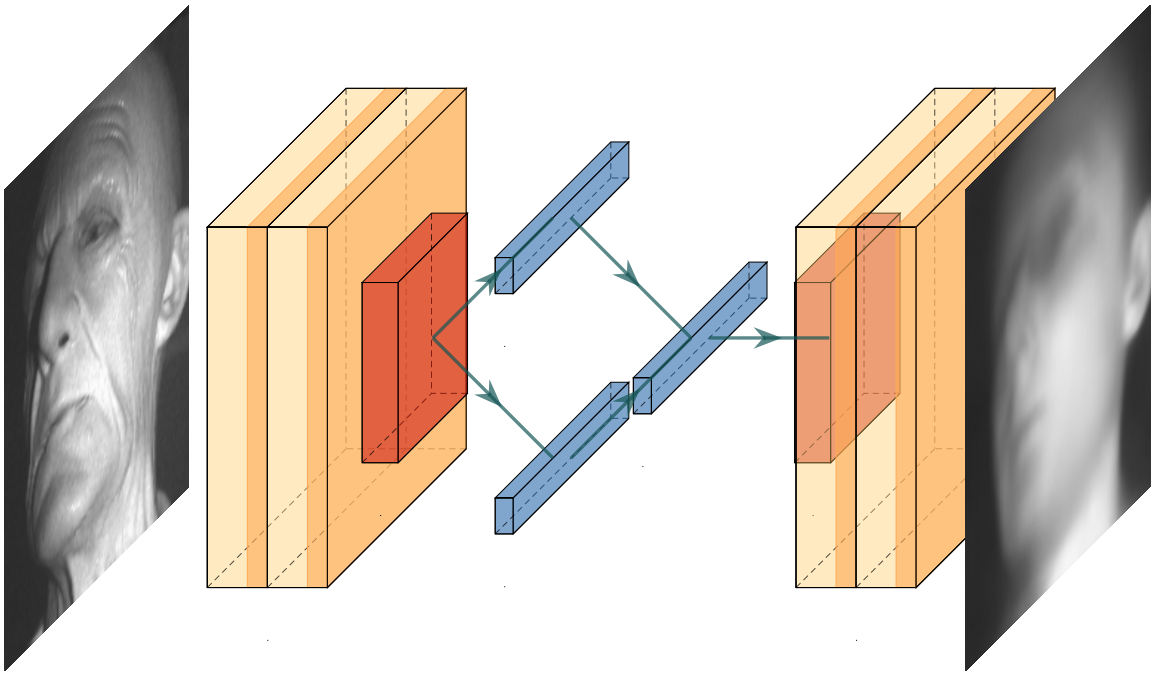


Unsupervised Learning for Face Anti-Spoofing Models

A generative approach using Autoencoders and Adversarial Networks

Master's thesis in Engineering Mathematics & Complex Adaptive Systems

GUSTAV MOLANDER, JENS NILSSON

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Unsupervised Learning for Face Anti-Spoofing Models

A generative approach using Autoencoders and Adversarial Networks

GUSTAV MOLANDER
JENS NILSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Artificial Intelligence
Chalmers University of Technology
Gothenburg, Sweden 2022

Unsupervised Learning for Face Anti-Spoofing Models
A generative approach using Autoencoders and Adversarial Networks
GUSTAV MOLANDER
JENS NILSSON

Supervisor and Examiner: Johan Jonasson, Department of Mathematical Sciences

Master's Thesis 2022
Department of Mathematical Sciences
Division of Artificial Intelligence
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of Variational Autoencoder architecture.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Unsupervised Learning for Face Anti-Spoofing Models
A generative approach using Autoencoders and Adversarial Networks

GUSTAV MOLANDER
JENS NILSSON

Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Distinguishing images of bonafide (genuine) faces from Presentation Attacks (PA) or spoofs attracts increasing attention in industry due to the wide application of reliable automatic authentication. The anomaly detection problem in high dimensional data such as images can be addressed as a supervised or unsupervised learning problem, but when spoof data is sparse and not all spoof domains can be represented in training the unsupervised model can generalize knowledge to unseen domains to a higher extent. The unsupervised anomaly detection paradigm is essentially a one-class learning problem to distinguish the distribution of bonafide images from everything else.

In this thesis we explore the mathematical framework and implementation of encoder-decoder-based deep learning models to learn the distribution of real images and classify spoofs. This thesis is preliminary work on the end goal of designing a lightweight unsupervised anti-spoofing model to run in Smart Eye automotive software. The researched models reproduce input images by utilizing a latent space embedding fitted to the distribution of bonafide images. The latent space along with the reconstructed image is used to classify spoofs.

The results of the experiments show promise but are not yet at the level feasible for implementation in production devices. The models have been evaluated in terms of the spoof precision and recall as well as the embedding of the spoofs in the latent space. Further, some models use Gaussian Mixture Models (GMM) of the latent space to determine the spoof affiliation, though these results are inconclusive.

Keywords: variational inference, deep learning, face anti-spoofing, variational autoencoder, Smart Eye, CNN, latent variable models

Acknowledgements

We would like to extend our thanks to Johan Jonasson whom have guided us through the theory and in the writing of this thesis. We would also like to thank our mentors at Smart Eye, Calle Ekdahl and Fredrik Walterson, for all their input and ideas. Finally we would like to thank Oscar and cother olleagues at Smart Eye for sharing their knowledge and helping us with big and small.

Gustav Molander & Jens Nilsson, 2022

Contents

List of Acronyms

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

AE	Autoencoder
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
ELBO	Evidence Lower Bound
FAS	Face Anti-Spoofing
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
IQR	Interquartile Range
KL	Kullback Liebler
ML	Machine Learning
MCMC	Markov Chain Monte Carlo
NIR	Near Infra Red
PAD	Presentation Attack Detection
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
t-SNE	t-distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection
VAE	Variational Autoencoder

List of Figures

1

Introduction

Ever since the beginning of computers, we have tried to make them learn on their own. In the 1950s, Arthur Samuel popularized the term Machine Learning when he developed a computer program for playing checkers. Machine Learning is a form of Artificial Intelligence (AI) that enables computers to learn from observed data rather than strict programming; The field has slowly progressed forward, ever limited by the hardware of its time.

1.1 Background

The term Deep Learning refers to the stacking of multiple layers of neurons in Artificial Neural Networks (ANN), which can progressively improve their ability through training. ANNs generally require minimal domain-specific programming and can learn complex and abstract patterns through highly non-linear models with millions of parameters. The learning process of networks is either *Supervised* or *Unsupervised* (or a combination), which are useful for different applications.

During training in a supervised setting the ANN has access to a target (correct answer) and improvements to the network is done by comparing the network predictions to targets. Supervised learning is often a more simple task and can yield high accuracy with simple models. A drawback is the amount of annotated data required to train the model which can be expensive or impractical to obtain. A critical weakness for supervised learning is the inability to generalize knowledge to unseen domains. This makes supervised learning unsuitable for anomaly detection because only anomalies represented in the training data will be detected.

Unsupervised learning attempts to cluster and find patterns in unlabeled data. During training the unsupervised methods exhibit self-organization that learns the internal domains of the data as probability densities, in contrast to supervised learning which has access to corrections during training. [1]

Face detection networks have reached a performance level where they are convenient for real-time authentication. Classical authentication such as regular passwords works because it requires an access key that only a verified user would know. Biometric security such as face detection relies on the fact that the access key is

very hard to imitate. This puts a high degree of trust in face detection systems to correctly identify when someone is trying to impersonate the authorized user.

1.1.1 Anti-spoofing

Face Anti-spoofing (FAS) refers to preventing security breaches in face recognition systems by detecting *spoofs* or *Presentation Attacks* (PAs), which means trying to fool the program by imitating a face through a mask, makeup, pictures, or other methods. As facial recognition systems become more popular for authentication, the sophistication of PAs will increase, and with it need for better Presentation Attack Detection (PAD).

As smart cars get more popular and more features are added the requirement for certain safety features becomes desired by customers, and sometimes even required by law. [2] One such feature is drowsiness and distraction detection, which makes sure the driver is focused on driving. These features require robust facial recognition software, which includes some accuracy that the recognized face is a face.

There are many ways of making sure faces are real. One prevalent design is to use neural networks to recognize the faces and imitations of faces. Using an unsupervised model, some performance on seen data is compromised for the hope of better generalization to new domains of spoofs.

The unsupervised approach to anti-spoofing is essentially understanding and estimating the distribution of bonafide face images. Once the true distribution of faces approximated, a PA or spoof can be discerned by recognizing it as anomaly in the bonafide distribution. In some sense this is an anomaly detection problem where spoofs or PAs are the anomaly.

1.1.2 Smart Eye

Smart Eye is a company located in Gothenburg, Sweden. It was founded in 1999 and has since grown into a company with over 100 employees. Their focus is mainly on developing driver monitoring systems for the automotive industry using artificial intelligence.

Main features of Smart Eye's software is the driver facial recognition and fake driver detection modules. By creating a new unsupervised anti-spoofing network, the goal is to supplement the current anti-spoofing solution for improved unseen domain adaptation.

The particular dataset used to test models in this thesis is provided by Smart Eye and captured using near-infrared cameras. The dataset contains images of live faces and different classes of spoofs. Training of ANNs has been done on processing units on Smart Eye servers.

1.2 Problem description

This thesis is concerned with developing and evaluating methods for unsupervised anomaly detection in images, more specifically to detect PAs from bonafide facial images.

The central task is to construct and train ANN models to learn the complex distribution of bonafide images by disentangling features of bonafide facial images. The models will be challenged to predict whether a new image belongs to the seen bonafide domain according to the disentangled representation.

The learning is done by forcing the model to compress and represent each facial image with a lower-dimensional vector containing all information about the original image, then reconstructing the image from that information. The hypothesis is that by studying the compressed image representation and the reconstruction, some anomaly score or other metric can be extracted to indicate domain affiliation.

How efficient can images be compressed using CNNs to still allow for adequate reconstruction?

Will the compressed vector have meaningful representations about the original image which can be used to determine domain affiliation?

With what accuracy can spoofs be singled out by studying the reconstruction error and compressed image vector?

1.2.1 Limitations

The competition for computational resources in a car is often high, even more so with the emergence of smarter cars with more and more features available. Therefore, any ANN models that run real-time inference have to be light, especially when such an ANN is only a subfeature in a facial recognition system. Making a lightweight network that could perform the anti-spoofing task was the first part of the project scope. But to make a computationally efficient high accuracy model, a teacher-student setup was required to condense the learned knowledge. This task was ultimately abandoned due to time.

One other way to speed up inference without compromising model size is to use quantization, which means restricting model weights and activation functions to integers, or some small quantity that is not continuous like floating points. This is something that was decided to be part of potential further research and thus was not used in the final models.

All FAS experiments will be limited to a particular single channel (grayscale) dataset provided by Smart Eye. The dataset has been cropped to center the face and minimize the background in the images. The ANN model will therefore be limited to categorizing PAs on similar datasets and is expected to have poor cross-domain

generalization.

1.2.2 Outline of report

The report will be divided into 6 chapters. Chapter 2 will review central theoretical concepts, establish a Bayesian framework for the models, and outline the fundamental structure of the ANN models used. The following chapters, Methodology, and Results will describe modifications and parameters of the individual models and their efficacy. The findings and results will be discussed in chapter 5 along with candidate model architectures for future research. Chapter 6, Conclusion, will be a summary of the project and report.

1.3 Related work

Face anti-spoofing is getting more attention as face recognition systems becomes wide spread. Most research focuses on supervised and semi-supervised approaches, and when unsupervised learning is used it is often supervised learning combined with unsupervised domain adaptation. Contemporary research on deep face anti-spoofing is summarized in a survey paper by Yu et al. [3].

The superior performance of supervised over unsupervised models has focused research on supervised models. The unsupervised approaches used in prior research has mainly used elements from autoencoders and GAN. A paper by Akcay et al. [4] proposes a fully unsupervised anomaly detector in images using autoencoder design patterns combined with a discriminator.

2

Theory

In this chapter we cover the main theory that is necessary to formulate the unsupervised approach to Face Anti-spoofing. Firstly, the Bayesian Framework will be introduced and the mathematical background of the models will be reviewed. Thereafter, specific Deep Learning architectures such as Variational Autoencoders (VAE) will be contextualized. Lastly, the classification process and associated mathematical theory will be detailed.

Most models in this thesis assume familiarity with how a *posterior probability distribution* is computed from a *prior distribution* and a *likelihood function* by using Bayes' theorem. The prior expresses ones beliefs before any data is observed and the likelihood function describes the probability of observing data given some parameters. Readers whom wish to further acquaint themselves with Bayesian statistics are referred to [5].

2.1 Probabilistic framework

In many machine learning tasks, we are interested in mathematically describing some phenomena or processes and creating models to make future predictions. Probabilistic models are used for understanding the process and to guide automated decision-making. A probabilistic model either uses a conditional probability distribution or a joint probability distribution to capture the uncertainty in the model. The joint probability distributions are the most thorough and capture all higher-order dependencies. [6]

Consider $\mathbf{x}$ as representing an observed random sample from a target process . The true distribution $p_{real}(\mathbf{x})$ or just $p(\mathbf{x})$ will be unknown, but can be approximated with some model $p_\theta(\mathbf{x})$ with parameters θ. The goal is to model $p_\theta(\mathbf{x})$ such that:

$$\mathbf{x} \sim p_\theta(\mathbf{x}). \tag{2.1}$$

The model can be constructed to incorporate knowledge about the data distribution that is known a priori while also adapting to new data. The process of *learning* in machine learning is the iterative search for a value of the parameters θ such that the

probability distribution of the model $p_\theta(\mathbf{x})$ better and better approximates that of the true distribution $p(\mathbf{x})$. The objective is to find a θ such that for any observed $\mathbf{x}$:

$$p_\theta(\mathbf{x}) \approx p(\mathbf{x}) \tag{2.2}$$

2.1.1 Generative modelling

In statistical classification one either uses generative or discriminative modeling. The formal definition is useful, but the dichotomy is inconsistent and falls apart under scrutiny.[7] The discriminative models take a more direct, and arguably simpler approach, to classify the data by emulating the conditional distribution $P(y|\mathbf{x})$ for each class label y. Thus, discriminative models have the sole objective of correctly assigning labels to new observations, given previously seen data.

The generative model, in contrast, studies the joint distribution $P(\mathbf{x}, y)$ and attempts to learn the underlying distributions of the data itself. The generative model is therefore solving a more general problem than is necessary for the classification task which brings higher bias and asymptotic errors [8]. However, knowledge of the underlying distributions can be very valuable when understanding the process which creates the data. Once the joint distribution is found, one can turn it into a discriminator by using Bayes' theorem to predict the possibility of label $\hat{y} \in Y$ for an unseen observation $\hat{x}$.

Knowing the joint distribution also enables us to sample from it to *generate* new data. The learned features and representations of the generative model can be more interpretable and can also easier be generalized to data that arise from different settings. [9] Further, the generative model can be used in an unsupervised manner that the discriminative models by design can not.

The Variational Autoencoder (VAE) which will be introduced later, has been used to learn representations of the data in an unsupervised manner by forcing it to perform the inverse of the feature extraction process, namely to generate the data from the representation. This seemingly redundant task helps disentangle meaningful lower-dimensional representations and learn abstract features that make way for better predictions downstream, especially on domains that are related but not explicitly trained on. Since one of the main concerns of Face Anti-Spoofing is the handling of new domains of spoofs, the unsupervised generative modeling is better suited to tackle the problem.

2.1.2 Maximum likelihood

When some observed data points $\mathcal{D}$ are sampled from the true distribution $p(\mathbf{x})$ we would like our approximated distribution $p_\theta(\mathbf{x})$ to have a high probability of sampling those same data points. By maximizing the probability $p_\theta(\mathcal{D})$ we are maximizing the likelihood of samples $\mathcal{D}$ from the distribution $p_\theta(\mathbf{x})$ with respect to the parameters θ. The likelihood and the logarithm of the likelihood share optimal

parameters, so for practical reasons, we use the log-likelihood instead.

The *maximum log-likelihood* (ML) objective is often used for bayesian modeling and can without loss of generality be converted to a minimization problem of the negative log-likelihood with respect to the parameters θ.

$$-\log p_\theta(\mathcal{D}) = -\log \prod_{x \in \mathcal{D}} p_\theta(\mathbf{x}) = \sum_{x \in \mathcal{D}} -\log p_\theta(\mathbf{x}) \tag{2.3}$$

Here we assume $\mathcal{D} = \{x_i\}_{i=1}^N$ to be sampled from the same, unchanging distribution and are said to be *independently and identically distributed*. Under this assumption, the product of individual probabilities is converted to a sum under the logarithm, which is more computationally tractable.

2.1.3 Latent variable models

Some large models have intermediate variables or representations of the input data before the output is created. These *hidden* intermediate representations are sometimes called *latent variables* and are parts of the model that are inferred from the input rather than observed. A common example of latent variables is the values of hidden layers in a deep neural network.

Modeling $p(\mathbf{x})$ for a high complexity distribution such high-resolution images can be very challenging, especially when the number of observations is limited. By introducing latent variables which transform the complex datapoints $\mathbf{x}$ to a lower-dimensional space, denoted $\mathbf{z}$, we attempt to explain the important features of $\mathbf{x}$ using only $\mathbf{z}$. By attempting to recreate $\mathbf{x}$ from the latent variables, $\mathbf{z}$ acts as a bottleneck through which all the information has to be compressed.

The compressed representation of the data should in theory contain all the information needed to reproduce the original data. This assumption stems from the *manifold hypothesis* that high-dimensional data lie on a lower-dimensional manifold embedded in the higher dimensional space. [10]

For some model with parameters θ we can now define the *prior* distribution of the latent variables $p_\theta(\mathbf{z})$ and the *likelihood* distribution $p_\theta(\mathbf{x}|\mathbf{z})$ that an input $\mathbf{x}$ will give rise to a latent variable $\mathbf{z}$. Further, the prior and the likelihood can be combined to form the joint distribution

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}). \tag{2.4}$$

Note that there are two kinds of parameters in this model, the latent variables $\mathbf{z}$ and the parameters θ. In a Bayesian framework, the model parameters are the parameters of the prior and posterior, which here are the latent variables $\mathbf{z}$.

The parameters θ governs the relationship between input data $\mathbf{x}$ and the latent variables $\mathbf{z}$ through the model p_θ. But since these parameters are fixed under the Bayesian calculations, they can be considered *hyperparameters* in the Bayesian formulation. The hyperparameters θ will be shown to manifest as neural network weights and are not to be confused with neural network hyperparameters such as the learning rate.

The joint distribution is related to the sought after marginal distribution $p_\theta(\mathbf{x})$ by the integral:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \tag{2.5}$$

which lacks analytic solution and is typically intractable to compute for non-trivial likelihood functions $p(\mathbf{x}|\mathbf{z})$ e.g neural networks with non linear hidden layers. [11]

Using Bayes rule we can also define the *posterior* distribution $p(\mathbf{z}|\mathbf{x})$ which expresses the beliefs about the latent variables after observing a data point. The posterior is related to the marginal, prior, and likelihood with:

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})}. \tag{2.6}$$

With the definitions in table 2.1 we can further formulate a Bayesian model with a *generative* and an *inference* component. Generation is when we sample $\mathbf{z}$ from the prior $p(\mathbf{z})$ then relate them to observations $\mathbf{x}$ through the likelihood $p(\mathbf{x}|\mathbf{z})$. Inference is conditioning on data $\mathbf{x}$ and infering latent variables from the posterior $p(\mathbf{z}|\mathbf{x})$. The flowchart of this process is shown in figure 2.1.

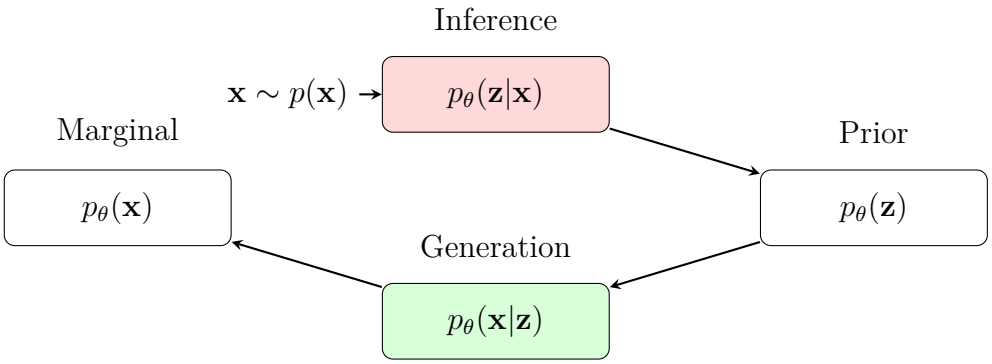


Figure 2.1: Generation (green) starts at the prior for the latent variables and samples $\mathbf{z} \sim p_\theta(\mathbf{z})$ then $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$ to generate a new data point. Inference (red) samples from $\mathbf{x} \sim p(\mathbf{x})$ then $\mathbf{z} \sim p_\theta(\mathbf{z}|\mathbf{x})$ to infer latent variables.

To avoid having to compute the intractable integral for the marginal distribution in 2.5 we can leverage the divided structure of inference and generation. By arbitrarily

Table 2.1: Bayesian Statistics definitions.

Symbol	Description
$\mathbf{z}$	Latent Variable
$\mathbf{x}$	Data
$p_\theta(\mathbf{x})$	Marginal Distribution
$p_\theta(\mathbf{z})$	Prior Distribution
$p_\theta(\mathbf{z}\|\mathbf{x})$	Posterior Distribution
$p_\theta(\mathbf{x}\|\mathbf{z})$	Likelihood Distribution

choosing the prior and likelihood as part of the model, we can reduce the problem to an approximate inference of the posterior. There are different approximate inference techniques, but the one considered in this thesis is variational inference.

2.1.4 Variational inference vs. MCMC

There are two main approaches to the approximate inference of the posterior; *Markov Chain Monte Carlo* (MCMC) sampling, and *variational inference*.

MCMC has been dominant in Bayesian modeling with algorithms such as the *Metropolis-Hastings* and the *Gibbs sampler*. In these algorithms, we construct an ergodic (recurrent) Markov chain with the posterior $p(\mathbf{z}|\mathbf{x})$ as the stationary distribution. We then sample from the chain and approximate the posterior from the collected samples.

For complex models or very large datasets, MCMC algorithms can fail to converge in a reasonable time which makes variational inference a good alternative. The idea behind variational inference is to turn a sampling problem into an optimization problem. First, a family of distributions is chosen for the density of the latent variables, often a mixture of Gaussian is sufficient. Then we find a distribution from that family which minimizes the *Kullback-Leibler divergence* to the exact posterior.

The complexity of the family of distributions used for the latent variables $p(\mathbf{z})$ creates a manageable scope for the optimization problem of approximating the posterior $p(\mathbf{z}|\mathbf{x})$. The idea is to regularize the distribution of the latent variables such that it is sufficiently malleable to approximate the true posterior well while still being simple enough for efficient computation. In short, variational inference does not guarantee reproducing (asymptotically) exact samples from the target density like MCMC but tends to be faster due to the advantage of stochastic optimization. [12]

2.1.5 Kullback-Leibler divergence

To measure the similarity between two distributions we can use the Kullback-Leibler divergence, D_{KL}. It measures how one probability distribution q differs from a reference distribution p in terms of the information provided. The intuition is that observing something probable yields low information, while the knowledge of a rare event occurring yields high information. In some way, the D_{KL} measures the expected surprise (*Shannon information*) from using q as a way to model p.

Since information is inversely related to the probability of an event x, we choose to model information of x with respect to some distribution p as $-\log p(x) = I_p(x)$. The difference in information between the two distributions is, therefore:

$$\Delta I = I_p - I_q = -\log p(x) + \log q(x) = \log\left(\frac{q(x)}{p(x)}\right)$$

Thus, the expectation of the difference in information is the D_{KL}:

$$D_{KL}\Big(q(x)||p(x)\Big) = \mathbb{E}_q[\Delta I] = \int q(x)\log\left(\frac{q(x)}{p(x)}\right)dx \tag{2.7}$$

D_{KL} is asymmetric and is therefore called a divergence rather than a metric, and by noting $\log x \leq x - 1$ we can show that the D_{KL} is non-negative.

$$\begin{aligned} &- D_{KL}\Big(q(x)||p(x)\Big) = \int q(x)\log\left(\frac{p(x)}{q(x)}\right)dx \\ &\leq \int q(x)\Big(\frac{p(x)}{q(x)} - 1\Big)dx = \int q(x)\frac{p(x)}{q(x)}dx - \int q(x)dx = 1 - 1 = 0. \end{aligned} \tag{2.8}$$

2.1.6 Evidence lower bound (ELBO)

When the prior $p_\theta(\mathbf{z})$ and the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ are arbitrarily chosen, the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is computed by an intractable integral. Therefore we further approximate the posterior using approximate variational inference denoted $q_\phi(\mathbf{z}|\mathbf{x})$ with new parameters ϕ.

To express the end goal of maximizing the likelihood in terms of the approximate posterior we use an alternative objective function that acts as a proxy to the ML-objective; the *evidence lower bound* (from the ML objective integral being called model evidence). Remember the ML objective (the marginal distribution) is not dependent on the latent variables, so taking the expectation with respect to $\mathbf{z}$ does not affect the expression. The true posterior given by the model (prior and likelihood) under the data $\mathbf{x}$ is $p_\theta(\mathbf{z}|\mathbf{x})$ and approximated posterior is $q_\phi(\mathbf{z}|\mathbf{x})$ which will be fitted to the true posterior using a neural network.

$$\begin{aligned} \log p_\theta(\mathbf{x}) &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x})] \\ &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \\ &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \\ &= \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]}_{\text{ELBO}} + \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]}_{D_{KL}} \end{aligned} \tag{2.9}$$

Here we have first used Bayes theorem to expand the marginal, then extended the fraction with the approximate posterior. Note that the second term is by definition the KL divergence of q_ϕ with respect to p_θ:

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\Big] = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) \geq 0.$$

The first term in 2.9 is what we refer to as the ELBO:

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\Big] = \mathcal{L}_{\theta,\phi}(\mathbf{x}) \tag{2.10}$$

By solving for the ELBO we get:

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \log p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) \tag{2.11}$$

From this, we can note that the maximization of ELBO achieves two things. It improves the generative model in the sense that the marginal likelihood $p_\theta(\mathbf{x})$ will increase, and at the same time it will minimize the KL divergence between the approximate posterior to the true posterior. [6]

Decomposing $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ and applying Bayes theorem yields:

$$\begin{aligned}
\mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x},\mathbf{z})\Big] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log q_\phi(\mathbf{z}|\mathbf{x})\Big] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{z})\Big] + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log q_\phi(\mathbf{z}|\mathbf{x})\Big] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})}\Big] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))
\end{aligned} \tag{2.12}$$

which can be seen as two terms, one concerning the reconstruction likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ of the data and one as a regularizer on the approximate posterior (D_{KL}). [13] The next section will show that the ELBO allows for stochastic optimization with respect to the model parameters.

2.2 Artificial Neural Networks

Differentiable feed-forward neural networks, or just *networks*, are a particularly flexible and scalable function approximator. *Deep learning* is another name for neural networks with multiple "hidden" layers, which have proven remarkable performance on a wide variety of tasks, one of them being classification. [14]

ANNs will be used through this paper to model the processes described in figure 2.1. Even though the architectures of neural networks differ, they all have in common that they are optimized with respect to some main objective function, often called the *loss function*. The loss we ideally would use is the ML-objective, but as previously explained, proxies to the ML objective such as ELBO are more efficient, stable, and produce the same results.

2.2.1 Stochastic gradient descent

Neural networks and their success relies on the fact that they are differentiable with respect to the parameters θ. The reverse-mode automatic differentiation algorithm proposed by Rumelhart *et al.* in 1988, most commonly known as backpropagation, allows efficient calculation of the value and gradient of the objective function [15]. The gradients can be used to iteratively find local optima of the objective by updating the parameters:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta) \tag{2.13}$$

where $L(\theta)$ is an unbiased estimate of the objective (or loss) function and η is the *learning rate*. The objective can for example be the minimization of the negative ML. The learning rate can be dynamically chosen during experiments with optimization methods such as Adam [16]. In general, the learning rate is high at the beginning of training and decays as the improvements to the objective stagnate.

If we compute the gradients using a portion $\mathcal{M}$ of the dataset, it would be referred to as a *batch* gradient descent. Using large batches yields smoother training at the cost of computational complexity since the operation scales linearly with the batch size N. A method known as *stochastic gradient descent* (SGD) uses randomly sampled batches of size N from the total datapoints $\mathcal{D}$, yielding unbiased (scaled) stochastic gradients: [6]

$$\frac{1}{N_\mathcal{D}} \nabla_\gamma L_\gamma(\mathcal{D}) \approx \frac{1}{N_\mathcal{M}} \sum_{x \in \mathcal{M}} \nabla_\gamma L_\gamma(\mathbf{x}). \tag{2.14}$$

where L_γ is the loss (objective) function with parameters γ. The right-hand side of equation 2.14 is an unbiased estimator of the desired gradients (over the whole dataset) on the left-hand side. Since the noise introduced in the sampling of $\mathcal{M} \in \mathcal{D}$ is unbiased, we can use these gradients to repeatedly update the parameters in the direction of the stochastic gradients and optimize the objective.

2.2.2 Activation functions

The output from a layer in an ANN is usually passed through what is called an *activation function*. Without any activation functions, the mapping from one layer

to another is linear. If we add a nonlinear activation function to the output we can capture non-linear behavior. Some examples of common non-linear activation functions are the ReLU function

$$\text{ReLU}(x) = \begin{cases} x, & \text{for } x \geq 0 \\ 0, & \text{for } x < 0 \end{cases} \tag{2.15}$$

the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.16}$$

and $\tanh(x)$. There are many more non-linear activation functions and each has its advantages and downsides. In some cases, it is not necessary with non-linear activation functions, such as in the last output layer. Layers without activation are called *linear* and simply become linear regression models.

2.2.3 Convolutional Neural Networks (CNN)

Regular ANN tends to struggle with the computational complexity of dealing with image data. Images tend to be fairly high dimensional, with each pixel adding a dimension. Dense layers in ANN connect every neuron with every neuron in the next layer which for high dimensional data creates an even larger number of dimensions for the weights between the layers. [17]

Usually when considering an image, one does not need to consider the image in its entirety, but one usually refers to its local *feature*. The features can for example be horizontal edges, vertical edges, dots, etc. The local features together construct the image.

CNN tries to extract these features and condense the image data to different feature maps. A convolutional layer is comprised of a *kernel* that is swept across the input image and element-wise multiplied with segments, of the same size as the kernel, of the image. This produces a value that is then stored in the output image. The kernels and biases are the trainable parameters of the convolutional layer and each kernel in a layer learns different features. It is common to pass the resulting output through an activation function.

Hyperparameters set during the creation of the layers are the kernel *stride*, filter size, and the number of filters. The stride is the step length the kernel uses to sweep (and multiply) over the image. The filter size commonly referred to as the *receptive field size* [17], describes how large of a portion of the input will have an impact on the output. The number of filters is how many channels the output will have and as described previously will learn different features of the input.

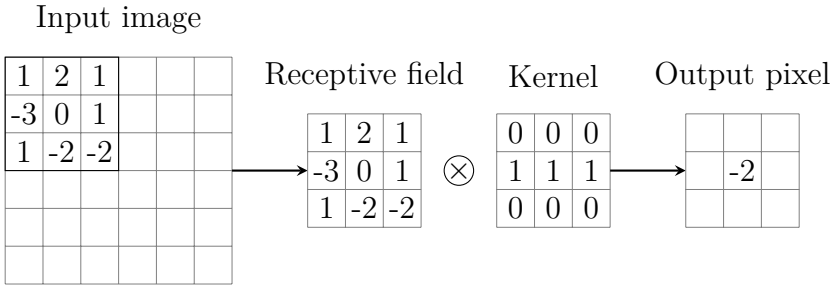


Figure 2.2: A visualization of the convolution operator in a convolutional layer. First, a segment with the size of the receptive field is taken from the input image. Then the segment is multiplied element-wise with the kernel and a sum is taken over the resulting elements and a bias is added to calculate the pixel value in the resulting output image.

Mathematically the convolution layer operator on an image can be described by

$$V_{ij} = g\Big(\sum_{p=1}^{P}\sum_{q=1}^{Q} w_{pq}x_{p+s(i-1),q+s(j-1)} - \theta\Big) \tag{2.17}$$

where V_{ij} is the output image, g the activation function, w_{pq} is the kernel, (P, Q) the receptive field size, s is the stride size, and θ is the bias/threshold [18].

2.2.4 Pooling layers

In a convolutional layer, each pixel in the input influences several pixels in the output due to the construction of the convolutional operator. It can be reasonable to assume that some information in the output is redundant and that the dimensions can be reduced without major information loss. This is the goal of pooling layers.

Pooling is done by sweeping a kernel over the input image and performing a pooling operation on the whole perceptive field. The pooling operator not only mutates but also reduces the dimension of the output. Some common pooling layers are *max pooling* which reduces the receptive field to the maximum value of the field and *average pooling* converts the receptive field to the receptive field average.

Pooling layers are similar to convolutional layers, but what distinguishes the pooling layers are the dimensional reduction of the receptive field and the lack of trainable parameters.

A pooling layer with stride $s = 2$ and receptive field size $(2, 2)$ operating on an image of size 128×128 would produce an output of size 64×64, effectively downsampling the image and reducing the number of pixels by a factor of 4.

2.3 Variational Autoencoder

A variational autoencoder is a type of ANN that provides a probabilistic approach for describing an observation in latent space and modeling the structure shown in figure 2.1. The autoencoder is made up of two major parts, the encoder and the decoder, responsible for inference and generation respectively. The encoder outputs probability distribution for the latent variables $\mathbf{z}$, while the decoder generates a new data point from the latent variables.

Any distribution could be used to model the latent variables, but with quite simple distributions such as a Gaussian distribution (making the *latent space* a mixture of Gaussians), we can model arbitrarily complex marginal distributions. As will be shown, using Gaussian priors for the latent variables enables analytical tricks to provide more efficient optimization steps. By using Gaussian distributions, the encoder generates means μ and variances σ^2 from which the latent variable $\mathbf{z}$ is drawn. This assumes that each $\mathbf{z}$ is independent of the previous $\mathbf{z}$ but also that each value in $\mathbf{z}$ is independent of the others, much like *Mean Field Theory*.

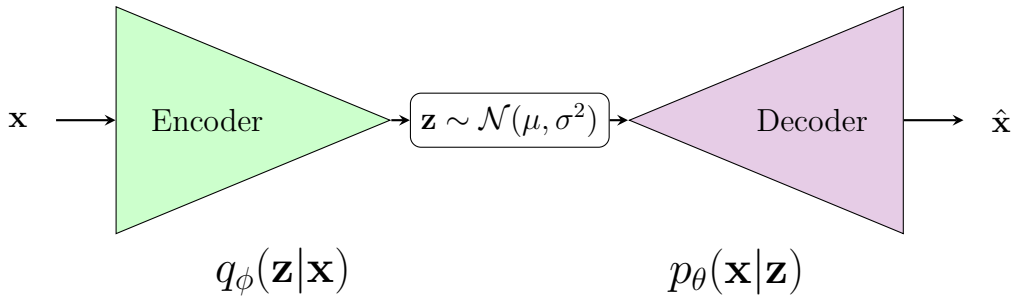


Figure 2.3: The structure of the variational autoencoder. The encoder approximates the posterior by generating means μ and variances σ^2 for each latent dimension. This makes every single $\mathbf{z}$ sampled from a unique Gaussian distribution, making the $\mathbf{z}$-space a mixture of Gaussian distributions. Further, the covariance matrix for $\mathbf{z}$ is diagonal meaning each value of $\mathbf{z}$ is independently sampled. The decoder generates new samples from the latent variables $\mathbf{z}$.

The Bayesian framework of Variational Autoencoders (VAE) assumes that the input data $\mathbf{x}$ is sampled from an unknown probability distribution $p(\mathbf{x})$. The goal is to model the parameterized distribution $p_\theta(\mathbf{x})$ using the encoder and decoder as two connected but independently parameterized models. As mentioned, when the posterior is approximated by variational inference to $q_\phi(\mathbf{z}|\mathbf{x})$ the overall problem is converted into the autoencoder domain and parameterized on parameters in the form of neural network weights θ and ϕ. The model parameters θ are all weights concerning the generative encoder module and ϕ are the *variational* parameters of the inference decoder module.

2.3.1 Reparametrization trick

The VAE seeks to maximize the ELBO $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ objective that acts as a proxy for the ML objective for reasons mentioned in previous sections. The computation of

joint optimization with respect to all parameters is an important attribute of the ELBO and one that requires a small reformulation of the framework in the decoder. Recall that for successful backpropagation the gradients for all parameters have to be computed.

Generator parameters (decoder)

To compute the gradients with respect to the generator parameters θ, the argument is quite straight forward.

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \nabla_\theta \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z}) - \nabla_\theta \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z})]\end{aligned} \tag{2.18}$$

Here the definition of ELBO is taken from equation 2.10, then the gradient can be moved inside the expectation since $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ is not dependent on θ, and for the same reason the second term is removed. To estimate the expectation, samples are drawn from $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ (mini-batch used in SGD) which are then used to compute the approximate (unbiased) batch gradients.

Inference parameters (encoder)

Sampling unbiased estimates of the ELBO gradients w.r.t. the variational parameters ϕ can not be done quite the same since:

$$\begin{aligned}\nabla_\phi \mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &\neq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\nabla_\phi \log p_\theta(\mathbf{x},\mathbf{z}) - \nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x})]\end{aligned} \tag{2.19}$$

The gradients can not be moved inside the expectation since $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ is dependent on ϕ. We would need some alternative way of representing the encoder such that it is differentiable.

By reformulating the stochastic process of inference of the latent variables using the so-called *reparametrization trick* [19] the randomness is outsourced to a separate variable ε independent from $\mathbf{x}$ and ϕ such that:

$$\mathbf{z} = \mu_\phi + \sigma_\phi \odot \varepsilon \tag{2.20}$$

The backpropagation with respect to the variational parameters ϕ through the latent variables can now be done due to $\mathbf{z}$ being deterministic with respect to $\mathbf{x}$ and ϕ. In the illustration figure 2.4, the original formulation of the objective is dependent on the random variable $\mathbf{z}$ which can not be differentiated with respect to ϕ because of the probabilistic nature. The reparametrized form allows differentiation of the objective to be done with respect to the now deterministic dependency on ϕ.

First introduced as "Stochastic Gradient Variational Bayes (SGVB) estimator." by Kingma and Welling 2014 [11].

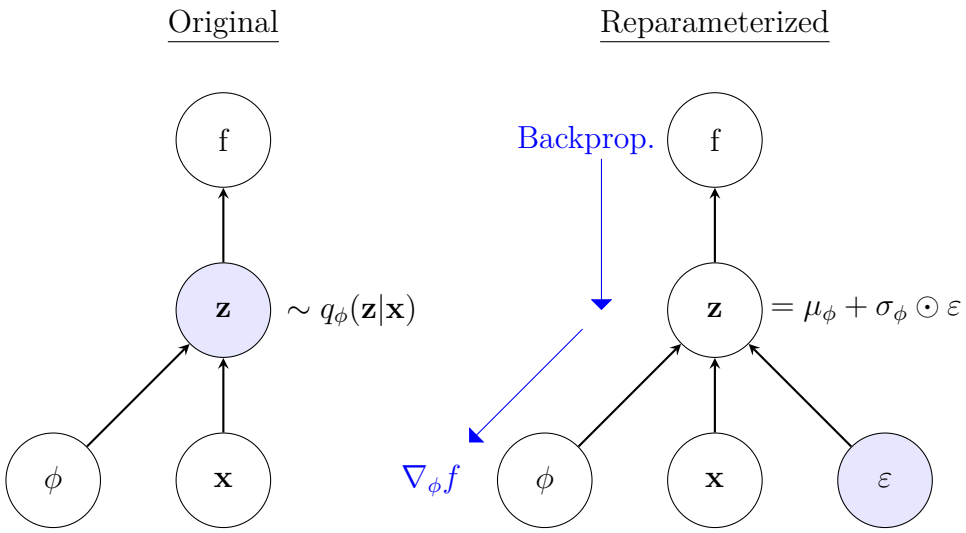


Figure 2.4: Reparametrization of the stochastic calculation of $\mathbf{z}$ to allow for differentiation with respect to ϕ of the objective f, a necessity for backpropagation. The blue nodes are stochastic nodes that can not be backpropagated through.

2.3.2 Closed form loss with gaussian latents

The assumption that the latent variables and the posterior follow Gaussian distributions are valid and can still produce flexible model behavior as mentioned in previous sections. A closed-form solution for the regularization term $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$ can be found under this assumption.

$$\forall z \in \mathbf{z} \qquad p(z) = \frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\Big(- \frac{(x-\mu_p)^2}{2\sigma_p^2}\Big) \tag{2.21}$$

$$\forall z \in \mathbf{z} \qquad q_\phi(z|\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\Big(- \frac{(x-\mu_q)^2}{2\sigma_q^2}\Big) \tag{2.22}$$

where z is an element of $\mathbf{z}$ and μ_p and σ_p are the means and variances generated by model $p(\mathbf{z})$ for each element z in $\mathbf{z}$, extends similarly to the the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$. By the definition of D_{KL} and expanding using logarithmic laws:

$$\begin{aligned} &\forall z \in \mathbf{z} \qquad - D_{KL}(q_\phi(z|\mathbf{x})||p(\mathbf{z})) = \frac{1}{\sqrt{2\pi\sigma_q^2}} \\ &\int \exp\Big(- \frac{(x-\mu_q)^2}{2\sigma_q^2}\Big)\Big(- \log\Big(\frac{\sigma_q}{\sigma_p}\Big) - \frac{(x-\mu_p)^2}{2\sigma_p^2} + \frac{(x-\mu_q)^2}{2\sigma_q^2}\Big) \end{aligned} \tag{2.23}$$

which can be expressed as an expectation with respect to with respect to q:

$$\begin{aligned}
&\forall z \in \mathbf{z} \qquad - D_{KL}(q_\phi(z|\mathbf{x})||p(\mathbf{z})) = \\
&= \mathbb{E}_q\Big[\log\big(\frac{\sigma_q}{\sigma_p}\big) - \frac{(x-\mu_p)^2}{2\sigma_p^2} + \frac{(x-\mu_q)^2}{2\sigma_q^2}\Big] \\
&= \log\big(\frac{\sigma_q}{\sigma_p}\big) - \frac{1}{2\sigma_p^2}\mathbb{E}_q\Big[(x-\mu_p)^2\Big] + \frac{1}{2\sigma_q^2}\underbrace{\mathbb{E}_q\Big[(x-\mu_q)^2\Big]}_{\sigma_q^2} \\
&= \log\big(\frac{\sigma_q}{\sigma_p}\big) - \frac{1}{2\sigma_p^2}\mathbb{E}_q\Big[(x-\mu_q+\mu_q-\mu_p)^2\Big] + \frac{1}{2}
\end{aligned} \tag{2.24}$$

which can be furthers simplified by rewriting the term in the expectation as a binomial square of $x - \mu_q$ and $\mu_q - \mu_p$ and using the definition of σ_q^2.

$$\begin{aligned}
&\forall z \in \mathbf{z} \qquad - D_{KL}(q_\phi(z|\mathbf{x})||p(\mathbf{z})) \\
&= \log\big(\frac{\sigma_q}{\sigma_p}\big) - \frac{\sigma_q^2 + (\mu_q-\mu_p)^2}{2\sigma_p^2} + \frac{1}{2} \\
\text{when:} \quad &\sigma_p = 1, \qquad \mu_p = 0 \\
&= \frac{1}{2}\Big[1 + \log(\sigma_q^2) - \sigma_q^2 - \mu_q^2\Big]
\end{aligned} \tag{2.25}$$

which is a closed expression for the Kullback-Leibler term used in ELBO. In practice this analytical expression is used to speed up computations.

2.4 Generative Adversarial Networks

Generative Adversarial Networks were first proposed in 2014 [20] with the idea of two agents competing against each other in an imitation game. The *generator* produces novel images imitating some target dataset, while the *discriminator* tries to distinguish which images are generated. The two models, the generator and the discriminator are adversarial during training with incompatible goals, thus their contest is a zero-sum game where the improvement of one comes at the expense of the other.

The core idea is to use the discriminator classifications to train the generator and thus avoiding the need for outside labels. Since the generator is only trained to deceive the discriminator, the training can be done unsupervised.

2.5 Anomaly detection

When the data distribution is adequately approximated, some method or process has to be applied to classify samples as bonafide or spoof.

There are two main areas where the spoof detection can be made, the first is looking at the location in latent space, and the other is looking at the reconstruction of the image. For example, the latent space can be gitted to a *Gaussian mixture model* (GMM) or the reconstructed image can be analyzed in terms of reconstruction error or by a *discriminator* network.

2.5.1 GMM

Every input image is given a coordinate (or vector) in the latent space, the combined locations for these embeddings of all the live images can be used to model the latent distribution of live images using a GMM model.

Mixture models are used to represent subpopulations within a larger population using a mixture of probability densities. In this work, only mixtures of Gaussian distributions are concerned, hence GMM.

The GMM algorithm only has access to the number of subpopulations and attempts to find subpopulations with Gaussian shapes in the latent space, and through them represent the whole population. As the number of subpopulations increases, GMMs can model more and more complex distributions and at the limit, each observation is its subpopulation.

For new observations, GMMs calculate the probability of the sample belonging to each of the individual modeled subpopulations, and through those probabilities, a spoof detector can be constructed. The spoof detector will have a confidence measure relating to the ratio of the probabilities of each subpopulation.

In this work, GMMs will be used on the latent variables $\mathbf{z}$ in the hope that two (or more) subpopulations will emerge containing bonafide samples in one and different kinds of spoof attacks in the others.

2.5.2 Discriminator

A discriminator network is a classifier used in adversarial training that distinguishes real from fake. A discriminator can be used in adversarial training where a generator and discriminator compete against each in a zero-sum game. The generator creates images resembling the target distribution and tries to fool the discriminator.

In an adversarial training setting, the goal of the discriminator is often just to train the generator, and when training is complete, the discriminator is discarded and the generator is the final product. The Generative Adversarial Network (GAN) is a simple way of utilizing adversarial training. Though the discriminator primarily is a tool to improve training, it could in theory be used as a spoof detector module once training is done.

2.5.3 Anomaly score by reconstruction error

A very straightforward approach to classifying spoofs is to give each input image an anomaly score depending on how well the image is reconstructed. A generator network will be better at reconstructing images that are trained on, which in this case is live images. When a spoof (anomalous) image is presented and reconstructed, that reconstruction could be worse since the network is not trained for that task. The ℓ_2 norm is used in this thesis for anomaly score based on reconstruction error.

3

Methodology

This chapter presents the details of the specific models used in the thesis, as well as the preprocessing of the data. The details include the training procedure, hyperparameters and evaluation of the individual networks.

3.1 Data acquisition

The dataset used is collected by Smart Eye by recording subjects and objects in an environment that simulates the inside of a car. All videos were filmed in the Near Infra Red (NIR) spectrum of light, a spectrum invisible to humans, as to not disturb the driver and to reduce the influence of daylight ambience. The video data was sliced into image frames at different times and these images served as the input for our networks.

One of our goals was to evaluate the potential of using different unsupervised methods for Presentation Attack Detection (PAD) of which the objective is to recognize whether the subject in the recording is a real person. This essentially means the network does not have access to the image label (live/spoof) during training and is prompted to find intrinsic groupings in the data.

Another goal was to explore the possibility of only training the networks using live data and use the spoofs exclusively for testing purposes. This is because there is generally an abundance of data from live subjects and it is much easier to collect. There are also multiple domains of spoofs, and creating spoof data from all possible spoof domains is impossible, so having a network that can learn without seeing spoofs is a definitive advantage. We also restricted ourselves to only use data from a single camera as input to the networks. This would make an eventual commercial product easier and cheaper to implement, and the inference less computationally heavy.

3.1.1 Data subsets

The dataset contains grayscale images of size 128×128 and auxiliary data such as head rotation angles, live/spoof labels etc. The auxiliary data was used for filtering the data into new datasets, but never used during training and inference. Further,

the dataset is divided into subject IDs and each ID is divided into different recording sessions. Each subject ID may or may not contain data from the two spoof categories and the live category. *mask* and *paper prints*.

Taking our goal, to evaluate the potential of only training on live data, into account we decided to create the follow subsets of the data.

- **Training data**: Contains subject IDs of which there are recordings of either mask, paper prints or both.
- **Test, live only**: Contains subject IDs of which there are only live recordings. Because of this there are no common IDs in this set and any other set.
- **Test, seen paper**: Contains subject IDs of which there are both recordings of live and paper prints. All IDs in this set are also contained in the training set.
- **Test, unseen paper**: Contains subject IDs, not contained in the training set, of which there are recordings of paper prints.
- **Test, seen mask**: Similar to the seen paper set, but with recordings of masks.
- **Test, unseen mask**: Similar to the unseen paper set, but with recordings of masks.

It is worth pointing out that none of the latex masks are made to look similar to any of the live subjects and it can therefore be argued that the distinction between *seen mask* and *unseen mask* is unnecessary since none portrays individuals in the training data. The unseen mask dataset contains some images of mannequins with masks, and for that reason the subset division was kept as provided.

Training & Test, live only sample

Figure 3.1: Samples from the training data. Due to privacy concerns we will not include data from the subset ”Test, live only” directly. It is however very similar to the training data since it is composed of live only images.

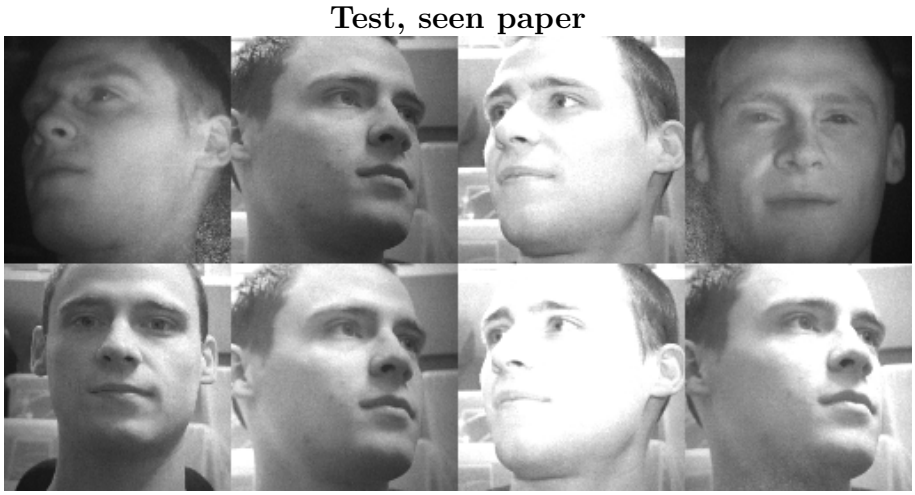


Figure 3.2: Samples from the subset "Test, seen paper".

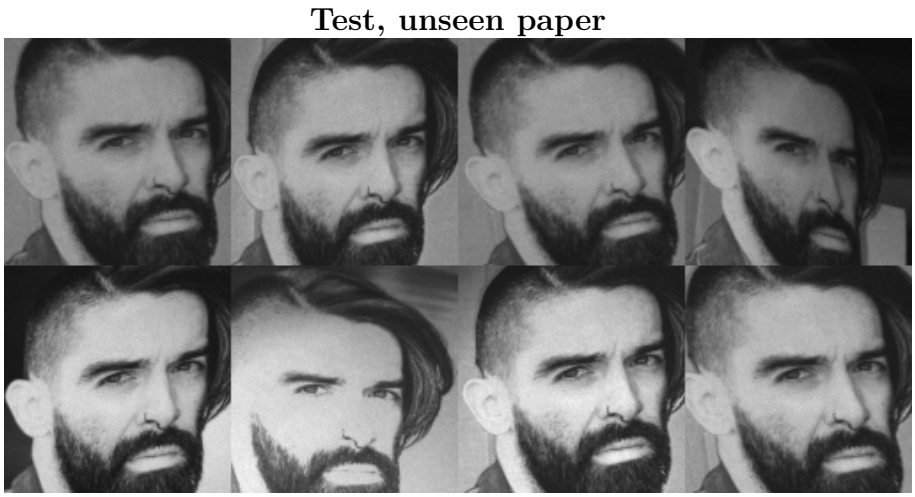


Figure 3.3: Samples from the subset "Test, unseen paper".

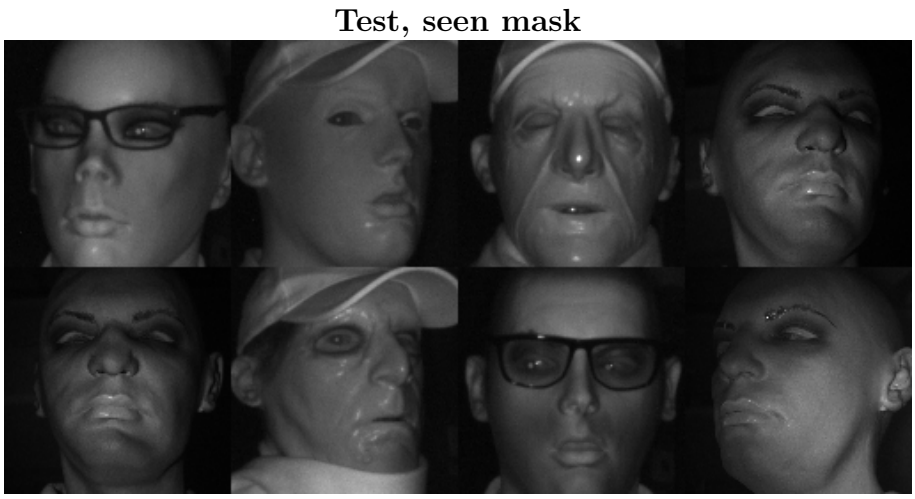


Figure 3.4: Samples from the subset "Test, seen mask".

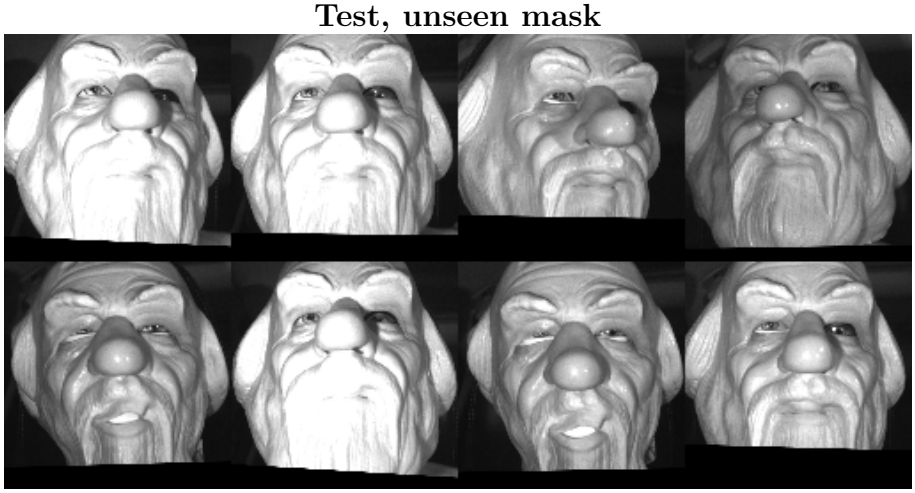


Figure 3.5: Samples from a single ID from the subset ”Test, unseen mask”.

3.2 Preprocessing

Some experiments, especially the first ones, used all the available image data. The full dataset contains images of subjects from a variety of head pose rotations and thus requires the model to capture a larger data distribution. The large variance may make it harder for the network to learn the data distribution so later experiments used a filtered dataset where head poses were limited to a certain angle range. Some session outliers were also filtered out to stabilize the training.

3.2.1 Head pose filtering

Each observation has a head pose in three dimensions associated with it. These can be though of as the head rotation from "side to side", "up and down", and "head tilt". To limit the size of the training set and the target distribution, only faces with a general "forward" facing direction were trained on. The angle ranged used were:

min angle	axis	max angle
-10°	x	10°
-10°	y	10°
-10°	z	10°

Table 3.1: Head position angle filtering in degrees. The x, y, z axes correspond to "side to side", "up and down", and "tilt" respectively.

Head position filtering reduced the dataset of 5 928 000 images down to 596 000, about 10% of the original. The the head position ranges were chosen to be small while still maintaining a large dataset.

3.2.2 Session outlier filtering

As described previously the data was divided into different recording sessions with all images from a session originating from the same video. Some images might be considered outliers in a session, for example, a frame where the lighting is significantly different or a hand covers the face etc. And since the dataset is to large for manual filtration it was decided that images with an mean pixel value outside of 1 IQR (Interquartile Range) were potential outliers, and was removed. The threshold of 1 IQR is quite low, and presumably some seemingly normal images were removed. But because of the abundancy of training data, uniformity of the dataset was valued over size. IQR filtration accounted for a 7% reduction in dataset size down to 557 000 images.

3.2.3 Augmentations

Since the data was generated by slicing raw video data they are part of a sequence and some images might be very similar to each other, especially if the subject has not moved significantly between frames. This can make the dataset contain virtual duplicates and be too uniform. Data augmentations combats this by adding transformations such as rotation, reflections, scaling etc. to the data and increases the variance. Augmentations were implemented in some experiments in the hope of boosting performance.

(a) Unaugmented images **(b)** Augmented images

Figure 3.6: The effect of all augmentations. (a) shows an unaugmented subset of the data while (b) shows a different subset with augmentations. All images from the same subject ID.

One augmentation that was always performed was *standard scaling*, where we scale the images to have a mean of $\mu_{image} = 0.5$, standard deviation of $\sigma_{image} = 0.25$, and the value range clipped to the range $X \in [0, 1]$. The reason for this augmentation was to make training easier, since differences in lighting during the recording session

won't have as much of an impact on the augmented image. The range was set so the network output range will be the same as the input range.

3.3 Deep Convolutional Variational Autoencoder (DCVAE)

This network is a standard sequential convolutional network, except for the stochastic latent interface between the encoder and the decoder. The number of filters are growing towards the encoder/decoder interface, while the image size is shrinking. Inspiration for these specific hyperparameters came from a previous project at Smart Eye where it showed some success at image decoding.

Encoder	Act.	Output shape
Image input	-	$1 \times 128 \times 128$
Conv 3×3	ReLU	$32 \times 128 \times 128$
Conv 3×3	ReLU	$64 \times 128 \times 128$
MaxPool	-	$64 \times 64 \times 64$
Conv 3×3	ReLU	$128 \times 64 \times 64$
MaxPool	-	$128 \times 32 \times 32$
Conv 3×3	ReLU	$256 \times 32 \times 32$
MaxPool	-	$256 \times 16 \times 16$
Conv 3×3	ReLU	$512 \times 16 \times 16$
MaxPool	-	$512 \times 8 \times 8$
Fully connected	-	$1024 \times 1 \times 1$
Sampling	-	$512 \times 1 \times 1$

Decoder	Act.	Output shape
Latent input	-	$512 \times 1 \times 1$
Reshape	-	$8 \times 8 \times 8$
Upsampling	-	$8 \times 16 \times 16$
Conv 3×3	ReLU	$512 \times 16 \times 16$
Upsampling	-	$512 \times 32 \times 32$
Conv 3×3	ReLU	$256 \times 32 \times 32$
Upsampling	-	$256 \times 64 \times 64$
Conv 3×3	ReLU	$128 \times 64 \times 64$
Upsampling	-	$128 \times 128 \times 128$
Conv 3×3	ReLU	$64 \times 128 \times 128$
Conv 3×3	ReLU	$32 \times 128 \times 128$
Conv 1×1	Sigmoid	$1 \times 128 \times 128$

Table 3.2: Architecture of the net we chose to call "Deep Convolutional Variational Autoencoder".

Evidence Lower Bound (ELBO) loss

The VAE models use the probability framework described in section 2.3 and the goal during training is to maximize the ELBO. However, since the optimizer of the networks use SGD we need to find an objective function to minimize. To do this, just change the signs in the ELBO and define the ELBO *loss* as

$$\mathcal{L}_{ELBO} = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) = \mathcal{L}_{rec} + \mathcal{L}_{KL}, \tag{3.1}$$

where $\mathcal{L}_{rec} = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ and $\mathcal{L}_{KL} = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$.

Reconstruction loss

The term $\mathcal{L}_{rec}$ is called the reconstruction loss, since the more similar the output of the VAE model is to the inpus, the lower the reconstruction loss. We can approximate the expected value in the term by taking the average over a sample

$$\mathcal{L}_{rec} = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \approx -\log p_\theta(\mathbf{x}|\mathbf{z}) \tag{3.2}$$

It is then calculated by taking the binary cross entropy between the original image, X, and the reconstructed image, $\hat{X}$, and averaging over the entire image.

$$\mathcal{L}_{rec} = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}\text{BinaryCrossEntropy}(X_{ij}, \hat{X}_{i,j}), \tag{3.3}$$

where H, W are the height and width of the image.

Kullback-Liebler loss

This term, $\mathcal{L}_{KL}$, is calculated as follows

$$\mathcal{L}_{KL} = -\frac{1}{2}\sum_{i=1}^{N}(1 + {}^i\mathbf{z}_{\log\sigma^2} + {}^i\mathbf{z}_\mu^2 - \exp({}^i\mathbf{z}_{\log\sigma^2})). \tag{3.4}$$

The terms ${}^i\mathbf{z}_\mu$, ${}^i\mathbf{z}_{\log\sigma^2}$ are the mean and log-variance of a latent sample in each dimension and are outputs of the two layers prior to the latent output.

3.4 ResNet Variational Autoencoder (ResVAE)

We saw some indications from early experiments that more filters yielded better reconstructions. One way to increase the number of filters is to use a deeper model with more convolutional layers. However, by adding more layers one may encounter a problem known as *vanishing gradients* [21], where the weight update from layers closer to the input becomes small compared to the layers closer to the output. One way to combat this is to use *residual networks* [22] that add skip connections between layers. This effectively makes part of the network not as deep and can speed up the learning.

However, since the number of channels generally differ between convolutional layers used in our models, we add a convolutional layer at each skip connection to match the number of channels before adding. Figure 3.7 shows the residual blocks (ResBlocks) we used. This layout took inspiration from another thesis done at Smart Eye [23].

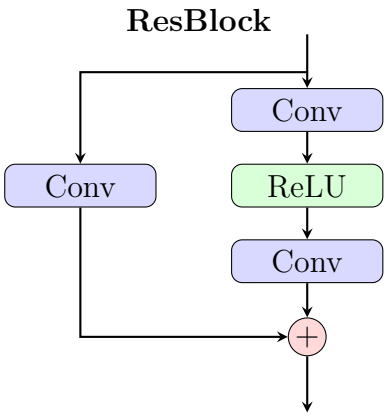


Figure 3.7: Layer structure of a residual block used in this paper. The output from the connection, with two convolutional layers and a ReLU activation function, is added with the output from the skip connection, that only contains one convolutional layer. The convolution at the skip connection is done so the number of channels match when adding the outputs. The skip connection allows for a shorter path during backpropagation which mitigates the problem with vanishing gradients.

Encoder	Act.	Output shape
Image input	-	$1 \times 128 \times 128$
Conv 3×3	-	$32 \times 128 \times 128$
ResBlock 3×3	-	$64 \times 128 \times 128$
AvgPool	-	$64 \times 64 \times 64$
ResBlock 3×3	-	$128 \times 64 \times 64$
AvgPool	-	$128 \times 32 \times 32$
ResBlock 3×3	-	$256 \times 32 \times 32$
AvgPool	-	$256 \times 16 \times 16$
ResBlock 3×3	-	$512 \times 16 \times 16$
AvgPool	-	$512 \times 8 \times 8$
ResBlock 3×3	-	$512 \times 8 \times 8$
AvgPool	-	$512 \times 4 \times 4$
Fully connected	-	$1024 \times 1 \times 1$
Sampling	-	$512 \times 1 \times 1$

Decoder	Act.	Output shape
Latent input	-	$512 \times 1 \times 1$
Reshape	-	$32 \times 4 \times 4$
Upsampling	-	$32 \times 8 \times 8$
ResBlock 3×3	-	$512 \times 8 \times 8$
Upsampling	-	$512 \times 16 \times 16$
ResBlock 3×3	-	$256 \times 16 \times 16$
Upsampling	-	$256 \times 32 \times 32$
ResBlock 3×3	-	$256 \times 32 \times 32$
Upsampling	-	$128 \times 64 \times 64$
ResBlock 3×3	-	$64 \times 64 \times 64$
Upsampling	-	$64 \times 128 \times 128$
ResBlock 3×3	LReLU	$32 \times 128 \times 128$
Conv 1×1	Sigmoid	$1 \times 128 \times 128$

Table 3.3: Architecture of the ResNet used. Notice that the last ResBlock uses LeakyReLU output activation.

ELBO loss

This model is also a VAE and thus uses the same losses as the Deep Convolutional Variational Autoencoder.

3.5 GANomaly

The architecture proposed by Akcay et al. [4] features a deep encoder-decoder structure coupled with a discriminator and adversarial training. The architecture includes

a classic encoder-decoder to produce latent representations and image reconstruction, a second encoder is added after the decoder to produce representations of the reconstructed images. A discriminator network is connected to the input image and the reconstructed image in order to penalize bad reconstruction.

Like the name GANomaly might suggest, the approach does not follow a conventional GAN architecture and is strongly influenced by the autoencoder pipeline. In figure 3.8 a flowchart of the GANomaly architecture is shown containing an encoder-decoder-encoder a discriminator adversary.

Prior work on adversarial autoencoders and GANs shows promising results in anomaly detection problems.[24] By adding the second encoder module to the generator the hope is to further improve the latent representation by imposing a loss on the latent representation of the input to the generated input. The discriminator is trained to output a probability measurement of an image authenticity and through an adversarial loss the generator is further incentivized reconstruct better images. The trained GANomaly network is hypothesized to have a noticeable difference in reconstruction proficiency between bonafide and spoof images through these added modules.

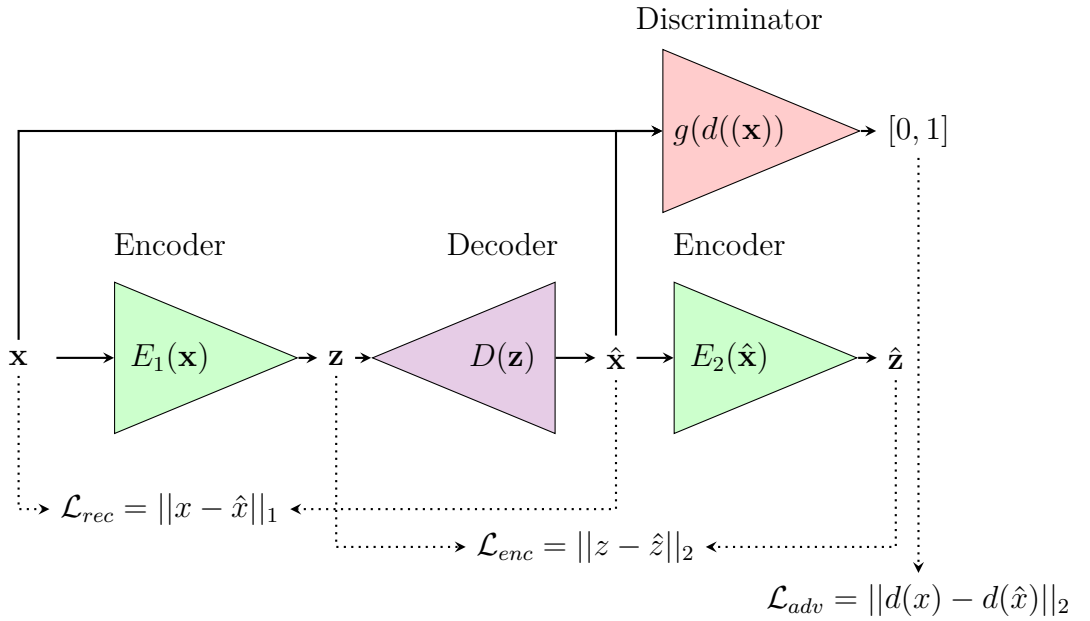


Figure 3.8: The GANomaly architecture. The input data is fed through through the encoder E_1 then the decoder D to generate latent variables and a new generated image, comprising the *generator*. The generator is primarily trained using the reconstruction loss $\mathcal{L}_{rec}$. The generated image is passed through a second encoder to observe what latent features remain in the generated image, this is regulated through the encoder loss $\mathcal{L}_{enc}$. The discriminator is denoted $g(d(.))$ and is trained through the adversarial loss $\mathcal{L}_{adv}$ that uses the last layer of the decoder d, and skips the softmax layer g that produces the spoof probabilities.

Encoder	Act.	Output shape
Image input	-	$1 \times 128 \times 128$
Conv 4×4	L-ReLU	$128 \times 64 \times 64$
Batch Norm.	-	$128 \times 64 \times 64$
Conv 4×4	L-ReLU	$256 \times 32 \times 64$
Batch Norm.	-	$256 \times 32 \times 64$
Conv 4×4	L-ReLU	$512 \times 16 \times 16$
Batch Norm.	-	$512 \times 16 \times 16$
Conv 4×4	L-ReLU	$1024 \times 8 \times 8$
Batch Norm.	-	$1024 \times 8 \times 8$
Conv 4×4	L-ReLU	$2048 \times 4 \times 4$
Batch Norm.	-	$2048 \times 4 \times 4$
Conv 4×4 (s:1)	L-ReLU	$512 \times 4 \times 4$
Fully connected	-	$512 \times 1 \times 1$

Decoder	Act.	Output shape
Latent input	-	$512 \times 1 \times 1$
Deconv 4×4 (s:1)	ReLU	$2048 \times 4 \times 4$
Batch Norm.	-	$2048 \times 4 \times 4$
Deconv 4×4	ReLU	$1024 \times 8 \times 8$
Batch Norm.	-	$1024 \times 8 \times 8$
Deconv 4×4	ReLU	$512 \times 16 \times 16$
Batch Norm.	-	$512 \times 16 \times 16$
Deconv 4×4	ReLU	$256 \times 32 \times 32$
Batch Norm.	-	$256 \times 32 \times 32$
Deconv 4×4	ReLU	$128 \times 64 \times 64$
Batch Norm.	-	$128 \times 64 \times 64$
Deconv 3×3	Tanh	$1 \times 128 \times 128$

Table 3.4: The details of the GANomaly architecture used in this thesis, notice the LeakyRELU and Tanh activation functions. The kernel size is 4 with a standard stride (step length) of 2, this is what reduces image dimensions in each layers. It is explicitly stated if the stride is changed to 1 for a particular layer. The discriminator has the same configuration as the encoders with the exception of output dimension, which is 1 for the discriminator.

3.5.1 GANomaly losses

The three loss functions used to optimize the GANomaly architecture are shown in figure 3.8 and are all through different means concerned with improving reconstruction of the target distribution.

Reconstruction loss

The classical loss function used by autoencoders is the reconstruction loss $\mathcal{L}_{rec}$. It aims to simply penalize the difference between the input image and the image reconstructed from the latent variables. There is a debate on whether linear or quadratic penalty is to be preferred in image processing with ℓ_1 or ℓ_2 norms, though some studies show less blurry reconstruction using the ℓ_1-norm which will be used here. [25]

$$\mathcal{L}_{rec} = ||\mathbf{x} - D(E_1(\mathbf{x}))||_1 = ||\mathbf{x} - \hat{\mathbf{x}}||_1 \tag{3.5}$$

The loss is taken and averaged over a whole batch where $\mathbf{x}$ is the input and $\hat{\mathbf{x}}$ is the reconstructed image by the generator.

Encoder loss

The purpose of the second encoder and the loss function connected to the latent variables is improve the latent variable embedding. The encoder loss is defined as the ℓ_2 distance between the latent vectors of the input and the generated image.

$$\mathcal{L}_{enc} = ||E_1(\mathbf{x}) - E_2(D(E_1(\mathbf{x})))||_2 = ||\mathbf{z} - \hat{\mathbf{z}}||_2 \tag{3.6}$$

This loss will enforce the capacity of the generator to reconstruct images with the same contextual features (latent representation) as the original image. The distance in feature space between the input and generated images will be minimized for the target distribution only and will not extend to anomalous samples. [4]

Adversarial loss

The discriminator module in figure 3.8 takes an image $\mathbf{x}$ and outputs a probability that the image is generated. The discriminator can be further divided into two functions:

- d: outputs the representation of the input $\mathbf{x}$ from the last layer in the discriminator
- g: softmax layer to convert the last layer into a probability

where the full discriminator $\text{Disc}(\mathbf{x}) = g(d(\mathbf{x})) \in [0, 1]$.

The adversarial loss measures the ℓ_2 distance in the latent space of the last layer the discriminator $d(x)$ between the input and the reconstruction. This is in contrast to the common approach of using the cross-entropy loss of the discriminator output $g(d(\mathbf{x}))$ as adversarial loss.

$$\mathcal{L}_{adv} = ||d(\mathbf{x}) - d(\hat{\mathbf{x}})||_2 \tag{3.7}$$

The total loss function is achieved by adding the losses with weighting:

$$\mathcal{L} = w_{rec}\mathcal{L}_{rec} + w_{enc}\mathcal{L}_{enc} + w_{adv}\mathcal{L}_{adv}. \tag{3.8}$$

3.5.2 Anomaly detection

During testing, GANomaly can classify spoofs according to their reconstruction error. The anomaly detector calculates the average reconstruction error using ℓ_2 norms and ranks them from highest to lowest. Note that the reconstruction error used in training uses the ℓ_1 norm and not ℓ_2 on which it is evaluated. The two norms share optima (where the images are equal) with the difference that the ℓ_2 norm has stronger outlier penalty.

The spoof threshold can be adjusted to the desired number of spoofs to produce precision-recall curves or other desired metrics. A global spoof threshold for maximum accuracy can in theory be found, but such a threshold will depend strongly on the nature of the test sets and the spoof ratio. No global threshold will be sought

for evaluation of this architecture, instead metrics will be presented for varying thresholds.

3.6 Training parameters

We tried a range of different training parameters to try and improve learning and to generate better reconstructions. The parameters varied between experiments, so detailed information about parameters the reader is referred to the result section for that experiment. This section contains explanations of the parameters that were varied.

3.6.1 Training length

After some initial experiments with each model tried to train them for different number of epochs. The parameters that decides how long an epoch is are the *batch size* and *batches per epoch*. Since we had a large dataset with high similarity between images the batch size and batches per epoch were set such that each epoch contained roughly 10% of the total images.

3.6.2 Optimizer

We decided to use the **Adam** optimizer [16] for the stochastic gradient descent. Some of the first experiments used a constant learning rate for the optimizer, but later experiments took inspiration from a previous thesis project at Smart Eye [23] and used a constant learning rate for the first half of the training, while later switching to a linearly decaying learning rate, that started at the constant rate and ended at close to zero.

3.6.3 Data augmentation

As mentioned previously, we always used augmentations in the form of standard scaling. Later experiments also used additional augmentations to get the network to hopefully generalize better. Table 3.5 shows the full set of augmentations and their frequencies. Specific experiments augmentations will be listed in the Results section.

Full augmentation set

Augmentation	Frequency
Standard scaling	1.0
Random brightness	1.0
Random Gaussian noise	0.47
Random contrast	0.17
Random cutout	0.51
Random rotation	0.3
Random translation	0.3
Random zoom	0.3
Right left flip	0.5

Table 3.5: Full set of augmentations used in the experiments and their frequency when used.

3.6.4 Loss weighting

The losses determines *how* the networks will learn. As an example, for the VAE networks, the $\mathcal{L}_{rec}$ puts emphasis on the ability to reconstruct the original image, while the $\mathcal{L}_{KL}$ determines the behaviour of the latent representation. It is possible to weigh each loss differently in order to stimulate the desired behaviour. A higher weight on the $\mathcal{L}_{rec}$ makes the reconstructions better while a higher weight on the $\mathcal{L}_{KL}$ makes the network better at generalizing. In general a balance between these two terms must be struck for the network to be good at both generalizing and to produce a faithful reconstruction. [26]

We mainly used the same weights for the $\mathcal{L}_{rec}$ and $\mathcal{L}_{KL}$ while training the VAE networks. If we used different weights it will be stated in the results section for that experiment.

For the GANomaly achitecture, no sweep over loss weighting was done for improved performance. Both the adversarial and encoder losses were weighted with unity and the reconstruction loss was varied.

3.7 Evaluation

Model evaluation was done by examining at the latent space and the error of the reconstructed images for signals that separated the live class from the spoof classes. The models themselves does not explicitly output live or spoof label as that would require a global threshold for the boundary between the live and spoof domains, and such a task requires further analysis and is itself quite a substantial problem. Instead, metrics are presented for a varying threshold.

For the VAE models we looked at the the latent output for each subset of the full dataset. Scatter plots were created of the first principal components to see separation in latent space. More intricate algorithms such as UMAP [27] was also

used in the hope of finding more highly correlated shapes in latent space. UMAP is a *deterministic* dimension reduction algorithm to unravel high dimensional spaces for clustering formations, it is similar to the stochastic t-SNE [28].

Gaussian mixture models were used on the latent output of the *training data* to see if they could detect a clustering. A number of mixtures $\{1, 2, 4, 8, 16, 32, 64, 128\}$ were tried to fit the data, which subsequently were used to measure log-likelihoods of the different datasets for possible separation.

The GANomaly architecture used anomaly scores calculated from the ℓ_2 norm of the reconstructed images for signals of anomalous images which were unrecognized by the generator (and thus reconstructed worse). The anomaly scores were used to classify spoofs, and Precision-Recall curves were shown for varying thresholds. Receiver Operating Characteristics were also created as complements to the Precision-Recall curves, despite claims that ROC-curves are less informative in evaluating binary classifiers on imbalanced data. According to Saito and Rehmsmeier, ROC curves can be visually deceptive, and wrong conclusions can be reached about the classification reliability. [29]

Further the loss distribution of the $\mathcal{L}_{rec}$, $\mathcal{L}_{KL}$, $\mathcal{L}_{ELBO}$ losses was analyzed to see if a difference between live and spoof datasets could be found.

4

Results

This chapter highlights the selected model configurations and their performance. Evaluations of the results and samples of reconstructed images will be shown along with analysis of desired model attributes.

4.1 DCVAE

4.1.1 Baseline training

This experiment can be considered to be our first iteration where an as simple as possible set of hyperparameters were used. No filtering or augmentations were used on the data during training.

Augmentation
Standardization

Table 4.1: The only augmentation used was standardization of the input.

Hyperparameter	**Value**
batches per epoch	10 000
epochs	15
batch size	64
$\mathcal{L}_{KL}$ weight w_{KL}	1.0
$\mathcal{L}_{rec}$ weight w_{rec}	1.0
Learning rate	0.0001
Adam optimizer β_1	0.9
Adam optimizer β_2	0.99

Table 4.2: Hyperparameters used for this experiment. The parameters were chosen to make the training as simple as possible while still allowing the model to learn efficiently.

4.1.1.1 Loss outputs

Some form of separation signal can be seen for some of the spoof subsets in the loss outputs. The signal is however small and the distributions overlap between subsets to a high degree.

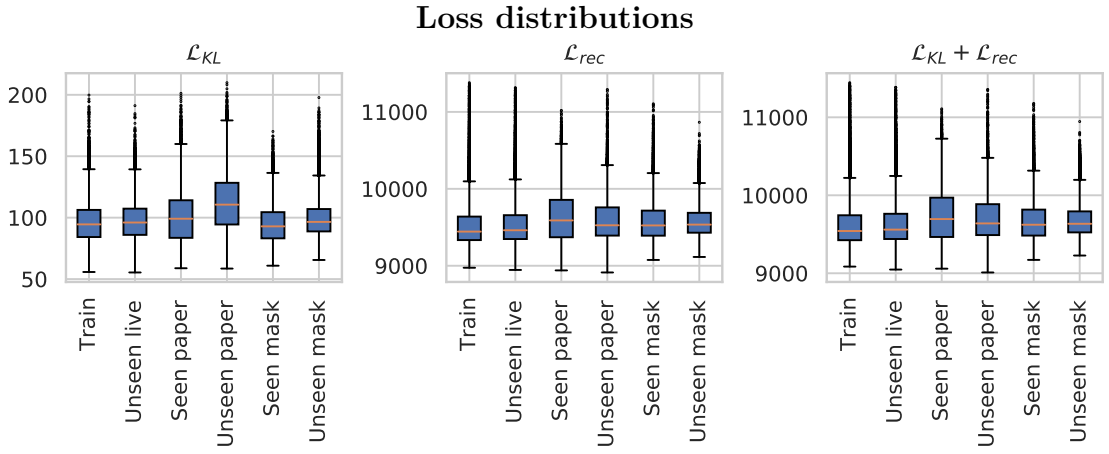


Figure 4.1: The loss distribution for between 30 000 - 50 000 samples from each subset. The mean and standard deviation of $\mathcal{L}_{KL}$ of the unseen paper subset is noticeably larger than the other subsets. For the other subsets the distributions look very similar and there is a large overlap between them. For $\mathcal{L}_{rec}$ the largest difference is in the seen paper subset, but not a very substatial difference. The total loss incorporates both of theses differences and the mean loss of the train and unseen live subsets is the smallest. There is still a large overlap.

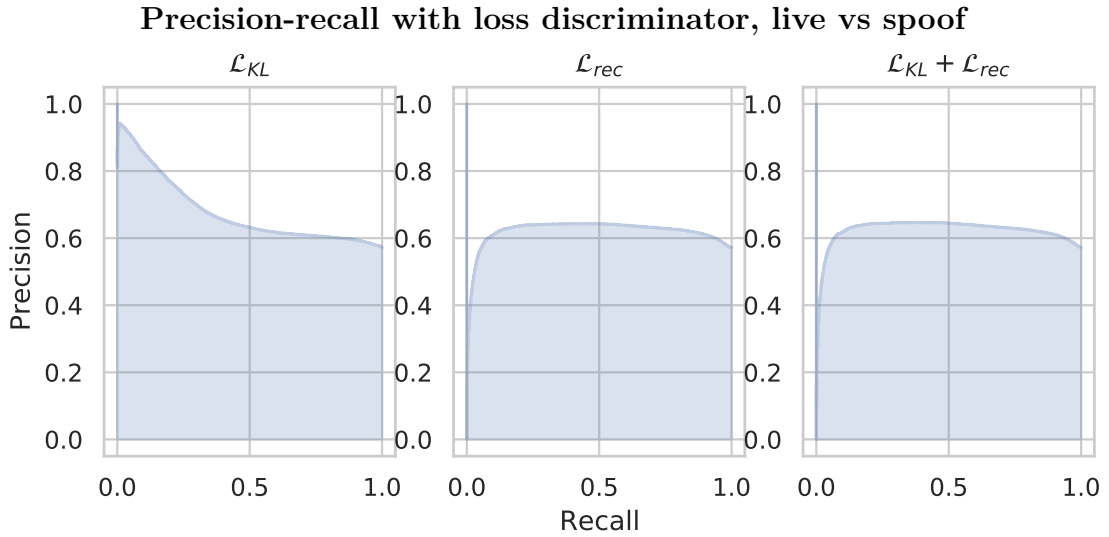
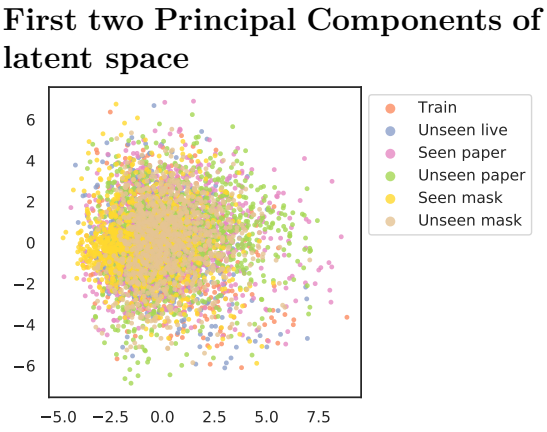


Figure 4.2: Precision-recall curves for various loss thresholds between live and spoof. The $\mathcal{L}_{KL}$ seems to perform the best for small threshold values but quickly becomes no better than chance. $\mathcal{L}_{rec}$ and the combination are slightly better than chance but only just. The data used is the same used for the loss distributions and contain around 55% spoofs.

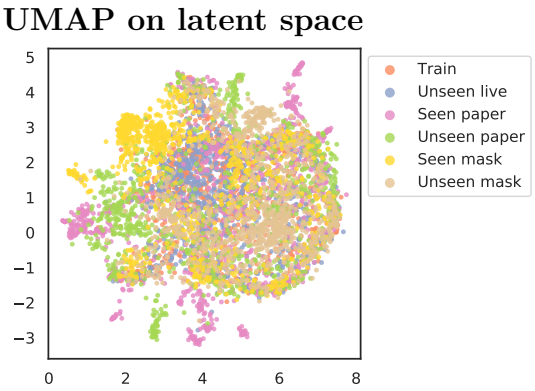
4.1.1.2 Latent space

In the latent space one can also see indications that the subsets do not disentangle much at all. Certain parts of the subsets appears to separate a bit from the rest of

the data, but again there seems to be quite a lot of overlap of the subsets.



(a) First two Principal components of the latent space. No separation can be seen but there seems to be a slight difference in the distributions between some of the subsets.



(b) The UMAP confirms what is seen from the Principal Component scatter plot. However, the UMAP takes all dimensions into account when performing dimension reduction and there seems to be parts of subsets that creates clusters.

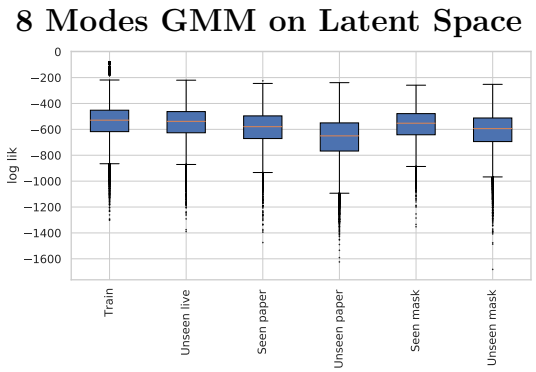


Figure 4.4: Mixture of eight Gaussians. All GMM we tried looked very similar in their ability to differentiate live from spoof, so there is no particular reason why we chose eight modes. This figure is confirming what we could see in the scatter plots, that there is some difference between certain subsets in the latent space. The difference is not very large however and no major separation can be found.

4.1.2 Training on filtered dataset

In this experiment we tried to see if the model could more effectively learn if it trained on the filtered dataset with less outliers.

Augmentation
Standard scaling

Table 4.3: The only augmentation used was standard scaling of the inputs.

Hyperparameter	Value
batch per epoch	3 000
epochs	20
batch size	64
$\mathcal{L}_{KL}$ weight w_{KL}	1.0
$\mathcal{L}_{rec}$ weight w_{rec}	1.0
Learning rate	0.0001
Adam optimizer β_1	0.9
Adam optimizer β_2	0.99

Table 4.4: Hyperparamters used during training for the experiment with the filtered dataset.

4.1.2.1 Loss outputs

Again a signal can be seen for some of the subsets and it appears to be stronger than the regular training. This is probably due to the fact that the filtered dataset only filters the live category and makes the differences between the live and spoof data more apparent.

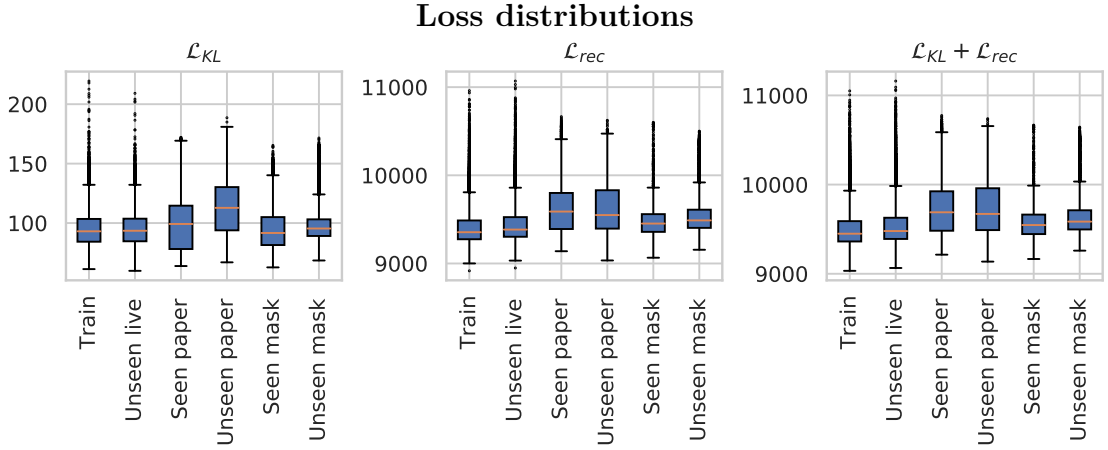


Figure 4.5: Training on the filtered dataset seems to improve the performance of our model to separate the paper subsets. This could however be due to the altered distribution of head-poses since only the training data was filtered.

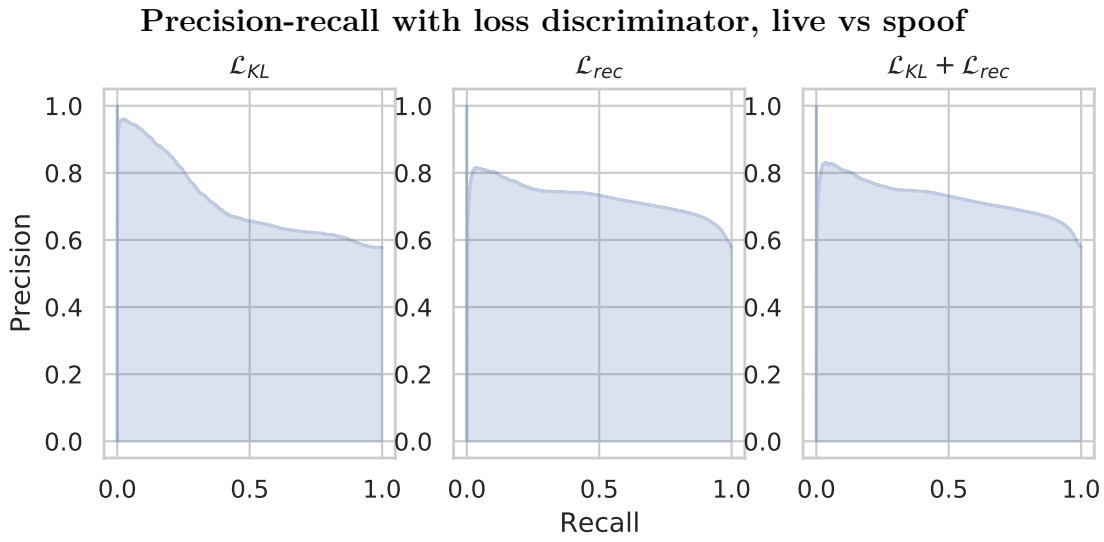
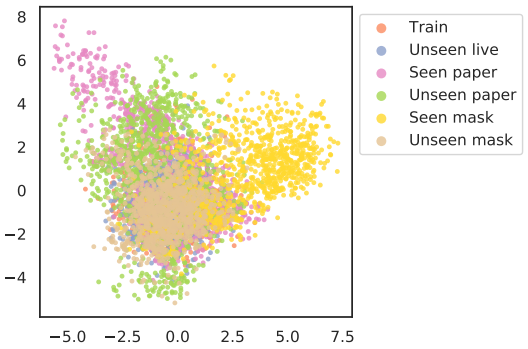


Figure 4.6: Training on the filtered dataset improved the performance to separate the live and spoof category based on reconstructions.

4.1.2.2 Latent space

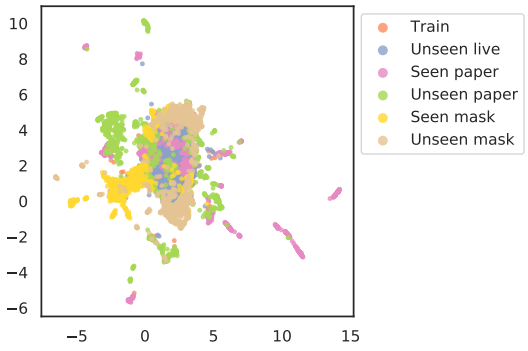
In the latent space we can see a larger separation but still some overlap. Worth to note again is that the filtration of the data was only done on the training data and not on the spoof categories so the larger separation should be due to the training process.

First two Principal Components of latent space



(a) This time we can see a greater separation from the first two Principal Components for some of the subsets. There is still a large overlap.

UMAP on latent space



(b) The UMAP confirms what we see from the Principal Components scatter plot. The live category subsets are still entangled with the spoof subsets since we can not see them.

4.1.3 Training on filtered dataset and with augmentations

This experiment looked to see if using the reduced image set together with augmentations to see if it had an impact on the performance.

Augmentation
Standard scaling
Random brightness
Random Gaussian noise
Random contrast
Random cutout
Random Rotation
Random translation
Random zoom
Right left flip

Table 4.5: This experiment used the full set of augmentations.

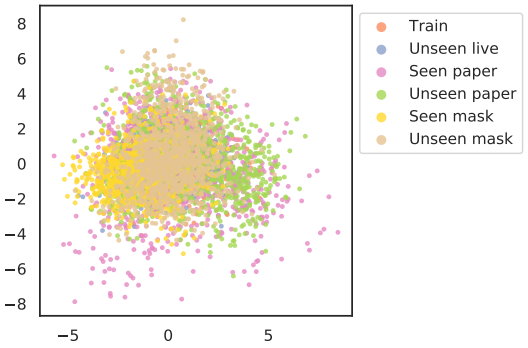
Hyperparameter	Value
batch per epoch	3 000
epochs	20
batch size	32
$\mathcal{L}_{KL}$ weight w_{KL}	1.0
$\mathcal{L}_{rec}$ weight w_{rec}	1.0
Learning rate	0.0001, decaying
Adam optimizer β_1	0.5
Adam optimizer β_2	0.999

Table 4.6: Hyperparameters used in this experiment.

4.1.3.1 Latent space

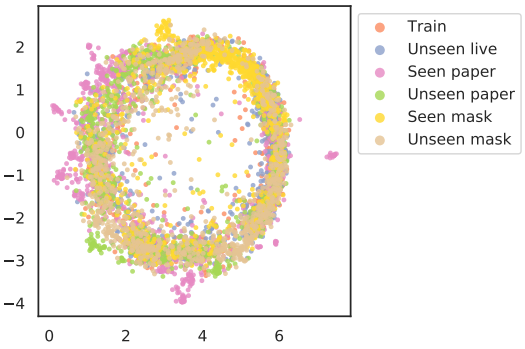
The augmentations did not increase the separation in the loss outputs and the latent space. The UMAP scatter plot has however formed a circular shape. This is probably just an artifact of the UMAP algorithm.

First two Principal Components of latent space



(a) Training with the augmentations did not seem to improve any separation, it looks very similar to the previous figures.

UMAP on latent space



(b) The UMAP looks different in shape from the previous ones, indicating that there might be a different distribution in the latent space. There still is no visible separation.

4.2 ResVAE

4.2.1 Baseline training

The residual net was trained on the filtered dataset to see how effective the training could be done. The network was also allowed to train for longer since the network was more complex, with residual blocks where DCVAE had only convolutional layers. A decaying learning rate was also used to see if the model would converge good set of weights.

Augmentation
Standard scaling
Random brightness
Random Gaussian noise
Random contrast
Random cutout
Random rotation
Random translation
Random zoom
Right left flip

Table 4.7: The only augmentation used was standardization of the input.

Hyperparameter	Value
batches per epoch	3 000
epochs	40
batch size	12
$\mathcal{L}_{KL}$ weight w_{KL}	1.0
$\mathcal{L}_{rec}$ weight w_{rec}	1.0
Learning rate	0.00005, decaying
Adam optimizer β_1	0.5
Adam optimizer β_2	0.999

Table 4.8: Hyperparameters used for this experiment.

4.2.1.1 Loss outputs

The ResVAE did not perform better than the DCVAE on the task of anti-spoofing. The model was more complex since every convolutional layer from DCVAE is at large replaced by residual blocks consisting of three extra convolutional layers, and the maximum depth of the network thus is around two times larger, while the minimum depth is around as large as DCVAE.

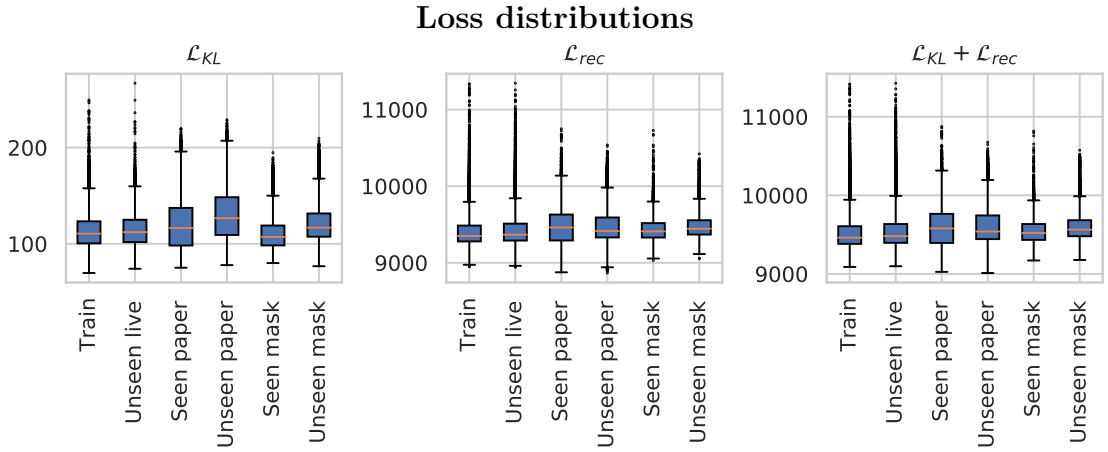
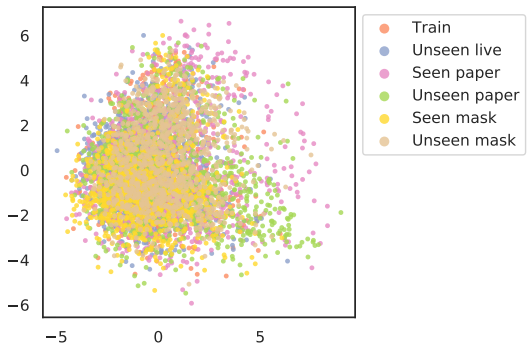


Figure 4.9: We see similar results for the ResVAE that we do for DCVAE, even though it was allowed to train for longer and with a smaller batch size.

4.2.1.2 Latent space

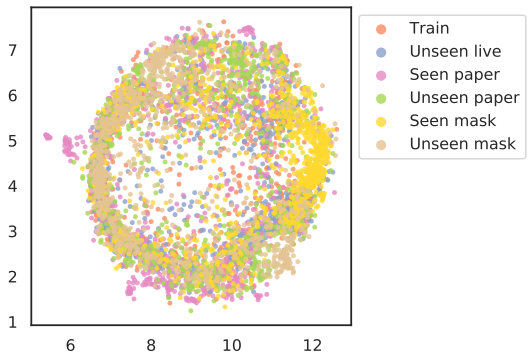
The latent space separation is very similar as the latent space from DCVAE.

First two Principal Components of latent space



(a) The latent space for ResVAE appears to be highly entangled.

UMAP on latent space



(b) The UMAP of the latent space for ResVAE seems to confirm the entanglement. Parts of subsets look like they are separating however.

4.3 Sample image output from VAE models

The actual image output, i.e. the reconstructions, from the VAE models can be of interest for determining their performance. A bad reconstruction can possibly indicate whether or not the input is sampled from the training distribution and is from the live category. The image output from all VAE models are similar and this section will contain sample output from each subset from different experiments. Which experiment will be mentioned together with the image. We chose images from experiments that shows some interesting behaviours for the model.

Live

Original **Reconstruction**

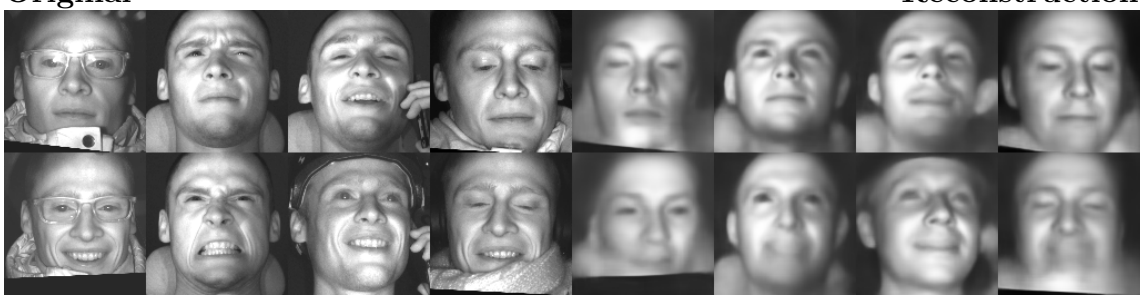


Figure 4.11: Sample reconstructions from the live category from DCVAE. In two images the subject has glasses but those are gone in the reconstructions. In one a hand is being reconstructed as an ear. An angry face is being reconstructed to something that looks like a big smile.

Figure 4.12: Sample reconstructions from the seen paper subset from DCVAE. Images similar in domain to the car interior are reconstructed the best. When the domain changes to a light background the network fails completely to reconstruct.

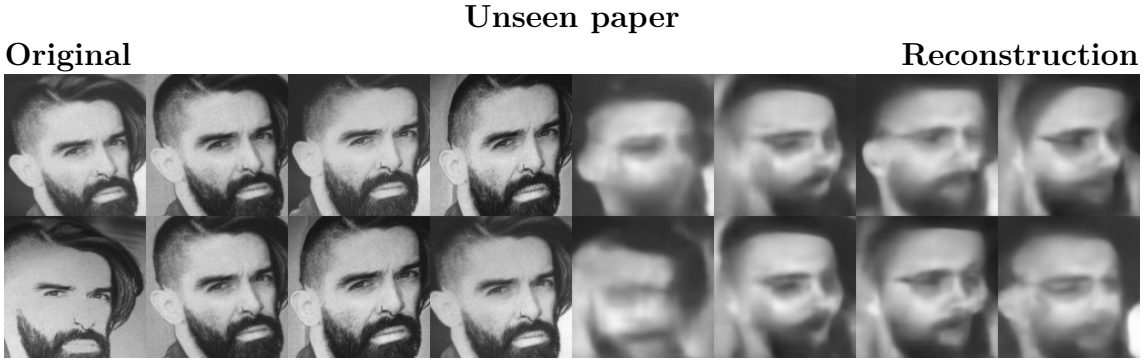


Figure 4.13: Sample reconstructions from the unseen paper subset from DCVAE. The network somewhat fails at reconstructing and in some reconstructions it looks like the subject is wearing glasses.

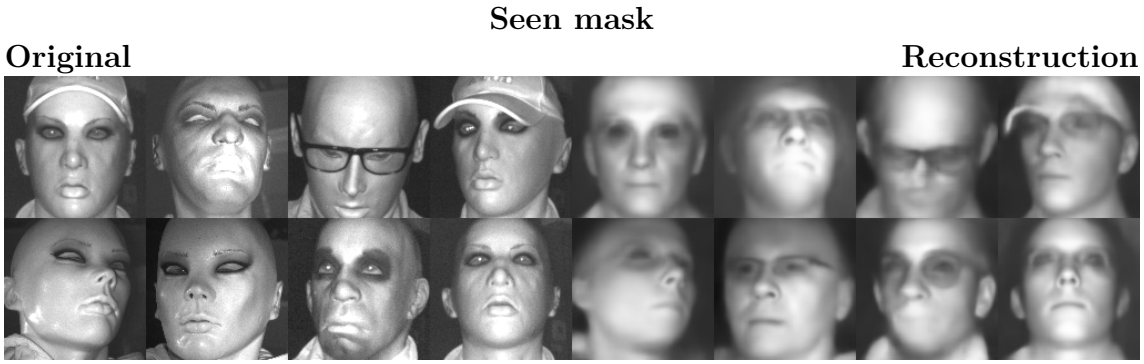


Figure 4.14: Samples reconstructions from the seen mask subset from ResVAE. The input images looks like they have been taken from the inside of a car and may be why the reconstructions are better than some of the paper reconstructions. The reconstructions are more human-like than the inputs and in some the subjects are wearing glasses.

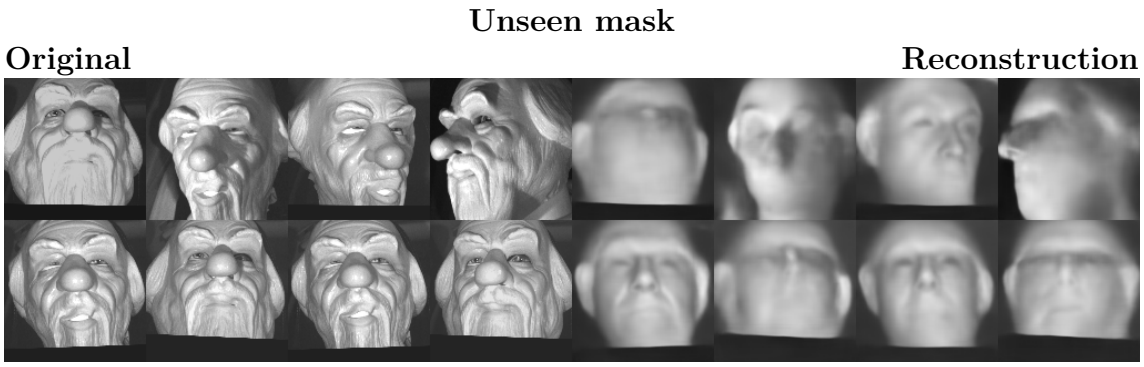


Figure 4.15: Sample reconstructions from the unseen mask subset from ResVAE. The reconstructions are not as good as the ones from the seen mask subset, but features from live faces are being reconstructed in some of the images. Some are also appearing to wear glasses.

4.4 GANomaly

The GANomaly architecture detailed in section 3.5 produced arguably the most promising results of all the tested models. The results are shown in terms of a Precision-Recall curve as well as a Receiver Operating Characteristic (ROC) with corresponding area under curve (AUC).

Different hyperparameters were tested for the GANomaly achitecture but no complete sweep over all parameters were performed. An exhaustive search for the best hyperparameters combined with training and testing for each configuration was not feasible in terms of resources and time. Instead, the final hyperparameters were found by varying one while keeping the others fixed. The relevant hyperparameters for GANomaly is presented in table 4.9.

Hyperparameter	Value
Learning rate	0.00002
batch per epoch	1500
epochs	50
batch size	16
Encoder loss weight w_{enc}	1
Adversarial loss weight w_{adv}	1
Reconstruction loss weight w_{rec}	50
Adam optimizer β_1	0.5
Adam optimizer β_2	0.99

Table 4.9: Displaying the final hyperparameters for the GANomaly architecture which produced the best results.

Figure 4.16 shows a Precision-Recall curve of the *test seen paper* and *test seen mask* datasets of the GANomaly model from the best performing epoch. For a high

threshold (low recall) we note a very high precision showing that the model is able to find the most obvious spoofs in the dataset. The threshold is lowered until all samples are categorized as spoofs, leading to a recall value of 1 and precision equal to the spoof ratio of the test set.

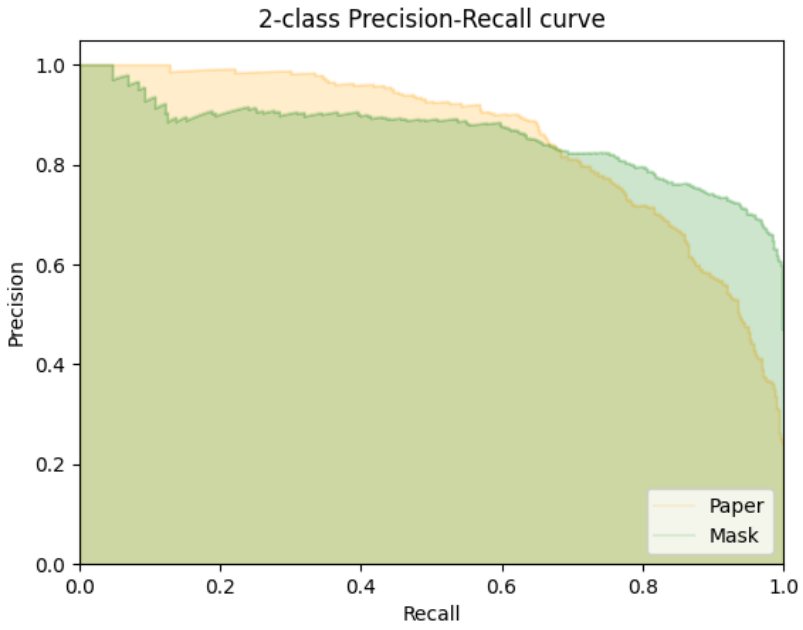
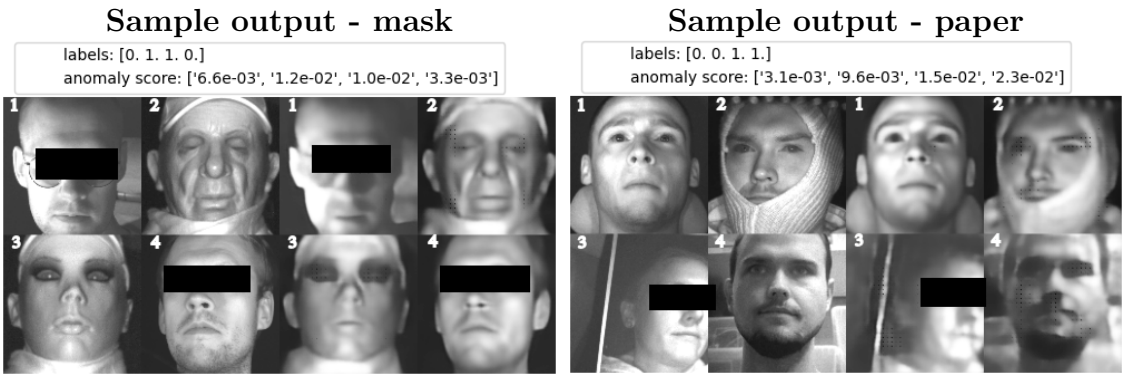


Figure 4.16: Precision-Recall cuve for *test seen paper* and *test seen mask* datasets. The paper spoofs have a higher precision for lower recall, indicating the most obvious spoofs are paper ones.

Figure 4.17a and 4.17b shows a sample output from the model, displaying four input images (left) and four reconstructed images (right). Spoofs have label 1 while bonafide samples have label 0 and anomaly scores are given to each of the reconstructed images.

The spoofs can be seen to have marginally worse reconstruction and the higher anomaly scores. Some bonafide images with high anomaly scores can be explained by unique features such as the ski-mask in image 2 in 4.17b.

Some behaviour of the model, such as the removal of glasses in image 1 in 4.17a suggest input images are projected onto the learned distribution of faces without glasses. This might give false signals in the anomaly score and affect performance.



(a) Output samples from the test seen mask dataset. Image 2 and 3 are spoofs and have a higher anomaly score.

(b) Output samples from the test seen paper dataset. Image 2 shows a quite high anomaly score which might be due to the ski-mask.

Figure 4.17: The figures show input samples and reconstructed output for the mask and paper datasets respectively. Labels and anomaly scores belong to image 1-4. Spoofs have label 1. Some images are partly covered due to privacy concerns.

Figure 4.18 displays the AUC of the ROC curve as the discriminative threshold is varied. The two datasets containing masks and paper spoofs are displayed. The overall precision and therefore a higher AUC is found for the paper spoof images.

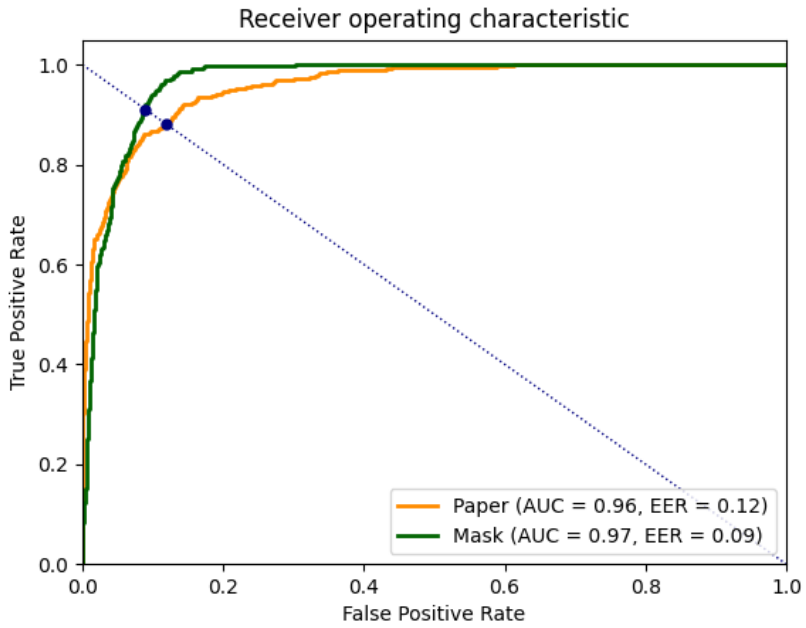


Figure 4.18: The ROC curve for mask and paper datasets showing dissimilar shapes. While they have similar EER and AUC, the paper dataset shows higher TPR for low FPR, while the mask dataset has higher TPR for higher FPR.

5

Discussion

In this chapter we discuss the performance of the individual configurations as well as the dataset and potential additions or changes.

5.1 Dataset

The dataset we had access to at Smart Eye was large in terms of the number of images available, but we do believe after having been working with this project, that the dataset is not extensive enough for the task of training a robust face anti-spoofing model unsupervised. This is common in general for the face anti-spoofing task, most often data from the spoof domains is lacking. There are many more spoof domains that we did not have access to, make-up spoofs for example. The models we focused on only used spoof data for evaluation so it had no implication on the training. There is an abundance of RGB face datasets from the live domain in existence that could be converted to grayscale and used to extend the live domain for training. The Smart Eye dataset also contained many similar images that were taken from the same sequence where the subject did not move much between the images, which could cause certain images to have a large impact on learning and make the dataset much larger than it has to be. The dataset also mainly consisted of caucasian faces and we believe that more diversity would make the dataset more robust and increase models real world performance.

Another potential weakness of the dataset is that the live category contains images where faces are partly obscured by hands, coffee mugs, etc. We say potential weakness since in most cases a human can still tell if it is a live person or a spoof behind the obscured, but it can be difficult to tell which person. If the network does not recognize obscured face images as at least anomalous then it could be a potential way to fool the models. We never fully explored this topic and believe it is something future research must take into account.

In an attempt to make the input images to the networks more suitable for learning, the dataset for some experiments was filtered from outliers and faces not looking toward the camera. The filtration based on head-pose can be seen as a quite large restriction since we effectively reduced the live domain that the models see. Our thought behind doing head-pose filtration was that if it worked it would be a

good proof of concept for further investigation while producing results with a less time-consuming training phase. We did see a signal that the models found some differences between the live and spoof categories. A better filtration technique might have been to remove images that are too similar according to some suitable metric, which would also reduce the amount of training data, while not restricting the live domain.

Included in the dataset is auxiliary information such as the presence of glasses, beards, hats, etc. This information could have been used explicitly to better train the latent space. Since the latent space represents the features of an input image, there should be one or more latent variables concerned with the glasses or facial hair. Training with auxiliary information could impact latent embedding positively.

5.2 Configuration

The original hypothesis was to use the latent space as a spoof indicator. During the project we realized that the networks generally learned to project spoofs onto regions in latent space associated with real faces. This made it so that spoof images were reconstructed more like real faces, with sometimes the spoof features being totally removed. This probably stems from the fact that the network is trained to recreate live images and will therefore project anything onto the live domain. This makes using the latent space not enough for spoof classification, instead the reconstruction of spoofs will have a higher median error since the image is projected on the live face domain. This is the reason the later models and the best performance is associated with anomaly detection on reconstruction error rather than latent domain analysis.

In some reconstructed images, especially for the GANomaly architecture, some aliasing could be seen for spoof images reconstruction. We did not observe this behaviour for any standard live images, however some anomalous sections, such as when a hand is covering the face, some alias could be present. We believe this aliasing comes from the network not having seen similar structures before, and is confused on how to embed and reconstruct it. This aliasing might not affect the reconstruction error significantly, but is still an interesting artifact we observed for the spoof domain.

It appeared that the GANomaly model performed the best during our attempts at FAS. It seemed to indicate anomalies for things that not directly are spoofs, for example a hand covering a face, head covering ski masks etc. This is something we still see as a somewhat desired result in some instances. A hand obscuring a face should be marked as a spoof, otherwise this would be a vulnerability and could be exploited to bypass the anti-spoofing.

The GANomaly architecture does not use stochastic sampling in the latent layer, and is therefore not a *Variational* autoencoder. Therefore we did not make any in depth analysis of the latent space of that model. There are still possibilities that the latent domain of the GANomaly architecture is as useful as the VAE, but the theory as to the representation of the latent domain is not covered in the theory

chapter and was thus omitted from the report.

We do however believe there is more exploration that can be done with the autoencoder models. There can be much more optimization done in terms of which layer configuration to use to best extract the desired facial features. We did some trials with simpler models with less filters for each convolution but generally it we got better reconstructions with more filters. We got inspiration for our current autoencoder architecture from an existing GAN generation at Smart Eye that had success at doing high resolution reconstructions. It should still be possible to get good results from a simpler architecture. The reconstructions of the training data was of a lower quality than we had hoped for and there are several reasons suspect causes this. One is the batch size during training. We never experimented much with it, but a smaller batch size is something we think could improve the networks ability to learn facial features. Looking at papers where facial reconstruction was done with a high resolution [23][30] they were done with a lower batch size for high resolution images. Another reason we believe that our reconstructions was of a lower quality is the learning rate, or rather, the learning rate together with the magnitude of our losses. The values we got was of order 10 - 10 000 which is much higher than what one usually see when using standard losses such as Mean Squared Error. The large losses is not caused by any fault in the model, but is an artifact from how they are calculated. A lower learning rate could increase the models learning performance. The autoencoder models are also less computationally complex than the GANomaly model since less internal networks are involved during training.

Recurrent Network for sequence handling

The dataset images are generated from videos, and are therefore part of sequences. When we as humans look at the dataset to determine spoofs, it can be quite hard for us only using a single picture, but with the use of the whole sequence, and how the image changes during the sequence we can make a much better prediction. In light of this, it would be useful for the model to have access to the sequence information as well, something the current model does not.

Using the sequence in the form of a *Recurrent* Neural Network (RNN) was something we discussed due to the huge possible upsides. In the end we determined it to be outside the scope of the project because of the major challenges of implementing a recurrent network for this purpose.

The main problems of RNNs stems from the inherent problems of the memory state when the network is starting inference on a new sequence. There are other difficulties such as what should happen if the continuous feed of images stops or becomes to low quality (driver outside frame etc.) or how much computational resources needs to be provided to a RNN for continuous spoof analysis.

5.3 Future work

Anecdotal evidence at Smart Eye suggests a partition of input images based on the regions of the face could improve performance. By having a separate preprocessing module that extracts the eyes, nose and mouth (and other) regions of the face, the inference for each region can be run separately for more stable results. Using this method a separate network for each region (or the same network with region label attached) can be used to analyze the features of each region. A spoof might have a very live-like mouth and nose but eyes that anomalous, something that might be easier for a partitioned model to recognize. This not experimented on but could be explored in future research.

One thing we noticed during training of the models was the emphasis put on the backgrounds and periphery. Objects in the periphery, such as steering wheels were usually reconstructed with a high fidelity and if the background was light up, as opposed to the dark background in the training images, the reconstructions would usually completely fail. To make the networks focus more on the facial features one might want to adopt facial crop masks for the inputs, where the backgrounds are removed from images and only facial information is present. Another method is to use an open source RGB dataset with various backgrounds to make the model learn that the backgrounds can vary and may not be all telling if the images are live or spoof. Or maybe a combination of both. This is something that has to be evaluated further.

As mentioned before, one could make use of auxiliary information from the dataset in for example Conditional Variational Autoencoders (CVAE), which inserts label information into the latent space [31]. This is something we never explored but could be interesting for future research. We also mentioned that the two terms in the ELBO loss could be weighted to get different behaviours for the VAE model [26], which we never explored to any depth either. These two VAE models do we think are worth more research for future projects in unsupervised face anti-spoofing.

Perceptual loss

The reconstruction loss used to train the network is based on a pixel-wise calculations of deviation between input and output and might not always be representative of how well the image is reconstructed. Imagine the network recreating the input image perfectly, but shifted 5 pixels over. To a human, that reconstruction would be close to perfect, but to the reconstruction loss could indicate a poor reconstruction.

The pixel-wise losses potentially penalizes the network in such a way that it can not disentangle the high dimensional correlations good enough. The network might then not be incentivized enough to guess, and instead plays it safe and returns a blurry image, all please the reconstruction loss' hard regularization.

Using *perceptual* or *contextual* losses is a way to circumvent this problem. Thesis losses does not work on a pixel-wise basis and instead tries to find structural

similarities in the input and output images that better represent a humans perceived reconstruction performance. These losses are more complex and can come in the form of separate networks to determine feature differences, or as statistical algorithms such as SSIM.

During the project, perceptual losses were experimented with in the hope of reducing reconstruction blur and improve overall performance. However, we did not manage to fully implement a perceptual loss network that could improve performance using a perceptual loss, so it was intentionally omitted from the thesis.

6

Conclusion

The experiments in this thesis explore the possibilities of using variational autoencoder and adversarial training to solve the unsupervised face anti-spoofing problem, that is to distinguish live images from presentation attacks. The models utilized a latent space to embed input images into a latent space and then reconstruct them, the reconstruction performance and the latent space embedding were used to classify spoofs.

The thesis proposes three different architectures to solve the unsupervised anti-spoofing problem, with the best results generated by an encoder-decoder network connected to a discriminator for adversarial training. The hypothesis of latent space separation between live and spoof domains was investigated thoroughly, but ultimately unable to sole perform adequate spoof classification. An overarching trend was the reconstruction error signals being the more indicative cue for spoof recognition compared to that of the latent space.

Altogether, the thesis concludes that unsupervised anomaly detection using encoder-decoder neural networks is a viable approach to classifying live images from spoofs. The challenges come in the form of creating comprehensive datasets to fully learn the distribution of real faces, and the entanglement of the spoof domains to the live domain. Further research and testing are required to find a comprehensive spoof threshold and exclude over-fitting to used datasets.

This primary investigation shows that the models show promise but are not ready to be implemented in production devices. With some additions and more rigorous training, the models should be able to assist as anti-spoofing software.

Bibliography

[1] Geoffrey Hinton and Terrence J. Sejnowski. *Unsupervised Learning: Foundations of Neural Computation*. The MIT Press, 05 1999.

[2] Road safety: Commission welcomes agreement on new eu rules to help save lives.

[3] Zitong Yu, Student Member, Yunxiao Qin, Xiaobai Li, Chenxu Zhao, Zhen Lei, Senior Member, and Guoying Zhao. Deep learning for face anti-spoofing: A survey.

[4] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training.

[5] J. M. Bernardo and Adrian F. M. Smith. Bayesian theory. page 586.

[6] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12:307–392, 6 2019.

[7] The generative-discriminative fallacy.

[8] Arindam Banerjee. An analysis of logistic models: Exponential family connections and online performance. *Proceedings of the 7th SIAM International Conference on Data Mining*, pages 204–215, 2007.

[9] Jianwen Xie. Generative modeling and unsupervised learning in computer vision. 2016.

[10] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes.

[12] David M Blei, Alp Kucukelbir, and Jon D Mcauliffe. Variational inference: A review for statisticians. 2018.

[13] Stephen Odaibo. Tutorial: Deriving the standard variational autoencoder (vae) loss function. 7 2019.

[14] Deep learning - ian goodfellow, yoshua bengio, aaron courville - google böcker.

[15] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature 1986 323:6088*, 323:533–536, 1986.

[16] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12 2014.

[17] Keiron O'shea and Ryan Nash. An introduction to convolutional neural networks.

[18] Bernhard Mehlig. Machine learning with neural networks. 2021.

[19] The reparameterization trick in variational autoencoders | baeldung on computer science.

[20] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets.

[21] Hong Hui Tan and King Hann Lim. Vanishing gradient mitigation with deep learning neural network optimization. *2019 7th International Conference on Smart Computing and Communications, ICSCC 2019*, 6 2019.

[22] Mohammad Sadegh Ebrahimi and Hossein Karkeh Abadi. Study of residual networks for image recognition. *Lecture Notes in Networks and Systems*, 284:754–763, 2021.

[23] Jonathan Bergqvist. Multimodal image-to-image translation for driver monitoring system development synthesizing diverse human faces in the near-infrared domain using conditional generative adversarial networks master's thesis in complex adaptive systems.

[24] Houssam Zenati, Chuan-Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection *. 2019.

[25] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks.

[26] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, Alexander Lerchner, and Google Deepmind. β-vae: Learning basic visual concepts with a constrained variational framework.

[27] Leland Mcinnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. 2020.

[28] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[29] Takaya Saito. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. 2015.

[30] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[31] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY