Linnéuniversitetet
Kalmar Växjö

Master Thesis Project

# Image generation through feature extraction and learning

*Using a deep learning approach*


*Author:* Tibo Bruneel
*Supervisor:* Prof. Dr. Welf Löwe
*External Supervisor:* Dag Björnberg
*Examiner:* Dr. Jonas Lundberg
*Reader:* Dr. Rafael Messias Martins


*Semester:* VT 2023
*Course Code:* 5DV50E
*Subject:* Computer Science

**Abstract**

With recent advancements, image generation has become more and more possible with the introduction of stronger generative artificial intelligence (AI) models. The idea and ability of generating non-existing images that highly resemble real world images is interesting for many use cases. Generated images could be used, for example, to augment, extend or replace real data sets for training AI models, therefore being capable of minimising costs on data collection and similar processes. Deep learning, a sub-field within the AI field has been on the forefront of such methodologies due to its nature of being able to capture and learn highly complex and feature-rich data. This work focuses on deep generative learning approaches within a forestry application, with the goal of generating tree log end images in order to enhance an AI model that uses such images. This approach would not only reduce costs of data collection for this model, but also many other information extraction models within the forestry field. This thesis study includes research on the state of the art within deep generative modelling and experiments using a full pipeline from a deep generative modelling stage to a log end recognition model. On top of this, a variant architecture and image sampling algorithm are proposed to add in this pipeline and evaluate its performance. The experiments and findings show that the applied generative model approaches show good feature learning, but lack the high-quality and realistic generation, resulting in more blurry results. The variant approach resulted in slightly better feature learning with a trade-off in generation quality. The proposed sampling algorithm proved to work well on a qualitative basis. The problems found in the generative models propagated further into the training of the recognition model, making the improvement of another AI model based on purely generated data impossible at this point in the research. The results of this research show that more work is needed on improving the application and generation quality to make it resemble real world data more, so that other models can be trained on artificial data. The variant approach does not improve much and its findings contribute to the field by proving its strengths and weaknesses, as with the proposed image sampling algorithm. At last this study provides a good starting point for research within this application, with many different directions and opportunities for future work.

**Keywords: Deep Learning, Neural Networks, Deep Generative Learning, Variational Autoencoders, Generative Adversarial Networks, Flow-based Models, Triplet Image Generation, Triplet Loss, Tree Log End Generation, Forestry Application.**

## Preface

I would like to take the opportunity here to thank the people that made my master thesis possible. First and foremost, I would like to thank both of my thesis supervisors, Prof. Dr. Welf Löwe and Dag Björnberg. I have had the pleasure of participating in many of Prof. Dr. Löwe's machine learning courses over my master studies, which got me inspired and interested in the field of AI in the first place. Over all the years of higher education studies, I always struggled finding the direction that I really wanted to continue in, these courses and the many conversations with Prof. Dr. Welf Löwe completely changed that. I have also been very fortunate to work together with Dag for the past year at Softwerk AB and learn a lot. Thank you for guiding me all the past months with much expertise, critical feedback and many suggestions, which made my thesis what it is today.

I would also like to thank my thesis reader Dr. Rafael Messias Martin for the suggestions and discussions, with great ideas for future work following up on this thesis. And Dr. Tobias Ohlsson for the guidance throughout the course.

On top of this I would like to thank my partner for reading and reviewing my thesis extensively, and pushing me further throughout my studies and goals. I would like to thank my parents for making my studies and living in Sweden possible in the first place, I would not be here without their support.

Finally I would like to thank Björn Lundsten for making this thesis with Softwerk AB possible and the many opportunities I have received working at Softwerk AB during my master degree studies. I would also like to thank my colleagues at Softwerk AB for making it such a nice work environment and team to be part of. Thank you all!

And thank you for reading my thesis, I hope you enjoy and learn something reading it.

*Tibo Bruneel*
*May 31, 2023*

# Contents

# 1 Introduction

The generation of artificial images is one of the most challenging tasks in the field of computer vision. Different machine learning techniques have proven to be on the front of this, and over time more approaches have resulted in promising solutions. The purpose of this thesis project is to explore and study the space of deep learning approaches in artificial image generation, and how such a deep generative model can be applied to an industrial project. The thesis consists of a literature review on the state-of-the-art deep generative models. Followed by a controlled experiment with an application of a generative model and a proposed variant, to further use the generative models to sample and generate images using a proposed algorithm. And in the final stage train a tree log end image recognition AI model.

## 1.1 Background

Image generation is the process of artificially generating non-existent images, based on a model that has learned how to generate them through the model goal and given data. The model attempts to learn a feature distribution that resembles the real world data distribution as well as possible. A feature can be seen as a certain aspect of an object on an image, that is extracted in the case of generative artificial intelligence (AI). Some general examples of features can be shape, colour, or curvature. These features can go from general to highly detailed. Learning a distribution furthermore has to do with how such a feature is mapped over a full set of data. These distributions are of high importance for learning features, and how the data behaves/evolves. By extracting features and learning their distribution over the full sample of data, it then becomes interesting to generate new non-existing data with these models.

Generative models attempt to learn how to generate artificial images based on the training data, by minimising the difference between the real world data distribution and the generated/reconstructed images of the model. Different generative models optimise in different ways, but the end goal of all the various state-of-the-art models is to generate as accurately as possible. As almost all problems are not closed-form and data is usually only a sample of the full possible data set and real world, it is hard to learn the underlying distribution exactly. This is why generative models attempt to learn it as well as possible, but in practice it would eventually always be an approximation of the real world.

Image generation has many useful applications and has proven to be quite important in different use cases, with some of the most common ones being data extension, data augmentation, anomaly detection, and text-to-image generation. In all the above mentioned use cases, images are used as model input, output, or both. In data extension, artificial images can be used to extend the real images in a data set, which is especially interesting in cases where the data set is rather small and insufficient for using it in other algorithms. In data augmentation, it can be useful to extend the existing data set to generalise any model better and to prevent any overfitting on real data. Although extension and augmentation are very similar in their use and goals, in data extension the focus is purely on extending the data set with more data. While with augmentation the focus is on augmenting the real data with generated versions, to introduce variants with augmentations and commonly some noise. Anomaly detection is the task of finding/detecting abnormal observations in data, based on the knowledge of what a non-anomaly or normal behavior is being

considered as. However, sometimes the exact anomaly is known, which makes detection and training easier. Detecting these outlier observations and patterns in the data is interesting and useful for many different reasons and use cases. Outlier detection is a commonly occurring manually solved problem, to which an automated solution can have a major impact to businesses on financial and time aspects, but also on people and environments. Image generation can prove to be very useful for detecting unexpected behavior, as the generative model would perform differently on those anomalies, since it learned well how the features are distributed on normal behaviour. [1, 2] Finally, text-to-image generation has become quite a hot topic over the past years within the field of AI, with generators as DALL-E [3], DALL-E-2 [4] and Stable Diffusion [5], where the goal of these models is to generate an artificial image based on given textual captions, i.e. a prompt. These have of late become very applicable in combination with large language models such as GPT-4 [6]. The automatic generation of new artificial data, that resembles real data has found to be useful for AI that is applied in many different sectors. These different sectors mainly exist out of production, analytics, security, and medical environments.

Within the deep learning space for image generation in computer vision, there are a few leading approaches. These include Variational Autoencoders (VAE) [7], which is based on the concept of Autoencoders [8], Generative Adversarial Networks (GAN) [9], and Flow-based generative models [10].

### 1.2 Motivations

There has been a lot of active research in feature extraction/learning, and image generation in the past decades. A lot of approaches have been very successful in their own use cases, especially in deep learning with different variational autoencoders [7, 11, 12, 13], various generative adversarial networks [9, 14, 15, 16], and flow-based generative models [10, 17, 18]. These architectures and algorithms have been achieving remarkable results in generating artificial images. This generation process and its results can contribute to significantly improving other fields within AI and other industries. Within anomaly detection the automatic unsupervised detection of anomalies can reduce a lot of costs and time in production environments, assist in medical and security environments, and in various analytics use cases like network analysis [1]. Additionally, it can also assist in improving and regularising existing models through data augmentation. As well as extending data sets where the data set is small, and the data collection process is expensive on financial and time aspects. It can also contribute heavily with approaches like text-to-image generation or other similar content-to-image methodologies.

The motivation for this thesis project is to construct a pipeline using a generative model, with a sampling algorithm, to potentially improve a recognition model. On top of this, the aim is to also construct a variant methodology in the form of a neural network architecture, similar to the existing approaches. The aim of the variant and the difference with the base approach is on learning the data more effectively and accurately in an unsupervised way, through more targeted learning on individual features. A forestry related project within Softwerk AB on generating artificial tree log ends, is a good application to test the generative model approach and the variant on, in order to further assess the usability and potential within the AI pipeline in the project. Softwerk AB is a software development company based in the south Swedish city of Växjö. Softwerk AB offers a wide range of technical excellence

over a large variety of projects, as software development, machine learning, system development and others. The proposed architecture and methods can contribute to the field by learning how effective and applicable it is. With success, the proposed variant approach provides better results and learning than the standard generative approach, and it can therefore improve the application by using this architecture. Furthermore it could contribute within the computer vision field, and impact the industry and society by making it applicable to different use cases. On top of that, if either the variant or standard generative model approach works well in the image synthesis within the forestry application, artificially generated images can be used in order to improve other models related to the same application. It can therefore also impact the industry application and related research within the field of wood processing and forestry.

## 1.3 Problem Statement

### 1.3.1 Problem and solution proposal

The research problem of this thesis project is on applying a standard variational autoencoder architecture and a variant of it, to an existing problem and data set, and with this application measure its performance and usability on quantitative and qualitative basis. The goal is to further use the generative models to sample images for measuring its capability of improving a recognition model with artificial data.

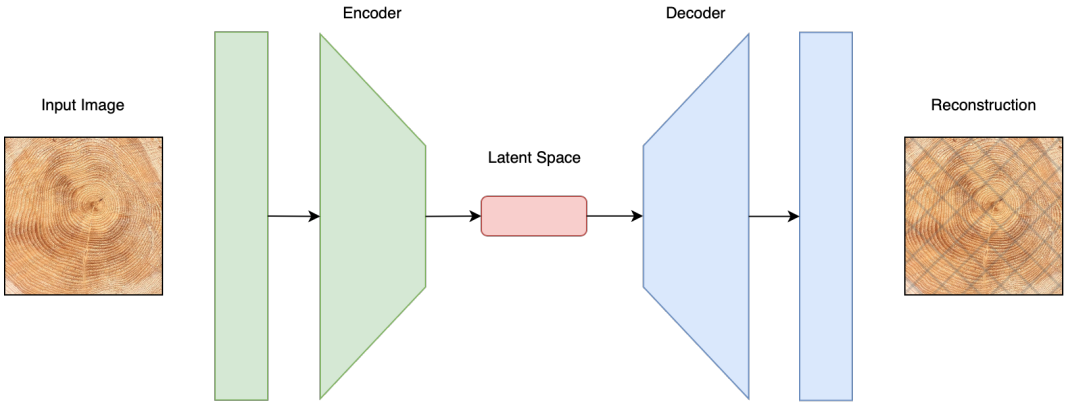


Figure 1.1: Vanilla VAE architecture.

The proposed variant is a network that is based on the autoencoder and variational autoencoder architectures by maintaining the encoder and decoder networks in the model. The vanilla variational autoencoder architecture with encoder, latent space with latent distributions, and decoder is visualised in the above Figure 1.1. The difference with the architecture proposed within this thesis project and the standard vanilla VAE, is that the proposed architecture learns a 1D latent space for every $D$ individual feature separately, instead of learning one latent space of all the extracted features together at the same time. In this architecture an extra set of hidden layers is added between the latent space training and the encoder network for the individual features. This is the key difference between the vanilla VAE architecture and the proposed VAE. The goal of applying this architecture is to experiment if such an architecture with an extra sub-network for all the features and

separated latent spaces can learn individual data distributions better compared to standard vanilla VAE architectures, to furthermore, with these sub-network latent dimensions, generate better, more creative and feature-rich images when sampling the latent space. On the Figure 1.2, the individual latent dimensions with their own sub-networks is visualised, this easily shows the difference between this architecture and the vanilla architecture in Figure 1.1. The main final goal of the architecture is to learn individual specific Gaussian distributions better from the images fed into the model, which in turn is interconnected with the quality of image generation/reconstruction, as better distributions would lead to better outputs of sampling the latent spaces. By applying both the standard VAE architecture and the proposed variant, to an existing problem and data set, the quality of the methodologies can be thoroughly tested and measured, while also being be compared.

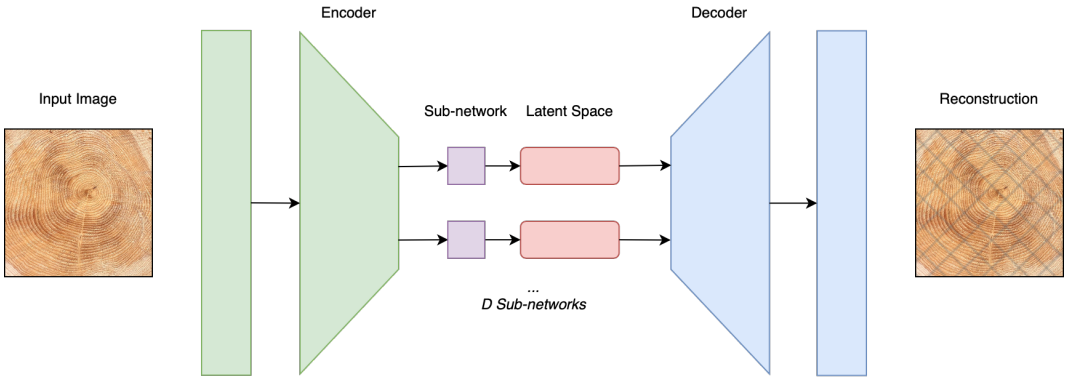


Figure 1.2: Proposed VAE variant architecture.

The application for this thesis project is a forestry application within Softwerk AB, which is on extracting information out of images of cross-sections of tree logs. Information extraction is done through different models such as log recognition. The generation of artificial images of tree log ends of high quality and high resemblance to real log ends, can significantly reduce the need for real data, which can save major costs and time, as data collection is often a tough, expensive and long process. It is especially interesting if the real data can be fully replaced by generated data in different models, and it is not just used as a data augmentation or data extension approach. The focus of this thesis project and its problem statement is to, next to a literature review on generative models, apply the above VAE architectures to the application of generating artificial images of tree log ends and optimise these architectures. The goal is then also to further down in the pipeline sample artificial log ends, using a proposed triplet sampling algorithm with the learned latent spaces, and use these images for a tree log end recognition model training. In this way, the fit of VAE architectures on this problem can be measured, and with success can improve other information extraction models through training on artificial data. The experiment can on top of the generative ability, improve other problems significantly in future work and research.

### 1.3.2 Research questions

With the above problem statement, this research aims to answer the following three research questions in Table 1.1 as accurately as possible:

| | |
|---|---|
| **RQ1** | What generative variant can be constructed with the goal of learning individual data distributions better than the standard approach? |
| **RQ2** | How does the applied generative variant and base model compare to other generative deep learning approaches in image synthesis? |
| **RQ3** | How can the proposed variant and base model contribute for triplet generation, for improving a recognition model on artificial data? |

Table 1.1: Thesis research questions

## 1.4 This Thesis Report

In this thesis the research questions are answered through two scientific approaches, that is a literature review and a controlled experiment. A forestry application is used for the image generation, where the goal is to generate artificial tree log end images, and with generated images improve a log end recognition model. A pipeline is presented in the implementation for the different stages from image generation to training recognition models.

The first stage of the pipeline, was applying a variational autoencoder architecture and train it on data sets of different sizes of log end images. An alternative variational autoencoder is proposed and also experimented with in this thesis, with the same circumstances as the base model. The idea behind this architecture is to introduce sub-networks per latent dimension, in order to improve the learning of features and to consequently improve the feature-richness of synthesis. The results show some strengths and weaknesses of both models, with the variant achieving better feature learning losses, but worse generation quality losses. A common problem over both architectures is the blurriness appearing in the image synthesis.

A literature review is done to study the existing state of the art on deep generative models to identify strengths and weaknesses. To eventually be able to make a comparison and analysis of the VAE and variant approach to other deep generative models. The blurriness and results achieved in the VAE stage are found in other research too and are therefore not specific to this problem in the forestry field.

The trained VAE models are further down in the pipeline used for the sampling of triplet images, where based on an anchor image, positives and negatives are sampled through a proposed algorithm. Creative and feature-rich triplets are sampled with the selected parameters. In the final stage of the pipeline the sampled triplets are used to train a log end recognition model. Unfortunately the recognition models trained on triplets perform poorly on real data and the VAE generated images with sampled triplets are at this stage not able to contribute to improving a recognition model based on sampled images.

All the results found in this research construct a good framework for future research and development.

## 1.5 Contributions

This thesis project contributes to the field of log end image generation and recognition, but also and especially to the field of generative models with deep learning.

This thesis project presents a variant VAE approach for improving the latent space and a triplet sampling algorithm on the VAE latent space. And all of this applied on the generation of artificial images on tree log ends. To the best of my knowledge and findings, these proposed approaches are novel and therefore their results provide an idea of its' capabilities within the field.

### 1.6 Target groups

This thesis project and research can be of interest to AI researchers, especially researchers working on generative models and sampling. It can also be of interest to AI focused companies working with computer vision, as the introduction of generative approaches could lead to more data generation and less costs on data collection if and where possible. On top of that, since the application of this thesis project is on the generation of artificial images of tree log ends, another target group are the researchers and companies working in the wood-processing and forestry industries. As with future work, the implementation could prove to be highly useful in many applications, not only restricted to the generation of tree log end images, but in any image synthesis.

### 1.7 Ethical Considerations

This thesis project and the scientific approaches did not include people and therefore no identity related data has been collected. Thus on that basis, there are no ethical concerns. However, the research done in this project contributes to the field of generative models and deep learning, and because of this there are some ethical concerns to be taken in account.

The first ethical consideration found within generative models is the issue regarding that these models may contribute to discrimination in any form, and that they have the capability of generating content that reproduces harmful stereotyping. This would be caused by being trained on data that already contains any real world stereotypes or biases. Even if this is not the intention with the model in the first place, the explainability of AI, and especially deep generative models, does not allow for much insight in this problem. The second ethical consideration on generative models is the problem of misinformation and applications, such as deepfakes. Deep generative models have become very good at synthesis with highly realistic outputs. This could be used with bad intentions where people may attempt to generate outputs with the goal of spreading misinformation or attacks on other people or institutions. An example of misinformation generation could be political or economical propaganda. An example of any attacks is the problem of deepfakes, where content is generated using individuals that often did not consent to be in the content. In generative models there have been many suggested approaches regarding tackling privacy concerns. An example of this is DALL-E [3] being unable to generate images with celebrities in it, to avoid deepfakes and spreading of misinformation on celebrities. Research in AI safety is a very important part of the field and helps mitigating these problems. Governments also start to recognise and tackling these problems with laws and studies, such as the study on tackling deepfakes in European policy [19]. In the study of the European Parliament [19], three different types of harm are described with risks: psychological, financial and societal harms. As further described on the list in Figure 1.3 below.

| Psychological harm | Financial harm | Societal harm |
|---|---|---|
| • (S)extortion<br>• Defamation<br>• Intimidation<br>• Bullying<br>• Undermining trust | • Extortion<br>• Identity theft<br>• Fraud (e.g. insurance/payment)<br>• Stock-price manipulation<br>• Brand damage<br>• Reputational damage | • News media manipulation<br>• Damage to economic stability<br>• Damage to the justice system<br>• Damage to the scientific system<br>• Erosion of trust<br>• Damage to democracy<br>• Manipulation of elections<br>• Damage to international relations<br>• Damage to national security |

Figure 1.3: List of different types of risks associated with deepfakes. [19]

Both the ethical considerations mentioned above do concern with the outputs of generative models, but the intentions are different. As with the output discrimination, there might not be a bad intention behind it. However with the misinformation there is a bad intention behind the usage of such deep generative models. Both of these risks are concerning and we recognise that there is a high need and importance for AI safety and methodologies recognising and mitigating these problems as much as possible. Especially towards the future with these generative models becoming stronger and better. There will be harmful sides with any technology, and it is something that should be always discussed and approached in research. We recognise that the good sides do outweigh the bad sides, as the powers of these models could simplify many problems appearing in humanity in the future and with this, steps can taken towards a better world where AI plays a large role in automation.

## 1.8 Report Structure

This thesis report is structured as follows: Section **2** provides a complete and base background on deep learning, deep generative models, latent sampling and the triplet loss, the main concepts connected to the research. Section **3** describes the scientific approaches taken in this research, and discusses the reliability and validity of this research. Section **4** contains the literature review done on deep generative models and describes the related work findings with a comparison. Section **5** presents the implementations of this thesis project, with the pipeline including multiple neural networks and a proposed sampling algorithm. Section **6** describes the experiments done in the thesis project and the assessment of the pipeline. Section **7** discusses the research questions again and the results that have been found. At last, Section **8** provides the conclusions of the research, implementation and experiments. Afterwards, directions and suggestions for future work are given.

# 2 Background

In this section the goal is to provide an understanding of the state of the art, in order to advance to the scientific approaches and practical implementations of this thesis project. This section: **2.1** introduces the basics of deep learning, while **2.2** covers an introduction to the field of generative artificial intelligence. Afterwards the most well-known and used deep generative models are presented, with the **2.3** Variational Autoencoders, the **2.4** Generative Adversarial Networks, and the **2.5** Flow-based Generative Models. Next, in **2.6** an introduction to latent sampling is given. Finally, **2.7** delves into the concept of Triplet Loss and Generation.

## 2.1 Deep Learning

In this subsection a basic introduction is given to the field of deep learning and neural networks. A base understanding of this is expected for the introduction of generative models, as these are based on neural networks. This subsection introduces the relevant deep learning basics for this thesis project, and is therefore no complete introduction on deep learning. If one already has basic knowledge on the field of deep learning, this subsection can be skipped and one can immediately go to the section of **2.2** Generative Artificial Intelligence.

### 2.1.1 Neural Networks

A neural network is a type of machine learning algorithm that was originally inspired by the human brain. The network represents a structure of layers existing out of interconnected neurons. Just as the human brain, the neurons/brain cells are found in highly complex networks with many connections, and are used to process information by sending electrical signals through these networks. However, a neural network in machine learning, although inspired by the human brain, still differs and does not resemble the same complexity and understanding of the world as the human brain does.

The goal of a neural network, is to learn a mapping or function $f*$ that approximates the output $y = f(x; \theta)$ as accurately as possible, where $x$ is the input data, and $\theta$ represents the network parameters, eventually trained to estimate the best approximation. The parameters $\theta$ are learned with a technique called stochastic gradient descent, using a cost/loss function that minimises a certain error or maximises a certain score. Backpropagation is used to adjust $\theta$ during the learning. In this section, the methodologies and techniques are further explained. A neural network, as represented in deep machine learning can be seen in Figure 2.4. It exists out of three main different components as layers:

- *Input layer*
  The input layer is the first layer of the network and is used to process the input raw data into the network, in the correct shape and form, to further pass it on to the hidden layer(s), in a computable way.

- *Hidden layer(s)*
  A hidden layer, or set of hidden layers, can be found between the input and output layer of the neural network. There are many types of hidden layers, such as Dense, Convolutional, LSTM, and others that can be used depending on the data to be processed and the problem statement at hand. The main goal

of hidden layers is to learn a certain representation and structure in the data by tuning the parameter values of the neurons and layers. The parameters are learned during the training of the neural network. The reason why these layers are called "hidden" is because of their goal, they estimate and learn a structure, without knowing the function or distribution of the data itself. The outputs of these layers are often difficult for interpretation and hard to correlate with a real meaning or output data type. Therefore, neural networks also heavily suffer from explainability issues.

- *Output layer*
  The output layer is the final layer of the neural network architecture and is used for the final processing of the output, coming from the final hidden layer. The output layer processes and returns an output that is interpretable for a human to use. This can be a numeric value, a set of probabilities, an image or even another output, depending on what is set and on the problem at hand.

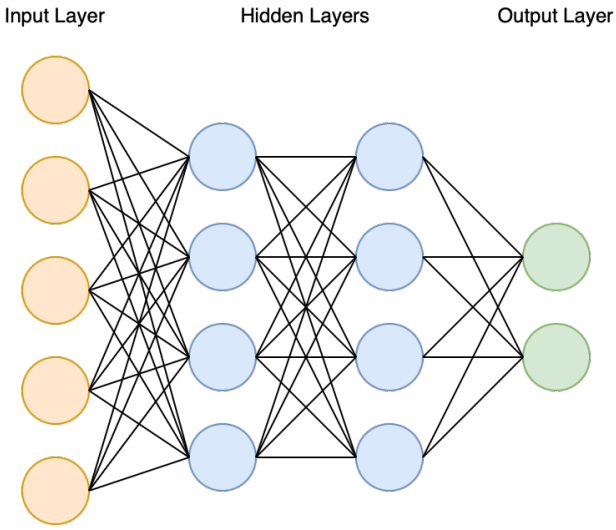


Figure 2.4: Basic feed-forward, densely connected, neural network architecture with an input layer, hidden layers and an output layer.

A layer itself can be of different types, such as densely connected (dense) or convolutional, each of these types of layers have their own set of hyperparameters that can be defined to affect and direct the learning in a certain way. A layer in the end exists out of different output units/neurons that have learnable parameters, weights and bias. On top of that it has an activation function that takes in the parameters and computes the output value of the neuron. During the training of the neural network, these parameters are adjusted all the time, based on a defined optimisation function. The formula of the output of a neuron (Equation 1) can be expressed as:

$$y = f(w^T h + b) \tag{1}$$

Where $y$ is the output, $f$ is the activation function, $w$ is the weight vector, $h$ is the input, and $b$ is the bias parameter.

- *Weight*
  The weight parameter of a neuron is a learnable value representing the strength of a connection between two neurons, and therefore how much a neuron should weigh through. The weights a certain neuron receives determine its own value and affects the output. This parameter is used for computing a weighted sum of inputs.

- *Bias*
  The bias parameter is a single learnable value that each neuron has, it is used for adding a bias to the weighted sum of inputs.

- *Activation*
  The activation is a function that defines the final output of a neuron by inputting the weighted sum of inputs plus bias through a predefined mathematical function. The activation is used to add non linearity to the neurons, in order to learn more complex structures and patterns in the data, which would not be possible while being restricted to linearity. Common activation functions are ReLU (Rectified Linear Unit), Sigmoid, and Tanh.

Weights and biases are usually randomly initialised, often through a Uniform or Gaussian initialiser, or are zero-initialised. The initialisation depends on the network and layers as well. The activation unit is commonly equal for the whole layer and typically predefined by the researcher, otherwise a default or no activation function is used.

### 2.1.2 Training

With the knowledge on layers, parameters and activation's, the next step is the training of a neural network architecture. For the training of neural networks, a technique called stochastic gradient descent is used, as often used in many other machine learning methods.

**Cost Function**
In order to evaluate the quality of the model, and compute the difference in the approximations made by the neural networks with the actual output that it should approximate as closely as possible to, a cost function is necessary. The choice of cost functions highly impacts the learning and is therefore an important choice that one has to make when training networks. There are many types of cost functions, where the goal is either to minimise the cost, such as error cost functions, like the MSE (Mean-Squared Error) and the MAE (Mean Absolute Error). Or to maximise the cost, such as an accuracy or correctness function. The choices are very dependent on the type of network and the problem statement/use case. A combination of different cost functions in one function is also common, where there is some adversarial or multi-objective learning goal. A combination of cost functions can also be used as regularisation of the training, to prevent any overfitting, a possible occurrence during training where the network learns too well how to approximate the training data, such that it no longer becomes generalisable over unseen data. This is why validation and test data sets are also used. The cost function is used in the gradient descent algorithm in order to optimise it and descent the gradient so it achieves a better cost.

**Gradient-Based Optimisation**
Gradient descent is an optimisation technique, that based on the cost function descents the gradient to minimise or maximise the cost function. The technique is used to iteratively adjust the parameters of the network, in other words the weights and biases, in the direction of the steepest descent using a certain learning rate. The learning rate is a step size used to descent the gradient. It is a parameter that allows to take smaller steps down the gradient in order to avoid overshooting the gradient, but it also allows to take large enough steps to not learn too slowly or end up stuck in local minima/maxima. The gradient is computed using the partial derivatives. The formula for gradient descent (Equation 2) is expressed as:

$$\theta^* = \theta - \alpha \cdot \nabla J(\theta) \tag{2}$$

Where $\theta^*$ represents the next set of parameters, $\theta$ represents the current set of parameters, $\alpha$ represents the learning rate, and $\nabla J(\theta)$ represents the gradient of the cost function $J$ over the current parameter set. Therefore, by iteratively computing the gradient and cost functions, the parameters $\theta$ of the neural network are adjusted during training. To mitigate problems occurring during the gradient descent, many optimisation techniques [20] have been proposed, each with their own specifics. The most common ones are Adam [21], AdaGrad [22], and RMSProp. Therefore, for training neural networks, one would always opt for one of these optimisation techniques. This is nonetheless, together with the learning rate, quite an important decision to make, as the adjustment of either the learning rate or optimisation technique highly affects the learning. One can however rely on the default settings used in other networks. In Figure 2.5, four gradients for four different networks and their loss functions are visualised. This illustrates that gradient descent is not always as straight forward as a convex gradient and that gradient-based optimisation can definitely easily end up in a local minima.

**Initialisation and Iterative Optimisation**
As neural networks introduce non-linearity with their nature, the optimisation becomes a non-convex problem. It therefore has no guarantee of convergence, can end up in local minima/maxima and is highly sensitive to the initialisation. This is also why randomised initialisation highly impacts the learning and thus two training's with different randomised initialisations can end up with different training progress and final loss results. The gradient descent algorithm is ran iteratively, this either with a predefined number of iterations or with a stopping condition. In the case of predefined iterations, a number of *k* epochs is set. An epoch represents a complete iteration of the training data through the network. Or until a defined stopping condition is reached; this is often used as regularisation of the network. A common example of this, is early stopping, a technique that interrupts the training of the neural network after *k* epochs because of insufficient improvement in the training over either one or multiple epochs. Although a global minimum/maximum is the ultimate goal of gradient learning, reaching this is most likely not going to be the case. The complexity can become very high, especially the larger the network gets, as the gradient usually becomes non-convex, causing possible problems that can occur during learning. Often the network settles for some local optimum. Figure 2.5 can help understanding that an initial random initialisation can very easily affect where the gradient-descent algorithm ends up, especially with VGG-56 and VGG-110, where there are many local minima.

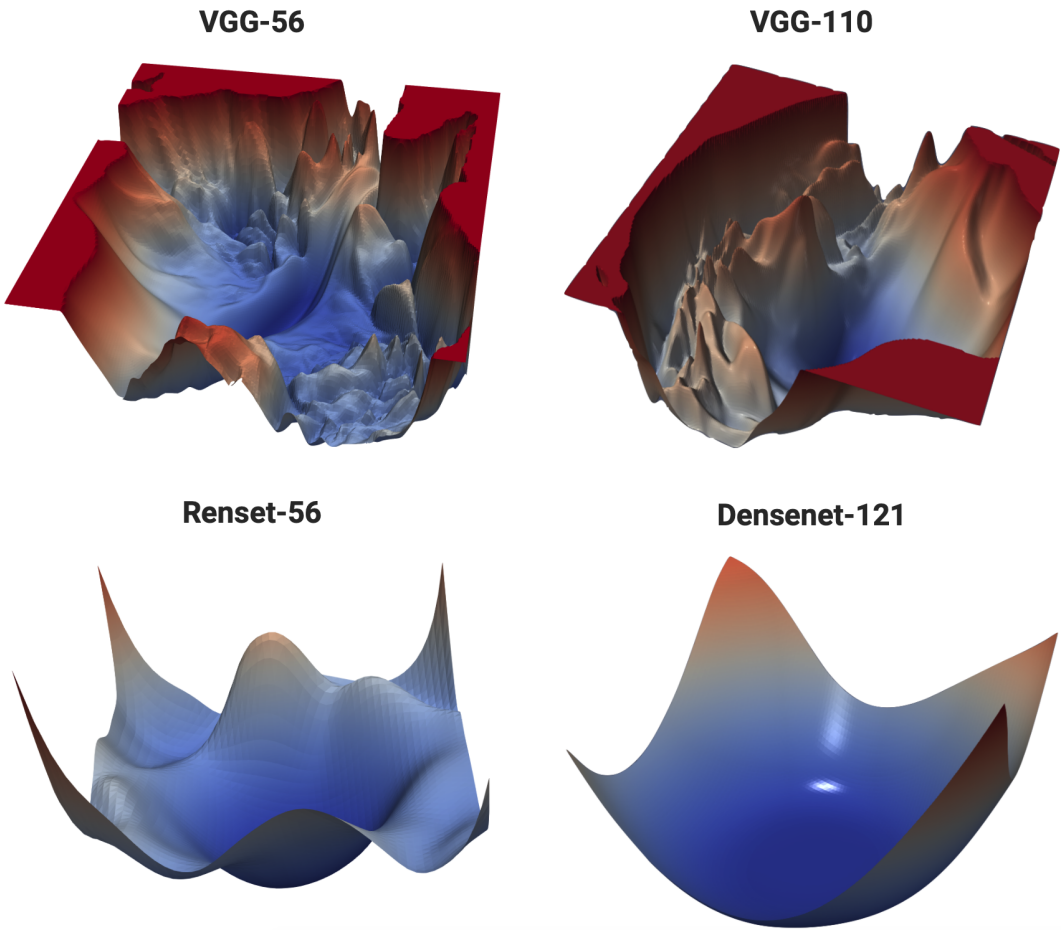


Figure 2.5: Gradients of the loss functions of different neural network architectures. Figure obtained from: https://www.cs.umd.edu/ tomg/projects/landscapes/

**Training Batches**
Gradient descent is usually applied using a batch version of the algorithm, meaning that subsets and thus smaller samples of the full training data are used for descending the gradient, instead of the full data set. In every epoch, gradient descent is applied for $TrainingDataSize/BatchSize$ steps. A batch version of gradient descent is mainly applied for computational efficiency, with the trade-off of a less accurate and generalisable gradient that represents the full training data. An example of this is the usage of large image input into a neural network, without any batch gradient descent, the approach would run out of memory very quickly. While with a certain batch size, it becomes computationally feasible. It is therefore of high importance to adjust the batch size hyperparameter accordingly when training neural networks. A very low batch size would give less generalisable gradients for the full training data, while a too high batch size would run into computational problems.

### 2.1.3 Backpropagation

Backpropagation is an essential part of deep learning and having a notion of it is important for understanding what happens during the training of architectures. Backpropagation is an algorithm used for training a network together with an opti-

misation algorithm such as gradient descent. The algorithm is used for propagating backwards through the architecture, i.e. from output to input, to compute the gradient (derivatives) based on the cost function for a single training example, using the current weights of the network. The algorithm outputs the gradient for every weight in the network and defines how a single training example would want to adjust the parameter in order to minimise the cost function. Doing this for a whole batch, the average of these gradients can then be used for the gradient descent algorithm to descent towards the lowest error and thus optimum with a certain learning rate. The algorithm thereby allows propagation through the network backwards, and through that computing the relative changes to decrease the error through the gradients for the learnable parameters.

### 2.1.4 Layers

There are many different types of layers available for neural networks. Below only the layer types relevant for the implementation of this thesis project are described.

*Dense layers*
A dense layer, also densely connected layer, is a type of layer where each unit/neuron in the layer is connected to every unit/neuron in the previous layer of the network. It is one of the more common deep learning layers, and can be used in many use cases.

*Flatten layers*
A flatten layer is a type of layer where a multidimensional input is outputted to a single dimensional output, in other words one larger vector. This layer is often used between convolutional layers and dense layers to handle the differences in output/input.

*Convolutional layers*
A convolutional layer is a type of layer that is quite different compared to the dense layer. The application of it is in the Convolutional Neural Network (CNN), which is used for computer vision applications such as images or time series problems, as it is great for learning and detecting features due to its nature. A convolutional layer and network is based on the idea of convolution itself and has a more grid-like topology. Convolution is the mathematical operation where a new function is computed based on two given functions. The computation itself is defined as the integral of the product of the first function and the reflected and shifted second function. In image processing and deep learning, this is seen as the data being the first function and a grid-like kernel of size *n x m* being the second function. In image processing, such as blurring or edge detection, the kernel shifted over the data contains certain hard-coded values over the kernel cells, while in deep learning this kernel is learned during training. There are a few main parameters on convolutional layers:

- *Kernel*
  The kernel parameter is equal to the size of the grid-like kernel to be learned. The dimensions of this kernel is dependent on the type of layer and the input data, for images this would commonly be two-dimensional. In the case of

multi-dimensional kernels, the sizes are often equal and thus square. Common sizes of kernels are 3x3 and 5x5. Odd-sized kernels are also preferred over even-sized kernels, because they have an exact center grid.

- *Filters*
  The filters parameter is equal to the amount of output dimensions of the convolutional layer, this thus represents how many filters or kernels one wants to learn in a single layer.

- *Strides*
  The strides parameter represents the number of steps the kernel takes when shifting over the data. Default this would be equal to one and the kernel filter then moves normally, if a higher stride is set, the kernel skips $x$ grid cells. This behaviour can be preferable in certain use cases such as very large dimensional data. Although it can speed up computation and learning, it does have a trade-off for the accuracy and detail.

- *Padding*
  One can understand that when a kernel moves over an image for example, the least information is captured at the borders, as only the borders of the kernel move over it. While towards the center, the kernel will move over every cell completely. Meaning information can be lost at the borders of images. A padding parameter allows to still maintain proper detection at borders and controlling the output space, as the sides of the images can be padded. Usually those padding borders are filled with zeros.

- *Activation Function*
  The activation parameter can also be chosen to further process the output filters. ReLU is a common activation function for convolutional layers, with Sigmoid and Tanh being popular choices for final convolutional layers in CNN's.

## 2.2 Generative Artificial Intelligence

Generative Artificial Intelligence refers to the field of AI models being capable of generating non-existent data, by mimicking a real world data distribution. This is different from the usual AI models that attempt to make a prediction $y$ based on a given input $x$. With generative models, the goal is to generate new data. There are generative models that do not require any input, but there are also models that can take in certain input/conditions to steer the model into certain directions. One can think of image generation based on a text prompt. Generative AI has many applications such as text-, audio-, video-, image- and molecule-generation. Deep learning has proven to significantly advance the field of generative AI with different approaches in the different types of generation. These introduced methodologies belong under the category of deep generative models. Deep generative models have received more and more attention with the uprising amount of introductions for public/commercial use, with models such as text-to-image generation approaches of DALL-E-2 [4] and Stable Diffusion [5] or image editing and enhancement approaches with Generative Adversarial Networks.

Image generation or synthesis itself is the process of artificially generating images by learning some underlying distribution and features from a training set. The

focus of this thesis project is on image synthesis based on image training data, where the aim is to learn an underlying distribution from images and being able to translate and sample from this learned distribution. The main deep generative models within this field are variational autoencoders, generative adversarial networks, and deep flow-based generative models.

## 2.3 Variational Autoencoders

### 2.3.1 Introduction to Autoencoders

An autoencoder [8] is a type of neural network architecture. The main goal of an autoencoder architecture is to learn a compressed and lower-dimensional representation of the input data, e.g. an image, as well as possible. One can understand this as an algorithm that attempts to learn a dimensionality reduction algorithm as accurately as possible, so that it can retain as much information as possible in a reduced lower level representation. The architecture exists out of two different networks, an encoder and a decoder, both of these network components are necessary for autoencoders to learn a lower dimensional representation. In the encoding network, a given vector of $n$ size is transformed into a vector of $p$ size, where $n > p$. The goal of this autoencoder component is to learn the mapping from a high-dimensional input to a lower-dimensional representation. The decoding network on the other hand, transforms the learned lower-dimensional vector of size $p$ into a higher-dimensional vector of size $n$ again. Thus the goal of this component is to do the reverse of dimensionality reduction. This is necessary in order to learn how good the lower-dimensional representation actually is, as now both the input for the encoder and output of the decoder can be compared. The autoencoder network can learn to map the output of it as closely to the input. The standard autoencoder is visualised on Figure 2.6 using MNIST [23] as example data. The reconstructed output is slightly different than the given input due to the reconstruction of it's encoding. The reconstruction error is a commonly used loss function for autoencoder architectures, where the error roughly represents the difference between the input vector and output vector of the architecture. The specific reconstruction error can be different based on use cases, commonly the mean squared error (MSE) (Equation 3) is used as the loss.

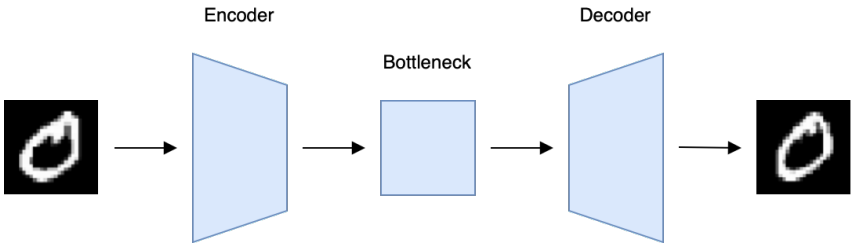


Figure 2.6: Autoencoder architecture using MNIST [23] as example data.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2. \tag{3}$$

The hidden layer in a neural network which represents the low-dimensional representation of size $p$, between the encoder and decoder, is commonly called the

bottleneck. In both the encoder and decoder, multiple hidden layers are sequenced and connected with this bottleneck. In an encoder, the layer size would sequentially output smaller and smaller output representations. In a decoder, the reverse happens where sequentially output representations of the layers grow larger, until the output of size $n$ is reached again. While training, the loss function is used to tune the weights and biases of these layers, in an attempt to learn a model where the hidden layers output a minimised loss function.

The autoencoder has been applied to different problems in various fields, especially in the field of optimisation, recognition and anomaly detection. In optimisation, the network can be useful for denoising data, e.g. denoising of medical images [24] or blind denoising [25]. Or optimisations as in enhancement data, e.g. a coupled deep autoencoder for enhancing image resolution [26]. Also recognition and anomaly detection have been proven to be a useful case of deep autoencoders [2] [27] [28].

### 2.3.2 Variational Autoencoder Architecture

A base autoencoder has not been considered as a deep generative model, due to the model just reconstructing based on a lower dimensional version. The variational autoencoder [7] architecture however has been. The VAE, short for variational autoencoder is a generative model based on the autoencoder, with the difference that it not just learns a bottleneck. Instead it aims to learn a latent space with latent distributions, i.e. the underlying true distribution of the inputted data. In the architecture, an input vector of size $n$ is compressed to a lower dimension in the encoder, of which the latent space of $D$ dimensions are learned during training. The decoder then decompresses the latent vector, which is sampled from the latent space, into a vector of the original size $n$. The variational autoencoder is visualised in Figure 2.7, with the latent space component added to the architecture.

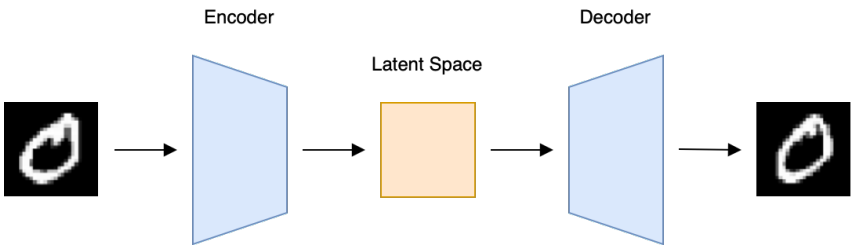


Figure 2.7: Variational Autoencoder architecture using MNIST [23] as example data.

There is more to it than just a rather simple architecture, learning a true underlying distribution of a data set is a problem that can not just be easily solved, especially with a VAE architecture due to its nature. The goal of a VAE model is to learn a proxy distribution which is an approximation of the posterior distribution, by maximising a lower bound on the log likelihood. The learning and specifics of the VAE will be further discussed in this section, but will not go into all the details, especially regarding all the mathematics and fine-grained specifics. As the focus here is on apprehending an understanding of the architecture itself. For more details than described, the following references are suggested. Kingma and Welling [29] present an extensive and detailed introduction to variational autoencoders. Cinelli

[30] presents a book on variational methods for machine learning with applications on deep networks, which covers extensive details on learning a variational method such as the VAE model.

### 2.3.3 Loss function of a VAE architecture

**Distributions terminology**

- $q_\phi(z|x)$: Inferenced and approximation of the posterior distribution. Related to the inference/encoder model.
- $p_\theta(z|x)$: Posterior distribution, often intractable due to complexity and dimensions.
- $p_\theta(z)$: Prior distribution.
- $p_\theta(x|z)$: Stochastic decoder for reconstruction based on the sampled vector from the latent space.

**Maximum log-likelihood**
The most common metric used for probabilistic models is the maximum log-likelihood function, it is an important way of evaluating the quality of a probabilistic model and how well the model represents the observed data. The parameters $\theta$ for a probabilistic model can thus be optimised by maximising the log-likelihood, also equivalent to minimising the Kullback-Leibler divergence, between the observed data and the model distributions. The maximum log-likelihood estimation (MLE) is usually optimised in deep networks through the technique of batch gradient descent. MLE (Equation 4) is therefore the estimation of the parameter set $\hat{\theta}$ through the maximisation of the log-likelihood loss function $\mathcal{L}_n(\theta\,;\mathbf{y})$ of the model distributions and the observed data, where $\theta \in \Theta$ and $\Theta$ represents the parameter space.

$$\hat{\theta} = \arg\max_{\theta\in\Theta} \mathcal{L}_n(\theta\,;\mathbf{y}) \tag{4}$$

The goal of the encoder network is to approximate the posterior distribution by optimising the maximum log-likelihood. The encoder, also called parametric inference model, aims to to learn a parameter set $\phi$ parameters of the latent space as well as possible. This through a process called variational inference or posterior inference (Equation 5), the process that attempts to find a set of parameters that minimises the difference with the posterior distribution as much as possible.

$$q_\phi(z|x) \approx p_\theta(z|x) \tag{5}$$

The neural network learns $\phi$ through the weights and biases of the hidden layers, usually dense hidden layers.

**Kullback-Leibler Divergence**
The KL-Divergence, short for Kullback-Leibler Divergence, is a commonly used measure for determining how different a certain probability distribution $P$ is from another probability distribution $Q$. For discrete distributions the measure (Equation 6) is given as:

$$D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left( \frac{P(x)}{Q(x)} \right) \tag{6}$$

For continuous distributions, the measure is transformed to instead use integrals (Equation 7):

$$D_{KL}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \left( \frac{p(x)}{q(x)} \right) dx \tag{7}$$

Simply put, the KL Divergence is the expectation of the log difference between $P$ and $Q$, using the probabilities $P$. It is also important to note that the Kullback-Leibler is a distance, but is not considered a distance metric. This because the distance does not meet the symmetric requirement of distance metrics, as switching $P$ and $Q$ will result in a different distance.

**Evidence Lower Bound**

The optimisation function or the loss of a VAE architecture however, is the Evidence Lower Bound, ELBO function, it is also called the variational lower bound. It is an optimisation function often used for variational inference. As mentioned above, computing the posterior distribution directly is often found to be intractable, due to complexity and dimensions of data. Therefore with the ELBO metric, one can obtain a lower bound on the maximum likelihood of the data, and obtain an approximation on the posterior distribution. Through the metric, the parameters of the chosen distribution can be optimised, resulting in an estimated distribution that attempts to be as close as possible.

$$L_{\theta,\phi}(x) = \log p_\theta(x|z) - D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) \tag{8}$$

$$L_{\theta,\phi}(x) \leq \log p_\theta(x|z) \tag{9}$$

Full derivations of the ELBO can be found in [29]. Essentially, the formula (Equation 8) provides the maximisation of the log-likelihood (first part of the formula) and the minimisation of the KL-Divergence (second part of the formula). The KL-Divergence is non-negative, therefore the ELBO loss is always smaller than the log-likelihood (Equation 9). However, the KL-Divergence can equal zero, in the case that both distributions are exactly the same, and the approximation equals the posterior distribution, making it no longer an approximation. In the case of a zero KL-Divergence, the ELBO equals the maximum log-likelihood. This is only possible if the exact posterior has been found; in VAE training cases, this is usually an unseen case. As the ELBO also uses the KL loss as regularization for not just directly mapping the data with the reconstruction loss. Trade-offs between reconstruction and KL loss are constantly done during learning.

### 2.3.4 Learning and optimisation of the ELBO

The Evidence Lower Bound loss function can be optimised with the deep neural network, through gradient descent. During the gradient descent, the full network adjusts the parameters of the hidden layers, aiming to minimise the loss function and descending the gradient in order the approximate or find the global (or often a local) minima. Computation with VAE's can be rather complex, especially when

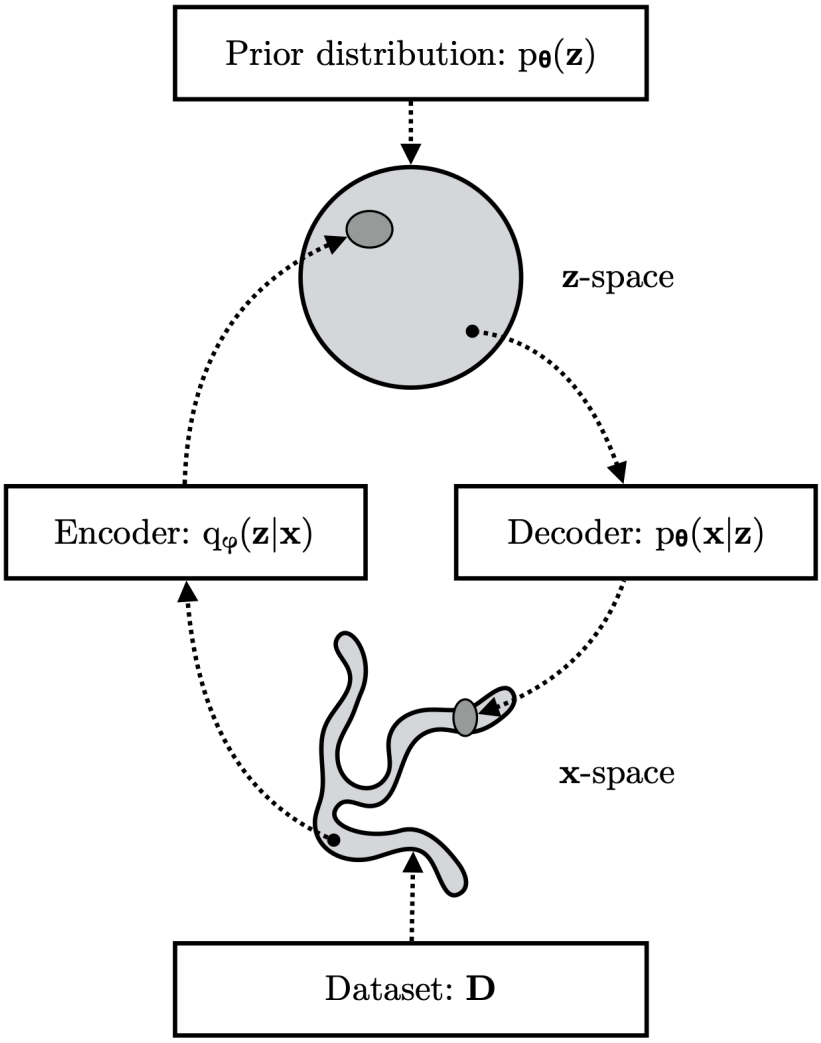


Figure 2.8: The process of VAE learning. In the encoder an approximation $q_\phi(z|x)$ of the intractable posterior distribution is optimised through the learning of distribution parameters in the hidden layers of the deep neural network. This results in a prior distribution $p_\theta(z)$ in the latent space or z-space. The stochastic decoder $p_\theta(x|z)$ further reconstructs by sampling from the z-space. Figure obtained from [29].

using large data or images. Usually a converge rule is set to stop the learning when it has reached predefined criteria. The ELBO over a data set is the sum or average, depending on the implementation, over all the individual ELBO functions over the data-points in the data set. The individual ELBO over a data-point $x_i$ in a data set $X$ is usually intractable, however good unbiased gradients can be obtained. In discrete distribution cases, this is easier to compute. In continuous latent spaces, this is not as straight forward, the parameters can then be computed through the reparameterisation trick. The reparameterisation trick is a methodology for the network that allows for backpropagation while still sampling.

Lastly, important to note on the learning of the latent dimensions is that usually a multivariate Gaussian model with diagonal covariance is applied. Although other distribution models can also be used and have been depending on the application.

In the case of a Gaussian model, the network aims to learn two parameters per latent dimension: the mean $\mu$ vector and the log variance $log(\sigma^2)$ vector. The learning of the VAE is visualised in Figure 2.8.

### 2.3.5 Reparameterisation trick

In order to allow backpropagation in the neural network and optimise the ELBO function with it, the reparameterisation trick is introduced. This includes the introduction of a new variable $\epsilon$ on top of the other variables computed for the Gaussian latent dimensions. In the original form, it is impossible to run backpropagation through it, as the random node does not allow this. In the reparameterised form, the random node is moved to a new variable $\epsilon$, allowing for the backpropagation through the distribution parameters. The random variable $\epsilon$ is sampled from a Gaussian distribution with mean $\mu$ zero and standard deviation $\sigma$ one. The reparameterisation trick with the introduction of a new variable $\epsilon$ is visualised in Figure 2.9, where the original form and reparameterised form of the latent sampling is illustrated.

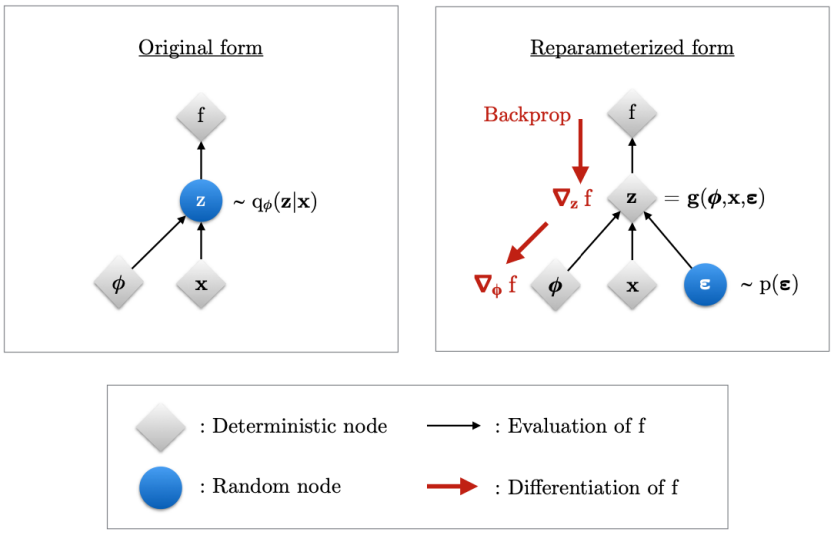


Figure 2.9: reparameterisation Trick. Figure obtained from [29].

### 2.3.6 Variants

Over the years of research, many variants have been proposed and introduced in the field of variational autoencoders. Some of the more commonly used and impactful variants are discussed below, found in the search for related studies and literature review. Overall, many variants also have their own variants. This to improve its architecture, loss functions, learning or other found weaknesses/issues.

**VQ-VAE and VQ-VAE-2**

The VQ-VAE [11] and VQ-VAE-2 [12], short for Vector Quantised-Variational Autoencoder, are VAE variants that have resulted in significantly high quality image synthesis. The difference between the VQ-VAE and standard VAE, is that the VQ-VAE learns discrete latent codes, instead of the standard VAE learning continuous latent codes, and the VQ-VAE learns the prior static. Having a discrete latent space,

instead of continuous, allows for more structured and interpretable representations. And furthermore easier sampling for synthesis. The Vector Quantised-Variational Autoencoder also mitigates the issue of mode collapse, which can appear with the base architecture. Due to the model learning discrete spaces, the VQ-VAE architecture definitely works well on data with more structure and discrete nature. [11] [12]

**Conditional VAE**
The conditional VAE [13] architecture is another variant of the VAE that is important to mention, as this extension has opened up for new applications within the field of the VAE. The base variational autoencoder does not have any conditional properties, therefore making it impossible to perform conditional synthesis of any kind. The addition of conditional properties allows for generation based on predefined inputs to the sampling model, such as a category, or data attribute. It has also found to be great for applications as image inpainting where it can conditioned on an incomplete image, generate the missing part and complete the image.

**$\beta$-VAE**
$\beta$-VAE [31] is a variant of the standard variational autoencoder with a minor change in the learning of the algorithm, that has significant impact on the latent space. The researchers of this approach introduced this variant with a focus on disentangling and organising the latent dimensions. This is done through a modification of the ELBO with the introduction of a new parameter $\beta$. The parameter multiplies the KL Divergence, putting more weight on that parameter of the ELBO. This results in more structured and disentangled learning of the latent space, with the trade-off on reconstruction quality of the network.

**NVAE**
NVAE [32], short for Nouveau VAE, is a deep hierarchical variant of the base architecture. Through the hierarchical structure of the convolutional blocks in the model, it learns the data to a deeper level. Often able to learn more complex features and attributes in that data, that a normal VAE would look over. Because of this, the reconstruction quality and learning is more stable and better.

## 2.4 Generative Adversarial Networks

### 2.4.1 Model

The generative adversarial network model, GAN shortly, is a generative model proposed by Goodfellow et al. [9]. Of all the deep generative models, GAN's have been applied the most with their very successful and high quality generation, although the other models are starting to catch up now. The GAN model itself exists out of two neural networks, a generator *G* and a discriminator *D*. The discriminator *D* is a network that takes in a generated input and a real input, and learns to correctly classify which input is real and which one is fake/generated. The generator *G* is a network that learns to generate data from random noise, it learns through the discriminator classifications. The GAN network with the generator and discriminator networks is visualised below in Figure 2.10.

So the goal of the generator is to generate output that comes from the same distribution as real data. The generator attempts to fool the discriminator and increase its classification error. The goal of the discriminator is to learn to classify real and

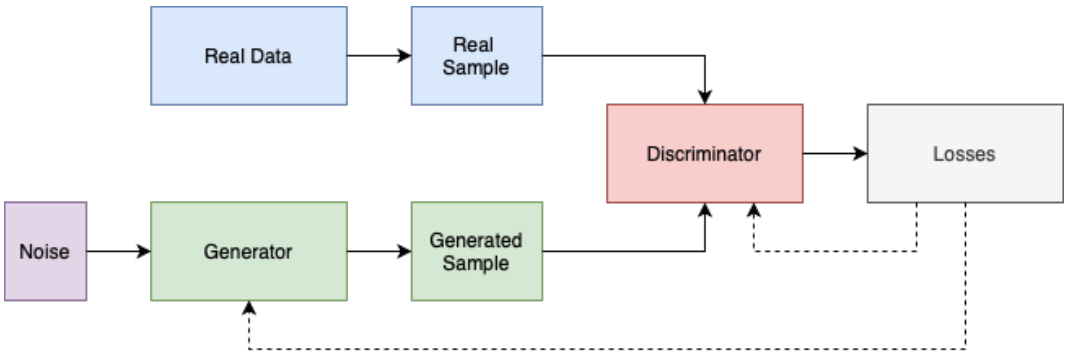


Figure 2.10: Generative Adversarial Network Architecture with the Generator and Discriminator networks.

generated as well as possible and achieve a classification error as low as possible. The generator network improves with the discriminator classifying, by using the classification feedback. One can understand this model and its training as an adversarial process, where the two networks play the minimax game. The minimax game can be understood as the game between the generator and discriminator networks, as improved generation would lead to worse classification. Improved classification would mean it is easier to classify for the discriminator and the generation is less close and realistic compared to the real data. By both of these models training with this optimisation, they both constantly attempt to improve and learn as well as possible. The networks combined have the value function (Equation 10) as:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))] \quad (10)$$

### 2.4.2 Loss Function

In GAN architectures, different loss functions can often be found and tuned specifically based on the problem and data. The generator and discriminator have different loss functions, as they optimise different objectives. The discriminator attempts to classify as correctly as possible to identify the real and generated images. While the generator attempts to fool the discriminator as much as possible by improving the generated images to be as realistic as possible. In other words, the generator uses the discriminator as a loss function. The network usually trains until a certain balance or equilibrium is reached in the two networks of the GAN, and not until one of the networks performs very well. The discriminator will in the beginning of training be able to classify quite well as it is still very easy then, while the generator will perform badly.

**Binary Cross-Entropy**

The binary cross-entropy loss or log loss is one of the very common, if not most common loss function used for discriminators, as it is well fit for binary classification. The binary cross-entropy (Equation 11) measures the dissimilarity between two distributions, the predicted and true distribution. It does this by comparing the real binary classification with the predicted probability of the output being that class. It therefore penalises losses being away from the true probability.

$$L(y, \hat{y}) = -[y * \log(\hat{y}) + (1 - y) * \log(1 - \hat{y})] \tag{11}$$

**MiniMax GAN Loss**
The minimax GAN loss (Equation 12) is the loss presented in the GAN introduction paper [9] and has already be introduced above. The first part of the loss represents the average of the log probability of real data, while the second part of the loss represents the log of the inverse probability of generated data. The whole minimax loss is what the discriminator attempts to maximise, while the generator minimises the second part of the loss function.

$$L(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))] \tag{12}$$

In the Appendix Section **A.1**, alternate GAN loss functions and variants are described for the interested reader.

## 2.5 Flow-based Generative Models

### 2.5.1 Model

The flow-based generative model is another deep generative model. The model utilises the concept of normalising flows in order to learn the probability distribution of given data as accurately as possible, similar to what the VAE does. The framework was introduced in Tabak and Vanden Eijnden [33] and Tabak and Turner [34]. Normalising flows became more well-known with the introduction of variational inference with normalising flows by Rezende D. and Mohamed S.[35], and the NICE framework by Dinh et al. [10].

Normalising flows is a concept where a simple distribution is transformed into a more complex distribution, through a set of invertible and differentiable mappings, using the change of variables rule for the transformation. By having invertible mappings, the flow-based model can be trained with gradient descent, as forward- and back-propagation become possible, allowing for estimating the density of the distribution by inverting generated samples. So through these flows, one can approximate a significantly more complex and powerful distribution of the data compared to using a simple distribution such as Gaussian. This also means that a flow-based model in theory can estimate the exact density of a data distribution and therefore why the model maximises the exact log-likelihood instead of a lower bound as used in variational autoencoders. In practice and in case of large data set, it can achieve an accurate approximation of the exact log-likelihood. In Figure 2.11 an illustrative example of normalising flows is shown.

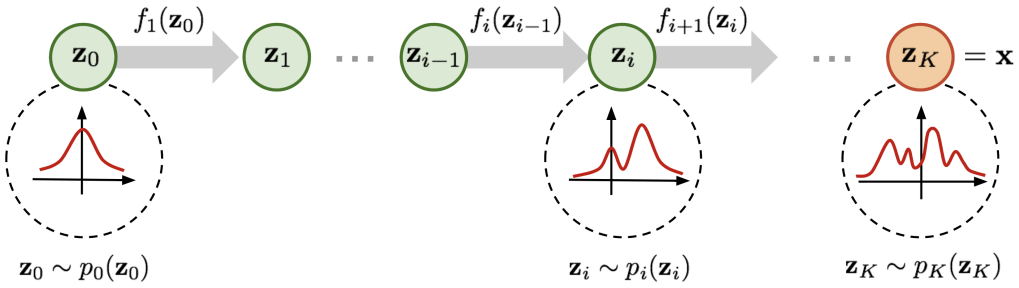


Figure 2.11: Concept of normalising flows, transforming a simple distribution into a complex distribution using invertible mappings. Figure obtained from: https://lilianweng.github.io/posts/2018-10-13-flow-models/

Different types of flow architectures have been proposed over the years, with each having their variants of models. The most prominent types of flows are coupling flows, autoregressive flows, residual flows and continuous flows. Coupling flows have proven to be the most commonly applied with variants as NICE [10], RealNVP [17], and GLOW [18]. Coupling flows have usually been an attractive approach due to their strengths of computational efficiency, scalability and flexibility in learning. Autoregressive flows have also been a powerful type of flows with variants such as Masked Autoregressive Flow [36], PixelRNN [37] and Inverse Autoregressive Flow [38].

In the Appendix Section **A.2**, some Flow-based model variants are described for the interested reader. For further introduction on flow-based models and these types of flows, one can read the introduction presented by Kobyzev et al. [39].

### 2.5.2 Loss Function

During the training of a flow-based model, the maximum likelihood estimation (MLE) is used as its loss function, as the density can be exactly estimated and is tractable with normalising flows. Therefore the maximising of log-likelihood is used, equivalent to the minimisation of the negative log-likelihood (Equation 13).

$$L(D) = \frac{1}{N} \sum_{i=0}^{N} -\log(p_\theta(x_i)) \tag{13}$$

The maximisation of the likelihood of the model under observed samples of the target distribution is equivalent to the minimisation of the Kullback-Leibler Divergence between the model's likelihood and the target distribution.

## 2.6 Latent Sampling

Latent sampling is the process of sampling from a learned latent space. Latent sampling is especially interesting in the case of variational autoencoder architectures, as the VAE offers a latent space that was learned during the training. And this VAE latent space of $D$ predefined dimensions, can be used to generate completely new samples, to then be decoded through the VAE decoder network, to

eventually return a newly generated image. As a VAE latent space is usually multi-dimensional, and contains more than 3 latent dimensions, it becomes harder for us humans to interpret such a latent space, as this can not be easily visualised. Especially for highly dimensional latent spaces, sampling techniques are required for more efficient generation. In this section, random Gaussian sampling and interpolation sampling are described, both are main sampling strategies used. These methods however can both return more worse and blurry results, or just overall bad generation, especially when the latent space was trained on smaller data. To solve problems appearing with the main sampling strategies, optimisations have been made in studies, such as presented in the research of White T. [40] through bias-corrected vectors with data replication and synthetic vectors with data augmentation.

**Random Gaussian Sampling**
A standard Gaussian distribution is often used for the latent dimensions of a VAE. Randomly sampling from every latent dimension in the $D$ dimensional space, following the mean and standard deviation parameters returned from the training, returns completely new latent vectors that can be decoded and turned into a new image. Through multivariate Gaussian sampling, vectors can be retrieved that contain learned features that do resemble the training data itself.

**Interpolation Sampling**
Interpolation sampling is another common sampling strategy, also known as linear interpolation. In interpolation sampling, two points, usually of training, validation or testing data, are chosen. Between these two points, encodings are sampled linearly in $x$ steps. With linearly, it is meant that exactly between two points, the shortest distance line is traversed in $x$ steps, and the steps are taken as the sampled encodings. These steps between the two prechosen points, can then be decoded and provide different samples, maintaining similar features as the prechosen steps. The strength of these features appearing depends highly on how close the point was sampled towards one of the reference points. On Figure 2.12 an example on linear interpolation is given, it is shown how towards both of the directions the images become more and more like the original reference image.

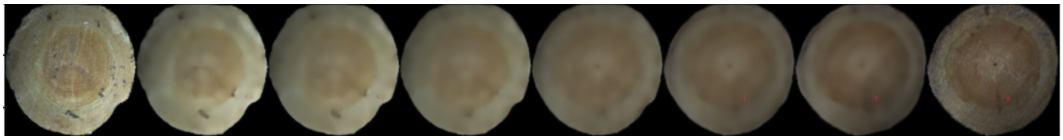

Figure 2.12: An example of linear interpolation retrieved from the generations further done in the implementation and controlled experiments on tree log ends.

## 2.7 Triplet Loss and Generation

The triplet loss [41] is a learning objective and loss function for improving the encoding space of a model, often for classification models. The goal of the triplet objective is to have the encodings (points in learned space) of the same classifications or similarity lie closer, while making unequal or dissimilar encodings more

distant. The loss function was introduced in FaceNet [41], an embedding for face recognition and clustering. The triplet exists out of 3 encodings:

- Anchor: the reference point in space, to which the positive and negative are compared.
- Positive: the point in space matching with the anchor, measured in distance from the anchor, aimed to be closer to the anchor than the negative.
- Negative: the point in space that does not match with the anchor, measured in distance from the anchor, aimed to be further away from the anchor compared to positive.

The goal is therefore to cluster similarities with these 3 encodings. While doing this over many triplets, the network learns to better identify and classify similarities and dissimilarities correctly. The loss has often been used for recognition models, to help the training more and improve the model quality and robustness. By making it harder to learn these triplets, the network becomes better. One can understand the loss function as the following goal (Equation 14):

$$||e(A) - e(P)||^2 \leq ||e(A) - e(N)||^2 \tag{14}$$

Where *A* represents the anchor, *P* the positive and *N* the negative, *e* represents the encoding function. The goal is to, at all times, have a smaller distance from the positive to anchor, than negative to anchor. The above goal can be easily transformed into the loss function (Equation 15).

$$max(||e(A) - e(P)||^2 - ||e(A) - e(N)||^2, 0) \tag{15}$$

If the loss is smaller than zero, the goal has been achieved. If the loss is equal, the two distances are the same. Note that equal distance does not mean it is the same point in space, but solely the same distance. Especially in multi-dimensional settings the distance can be the same, but the point can be different. If larger than zero, the negative is closer to the anchor than the positive, which is unwanted.

However, having both distances equal to zero, returns a loss of zero, so the network would not learn at all and encode all the triplets as the same point in space. All of this while the loss function would still be satisfied. To mitigate this learning behaviour, an extra parameter $\alpha$ is introduced, also called the triplet margin. This returns the full final loss (Equation 16).

$$max(||e(A) - e(P)||^2 - ||e(A) - e(N)||^2 + \alpha, 0) \tag{16}$$

The introduction of a triplet margin, enforces the model to learn better and separate dissimilarities more, on top of mitigating the above mentioned problem of no learning at all. The higher the parameter, the more stricter the model it becomes, and the more robuster the model can become. If the margin is set too high, it might learn to focus too much on the maximisation of the negative and anchor, that it no longer learns to minimise the distance between positive and anchor properly. If the margin is too little, the model might not learn sufficiently. A middle ground has to be found and the anchor parameter has to be more specifically tuned for optimal learning. Note that in the above formulas, the Euclidean distance is used, although this is one of the most common distance metrics, other distances can also be used if preferred.

# 3 Method

In this section the method of the thesis is described extensively in advance of the actual implementation. This section includes: **3.1** introduces the applied scientific approaches, with in **3.2** a description of the selected scientific methods. Eventually, in **3.3**, the reliability is assessed, and in **3.4**, the validity is covered.

## 3.1 Scientific approach

In this thesis project, two methods are applied in the scientific approach. First a literature review with qualitative focus is applied in order to identify the applications, variants, strengths and weaknesses of the generative models, to further compare the state of the art deep generative approaches. After a literature review, a controlled experiment is the next stage of the thesis. In the controlled experiment, quantitative results are produced in order to measure the generative quality of the VAE and the proposed variant, on the application of log end image generation. After that, a qualitative and quantitative assessment of the triplet sampling algorithm and the recognition model is applied, respectively.

## 3.2 Method description

### 3.2.1 Literature Review

As mentioned in **3.1** Scientific Approach, a literature review is applied for the identification of variants, applications, strengths and weaknesses. Especially the identification and analysis of variant architectures highlights the problems occurring with the base models. On top of this, the literature review helps answering the research question 2, on the comparison of variational autoencoders with other generative models. The literature review is however a rather short review of the current existing literature, as researching extensively into three different generative models in a correct scientific way, is impossible within the scope of a thesis project that also includes an implementation with a controlled experiment. The literature review is prepared with more specific goals/questions, a search procedure and inclusion/exclusion criteria for the extraction of primary studies. During the literature review, an extensive summary is provided per generative model, with more specific subsections and scopes of the primary studies per model. It is important to note that the literature review aims to follow the approach of a systematic review [42], but this review is smaller, less detailed and very much focused on qualitative directions instead of quantitative results. No quantitative results are reported in the literature review. More specifics can be found in Section **4** (Literature Review).

### 3.2.2 Controlled Experiment

In the controlled experiment phase, the models are tested on the application of generating images of tree log ends. The data returned from these experiments is quantitative data using different metrics in order to compare the quality of the proposed methodologies. First, a base VAE architecture is tested in the experimentation phase, after that the proposed variant is tested with. During the stage of the controlled experiment, both the models are improved based on the quantitative results and improvements through changes in the neural network, while still

maintaining the specific architecture itself. Here multiple quantitative results are returned, giving a good overview of the quality and learning of the models. After the models have been trained, the generation quality and sampling quality are also experimented with, through the training of a recognition model, returning even more quantitative metrics. In this way, not only the generative models can be compared, but the generative images can be compared with real data in the recognition model. This controlled experiment provides more specific analysis on the current state, and with this a good grasp for future work on this research. More specifics can be found in Section **6** (Controlled Experiment).

### 3.3 Reliability

The controlled experiments have some reliability weaknesses. Below these reliabilities are discussed, with arguments and counter-arguments.

The first reliability is the data collection for images. Log end images are retrieved from production pipelines. For this, a time frame of a week is set and all images from within the time frame are retrieved from storage. Of this larger set of images, from the selected week, a randomly smaller set of $n$ size is used for training the models. It could be argumented that this data collection approach might introduce any bias towards certain log ends due to random parameters and does not allow for reproducibility. A counter argument here, is that when scaling up in the data set size, it will reduce any possible bias towards certain types of logs as more data is used. The randomisation parameter does also remove any researcher bias as there is no control of the selected images for training the generative models. It could also be argumented that the selected time frame might introduce any bias, but overall the differences in different time frames can be considered minimal, and the trained generative model should be able to handle this. For future work, it could definitely be interesting to also use more data, and from within different time frames.

The second reliability is the usage of the data set itself, although the exact model and parameters for training are given in the Appendix **D**, the data set is private and everything is done within an industrial environment and setting. So the exact results and implementation are not reproducible, due to the privacy of the application. Obtaining a similar data set from a different source might return very similar results nonetheless.

A third reliability is the neural network. A neural network contains random parameters, and although one can set random seeds, to allow for exact replication, this was not done within this thesis project. The reason for this is that setting an exact random seed is good for reproducibility, but it does introduce a large bias for training the neural network and highly affects the training results. To avoid any large bias like this, the randomisation is kept. This does not allow for exact reproducibility and becomes a weakness on this part. However, to improve the validity and reliability of the approaches, the exact model architectures are discussed in Subsection **5.1** Design and Implementation and can be found in Appendix **D**.

The decision to reduce exact reproducibility of the training and the controlled experiment was done to avoid any researcher or seed bias. Nonetheless, with the exact parameters, the results of the deep learning process should not be too different with different randomisation's and initialisation, as the models have proven to be quite stable. So very similar results will be achieved when replicating the implementation and controlled experiment.

### 3.4 Validity

Validity and the possible initial threats in this thesis are extensively discussed below with the construct validity, internal validity and the external validity. At the end of the **6** controlled experiment, any other validity concerns found post-experimentation are discussed.

#### 3.4.1 Construct Validity

Construct validity concerns with the validity of the theoretical constructs and metrics. Within this thesis project, the default and most common metrics for the specific implementations are used. This includes the ELBO loss with the MSE reconstruction error and KL Divergence, the triplet loss and the accuracy of the recognition model. On top of this, the large amount of metrics and therefore different views on the generative quality, provides more validity to the constructs and interpretations. All the conclusions on results and performances of neural networks are fully based on the losses of the networks. The trainable parameters and network sizes are also given in the implementation section, with prediction time performances in the controlled experiment section. All the descriptions and conclusions are written in a way that avoids as much construct validity issues as possible.

#### 3.4.2 Internal Validity

Internal validity concerns with the validity of the results following the collected data and therefore how well the research rules out alternatives due to errors and biases.

A first measure taken to improve internal validity, is the decision of the usage of random parameters in the deep learning models to reduce any bias. As setting a random seed would bias the deep learning networks training and results wise, towards a better or worse results. To avoid any seed bias, the randomisation is completely kept in the deep learning network. Models are also trained for a larger amount of epochs, with all the same models having the same epochs, to avoid any initialisation bias, and therefore achieving consistent results when retraining networks. As mentioned above, this trade-off for internal validity is made with the reliability and thus the exact reproducibility of training these networks. As also mentioned in the reliability section, a small bias and therefore threat to internal validity might be introduced with the decision of the data collection time frame of log end images, but this is only considered to be a minor threat.

Another type of bias that can occur is the bias of strictly following metrics and quantitative results. During the implementation and experiments, qualitative visual analysis has also been done on the generative results and have been discussed with the supervisors, in advance of adaptations.

One other, rather large threat to internal validity, is the interpretability of neural networks and the latent space/features learned/trained with the generative models. Interpretability of neural networks is a rather large and unsolved problem, and many approaches have been taken towards more explainable and interpretable artificial intelligence. More specific to this project, is the interpretability and explainability of latent features of the variational autoencoders. To address this, a literature review is partly done on the disentanglement of the latent space of variational autoencoder architectures. This disentanglement allows for more separated and interpretable latent

features, in this **4.2** literature review section some primary studies within the field are covered to identify and address the threat to internal validity. On top of that, the proposed variant architecture is designed in an attempt to learn features better and more specifically. It is therefore also more focused on disentanglement and interpretability compared to the base architecture. Nonetheless, the explainability and interpretability of neural networks and deep generative models, still is considered to be a large validity threat.

### 3.4.3 External Validity

External validity concerns with the validity of the results on different applications and generalisability of the methodology and results. An important aspect of this thesis project is the very specific application of the generative models on the synthesis of log end images. The goal of this project is not in any case, to research the generalisability of the methodology on other research problems and applications. All the results are specifically documented and described on this specific problem, and is nowhere mentioned to return similar or better results on other data/applications. The standard VAE model has been applied and proven to work on other data sets/applications, as also noticeable in the literature review. The variant proposed in this thesis project, has not been tested on other applications, and therefore can not at all be proven to be generalisable or to return certain other results on different data. For future work, it could be interesting to investigate and report results on generalisability over different data sets.

# 4 Literature Review

In this section a literature review is performed in order to get an understanding of the generative models and the state-of-the-art research built on top of the base models. The literature review is partly based on the guidelines of "Procedures on performing systematic reviews" [42], which was written for systematic reviews within the software engineering field, but in turn was built on top of guidelines and procedures from the medical field. This section includes: **4.1** an introduction to the review protocol with the search procedure, inclusion and exclusion criteria, and the data extraction. After introducing the protocol, the literature review findings are discussed, with the **4.2** VAE findings, the **4.3** GAN findings, and the **4.3** Flow-based models findings. Finally **4.5** presents general comparison findings.

## 4.1 Protocol

### 4.1.1 Literature review research goals/questions

The literature review is done to answer the following thesis research question: *"How does the applied generative variant and base model compare to other generative deep learning approaches in image synthesis?"*, where the applied methods are the variant and base VAE. The research question can be answered by investigating the research space for the different methodologies, in order to further compare them with valid reasoning. This includes applications, variants, strengths, weaknesses, and image synthesis. The aim of the search for primary studies is to answer the following goals/questions per generative model:

**Variational Autoencoders**
The goal is to identify VAE variant architectures, the strengths of variant model and the application of it. On top of the VAE variants, identify hybrid variant architectures, their addition to a VAE model and the application of it. And as the VAE's have a learned latent space, which is also the focus of the thesis project itself, identify research on disentanglement of latent VAE space, and the optimisation on usage of latent dimensions.

**Generative Adversarial Networks**
The goal is to identify GAN variant architectures, the strengths of variant model and the application of it. On top of the GAN variants, identify hybrid variant architectures, their addition to a GAN model and the application of it. And identify other applications and studies of existing GAN (variant) architectures.

**Flow-based models**
As flow-based models are not at the research stage of VAE and GAN architectures, this primary study search is more limited. The goal here is to identify flow-based variant architectures and applications, the strengths of the models and the application of it. And identify other applications and studies of existing GAN (variant) architectures.

### 4.1.2 Search procedure of primary studies

Studies are collected through a search procedure in:

- Google Scholar [1]
- Arxiv [2]
- ACM Digital Library [3]
- IEEE Xplore [4]
- Proceedings of the IEEE International Conference on Computer Vision [5]
- Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition [6]
- The Conference and Workshop on Neural Information Processing Systems is a machine learning and computational neuroscience conference [7]
- International Journal of Computer Vision [8] journal

The search procedure is executed based on search terms with key term and Boolean ***AND/OR***, ******* required and ***?*** optional operators. The following search term is used:

(Applications of ***OR*** Assessment of ***OR*** A survey on)***?*** ***AND*** (Variational Autoencoders ***OR*** VAE ***OR*** Generative Adversarial Network ***OR*** GAN ***OR*** Flow-based Generative Model)******* ***AND*** (Image Generation ***OR*** Network ***OR*** Variants ***OR*** Generation)***?***

### 4.1.3 Inclusion and exclusion criteria

The following criteria are set in order to critically assess the primary studies, and filter the articles based on their relevance to this study:

1. The primary study must be at least applicable to a problem statement in computer vision with images as input, and images as output. Video, audio or other sources of data are excluded criteria.
2. The primary study must be on the application of one of the three generative models, a combination of them or any variant built on the base knowledge of them.
3. The primary study must contain at least one experiment on a data set.
4. The primary study must have been published after 2010. As these deep generative models were not introduced before 2010 [7] [9] [33].

In order to apply this criteria, the abstract is manually assessed. In case the abstract is not clear enough, the results and implementation are manually assessed too.

[1] https://scholar.google.com
[2] https://arxiv.org
[3] https://dl.acm.org
[4] https://ieeexplore.ieee.org
[5] https://ieeexplore.ieee.org/servlet/opac?punumber=9709627
[6] https://ieeexplore.ieee.org/servlet/opac?punumber=9878378
[7] https://nips.cc
[8] https://www.springer.com/journal/11263

### 4.1.4 Data Extraction

In order to accurately extract the information out of the primary studies, a data extraction strategy is designed. The following information is extracted and recorded:

- Title, authors, publication details, study version.
- Source, date of data extraction, search term.
- Type of generative model(s) applied.
- Number of data sets/problems for the application of the generative model and which data sets were used for training/testing.

In order to maximise validity and transparency, the list of primary studies with data extractions are presented in the Appendix **E**.

## 4.2 Variational Autoencoders / VAE Findings

Variational Autoencoders have since their publishing [7] gained more and more attention over the years, and a lot of research has been done on top of the base architecture, either specific to a certain problem or generalisable over different problems. Researching the VAE research space for variants, hybrid variants and disentanglement primary studies helps answering the research question and sub-questions. The main research question: *"How does the applied generative variant and base model compare to other generative deep learning approaches in image synthesis?"* can only be properly answered with valid reasoning, by not only investigating the research space of other generative models, but also the one of the VAE architecture. Through a literature review on these categories, the strengths and weaknesses of the variational autoencoder model can become clear and the applications and synthesis process can be clearly identified. This knowledge can then be used for a comparison of the generative models themselves.

### 4.2.1 Variants

Variants are considered to be VAE architectures that are built on top of the base architecture or another variant itself. These usually consist out of changes in the base architecture or in the loss function to make the model learn in different and specific ways. From the literature review of primary studies one can observe that the variants were mostly focused on extending the VAE with new properties.

Variational autoencoders do not have the ability of conditional generation, thus the synthesis of images based on conditional inputs that define and control the output. To resolve this issue, Harvey et al. [43] present a conditional VAE variant that allows for conditional synthesis. Variational autoencoder models have also proven to struggle with higher resolution synthesis, especially compared to GAN generated outputs, more blurriness is often found in the model generation. VQ-VAE-2 [12], VQ-VAE [11], RVQ-VAE [44], VD-VAE [45], and VAE with self-attention and mutual information [46] are found variants that attempt to improve the generation quality more. Improvements for training are also proposed with the Exemplar VAE [47], ByPE-VAE [48], eVAE [49], BooVAE [50] variants, that each propose their own improvements for robustness, complexity and learning. Loss functions are also

commonly adjusted for better training in certain use cases, such as the novel frequency loss introduced by Jiang et al. [51] or the $\beta$-VAE [31]. Many of the VAE variants found in the search for primary studies mainly focus on the application of image synthesis such as VQ-VAE-2 [12] and VD-VAE [45] architectures, with Dohi [52] applying a VAE for image synthesis on the problem of jet simulation. Image inpainting is also a common application for variational autoencoders like Tu and Chen [53] applying the model for face inpainting. Some variants are also focused on a specific problem, such as the $DP^2$-VAE variant by Jiang et al. [54] that tackles the problem of privacy concerns in training data. Wei et al. [55] further present a comprehensive and comparative evaluation of different state of the art variants of the standard VAE architecture.

### 4.2.2 Hybrid variants

Hybrid generative variants are also VAE variant architectures, but with the difference that they are hybrid generative models and consist out of two or more major generative models/networks/components. This to improve weaknesses found in the VAE architecture, that are however a strength of another (generative) model; by combining them, strengths can be combined and weakness can be reduced significantly. Therefore it is vital for learning strengths and weaknesses of the VAE, to find studies that present hybrid models, these can be found in this section.

Huang et al. [56] present an introspective variational autoencoder, IntroVAE, combined with a GAN, the architecture is capable for self-evaluation through the discriminator and allows for self-improvement during learning. Daniel and Tamar [57] propose a Soft-IntroVAE network, with a more stable and smooth loss function on top of the IntroVAE. Lu et al. [58] propose another variant, called the Adversarial Similarity Distance Introspective VAE, AS-IntroVAE, an architecture that attemps to solve the vanishing gradient and posterior collapse problems appearing with the original architecture. Pandey et al. [59] introduce a DiffuseVAE combined architecture, a VAE combined with a Diffusion model for tasks such as controllable image generation, which the default VAE does not offer. In order to improve generation quality, Lee et al. [60] propose the two stage hybrid framework consisting out of a Residual-quantised VAE (RQ-VAE) and an RQ-Transformer. And Xiao et al. [61] present the VAE-Info-cGAN, a hybrid model that exists out of a VAE and a conditional InfoGAN network. Both of the RQ-VAE and VAE-Info-cGAN are capable of higher quality generation. Imran and Terzopoulos [62] introduce Multi-Adversarial Variational Autoencoder Networks, MAVEN, a hybrid variant consisting of VAE and GAN components with an ensemble of discriminators, for better image generation and classifications.

### 4.2.3 Disentanglement

Disentanglement is considered to be the process of disentangling and organising of latent variables and dimensions of a VAE space. The ability of disentanglement of latent variables provides a lot of power and control on the synthesis and sampling of the latent space. Such control is similar to what is found in conditional generative models, as the conditional VAE [43] previously mentioned. The primary studies found under this section are considered to be studies investigating disentanglement, or the control of the latent variables.

For disentanglement of latent variables, Burgess et al. [31] introduce β-VAE, a VAE architecture that learns to disentangle latent variables more, compared to different implementations. This with quite a small change in the loss function resulting in a large impact on learning. One issue with the β-VAE model, is that it trades a better KL-loss, for a worse reconstruction and synthesis. Approaches as ControlVAE [63], GCVAE [64], PBT-VAE [65], and the approach by Ebrahimabadi [66] all focus on improving reconstruction errors while still maintaining high disentanglement qualities. Other approaches such as the Topographic VAE [67] and Spatial-VAE [68] are also disentanglement approaches, that are more applied on specific applications for disentanglement, such as the topographic and spatial features in data. Xu et al. introduce Multi-VAE [69], a VAE variant that uses a multi-clustering framework by learning disentangled representations, allowing for control with it. Finally, Pastrana [70] presents an experimental study that investigates latent space disentanglement with VAE models and compares different architectures.

### 4.3 Generative Adversarial Networks / GAN Findings

The GAN architecture [9] has been the most promising and applied network architecture of the three deep generative models. It has been especially used for high resolution synthesis. Much research on the GAN has been done with a large amount variants and improvements proposed on top of the base architecture. Therefore researching the GAN space helps identifying variants, hybrid variants and applications.

#### 4.3.1 Variants

Variants are considered to be GAN architectures that are built on top of the base architecture or another GAN variant itself. The changes the variants provide are often changes in loss functions and architecture for either or both the generator and discriminator networks.

Such as the well-known StyleGAN architecture proposed by Karras et al. [15] that applies an alternative generator architecture, based on style transfer. The StyleGAN leads to better disentanglement of high level features, stochastic variation in generation, and specific synthesis control. The StyleGAN network also has variants of itself such as a robust Style-GAN [71], StylEx [72], ReStyle [73], and a StyleGAN approach with an improvement focus towards disentanglement [74]. Lu et al. [75] present a contextual generative adversarial network to learn the joint distribution of sketches and images. Radford et al. [76] present the deep convolutional generative adversarial network, DCGAN. The architecture adopts convolutional layers in order to learn the representations well. The deep convolutional GAN learns a hierarchy of representations from object parts to scenes, in both the networks of the GAN. A variant of the DCGAN [77] focuses on image synthesis on multiple class conditions. Yang et al. [78] propose a Layered Recursive GAN, LR-GAN, a variant approach for learning scene structure and context. The approach is a generative adversarial network for learning the foreground and background of images separately and recursively, to further down the pipeline combine the layers again and generate a natural looking image. Zhang et al. [79] propose a Self-Attention GAN, SAGAN variant. The variant focuses on image generation of details using cues originating from all feature locations, compared to spatially local point in lower resolution

feature maps in traditional GANs. On top of that it provides improved training dynamics of the generator through spectral normalisation. Liu et al. [80] propose a Residual Block Based GAN variant for image synthesis, a GAN architecture where the generator and discriminator are modified and a residual block is added to the architecture. The implemented methodology aims to achieve more stable training and learn image features better. Huang et al. [81] propose Stacked Generative Adversarial Networks, SGAN, a generative model extended on the standard GAN. The model exists out of a top-down stack of different GANs, with the goal of learning lower level representations based on the higher level representations. The models are encouraged to align with each other through the discriminator and an introduced conditional loss function. Because of the stacked GAN architecture, the generative model is able to generate significantly better quality images and learn features better. Durugkar et al. [82] present Generative Multi-Adversarial Networks. A variant of a GAN that instead of implementing one generator and one discriminator, implements a generator with multiple discriminators. Liu et al. [83] introduce BlendGAN, a GAN variant approach to handle the issues of layer-swapping mechanisms not being able fit arbitrary styles in a single model and the necessity of style-consistent data for all the different styles. The BlendGAN architecture approaches this by implementing an encoder to extract the style representations and a controllable weight blending module, all in a unified model that reduces the necessity for data for each individual style. As in VAE's, disentanglement of features also has research within GAN's, such as the Branched Generative Adversarial Network, BSD-GAN [84], for Scale-Disentangled Representation Learning and Image Synthesis. Chan et al. [85] present the Periodic Implicit Generative Adversarial Network, pi-GAN, a variant for 3D image synthesis. The architecture involves periodic activation functions and volumetric rendering, to learn high quality image synthesis with a consistent 3D multi-view understanding.

A default GAN architecture does not allow for much controlled properties, some variants tackle this problem by allowing for controlled synthesis. Such as the ADGAN [86] for controllable person image synthesis, the conditional GAN architecture [14] and CWGAN [87], a variant of the Wasserstein GAN [88], a variant that on top of controlled synthesis, aims to mitigate the problem of gradient vanishing.

On top of improved synthesis, variance and disentanglement of features. Many architectures also attempt to mitigate problems appearing in the base architecture such as the InfoMax-GAN [89] for catastrophic forgetting of the discriminator network and mode collapse of the generator network. The LSGAN [90] attempts to mitigate the problem of vanishing gradients in the the sigmoid cross entropy loss function of the discriminator. The Lifelong GAN [91] for lifelong learning and mitigating catastrophic forgetting in the network. Dong et al. [92] present an approach to handle key challenges with state of the art generative models, caused by arbitrary person pose manipulation. To take on all the current challenges, the authors present a Soft-Gated Warping Generative Adversarial Network, Warping GAN. The variant model consist of two stages: the target pose segmentation map synthesis, and the modified GAN approach with a soft-gated warping block for feature-level mapping.

Variants also attempt to tune the network to occurring issues with the training data itself, such as the F2GAN [93] that attempts to optimise high quality diverse images on classes with limited data or the TrGAN, a variant that attempts to solve the issue of conditional GANs requiring a lot of labelled data. Choi et al.

[94] present StarGAN, a novel and scalable GAN framework for multi-domain high quality image-to-image translation. The unified architecture of StarGAN provides the opportunity for simultaneous training on multiple data sets.

An application that quite some variants also aim to approach is the up-scaling and super-resolution synthesis of images. Some of these variants are the SR-GAN [95] and the WDSRGAN [96]. Another approach is image improvement through impact inpainting, Wang et al. [97] introduce the Discriminative Region Proposal Adversarial Networks, DRPAN model, a GAN variant that works straightforward by finding the worst generated/fake region in an image using DRPnet, and implementing learned image inpainting on that region for a more realistic generated part.

### 4.3.2 Hybrid variants

Hybrid generative variants are also GAN variant architectures, but with the difference that they are hybrid generative models and thus consist out of two or more major generative models/networks/components.

Bao et al. [98] present a variational generative adversarial network, a combination of both a Controllable VAE and a GAN architecture. The hybrid model is focused on high quality image synthesis with controllable properties such as features or categories, through assymetric training. Gorijala and Dikkipati [99] introduce a Variational InfoGAN model, ViGAN, a hybrid conditional model existing out VAE and GAN networks. The model is introduced with the goal of solving the more blurry synthesis of VAE architectures and the distortions in generations found with traditional GAN architectures. Instead of a combination with a variational autoencoder, Zhao et al. [100] introduce an stylised autoencoder-based generative adversarial network, a hybrid generative model for image synthesis existing out of an autoencoder and a GAN model, allowing for more adjustments. The discriminator of the GAN is implemented as a multi-class classifier, resulting in better image generation. Lastly, a hybrid GAN model is presented with a Swin-transformer in a style-based architecture [101]. The transformer based GAN has to ability of scaling to higher resolutions with the strong expressiveness that transformers offer.

### 4.3.3 Applications

Under applications, other primary studies that are neither variants of the GAN architecture or a hybrid model, but rather applications of existing GANs on a certain problem or data set, are described. This also includes analyses and overviews on GANs.

When investigating the research space of deep generative models, two fields of applications appear quite a lot. First of all, the medical field, where generative models, specifically here, GAN's, can be applied to generate more rare or harder to collect data, obtain higher resolutions, or find features that a human would not detect. An example of this is the application of generative adversarial networks on image synthesis for Magnetic Resonance (MR) images [102], where the synthesis even passed a visual Turing test. Another common field is the one of anomaly detection, with the SDGAN [103], a GAN variant for surface defect image generation, or the application of a Cycle-GAN [16] variant for abnormal-to-normal generation [104]. A common application for reporting quantitative results, especially with GAN's, is face synthesis. An example of this is the study on DCGAN's [76] applied for face image synthesis with added and controllable attributes [105]. Zeno et

al. [106] implement a comparative analysis of a UNIT GAN, a hybrid model based on a GAN combined with a VAE, and a StarGAN architecture. The authors apply these architectures to the application of photo-realistic new face image synthesis, with preserving identity in the generation. There are many other applications as well, some examples of these are the following. Minaee and Abdolrashidi [107] apply generative adversarial networks for the image synthesis of high-quality fingerprints. Many existing models and approaches are not powerful enough to learn such complex representations as fingerprints, the GAN approach in this research has proven to work well for such complex tasks as synthesis of fingerprints. And Mustikovela et al. [108] present SSOD, an end-to-end analysis-by-synthesis framework using controllable GANs, for the task of self-supervised object detection. The authors apply their framework on the application of car detection. Zhao et al. [109] apply image augmentations to vanilla GAN architectures, to investigate and study the impact on results and training for image synthesis. The application of it is tested in a variety of settings. On top of that, Liu et al. [110] present an extensive overview of adversarial generative models for image and video synthesis, with related works, regularisation's/stabilisation's, improvements and limitations. Shamsolmoali et al. [111] provide a comprehensive survey and case studies on adversarial networks for image synthesis. The authors summarise the methodologies and applications, with extensive details on architectures, loss functions, metrics, and training. Tahmid et al. [112] present a comparative analysis of different GAN variants and quality image synthesis assessment.

### 4.4 Flow-based Generative Models Findings

The Flow-based architectures have been proposed in a similar time frame as the GAN and VAE architecture, but has only grown in attention and more research years later, and is really starting to grow in research as of the last couple years. Exploring the flow research space, helps identifying the main approaches with architectures, variants and applications.

#### 4.4.1 Architectures and variants

Kingma and Dhariwal [18] present GLOW, a generative flow-based model using 1x1 convolution. A flow-based approach that achieves significant results on the log-likelihood optimisation and on high resolution realistic image synthesis. The GLOW architecture has been one of the lead approaches for flow-based models and much research and approaches have been based on or inspired by the model. Such as DP-GLOW [113], a GLOW model combined with a local differential privacy (LDP) algorithm, for medical image generation with more privacy protection. Or DUAL-GLOW, a variant architecture based on two invertible networks, instead of a single invertible network, and a relation network that maps the latent spaces together. The architecture is applied on medical PET image synthesis and achieves state-of-the-art results on the data set. Just as the standard VAE and GAN, the base architectures in flows do not originally have the capability of conditioned synthesis on certain labels or classifications, CAGlow [114] tackles this architectural issue with an encoder, showing improved synthesis compared to the original GLOW architecture. Ma et al. [115] introduce Masked Convolutional Generative Flow, Macow, a flow-based generative model with masked convolutions, resulting in better density estimation compared to GLOW on standard image data sets. Mukherjee

et al. [116] present Attentive Contractive Flow, a model and plug-and-play flow for localised self-attention. The self-attention contractive flow is implemented for extending expressiveness while maintaining invertibility of the flows. The authors demonstrate representation power, faster training convergence and more realistic synthesis by using a generative model with attentive contractive flows. Another approach to improving flows expressiveness is the mAR [117] or Multi-Scale Autoregressive Priors approach and introducing channel-wise dependencies in the latent space. Using these mAR flow layers, the model results in better density estimation and improved image generation losses. To mitigate the three limits of uniform noise for dequantisation, inexpressive affine flows, and purely convolutional conditioning in coupling layers in prior work, Ho et al.[118] introduce Flow++, a variant flow-based generative model. The architecture achieves new state-of-the-art results due to the architectural changes, compared to the prior work that contained these limits. Chen et al. [119] propose GSMFlow, an flow-based architecture for tackling generation shifts. Generation shifts means the model synthesis shifts away from the real distributions of unseen data. The researchers find and attempt to mitigate three problems possibly causing generation shifts: semantic inconsistency, variance collapse, and structure disorder. The flow model achieves state-of-the-art results on the tested data sets on image synthesis.

### 4.4.2 Applications

Hajij et al. [120] implement the RealNVP [17] flow-based architecture on the application of medical image synthesis and show the effectiveness of the method on the data. Much research is focused on the quality of models based on their generation performance, but Pope et al. [121] study the adversarial robustness of flow-based models, with more specifically the GLOW [18] and RealNVP [17] models. The authors prove both the models are extremely sensitive to adversarial attacks, and a significant improvement in robustness with a more hybrid training procedure. Pires and Figueiredo [122] present a way of introducing discrete structure into the flow-based model framework, by applying a flow-based model combined with a mixture model. The authors demonstrate its application in density estimation, clustering, and semi-supervised learning. Tailanian et al. [123] apply a U-shaped flow-based model on the application of anomaly detection, in combination with a multi-scale image transformer network. The application of the presented framework on anomaly detection returns state-of-the-art results compared to all other models on the commonly used anomaly detection data set. Dong et al. [124] present the application of flow-based models on image super-resolution enhancement, of medical MRSI data. The flow-based enhancer shows flow-based models can be used for this application well, as the authors report better resolution enhancement compared to generative adversarial networks and base flow-based models. ClothFlow [125] and FVTN [126] are both flow-based model frameworks, that are applied on the problem of clothed person image synthesis for virtual try-on and pose-guided image generation. Both flow-based approaches returned state-of-the-art results on the synthesis, with especially, FVTN, that came after ClothFlow achieved high-quality try-on synthesis.

### 4.5 Comparison Findings

After covering the findings on the three deep generative models separately, a comparison can be made based on the qualitative observations on the primary studies.

Research wise, the GAN space seems to be the largest and most adapted for image synthesis, followed by the VAE and then the Flow-based models. Especially the last couple of years, all the three generative approaches have been very large within research with many variants and adaptations to improve the existing state-of-the-art architectures. Overall the GAN and Flow-based architectures with the many variants seem to be able to generate more higher quality and resolution outputs, while the VAE ends up in more blurry reconstructions. VAE variants attempt to solve the blurriness, but so far, it does not outperform the other models. The GAN however does end up in more noisy outputs, while the VAE and Flow-based models end up in more cleaner and smoother images. The VAE also offers a very accessible learned latent space that can be used for sampling other images with different combinations of features. This latent space furthermore allows for research and variants focusing on disentanglement and latent variable control. The different models optimise in different ways, where VAE learns a lower bound of the maximum likelihood, while the Flow-based models learn the exact maximum likelihood, therefore in theory being able to learn significantly more exact than the VAE. And then the GAN not optimising for the maximum likelihood at all, but instead optimising for the minimax-loss.

All the architectures do seem to have some problems during training, that is the GAN suffering from mode collapse, vanishing gradients, catastrophic forgetting and in-stable training. The VAE suffering from posterior collapse, disentanglement and over-regularisation leading to blurry reconstructions. And the Flow-based model suffering from mode collapse, and generation shifts. All three deep generative models do not have that capability of conditioned synthesis with the base architecture, in all three, variants as the conditional VAE [43], cGAN [14], and the CAGlow [114] architectures provide a conditional solution. Lifelong learning is also a problem within image synthesis which certain deep generative variants attempt to tackle with adjusted architectures.

In theory, the flow-based models can due to their nature, outperform the other models as they are able to explicitly learn the probability density function of real data, something that VAE and GAN do not learn. However in practice, the flow-based models are significantly more heavier to train compared to the VAE and GAN. Per epoch, the VAE seems to be lighter to train, as the GAN trains the discriminator and generator adversarially. But the performances highly depend on the implemented architectures and the complexity of the network.

As the individual deep generative models suffer from different problems in their architectures, hybrid variants are proposed, these combined deep generative models attempt to solve weaknesses in the individual architecture, such as the VAE-InfocGAN [61]. These hybrid models seem to be able to achieve promising results and proves that it might be possible to solve general problems by combining models.

Many details of this comparison have already been covered in the individual model sections. It is important to mention that this comparison is based on a sample of the research out there, thus it is not complete in any way. It does highlight some of the larger outlines in the differences and models. This comparison was again based on qualitative directions, no quantitative comparisons are done here.

# 5 Design and Implementation

In this section the implementation for the thesis project is described. This section includes: **5.1** introduces the experimentation pipeline. The pipeline exists out of the **5.2** VAE architectures, the **5.3** triplet sampling, and the **5.4** recognition model. **5.5** describes the data collection processes, and **5.6** the technical specifications.

## 5.1 Pipeline

The pipeline in this implementation is a sequence of three different stages and components, as visualised in Figure 5.13. There is an ordered sequence to the steps, due to the dependencies of the individual components. The first stage is the VAE model, this stage includes the training and experimenting with both the vanilla VAE architecture and the proposed variant VAE architecture. From this stage, the model is further used for sampling new images in the triplet sampling stage, where a proposed algorithm uses the learned distributions to sample triplets based on anchor images. This triplet data is then further used in the recognition model stage, where triplet samples are used to train a log end image recognition model.

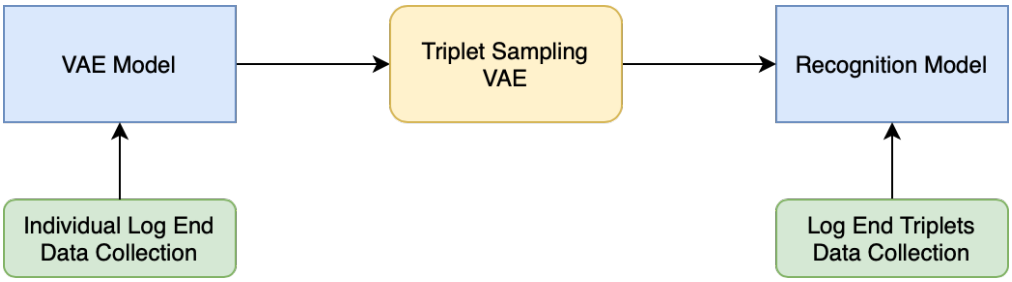


Figure 5.13: Pipeline of the implementation, with the different major steps.

On top of the three major components, there are two data collection components for both the deep learning models. For the VAE model stage, real log end images are collected for training, with different data set sizes used. For the recognition model real triplets are collected, that are further used for training a baseline model and testing the recognition model on sampled triplets. The exact order of stages was also followed in the development of this implementation. In the following sections, the stages are described more in depth.

## 5.2 VAE Architectures

Below the implemented VAE architectures are described. All the code is written in Python, using the Tensorflow and Keras library for the neural network building and training. Other libraries are also used in order to properly set up a pipeline and data processing. The code for the neural networks is added in Appendix **D**.

### 5.2.1 Vanilla VAE architecture

One can understand the vanilla VAE architecture, as the standard VAE [7]. As the application of the VAE models is on image learning, a convolutional VAE is implemented. This means that both the encoding and decoding networks exist out

of convolutional layers, in order to learn the image features. The *D* number of latent dimensions was one of the dependent variables and was adjusted based on the performances of the models and scaling of data set.

**Encoder**

The encoder of the vanilla VAE architecture exists out of 5 convolutional layers, a flattening layer, 3 dense layers, and a custom sampling layer. The 2D convolutional layers ("Conv2D" in TensorFlow) have filters of 32, 64, 128, 256, and 512, respectively. All Conv2D layers use a kernel of 3x3, a stride of 2x2, a "same" padding, and a ReLU activation function. Using this sequence of Conv2D layers, the first and second dimension are scaled down exponentially, originally the height and width dimensions of the image, while the third dimension, originally the RGB channels, is scaled up exponentially. The flattening layer is then used to flatten the output of the last convolutional layer. The flattened output further goes through a Dense layer of the *D* latent dimensions units size, without a defined activation function. This output is then used to train both the mean and log variance layers, which are using the same parameters as the first Dense layer. Afterwards both the mean and log variance outputs are used in the custom sampling layer and reparameterisation trick. The architecture and sequence of layers in the encoder are described in Table 5.2.1 with the example on a vanilla VAE with 1024 latent dimensions.

| Layer (Type) | Output Shape | Parameters |
|---|---|---|
| Input Layer | (256, 256, 3) | 0 |
| Conv2D Layer (01) | (128, 128, 32) | 896 |
| Conv2D Layer (02) | (64, 64, 64) | 18,496 |
| Conv2D Layer (03) | (32, 32, 128) | 73,856 |
| Conv2D Layer (04) | (16, 16, 256) | 295,168 |
| Conv2D Layer (05) | (8, 8, 512) | 1,180,160 |
| Flatten Layer | (32768) | 0 |
| Dense Layer | (1024) | 33,555,456 |
| zMean Dense Layer | (1024) | 1,049,600 |
| zLogVar Dense Layer | (1024) | 1,049,600 |
| Sampling Layer | (1024) | 0 |
| Total parameters: 37,223,232 | | |

Table 5.2: Layer architecture and specifics for the vanilla VAE encoder architecture with 1024 latent dimensions.

**Decoder**

The decoder of the vanilla VAE architecture exists out of a dense layer, reshape layer and 5 convolutional transpose layers. The Dense layer has the same amount of units as the flattening layer in the encoder architecture and uses the ReLU activation function. The output from the Dense layer is then reshaped using a Reshape layer, so it can be used for the 2D convolution transpose "Conv2DTranspose" layers. The 5 transpose layers have filters of 512, 256, 128, 64, and 3 respectively. All of the Conv2DTranspose layers use a kernel of 3x3, a strides of 2x2, a "same" padding, and a ReLU activation function. Besides the last Conv2DTranspose layer using a 1x1 stride. The output of the decoder is again an image of the same input size with the 3 RGB channels. The architecture and sequence of layers in the decoder

are described in Table 5.2.1 with the example on a vanilla VAE with 1024 latent dimensions.

| Layer (Type) | Output Shape | Parameters |
|---|---|---|
| Input Layer | (1024) | 0 |
| Dense Layer | (32768) | 33,587,200 |
| Reshape Layer | (16, 16, 128) | 0 |
| Conv2DTranspose Layer (01) | (32, 32, 512) | 590,336 |
| Conv2DTranspose Layer (02) | (64, 64, 256) | 1,179,904 |
| Conv2DTranspose Layer (03) | (128, 128, 128) | 295,040 |
| Conv2DTranspose Layer (04) | (256, 256, 64) | 73,792 |
| Conv2DTranspose Layer (05) | (256, 256, 3) | 1,731 |
| Total parameters: 35,728,003 | | |

Table 5.3: Layer architecture and specifics for the vanilla VAE decoder architecture with 1024 latent dimensions.

**Sampling**
As described above in the encoder, the network learns both the mean and log variance in the dense layers of *D* latent dimension units. The distribution type for the VAE, as most commonly used, is the Gaussian distribution. In order to sample from this, a custom sampling function is used. This is like the VAE reparameterisation trick, as described in Section **2.6**. The inputs of the sampling function are both the mean and log variance layers of *D* latent dimensions size, these represent the learned distributions. In the algorithm, a new parameter epsilon is computed based on the means, using a random Gaussian distribution. Further the standard deviation can be easily computed based on the log variances. Therefore based on the mean, standard deviation and epsilon, a point in the latent space is sampled. This algorithm allows for backpropagation, while still being able to learn the distributions.

**Algorithm 1** VAE Sampling

**function** SAMPLING($z_{mean}, z_{log_var}$)
    ▷ Generate a random normal distribution for the reparameterisation trick
    $epsilon \leftarrow$ random normal distribution with $\text{shape}(z_{mean})$
    ▷ Compute the standard deviation from the learned log variance
    $sigma \leftarrow \exp(0.5 * z_{log\_var})$
    ▷ Generate a sample in the latent space based on the learned distributions
    $z \leftarrow z_{mean} + sigma * epsilon$
    **return** $z$
**end function**

**Connected Architecture**
The final step of this model is to just connect the encoder and decoder. The losses can be computed based on both the outputs of the encoder and the decoder.

### 5.2.2 Variant VAE architecture

The variant network uses the same architecture as the vanilla VAE network, but with the difference that the variant architecture has a different approach in the

**Algorithm 2** Connected VAE

**function** VAE($data, latentdims$)
    ▷ Input the data and *n* latent dimensions through the encoder
    ▷ This will return the latent mean, log variance, and points
    $zmean, zlogvar, z \leftarrow encoder(data, latentdims)$
    ▷ Reconstruct the latent point through the decoder
    $reconstruction \leftarrow decoder(z)$
    **return** $reconstruction$
**end function**

learning of the mean and log variance, in the encoder network. As described above, the vanilla VAE learns all the *D* dimensions means and log variances in one layer each, with the previous layer being a dense layer connecting to these two layers. The variant approach is constructed in a way that the network learns every dimension parameters separately. For each dimension, a small sub-network of layers being added before the layers are learned. This is again visualised in Figure 5.14.

In the implementation within this thesis project, a sub-network of two sub-layers is implemented for each latent dimension. These layers are both densely connected layers with 8 and 4 units respectively, using a ReLU activation function. After the two sub-layers, the mean layer and the log variance layer, each containing 1 unit, are added. This is repeated for every latent dimension and all these layer parameters are constantly trained. The aim of this approach is to improve the learning of the features, resulting in more creative sampling and better reconstructions. One can understand an improvement in creativity as an improvement in the learning of features, therefore resulting in more different features upon sampling. Figure 5.15 visualises the implemented sub-network, with the two dense layers and the latent dimension parameter layers, for one single latent dimension. Besides the changes regarding latent dimensions in the encoder, the variant architecture is exactly the same as the vanilla architecture implemented and described above.

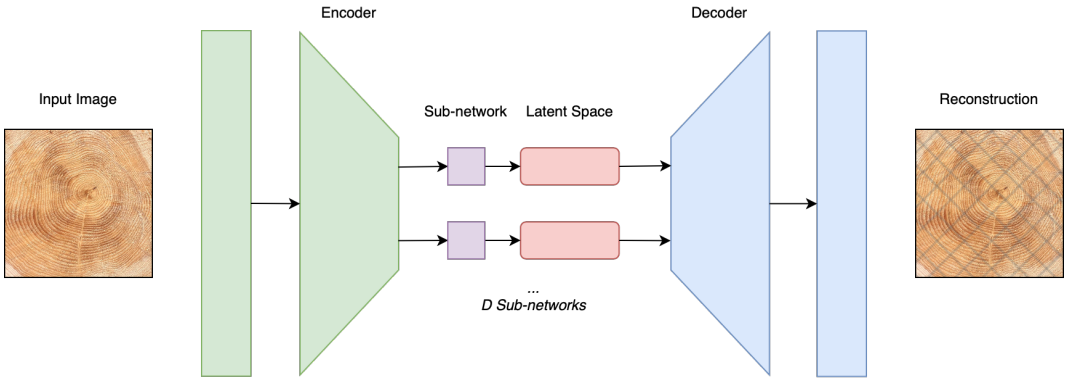


Figure 5.14: Variant VAE Architecture with *D* sub-networks each learning the mean and log variance dimension parameters.

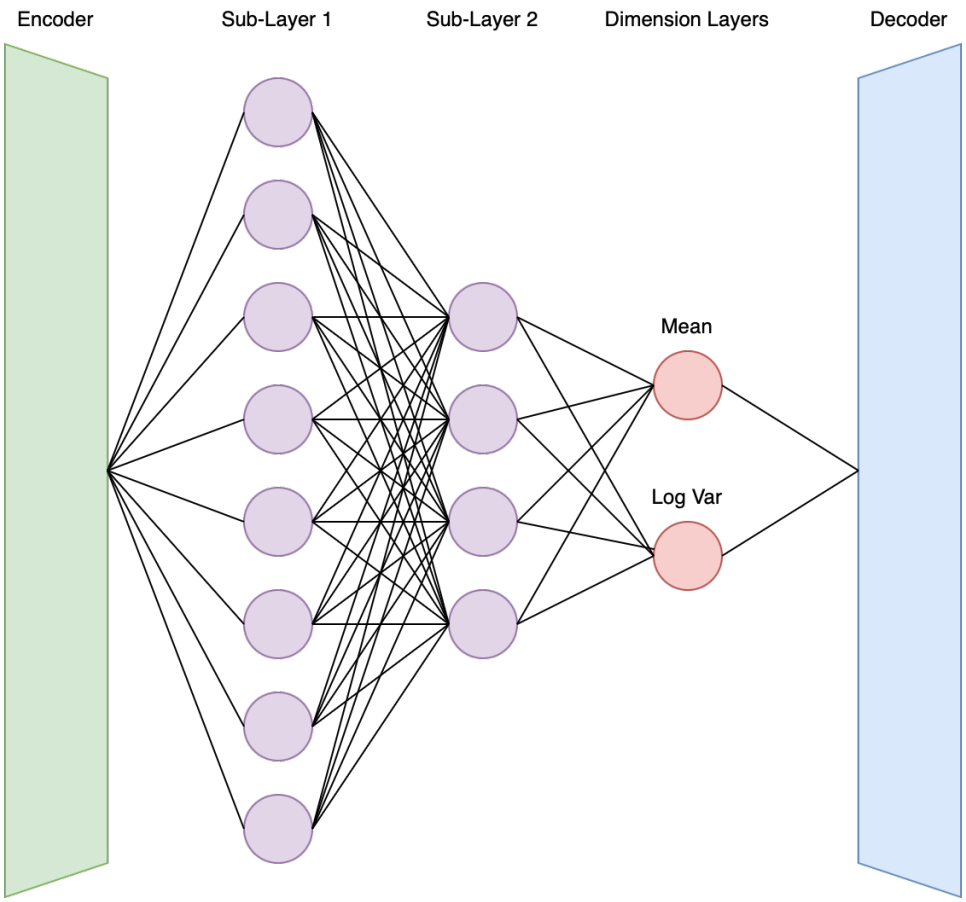


Figure 5.15: Visualisation of one sub-network for one latent dimension. This sub-network is repeated and trained for every latent dimension. Here the sub-network contains two sub-layers before the mean and log variance layers are trained.

### 5.2.3 Loss Function

The loss function for the VAE architectures is the ELBO loss (Equation 17), this loss exists out of the reconstruction loss and the KL divergence loss.

$$\textit{ELBO loss} = \textit{Reconstruction loss} + \textit{KL loss} \tag{17}$$

For the reconstruction loss, the sum of squared errors (SSE) (Equation 18) is used. Where $y_i$ is the real image, also the input of the encoder, and $f(x_i)$ the reconstructed VAE image, also the output of the decoder. This represents the pixel-wise discrepancy between the original image and the reconstructed image.

$$\text{Reconstruction} = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{18}$$

And the KL loss regularisation term (Equation 19), where *zLogVar* and *zMean* are the learned sets of latent parameters learned by the network.

$$\text{KL} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{d} [-0.5 * (1 + zLogVar - zMean^2 - e^{zLogVar})] \tag{19}$$

### 5.2.4 Network sizes

The network sizes are listed in Table 5.4. The networks largely scale up in network size and parameters based on the defined $D$ latent dimensions. The variant networks are also larger in trainable parameters compared to the vanilla architecture, with the difference growing larger with the latent dimensions.

| Model | Latent Dimensions | Total parameters |
|---|---|---|
| Vanilla VAE | 256 | 20,651,203 |
| Variant VAE | 256 | 21,057,731 |
| Vanilla VAE | 512 | 37,822,403 |
| Variant VAE | 512 | 39,421,891 |
| Vanilla VAE | 1024 | 72,951,235 |
| Variant VAE | 1024 | 79,295,939 |

Table 5.4: Networks sizes with different latent dimensions

## 5.3 Triplet Sampling

The goal of the triplet sampling algorithm is to input a log end image anchor, obtain a VAE encoded version of it, so that a positive and negative can be selected from the encoded point in space. All of the encoded images can then be decoded again, resulting in a formed triplet. This process can be done repeatedly to generate an $x$ amount of triplets. In this thesis, an algorithm for the triplet sampling is proposed and described below. The algorithm takes a few parameters, that can be tuned accordingly:

- Data: The data used for obtaining the VAE distributions. For this, the training data of the VAE can be used.
- Anchors: A set of anchor images of which for every image, an anchor can be encoded, and a positive and negative can be sampled. The test data of the VAE can be used.
- VAE: The trained variational autoencoder model, that exists out of the encoder and decoder networks.
- Latent dimensions: The size of the latent space used for the trained VAE model.
- Radius for positive: The distance in log variance from the encoded anchor that the positive is sampled from. Parameter has to be lower than the radius for negative.
- Radius for negative: The distance in log variance from the encoded anchor that the negative is sampled from. Parameter has to be higher than the radius for positive.
- Probability of sampling towards the mean for the positive sample: This is the probability of sampling towards the mean, for the positive encoding. Parameter is a value between 0 and 1, inclusive of both 0 and 1. Parameter is used

for tuning the amount of outliers and noise in the images. The higher the parameter, the less noise can be found in the image as it samples more towards the mean of the latent distribution.

- Probability of sampling towards the mean for the negative sample: This is the same parameter as above, but for the negative sample. A lower value than the probability for a positive sample is preferred, as the negative is preferred to have more different outlier features and noise.

With the parameters given to the function, the algorithm first computes the distributions from the VAE on the data. The latent distributions can be given as a parameter as well, as they are known after training the VAE model itself. In case this is not saved, it can be computed again. After the distributions are computed, the algorithm encodes the original anchor image into the encoded anchor, which can then further be used for sampling the positive and negative sample.

For the sampling of the positive and negative, it is computed per individual latent dimension. One can only sample in two directions in a single dimension, towards or away from the mean of the distribution. As doing this completely random returns more noisy results, a probability of sampling towards the mean has been added, allowing for more good features and less outliers. The probability of sampling in the direction of the mean can be different for the positive and the negative, where a negative is allowed to be more noisy than the positive. First the direction is decided through a uniform random variable, then the decided log variance distance from the point is computed. Doing this for every latent dimension, a new encoded sample is constructed based on the encoded anchor sample. This process is done for both the positive and negative, with their respective log variance distances and probability towards mean parameters. The algorithm is visualised in Figure 5.16 with the different parameters and encoding points. After the encoded triplet has been constructed, the encoded anchor, sampled positive and sampled negative, can be decoded with the VAE decoder.

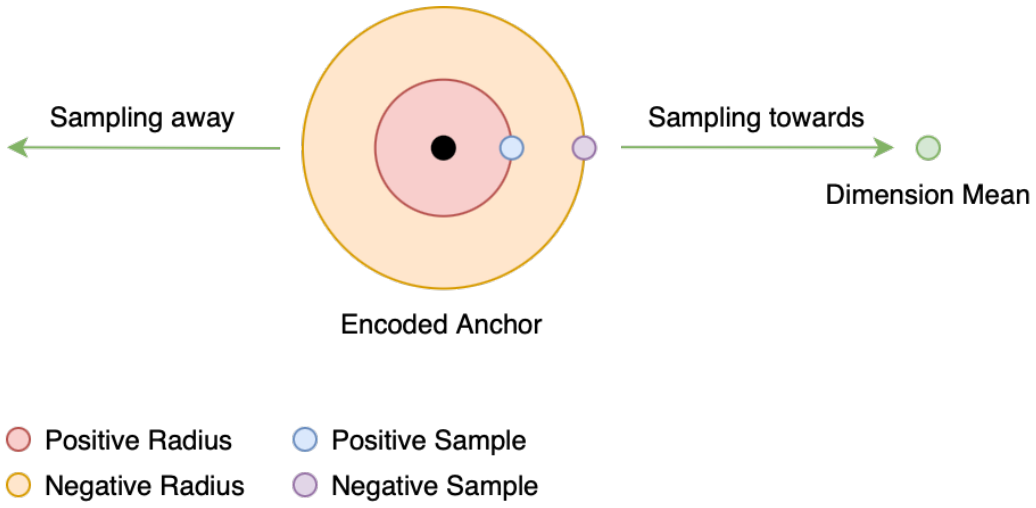


Figure 5.16: Visual representation of the triplet sampling algorithm.

**Algorithm 3** Triplet Sampling

**function** TRIPLETSAMPLING($data, anchors, vae, latentDim, radiusPos, radiusNeg, posSampleTowardsMeanProb, negSampleTowardsMeanProb$)
    ▷ Input the data and *n* latent dimensions through the encoder to obtain the distributions
    $zMean, zLogVar, z \leftarrow vae.encoder(data, latentDims)$
    ▷ Loop over the given anchor images
    **for** $anchor$ **in** $anchors$ **do**
        ▷ Input the anchor through the encoder to obtain the encoded point
        $imgZMean, imgZLogVar, imgZ = vae.encoder.predict(anchor)$
        $posSample, negSample = zeros((latentDim)), zeros((latentDim))$
        ▷ Loop over the latent dimensions
        **for** $dim$ **in** $latentDim$ **do**
            $globalMean = zMean[dim]$
            $mean, logVar = imgZMean[dim], imgZLogVar[dim]$
            ▷ Sampling the point based on the probability of sampling towards or away from the mean. Both for the positive and negative sample
            **if** $random(0, 1) <= posSampleTowardsMeanProb$ **then**
                **if** $mean > globalMean$ **then**
                    $posSample[dim] = mean - (radiusPos * abs(logVar))$
                **else**
                    $posSample[dim] = mean + (radiusPos * abs(logVar))$
                **end if**
            **else**
                **if** $mean > globalMean$ **then**
                    $posSample[dim] = mean + (radiusPos * abs(logVar))$
                **else**
                    $posSample[dim] = mean - (radiusPos * abs(logVar))$
                **end if**
            **end if**
            **if** $random(0, 1) <= negSampleTowardsMeanProb$ **then**
                **if** $mean > globalMean$ **then**
                    $negSample[dim] = mean - (radiusNeg * abs(logVar))$
                **else**
                    $negSample[dim] = mean + (radiusNeg * abs(logVar))$
                **end if**
            **else**
                **if** $mean > globalMean$ **then**
                    $negSample[dim] = mean + (radiusNeg * abs(logVar))$
                **else**
                    $negSample[dim] = mean - (radiusNeg * abs(logVar))$
                **end if**
            **end if**
        **end for**
        ▷ Decode the encoded points to obtain the triplet
        $triplet = vae.decoder.predict([imgZMean, posSample, negSample])$
        **return** $triplet$
    **end for**
**end function**

The proposed algorithm does have both deterministic and stochastic variables, but with high stochastic probability values, the algorithm will generate very similar logs upon each generation with the same anchor images. As a future approach, it would be good for the triplet generation to introduce a random parameter for the distance too, to add the capability of generating different triplets with the same anchor. For now, it is advised to use different anchors when using high probability of sampling towards the mean.

## 5.4 Recognition Model

The final stage of the pipeline is the recognition model. The application of this recognition model would be to recognise and identify the same log, at different stages in a wood processing pipeline. The triplet loss becomes a very good application here, as the anchor and positive can be seen as the same log, but at a different stage in the wood processing pipeline and therefore looking slightly different, for example at a sawmill and sorting station. While the negative is a completely different log. Thus based on the triplets, a recognition model can be trained to identify the same log ends over a production pipeline. This is also where the VAE has the potential to significantly improve the training of such a recognition model. As the data collection of triplets is a manual collection process, and therefore a very expensive cost. Being able to generate triplets with a VAE, for ideally a data set of infinite size, can help with the training of such a model by either being an addition to the real triplet data, or to replace the training data set.

### 5.4.1 Architecture

The architecture used for the recognition model is based on the EfficientNetB0 pretrained neural network. Using this, the weights are not randomly initialised and the recognition network trained on log triplets does not have to fully learn from scratch, as the EfficientNetB0 neural network has been pretrained on the large ImageNet data set. The EfficientNet architecture [127] was originally proposed for the reason that it is easily applicable for different convolutional classification problems and networks. The EfficientNetB0 network is combined with a MaxPooling, Dense, and Lambda output layer to form the full recognition model. The model can be considered rather small compared to other convolutional classifiers. There are also way heavier EfficientNet or other pretrained networks available [9], but the goal of this project was not optimising this architecture, instead test the application of sampled triplets on a recognition model. The weights of the EfficientNet architecture are also not set to be trainable. Table 5.4.1 visualises a simplified version of the architecture.

### 5.4.2 Loss Function

**Triplet loss with semi-hard mining**

For the learning of the network, the triplet loss is used as the cost function. The triplet loss comes with the triplet margin hyperparameter, during the learning in these networks a margin of 0.6 was used over all network training's. The most interesting type of negative encoding in the triplet loss is the semi-hard negative, as

[9] https://keras.io/api/applications/

| Layer (Type) | Output Shape | Parameters |
|---|---|---|
| Input Layer | (256, 256, 3) | 0 |
| EfficientNetB0 | (8, 8, 1280) | 4,049,571 |
| MaxPooling2D Layer | (1280) | 0 |
| Dense Layer | (1024) | 1,311,744 |
| Output Lambda Layer | (1024) | 0 |
| Total parameters: 5,361,315 | | |

Table 5.5: Layer architecture and specifics for the recognition network.

these semi-hard negative encodings are within the margin from the radius for positive, therefore further away from the anchor than the positive, but still producing positive losses. Using the semi-hard triplet loss, the network will only update the parameters based on the negatives that are semi-hard, not on easy or hard negatives. The algorithm therefore only selects the semi-hard negatives. The reason for this is that focusing the training on these forces the network to separate the encoding space more and learn features better.

**Accuracy**
The accuracy represents the accuracy to which the recognition model can correctly identify logs between anchors and positives. The accuracy is not used as the cost function, but is however used to measure the quality of the recognition model, as the accuracy of identifying the same log ends at different stages is vital.

### 5.4.3 Training

During the training of the recognition model, augmentations are used to improve the learning and robustness of the model. These augmentations include mirroring, affine transformations, perspective transformations and slight colour filters. These augmentations on the images also prevent the model from overfitting heavily, especially since small data sets are used. On every epoch, the images are augmented randomly and with different randomised augmentations parameters. The network also uses an Adam optimiser with an initial learning rate of 1e-04.

## 5.5 Data Collection

The data collection for the log ends was already set up, as prior research has already been done on different AI implementations. As visible in the 5.13 pipeline figure, there are two different data sets used.

**VAE Individual Log Data**
Individual log ends are collected from a log sawmill and log sorting station, from a period of end January until beginning of February. The data sets are collected 50/50 from sawmill and sorting, no matter the size of the data set, it is always half/half. The individual log selection process is stochastic, as first all the logs within the query parameters are retrieved, then the smaller data set is randomly constructed from all logs. No data preprocessing was done in constructing the data set, as the goal of the variational autoencoder training is to learn the logs well, but also be able to handle and reconstruct outliers and more special/rare logs. Therefore in the

pipeline, types of trees can be found in different conditions, shapes, and other visual features.

**Recognition Model Triplet Log Data**
The triplets are collected from a previously manually created triplet data set of 1470 triplets, where a positive represents the same log but at a different station, and a negative a different log. In this way a recognition model can be trained to match positives and therefore identify the same log-end pairs from different stations, allowing for tracking capabilities. Besides training augmentations, no preprocessing is done in any way.

### 5.6 Technical specifications

For computational reasons, all the networks were not trained on a local device. Instead, these were trained on virtual machines in the cloud. The technical specifications of the training device are the following:

- RAM: 32GB/64GB (More RAM for larger model/data training)
- Memory: 80GB
- VCPU: 8
- VGPU: 1
- Operating System: Ubuntu 22.04.1

# 6 Controlled Experiment

In this section the controlled experiment is described. This section includes: **6.1** where the different VAE model training's and results are described. **6.2** presents the triplet sampling results and selected hyperparameters. **6.3** presents the recognition model training and results, from both real data and VAE sampled data. **6.4** addresses additional aspects of experiment validity, post-experimentation.

## 6.1 VAE Models

In this subsection the different trained VAE models are covered over the different data sets, and how the models and independent variables were adjusted over time. More VAE reconstructions can be found in Appendix **B**.

### 6.1.1 Results - 1K data set Training's

The first data set is the 1K data set, using a 0.8/0.1/0.1 training/validation/testing split. This results in 800 images being used for training, with a 100 images for validation and testing each.

**Final losses**
Based on the results reported in Table 6.6, it can be concluded that the variant VAE with 512D achieves the best training losses, with the lowest reconstruction and KL loss, on the 1K data set. In validation losses, report in Table 6.7, the vanilla VAE achieves the lowest loss due to a higher reconstruction error on the variant VAE. The variant VAE however still achieves a lower KL loss on validation. The difference between the training and validation losses is quite large here, appearing due to the high variation in tree log ends with the combination of using a very small data set. This causes the training to generalise less over the validation and test sets, therefore returning higher reconstruction errors.

| Model | Epochs | ELBO | Reconstruction | KL Loss |
|---|---|---|---|---|
| Vanilla VAE - 256D | 500 | 9102.97 | 8439.91 | 741.06 |
| Vanilla VAE - 512D | 500 | 6632.79 | 5938.78 | 694.00 |
| Variant VAE - 512D | 500 | 5922.91 | 5270.83 | 652.08 |

Table 6.6: Models Trained on 1K data set - Training Losses

| Model | Epochs | ELBO | Reconstruction | KL Loss |
|---|---|---|---|---|
| Vanilla VAE - 256D | 500 | 22903.19 | 22170.41 | 732.77 |
| Vanilla VAE - 512D | 500 | 20533.95 | 19792.96 | 740.95 |
| Variant VAE - 512D | 500 | 22820.96 | 22189.00 | 631.95 |

Table 6.7: Models Trained on 1K data set - Validation Losses

**Losses during training**

The reconstruction training losses, as visualised on Figure 6.17, all follow the same trend over the epochs during training, where both the 512D models are very close to each other and keep on slowly dropping. The validation losses for the 512D models follow the exact same trend, with the vanilla VAE staying under the variant VAE in loss. The validation loss for the 256D vanilla VAE drops significantly slower and does not reach a plateau yet within the 500 epochs. This is also why the variant VAE was never trained on 256D, and 512 was used immediately.


Figure 6.17: Reconstruction Loss during the training of the models on the 1K data set.

The KL training loss for the 256D vanilla VAE drops a lot in the beginning of training, as visualised in Figure 6.18, but keeps on increasing over time. While both the 512D models stay quite stagnant and decrease very slowly after a large drop the first 100 epochs. The validation losses for the vanilla VAE models follow a very similar trend and reach the same plateau. The variant model does too, but at a lower level and therefore a better KL loss.


Figure 6.18: KL Loss during the training of the models on the 1K data set.

**Vanilla VAE - 256D Image Reconstructions**

The reconstructions of the 256D vanilla VAE on training data, as shown in Figure 6.19, contain some slight blurriness. However, the features seem to be captured well and although not perfectly captured, the latent space reconstruction still looks close to the original image. The reconstructions on the validation data, in Figure 6.20 are substantially poorer, where the blurriness is heavier and the features are not captured that well. The logs have slightly different shapes and non-matching structures, but the colours are however still quite close. These results show that the model did overfit on training data and does not generalise well for new data. To capture more features, the next step was to train the same model with an increased latent space.

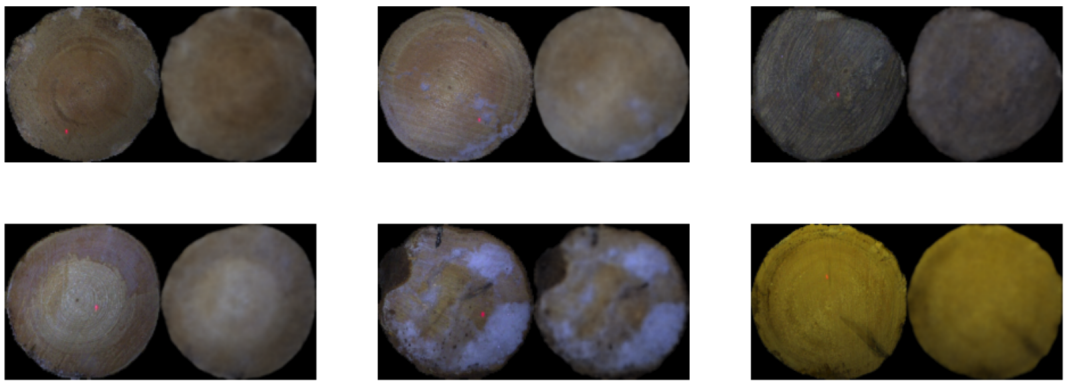

Figure 6.19: 256D Vanilla VAE reconstructions on random training samples.

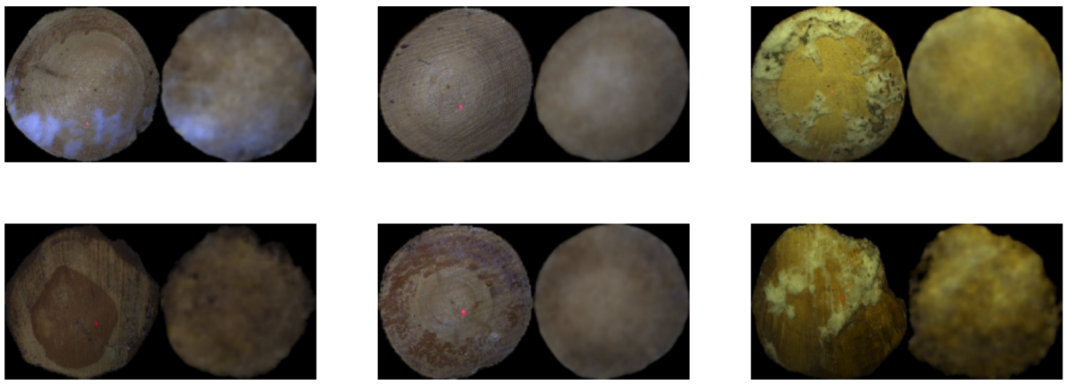

Figure 6.20: 256D Vanilla VAE reconstructions on random validation samples.

**Vanilla VAE - 512D Image Reconstructions**
After training a VAE on 256 dimensions, this model was trained on the same 1K data set using a double increase in latent space of 512 total dimensions. With this model the reconstructions on training data are very close to the original images. As shown in Figure 6.21, the features are captured well and the generation quality is close to the real images. It is even quite hard to distinguish the reconstructions from real images. The validation reconstructions, shown in Figure 6.22, are at a similar level compared to the 256 dimensions model where the features are not captured well and the generation quality is quite blurry. It is clear here that these models overfit heavily on training data, testing them with unseen data further results in reconstructions that are not as close to the original image. An increase in latent dimensions with this little data seems to improve training reconstructions, but not the generalisability of the model and latent space.

**Variant VAE - 512D Image Reconstructions**
The model results here are very similar to the vanilla VAE with 512 dimensions, where the training reconstructions, displayed on Figure 6.23, are very close to the original images. The validation reconstructions, in Figure 6.24, on the other hand do not capture the features well. Testing the model with a small data set is clearly insufficient for capturing features effectively and ensuring a proper level of generalisability. Therefore, increasing the data set size to include more features is vital and the next step.

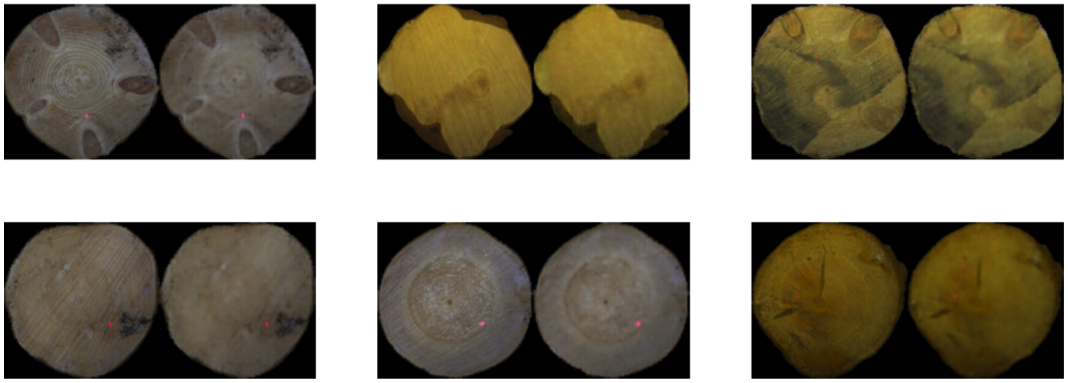

Figure 6.21: 512D Vanilla VAE reconstructions on random training samples.

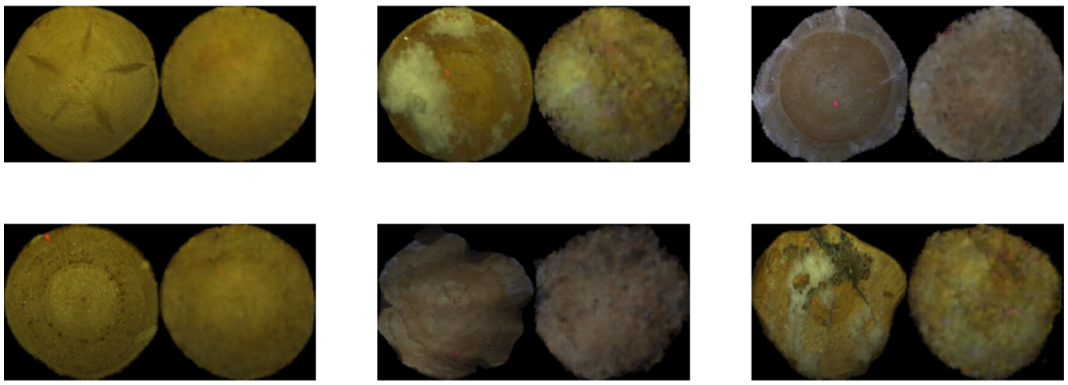

Figure 6.22: 512D Vanilla VAE reconstructions on random validation samples.

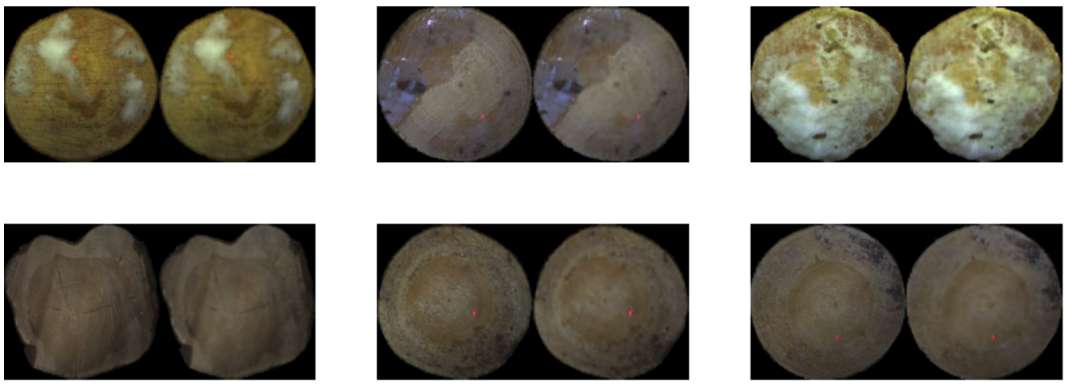

Figure 6.23: 512D Variant VAE reconstructions on random training samples.

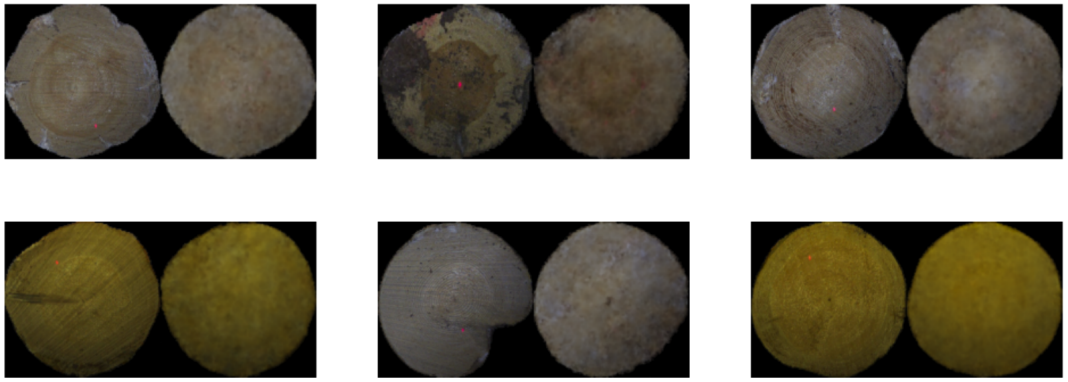

Figure 6.24: 512D Variant VAE reconstructions on random validation samples.

### 6.1.2 Results - 20K data set Training's

The second data set is the 20K data set, using a 0.8/0.1/0.1 training/validation/testing split. This results in 16K images being used for training, with 2K images for validation and testing each. The 20K data set was used for evaluating the impact of the 20 times data increase from 1K data, with the same latent space size, aiming to observe if it is learning anything at all. The models were not fully trained and therefore reconstructions are not covered on the two models trained on this data, the same their learning and losses over training time, as 100 epochs do not provide much insight in the learning.

**Final losses**
After 100 epochs on both models, the variant VAE achieves lower losses on all training losses compared to the vanilla VAE. The KL loss is also lower on validation data with the variant model, where the reconstruction and total ELBO loss are higher compared to what the vanilla VAE achieves. The KL loss of the variant is also noticeably under the KL loss of the vanilla model.

| Model | Epochs | ELBO | Reconstruction | KL Loss |
|---|---|---|---|---|
| Vanilla VAE - 512D | 100 | 10802.96 | 10001.71 | 801.25 |
| Variant VAE - 512D | 100 | 9906.49 | 9341.26 | 565.22 |

Table 6.8: Models Trained on 20K data set - Training Losses

| Model | Epochs | ELBO | Reconstruction | KL Loss |
|---|---|---|---|---|
| Vanilla VAE - 512D | 100 | 12999.11 | 12181.11 | 817.92 |
| Variant VAE - 512D | 100 | 14632.25 | 14115.57 | 516.66 |

Table 6.9: Models Trained on 20K data set - Validation Losses

### 6.1.3 Results - 100K data set Training's

The third and final training data set is the 100K data set, using a 0.8/0.1/0.1 training/validation/testing split. This results in 80K images being used for training, with

10K images for validation and testing each. The vanilla VAE model was trained in two sessions of 200 epochs each, 400 epochs in total. The variant VAE model on the other hand was trained in four sessions of 100 epochs each, also 400 epochs in total. The reason for this is that when training on data this large, the virtual machines need to train for a large amount of time. Where the variant VAE needs about 2-3x more time to train, due to all the sub-networks and the current implementation. Therefore training in sessions was necessary for mitigating memory issues with the available computing. This does however slightly affect the training, as restarting a session initialises the batches differently, causing some drops in losses with the new batches. The vanilla VAE was trained before the variant VAE, and therefore it was not initially planned to do the training in 4 sessions. Although this slightly biases the results, the impact of this in the end is minimal as the data set is very large and both were trained for 400 epochs in total.

**Final losses**
On the large data sets the model still achieved similar results for the KL losses as with the other data sets, where the variant achieves a lower KL divergence. But the reconstruction in training and validation are both lower with the vanilla model, something not appearing in training on smaller data sets. For this reason, the total ELBO loss is also higher for the variant VAE.

| Model | Epochs | ELBO | Reconstruction | KL Loss |
|---|---|---|---|---|
| Vanilla VAE - 1024D | 400 | 10258.25 | 9372.83 | 885.43 |
| Variant VAE - 1024D | 400 | 10885.14 | 10122.26 | 762.88 |

Table 6.10: Models Trained on 100K data set - Training Losses

| Model | Epochs | ELBO | Reconstruction | KL Loss |
|---|---|---|---|---|
| Vanilla VAE - 1024D | 400 | 12530.28 | 11661.82 | 868.60 |
| Variant VAE - 1024D | 400 | 13452.34 | 12719.42 | 732.96 |

Table 6.11: Models Trained on 100K data set - Validation Losses

**Losses during training**
The reconstruction training losses, as visualised on Figure 6.25, seem to follow both the same trend. Where the vanilla VAE seems to reach a flat plateau compared to the variant VAE, as the difference between reconstruction losses becomes smaller over training time. The reconstruction losses in validation both seem to drop heavily at first to then both reach a plateau and no longer improve.

The KL losses, as visualised on Figure 6.26, show very similar trends for both the training and validation sets of both the vanilla and variant models. The KL losses do reach a plateau and very lightly increase over time. The light increase in KL loss over training seems to correlate highly with the reconstruction slowly dropping, as the model is trading off KL loss for reconstruction loss. The variant VAE increases more than the vanilla VAE, aligning with the variant still dropping more in reconstruction loss compared to the vanilla VAE. It is still unsure here if this trend would continue over significantly more training and if the variant would surpass the vanilla model with a higher KL loss, but lower reconstruction loss. Or

if the models reach a plateau and the losses would not be affected much with more training.

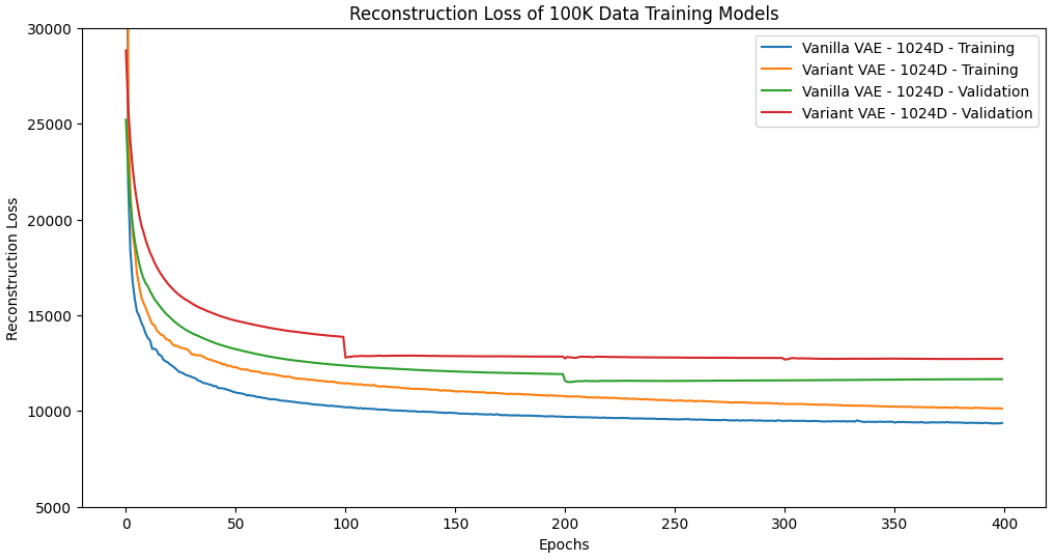


Figure 6.25: Reconstruction Loss during the training of the models on the 100K data set.


Figure 6.26: KL Loss during the training of the models on the 100K data set.

**Vanilla VAE - 1024D Image Reconstructions**

The reconstructions of the vanilla VAE on the large data set have evidently learned the features quite well, as visible on the training reconstructions in Figure 6.27 and the validation reconstructions in Figure 6.28. Even rather very small features such as dots or laser pointers on the logs are visible in the reconstructions and not lost in the training process. However the reconstructions contain a lot of blurriness, and

despite most features being present, a lot of detail is lost on the log rings. Especially in validation reconstructions where the blurriness is heavier.

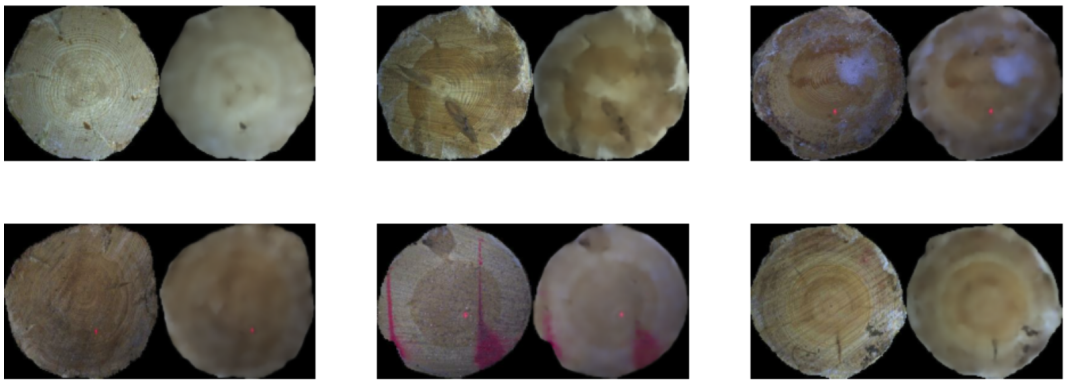

Figure 6.27: 1024D Vanilla VAE reconstructions on random training samples.

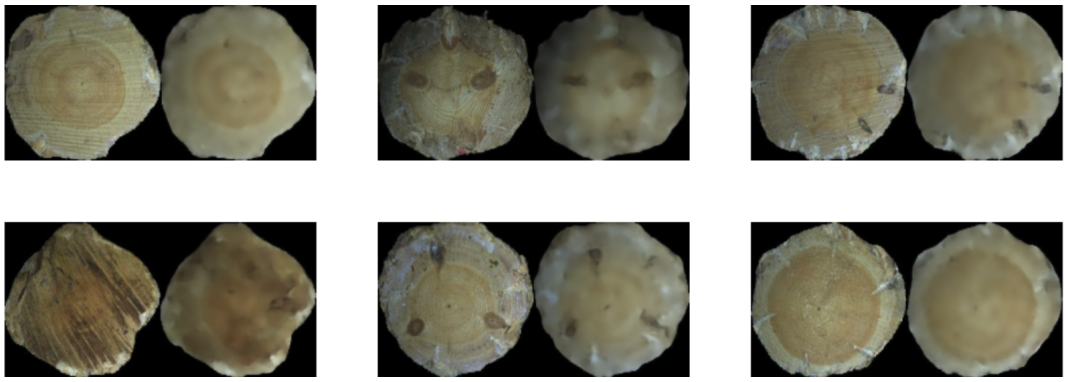

Figure 6.28: 1024D Vanilla VAE reconstructions on random validation samples.

**Variant VAE - 1024D Image Reconstructions**

For the reconstructions coming from the trained variant model on the large data set, there are not many differences with the vanilla VAE model reconstructions, as shown in Figure 6.29 for training reconstructions and in Figure 6.30 for validation reconstructions. Since the reconstructions also yield blurry images, it becomes challenging to visually compare them. With the variant resulting in higher reconstruction errors, it can be assumed that the blurriness is also slightly higher on these reconstructions. Besides the poor image generation quality, the features still seem to be well learned by the model.

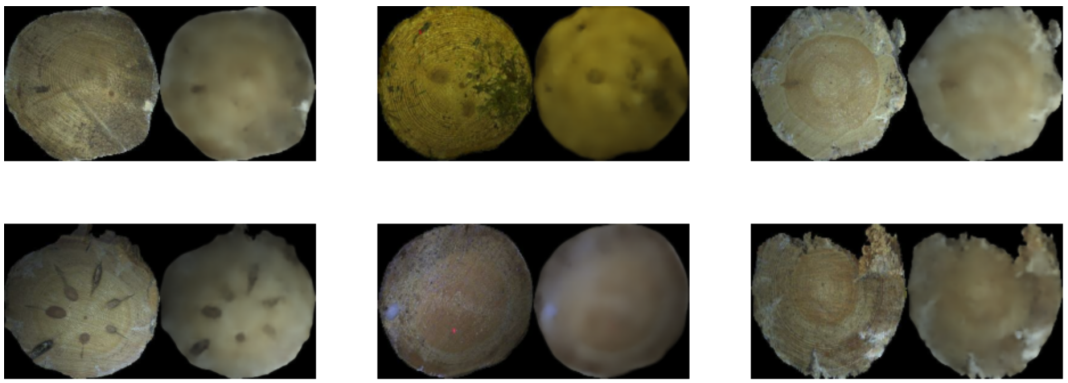

Figure 6.29: 1024D Variant VAE reconstructions on random training samples.

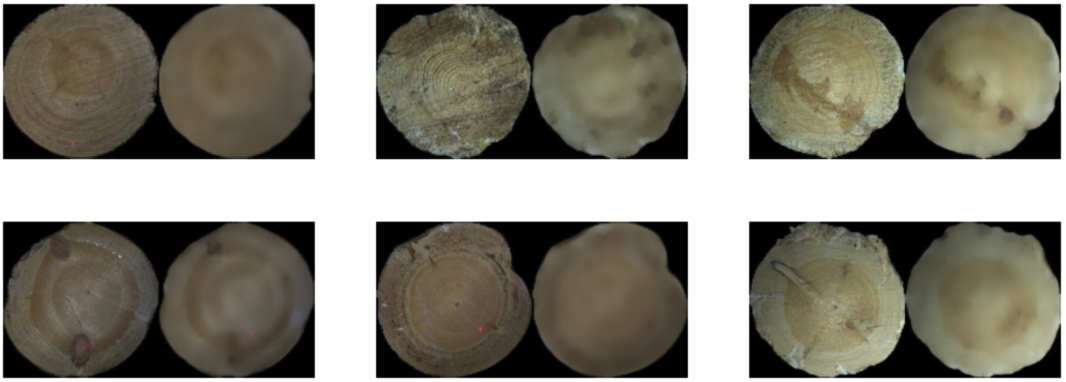

Figure 6.30: 1024D Variant VAE reconstructions on random validation samples.

**VAE Prediction Time Performances**

On top of just evaluating the performance of the model in generation quality and training losses, another important aspect is generation performance on a time basis. In some environments, e.g. a production pipeline, time can be very restricted. In such occasions prediction times play a large role. The process of a model predicting includes the complete network, thus the encoder and decoder, where one image goes through the network completely and is reconstructed at the end. To evaluate this, predictions were done on the 10K images test data set. The predictions were done 5 times in total per model to obtain an average performance, using GPU.

| **Model** | **Time in seconds** |
|---|---|
| Vanilla VAE - 1024D | 38.591s |
| Variant VAE - 1024D | 1108.081s |

Table 6.12: VAE Average Prediction Times on 10K Data

The vanilla VAE model shows a significantly faster prediction time, of about 28.71x times faster than the variant VAE. Although the variant VAE has a few million trainable parameters more, its architecture has many more layers. The current

implementation done in this thesis project could be more optimised and parallelised. These reasons lead to way slower prediction times for the variant. Although the prediction time per image is still rather low for the variant VAE, it might be too slow to deploy this architecture in production, as of now. Where the vanilla architecture could be more interesting to have deployed.

**Conclusion on VAE experiments**
Overall the VAE models, both vanilla and variant architectures, learned the features well. On small data sets the training reconstructions were quite good and of high quality. However the validation reconstructions with small data sets ended up very poorly, caused by the model not learning enough features due to limited data and therefore not generalising well over unseen data. When training on larger data sets, the gap between training and validation losses noticeably improved and grew smaller. Trained on the larger data sets, the models returned poor reconstructions on a quality basis, with a lot of information, especially regarding log rings, being lost in generation. On log ends with less variance in features, the reconstructions ended up worse compared to logs rich of different features. Nonetheless, all the VAE training's showed good feature learning with a lot of small details being extracted and learned during training. The variant architecture however seems to be performing significantly slower compared to the vanilla architecture, making it less interesting for implementation in highly time constricted environments.

## 6.2 Triplet Sampling

The triplet sampling algorithm has different parameters that affect the output of the positive and negative. For both of the sampled points there is the radius parameter and the sampling towards the mean parameter. The parameters for the algorithm were experimented with and tuned based on the visual quality and the feature-richness in the decoded samples. This was not done through any metrics, but rather in a subjective manner.

In the end, the following parameters were found to give the most appealing results on average, over the generated triplets, based on the visual assessment. For the positive, a radius of 0.15 and probability of sampling towards the mean of 0.95 is selected. The positive images still look quite similar to the anchor images, but with slight adjustments in visual features. Here colour is the main feature changing. For the negative, a radius of 0.30 and probability of sampling towards the mean of 0.85 is selected. The radius for the negative is expected to be further than the radius for positive, in order to sample a more "different" log end. The negative image also has a lower probability of sampling towards the mean, so more noise and extreme features are encoded. With the selected radius, more drastic changes to the log ends happen, where colours and shape change a lot. The inner structure of the log end also changes more when sampling further away from the anchor. Below in Figure 6.31, an example of a generated triplet with the suggested parameters is given.

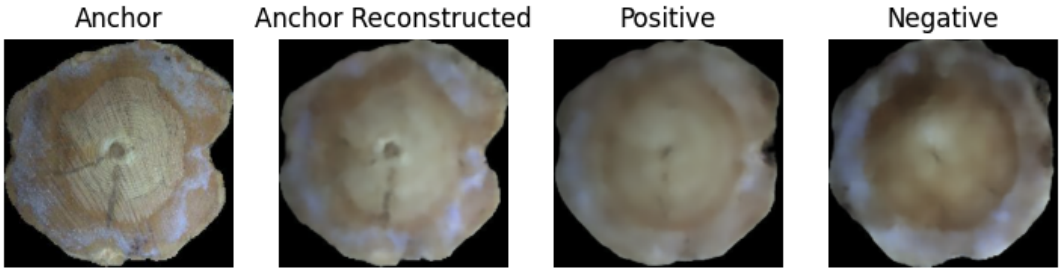


Figure 6.31: An example of a triplet with the chosen parameters.

On Figure 6.32, the chosen radii are used for sampling, but the probability of sampling towards the mean is just purely random, so with a 0.5 probability in the direction of the mean. This visually shows that these parameters highly impact the sampling and make a significant contribution to the proposed sampling algorithm itself.

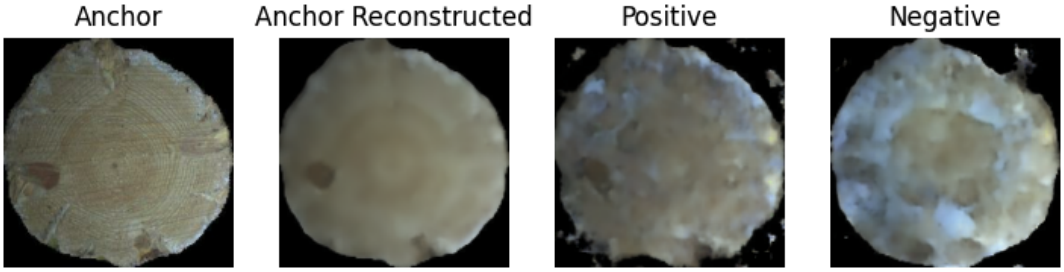


Figure 6.32: Bad Triplets

No metrics were used for tuning the parameters of the sampling algorithm. However, in future work the AI models that use the sampled images can be used for optimising the parameters. Where networks can be learned on different parameter sets, and the optimal parameter set can be found. More generated triplets can be found in Appendix **C**.

### 6.3 Recognition Models

After the experiments with training the VAE models and generating triplet data sets finished, proceeding to the experiments with the recognition model was possible. All of the recognition models were trained with the exact same configuration. In order to be able to evaluate and compare the performances of the models trained on VAE generated data, a baseline was initially trained on the real data triplets using a 0.7/0.15/0.15 training/validation/testing split, for 1470 real triplets. Afterwards for both the 100K data trained VAE models, two recognition models were trained, one on 1470 generated triplets and one on 10K generated triplets. In total four recognition models trained on generated log end images. The recognition models on generated data used a 0.7/0.3 training/validation split with all the 1470 real triplets as testing data. The real triplets were used as testing data to evaluate if the model was applicable on real log end images, since a difference between generated

and real images is definitely present with image quality. The same triplet margin was used for all models with a value of 0.6. All these training's allowed for comparison and evaluation on if such models are in any way applicable to real data, and can therefore be used as data extension/augmentation/replacement.

The baseline resulted in quite a good accuracy for being trained on only 1470 triplets, especially its validation and testing accuracy demonstrate that it can be used for unseen log ends as well and is robust to a certain degree. The 1470 sampled data set training's resulted in average training accuracies, proving there is some learning on recognition. It however seems to perform poorly on validation accuracy making it less applicable for unseen data, with further a very low accuracy on testing with real data. The same results are found with the 10K sampled images data sets. There seems to not be any improvement in learning with more data and the models trained on generated data seem to not be applicable on real testing data.

| **Model** | **Epochs** | **Training** | **Validation** | **Testing** |
|---|---|---|---|---|
| Baseline Real | 33 | 94.07% | 89.09% | 85.45% |

Table 6.13: Models Trained on 1470 Real data set - Accuracy Losses

| **Model** | **Epochs** | **Training** | **Validation** | **Testing** |
|---|---|---|---|---|
| Baseline Standard VAE | 49 | 76.19% | 41.95% | 0.68% |
| Baseline Variant VAE | 35 | 64.52% | 35.14% | 0.61% |

Table 6.14: Models Trained on 1470 Sampled data set - Accuracy Losses

| **Model** | **Epochs** | **Training** | **Validation** | **Testing** |
|---|---|---|---|---|
| Baseline Standard VAE | 43 | 58.90% | 40.96% | 1.36% |
| Baseline Variant VAE | 57 | 57.32% | 33.93% | 0.74% |

Table 6.15: Models Trained on 10k Sampled data set - Accuracy Losses

**Losses during training**

The losses during training indicate the same results as found above, where the training accuracy all seems to follow a similar trend, but still very dissimilar from real baseline model, as visualised in Figure 6.33. The models on 1470 data seem to perform and train better compared to the 10K data models, based on training accuracy, with also the vanilla VAE performing better than the variant VAE. The validation losses on Figure 6.34 do not seem to improve over training while also being quite low in accuracy. All the generated data trained models follow the same trend all together, with again the vanilla VAE performing better than the variant VAE. At last the testing accuracy, as visualised on Figure 6.35, demonstrates clearly that the model does not improve over time on real data and is not applicable on it.

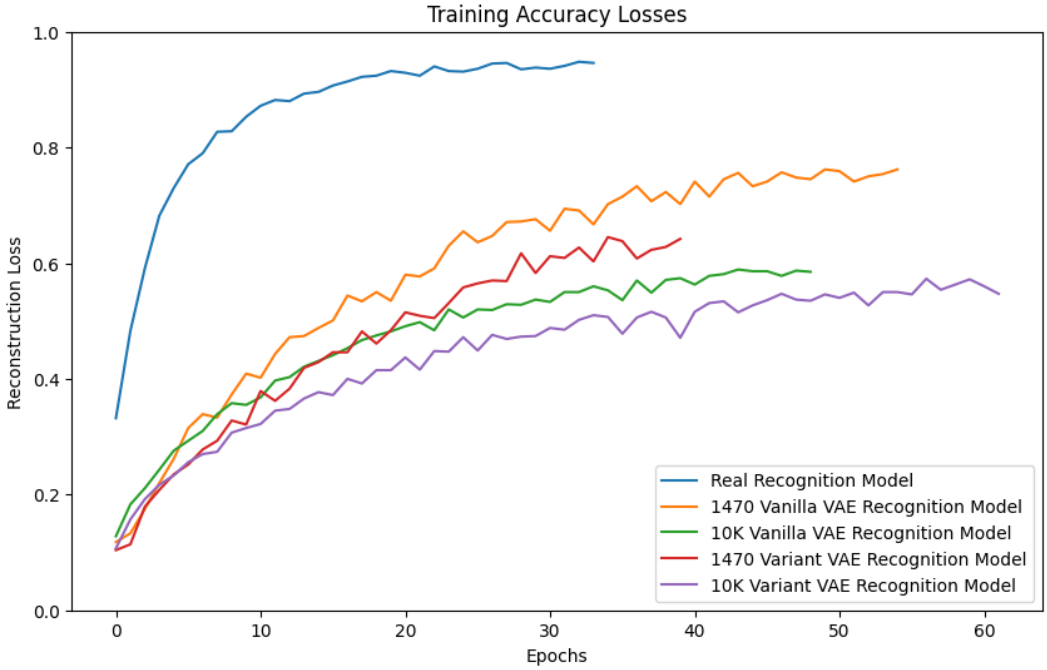


Figure 6.33: Image reconstructions on random samples from validation data set.

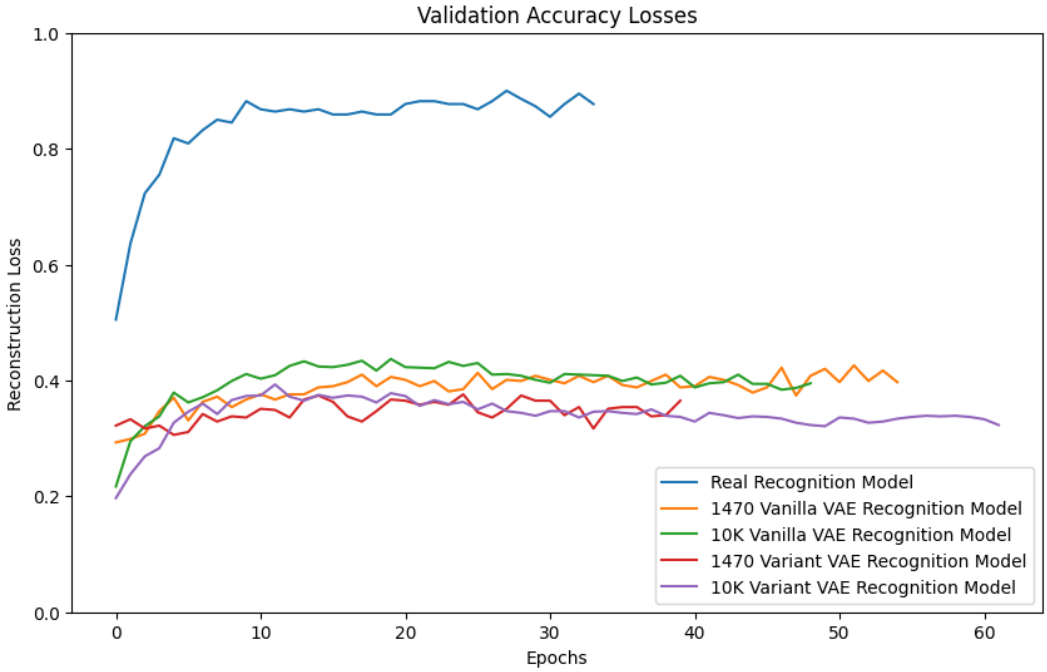


Figure 6.34: Image reconstructions on random samples from validation data set.

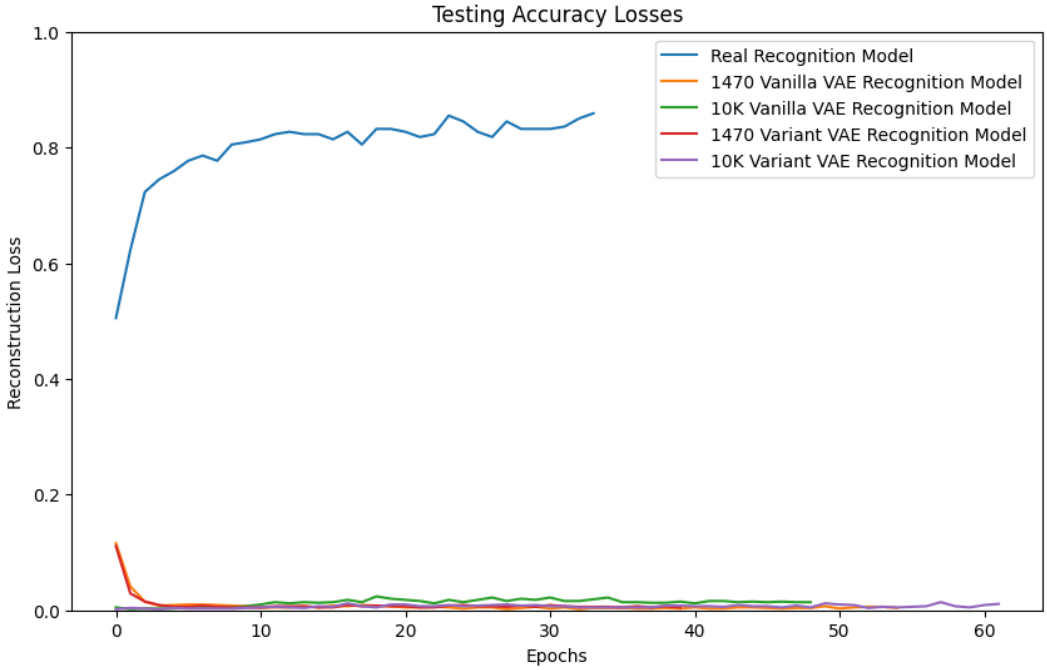


Figure 6.35: Image reconstructions on random samples from validation data set.

**Conclusion on recognition experiments**
The experiments on recognition models conclude that the models trained on VAE generated data do learn, but not sufficiently enough with the used parameters and model settings. On top of that, these trained models are not at all applicable on real data, making it impossible to use the data as extension or replacement at this point in research. It is assumed that the reason for this is the difference in image generation quality between the real data and the generated data, and the issues found in VAE generation propagating to the recognition model stage. In addition, the vanilla VAE performed slightly better than the variant VAE, but the variant VAE ended up with a worse reconstruction loss after 400 epochs, therefore making it hard to really compare the models when the image quality is highly affecting the results.

## 6.4 Experiment Validity

In Section **3** the reliability and validity of the project have been discussed. However, during the controlled experiment, another construct validity concern has been added. The validity concern is regarding the training sessions of the large models, where four restarts were done for the variant, but only two restarts for the base model. The restarts were necessary due to computational reasons. This concern only came up after the base model had been trained and the variant VAE was halfway training. As these models take quite some time to train, retraining them was not worth it. The impact of this is that the restart causes different random batch initialisations, as no seeds were selected to avoid any research bias. These batch initialisations impact the validation losses slightly, as these data sets are rather small compared to training data and sensitive to batch initialisation. Although the impact is minimal, it should be taken into account that the losses might be slightly biased in comparison due to the training sessions for computational reasoning.

Another validity concerns appears with the hyperparameter optimisation of the triplet algorithm parameters. The hyperparameters were chosen in a subject manner, instead of any quantitative basis. It would require a lot of time to optimise the parameters based on the recognition model results as well, therefore such an approach was not feasible within the thesis project. Due to the subjective manner in which the parameters were decided, a slight research bias could appear therefore slightly affecting the results.

# 7 Discussions

In this section the research questions are evaluated and discussed, based on the results achieved in this thesis project in the literature review and controlled experiments.

**RQ1: What generative variant can be constructed with the goal of learning individual data distributions better than the standard approach?**
In this thesis, we proposed a VAE variant approach, as visible on Figure 7.36, that adapts the base architecture with sub-networks for each latent dimension in the latent space. The goal of this variant was to learn the latent space more, learn better features and with this generate more feature-rich images. Qualitatively the differences between the vanilla VAE architecture and the proposed variant are hard to observe. An immediate impact on the generation of images itself is not really visible. On quantitative basis, the variant however achieved lower KL-Divergence losses on all variant models, on both training and validation. This while having little impact on the reconstruction qualities. We can conclude that the constructed variant achieved slightly better learning of individual data distributions compared to the standard VAE approach on this specific problem and data set. Due to the constructed variant being tested on only this problem and data, the results are not sufficient to be generalisable over other problems. The resource requirements with the current technical implementation are very high and about double of the standard VAE approach, the prediction time is also significantly larger. A trade-off and analysis on if this is worth it should be done for further use of the constructed variant approach, especially in environments where time is highly constricted.

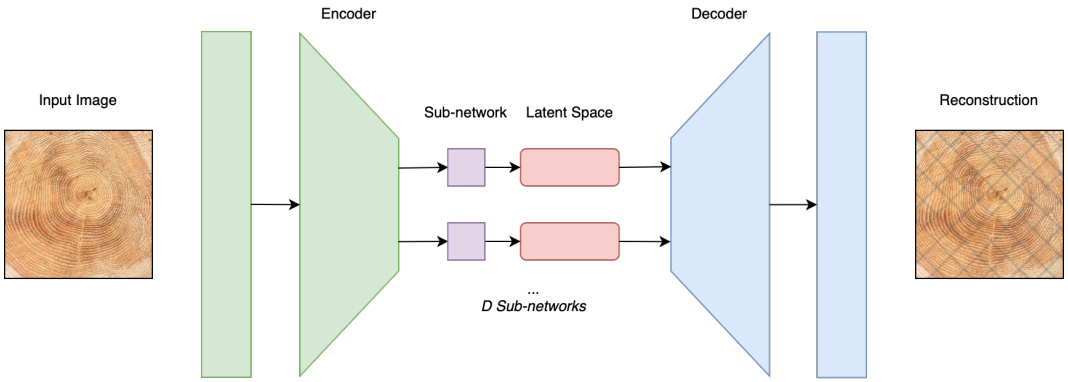


Figure 7.36: Variant VAE Architecture.

**RQ2: How does the applied generative variant and base model compare to other generative deep learning approaches in image synthesis?**
A literature review was done for findings on related work with the different deep generative models. During this process many strengths and weaknesses in the different models were identified, which variants and hybrid models attempted to approach. This search for related work allowed for detailed analysis of a sample of the research out there, and with this allowed the opportunity for a comparison between the deep generative models. The comparison findings are covered in Section **4.5** (Literature Review). In the literature review itself, the base model and variant

were taken as one architecture under the search for VAE, as at the point of research, the results of the models were not found. Eventually the weaknesses found in the base VAE model are also reflected in the generative variant and application within this thesis project, such as the blurriness and high reconstruction losses with the complex data. This opens up the opportunity for applying similar solutions as other variant and hybrid models out there, to in the future mitigate and solve the occurring problems regarding reconstruction qualities appearing now.

**RQ3: How can the proposed variant and base model contribute for triplet generation, for improving a recognition model on artificial data?**
The variant and base model learned latent spaces were used for triplet generation, where we proposed an algorithm to sample triplet images from the latent space. This was first done by obtaining an anchor, getting its encoding, and from the encoded point in space the positive and negative points in space were sampled. This was done based on a log variance distance from the anchor and a probability of sampling towards the mean of each latent distribution. In this way both the trained variant and base model were able to contribute for triplet generation, with in the end quite feature-rich tree log-ends. The obtained sampled triplets were then used for training a recognition model. Due to the high reconstruction losses and the appearing blurriness in the image synthesis, the recognition model trained on sampled triplets could not contribute for proper evaluation on real images, as the differences were too large between the real and generated data. Therefore the generated data, at this point in the research, is not capable of improving another model and consequently unable to mitigate costs such as data collection. We reason that this is because of the quality differences.

# 8 Conclusions and Future Work

In this section the final conclusions of the research and work are presented, followed by future work suggestions.

**Conclusions**

The motivations for this thesis project were to construct a full deep learning pipeline, apply a deep generative model, propose a variant generative approach and to introduce a sampling algorithm. All of this in the hopes of improving a recognition model on artificial data and evaluating the usability and capabilities on an industrial forestry application. In the end, the construction of these individual components was successful. However, the ideal situation where a VAE model can generate large amounts of high quality tree log end images and consequently improve a recognition model with artificial data was not reached. Nonetheless, there are still many interesting results to be found, opening many opportunities for future work.

The VAE image generation did prove quite good feature learning, where the proposed variant VAE in this application returned better losses and feature extraction through an adapted architecture using sub-networks for each latent space. Although the variant requires more resources, longer training and more total parameters, this trade-off can definitely be worth it to achieve better feature learning. The reconstruction and generation quality itself did not reach the ideal point, resulting in a large amount of blurriness appearing in the final decoded images. We reason that this is most likely due to the implemented architecture, meaning that the used data set requires more complexity in the architecture, especially in the decoder, in order to have better decoded reconstructions from the encoded representations. We do also recognise that tree log ends are not the easiest of image data to learn, due to a very large variance in the data and logs often containing very different features. Therefore, more data or the option of image augmentations might also be required for better learning. The appearing blurriness is a commonly appearing problem with VAE models, which was also noticeable here.

The presented triplet sampling algorithm was not able to evaluated properly on a quantitative basis due to the problems occurring in the VAE reconstructions and therefore also propagating through to the recognition model. The algorithm nonetheless did prove to be able to generate many triplets that were sampled in a creative way and showed very different features based on the sampling parameters.

The recognition models trained well on the sampled triplets themselves, and returned quite good losses for a first iteration in this research. Unfortunately, the recognition models trained on VAE sampled triplets proved to be not applicable to real triplet images as testing data, where the recognition model achieved very low accuracies on recognition when trained on generated data.

This thesis work contributes to the field of generative models and the forestry industry. Not only by proving that the variant VAE network, triplet sampling algorithm and individual recognition models still performed quite well, but also by proving that the current state of this research is lacking the generation complexity and quality in order to use the synthesis for improving recognition models. Future work might be able to solve this problem. On top of that, this thesis presents a literature review on the different deep generative models and the differences in achievements between them. Although the primary studies found in this literature search are only a sample of all the research out there, it still contributes through the exploration of

findings in the sample of studies, as well as the weaknesses, strengths, variants and applications found and identified in research of deep generative models.

**Future Work**

There are several directions for future work based on the results of this thesis project. The first step for future work on the VAE that we suggest is a significantly more complex decoder network and more experimentation with it, while also further investigating variants for improved quality. The image synthesis results ended up quite blurry, while the features were still learned quite well and were mostly reconstructed. A heavier architecture than used in this project could improve the reconstruction and decrease losses. On top of that, the encoder weights could be initialised to the already trained weights from the trained models on large data. A second suggestion for future work is the experimentation of different losses, a different loss might lead to improved reconstructions as well. The variant architecture implementation in this thesis was not the most optimised and built on quite a high level using the TensorFlow and Keras libraries, a lower level implementation might significantly increase performance and utilise resources better. Additionally, it would be interesting to experiment the variant on more data sets to test the generalisability and performances on other problems. A different approach to the log end image generation could also be an interesting direction with something that works better for a VAE, such as a combination with a GAN for optimised synthesis.

For the triplet sampling approach, it would be interesting to add more stochastic variables to the algorithm, in order to sample more different triplets, allowing to use anchors several times and still producing very different triplets containing creative feature generation. It would also be very interesting to analyse the latent space more, with the help of dimensionality reduction techniques, such as t-SNE, in order to generate a more structured mapping of the latent space and identify clusters. In this way, the sampling algorithm can also be more optimised towards the latent space. For example, by finding interesting points in the latent space and sampling triplets based on linear interpolations.

At last, it could be interesting to experiment more with the recognition model stage as well. Suggested options here are experimenting with different pretrained models and hyperparameters such as the triplet margin. This was out of the scope of this thesis due to time restrictions, but would still be very interesting to experiment with in advance of investing in VAE improvement approaches. On top of that, it could be very interesting to sample new triplets during training that are considered semi-hard triplets, as only these are used with the semi-hard triplet loss. This has the potential of learning faster and better, as more triplets would be used for training.

This thesis project has many directions for future work. The above are not the only ones, but at least the ones we think to be interesting and suggest to follow up on the research done in this thesis.

## References


[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, jul 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[2] I. Cloudera. (2020) Deep learning for anomaly detection. [Online]. Available: https://ff12.fastforwardlabs.com

[3] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8821–8831. [Online]. Available: https://proceedings.mlr.press/v139/ramesh21a.html

[4] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with CLIP latents," *CoRR*, vol. abs/2204.06125, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.06125

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 684–10 695.

[6] OpenAI, "Gpt-4 technical report," 2023.

[7] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6114

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press, 1986, p. 318–362.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[10] L. Dinh, D. Krueger, and Y. Bengio, "NICE: non-linear independent components estimation," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1410.8516

[11] A. van den Oord, O. Vinyals, and k. kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus,

S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf

[12] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Paper.pdf

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

[14] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[15] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2242–2251.

[17] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=HkpbnH9lx

[18] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf

[19] M. van Huijstee, P. van Boheemen, D. Das, L. Nierling, J. Jahnel, M. Karaboga, M. Fatun, L. Kool, and J. Gerritsen, "Tackling deepfakes in european policy," 2021.

[20] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[22] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html

[23] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[24] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 2016, pp. 241–246.

[25] A. Majumdar, "Blind denoising autoencoder," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 1, pp. 312–317, 2019.

[26] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao, "Coupled deep autoencoder for single image super-resolution," *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 27–37, 2017.

[27] J. Chow, Z. Su, J. Wu, P. Tan, X. Mao, and Y. Wang, "Anomaly detection of defects on concrete structures with the convolutional autoencoder," *Advanced Engineering Informatics*, vol. 45, p. 101105, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034620300744

[28] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[29] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019. [Online]. Available: http://dx.doi.org/10.1561/2200000056

[30] L. P. Cinelli, *Variational methods for machine learning with applications to deep networks*, 1st ed. Cham, Switzerland: Springer, 2021.

[31] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in $\beta$-vae," *CoRR*, vol. abs/1804.03599, 2018. [Online]. Available: http://arxiv.org/abs/1804.03599

[32] A. Vahdat and J. Kautz, "Nvae: A deep hierarchical variational autoencoder," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 19 667–19 679. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/e3b21256183cf7c2c7a66be163579d37-Paper.pdf

[33] E. Tabak and E. Vanden-Eijnden, "Density estimation by dual ascent of the log-likelihood," *Communications in Mathematical Sciences - COMMUN MATH SCI*, vol. 8, 03 2010.

[34] E. Tabak and T. Cristina, "A family of nonparametric density estimation algorithms," *Communications on Pure and Applied Mathematics*, vol. 66, 02 2013.

[35] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1530–1538. [Online]. Available: https://proceedings.mlr.press/v37/rezende15.html

[36] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6c1da886822c67822bcf3679d04369fa-Paper.pdf

[37] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1747–1756. [Online]. Available: https://proceedings.mlr.press/v48/oord16.html

[38] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf

[39] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, nov 2021.

[40] T. White, "Sampling generative networks: Notes on a few effective techniques," *CoRR*, vol. abs/1609.04468, 2016. [Online]. Available: http://arxiv.org/abs/1609.04468

[41] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015.

[42] B. A. Kitchenham, "Procedures for performing systematic reviews," Keele University, Department of Computer Science, Keele University, Kelee, UK, Tech. Rep., 07 2004. [Online]. Available: http://www.it.hiof.no/~haraldh/misc/2016-08-22-smat/Kitchenham-Systematic-Review-2004.pdf

[43] W. Harvey, S. Naderiparizi, and F. Wood, "Conditional image generation by conditioning variational auto-encoders," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=7MV6uLzOChW

[44] C. Lai, D. Zou, and G. Lerman, "Robust vector quantized-variational autoencoder," *CoRR*, vol. abs/2202.01987, 2022. [Online]. Available: https://arxiv.org/abs/2202.01987

[45] R. Child, "Very deep vaes generalize autoregressive models and can outperform them on images," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=RLRXCV6DbEJ

[46] L. Lin, X. Liu, and W. Liang, "Improving variational auto-encoder with self-attention and mutual information for image generation," in *Proceedings of the 3rd International Conference on Video and Image Processing*, ser. ICVIP 2019. New York, NY, USA: Association for Computing Machinery, 2020, p. 162–167. [Online]. Available: https://doi.org/10.1145/3376067.3376090

[47] S. Norouzi, D. J. Fleet, and M. Norouzi, "Exemplar vae: Linking generative models, nearest neighbor retrieval, and data augmentation," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 8753–8764. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/63c17d596f401acb520efe4a2a7a01ee-Paper.pdf

[48] Q. Ai, L. HE, S. LIU, and Z. Xu, "Bype-vae: Bayesian pseudocoresets exemplar vae," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 5910–5920. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/2e9f978b222a956ba6bdf427efbd9ab3-Paper.pdf

[49] Z. Wu, L. Cao, and L. Qi, "evae: Evolutionary variational autoencoder," *CoRR*, vol. abs/2301.00011, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2301.00011

[50] E. Egorov, A. Kuzina, and E. Burnaev, "Boovae: Boosting approach for continual learning of vae," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 17 889–17 901. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/952285b9b7e7a1be5aa7849f32ffff05-Paper.pdf

[51] L. Jiang, B. Dai, W. Wu, and C. C. Loy, "Focal frequency loss for image reconstruction and synthesis," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 13 899–13 909.

[52] K. Dohi, "Variational autoencoders for jet simulation," 2020. [Online]. Available: https://arxiv.org/abs/2009.04842

[53] C.-T. Tu and Y.-F. Chen, "Facial image inpainting with variational autoencoder," in *2019 2nd International Conference of Intelligent Robotic and Control Engineering (IRCE)*, 2019, pp. 119–122.

[54] D. Jiang, G. Zhang, M. Karami, X. Chen, Y. Shao, and Y. Yu, "Dp$^2$-vae: Differentially private pre-trained variational autoencoders," *CoRR*, vol. abs/2208.03409, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.03409

[55] R. Wei, C. Garcia, A. El-Sayed, V. Peterson, and A. Mahmood, "Variations in variational autoencoders - a comparative evaluation," *IEEE Access*, vol. 8, pp. 153 651–153 670, 2020.

[56] H. Huang, z. li, R. He, Z. Sun, and T. Tan, "Introvae: Introspective variational autoencoders for photographic image synthesis," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/093f65e080a295f8076b1c5722a46aa2-Paper.pdf

[57] T. Daniel and A. Tamar, "Soft-introvae: Analyzing and improving the introspective variational autoencoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 4391–4400.

[58] L. Changjie, Z. Shen, W. Zirui, D. Omar, and G. Gaurav, "As-introvae: Adversarial similarity distance makes robust introvae," in *Proceedings of The 14th Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. Khan and M. Gonen, Eds., vol. 189. PMLR, 12–14 Dec 2023, pp. 658–673. [Online]. Available: https://proceedings.mlr.press/v189/changjie23a.html

[59] K. Pandey, A. Mukherjee, P. Rai, and A. Kumar, "DiffuseVAE: Efficient, controllable and high-fidelity generation from low-dimensional latents," *Transactions on Machine Learning Research*, 2022. [Online]. Available: https://openreview.net/forum?id=ygoNPRiLxw

[60] D. Lee, C. Kim, S. Kim, M. Cho, and W.-S. Han, "Autoregressive image generation using residual quantization," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 11 513–11 522.

[61] X. Xiao, S. Ganguli, and V. Pandey, "Vae-info-cgan: Generating synthetic images by combining pixel-level and feature-level geospatial conditional inputs," in *Proceedings of the 13th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, ser. IWCTS '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3423457.3429361

[62] A.-A.-Z. Imran and D. Terzopoulos, "Multi-adversarial variational autoencoder networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 777–782.

[63] H. Shao, S. Yao, D. Sun, A. Zhang, S. Liu, D. Liu, J. Wang, and T. Abdelzaher, "ControlVAE: Controllable variational autoencoder," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 8655–8664. [Online]. Available: https://proceedings.mlr.press/v119/shao20b.html

[64] K. Ezukwoke, A. Hoayek, M. Batton-Hubert, and X. Boucher, "GCVAE: generalized-controllable variational autoencoder," *CoRR*, vol. abs/2206.04225, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2206.04225

[65] B. Estermann, M. Marks, and M. F. Yanik, "Robust disentanglement of a few factors at a time using rpu-vae," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 13 387–13 398. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/9b22a40256b079f338827b0ff1f4792b-Paper.pdf

[66] M. H. Ebrahimabadi, "Disentangled representation learning using ($\beta$-)vae and GAN," *CoRR*, vol. abs/2208.04549, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.04549

[67] T. A. Keller and M. Welling, "Topographic vaes learn equivariant capsules," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 28 585–28 597. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/f03704cb51f02f80b09bffba15751691-Paper.pdf

[68] T. Bepler, E. Zhong, K. Kelley, E. Brignole, and B. Berger, "Explicitly disentangling image content from translation and rotation with spatial-vae," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/5a38a1eb24d99699159da10e71c45577-Paper.pdf

[69] J. Xu, Y. Ren, H. Tang, X. Pu, X. Zhu, M. Zeng, and L. He, "Multi-vae: Learning disentangled view-common and view-peculiar visual representations for multi-view clustering," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9214–9223.

[70] R. Pastrana, "Disentangling variational autoencoders," *CoRR*, vol. abs/2211.07700, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.07700

[71] Y. Poirier-Ginter and J. Lalonde, "Robust unsupervised stylegan image restoration," *CoRR*, vol. abs/2302.06733, 2023. [Online]. Available: https://doi.org/abs-2302-06733

[72] O. Lang, Y. Gandelsman, M. Yarom, Y. Wald, G. Elidan, A. Hassidim, W. T. Freeman, P. Isola, A. Globerson, M. Irani, and I. Mosseri, "Explaining in style: Training a gan to explain a classifier in stylespace," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 673–682.

[73] Y. Alaluf, O. Patashnik, and D. Cohen-Or, "Restyle: A residual-based stylegan encoder via iterative refinement," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 6691–6700.

[74] G. Kwon and J. C. Ye, "Diagonal attention and style-based gan for content-style disentanglement in image generation and translation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 13 960–13 969.

[75] Y. Lu, S. Wu, Y.-W. Tai, and C.-K. Tang, "Image generation from sketch constraint using contextual gan," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[76] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.06434

[77] E. Bolluyt and C. Comaniciu, "Collapse resistant deep convolutional gan for multi-object image generation," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 1404–1408.

[78] J. Yang, A. Kannan, D. Batra, and D. Parikh, "LR-GAN: layered recursive generative adversarial networks for image generation," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=HJ1kmv9xx

[79] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 7354–7363. [Online]. Available: https://proceedings.mlr.press/v97/zhang19d.html

[80] K.-H. Liu, C.-C. Lin, and T.-J. Liu, "Image generation by residual block based generative adversarial networks," in *2022 IEEE International Conference on Consumer Electronics (ICCE)*, 2022, pp. 1–4.

[81] X. Huang, Y. Li, O. Poursaeed, J. E. Hopcroft, and S. J. Belongie, "Stacked generative adversarial networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 1866–1875. [Online]. Available: https://doi.org/10.1109/CVPR.2017.202

[82] I. P. Durugkar, I. Gemp, and S. Mahadevan, "Generative multi-adversarial networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=Byk-VI9eg

[83] M. Liu, Q. Li, Z. Qin, G. Zhang, P. Wan, and W. Zheng, "Blendgan: Implicitly gan blending for arbitrary stylized face generation," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 29 710–29 722. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/f8417d04a0a2d5e1fb5c5253a365643c-Paper.pdf

[84] Z. Yi, Z. Chen, H. Cai, W. Mao, M. Gong, and H. Zhang, "Bsd-gan: Branched generative adversarial network for scale-disentangled representation learning and image synthesis," *IEEE Transactions on Image Processing*, vol. 29, pp. 9073–9083, 2020.

[85] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 5795–5805.

[86] Y. Men, Y. Mao, Y. Jiang, W.-Y. Ma, and Z. Lian, "Controllable person image synthesis with attribute-decomposed gan," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 5083–5092.

[87] Q. Jin, X. Luo, Y. Shi, and K. Kita, "Image generation method based on improved condition gan," in *2019 6th International Conference on Systems and Informatics (ICSAI)*, 2019, pp. 1290–1294.

[88] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 214–223. [Online]. Available: https://proceedings.mlr.press/v70/arjovsky17a.html

[89] K. S. Lee, N.-T. Tran, and N.-M. Cheung, "Infomax-gan: Improved adversarial image generation via information maximization and contrastive learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2021, pp. 3942–3952.

[90] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[91] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori, "Lifelong gan: Continual learning for conditional image generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[92] H. Dong, X. Liang, K. Gong, H. Lai, J. Zhu, and J. Yin, "Soft-gated warping-gan for pose-guided person image synthesis," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/1700002963a49da13542e0726b7bb758-Paper.pdf

[93] Y. Hong, L. Niu, J. Zhang, W. Zhao, C. Fu, and L. Zhang, "F2gan: Fusing-and-filling gan for few-shot image generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2535–2543. [Online]. Available: https://doi.org/10.1145/3394171.3413561

[94] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[95] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 105–114. [Online]. Available: https://doi.org/10.1109/CVPR.2017.19

[96] M. Wang, X. Zhang, K. Shi, X. Zhang, D. Lei, and X. Yang, "Image super-resolution reconstruction algorithm based on improved gan," in *Proceedings of the 3rd International Conference on Data Science and Information Technology*, ser. DSIT 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 59–64. [Online]. Available: https://doi-org/10.1145/3414274.3414487

[97] C. Wang, H. Zheng, Z. Yu, Z. Zheng, Z. Gu, and B. Zheng, "Discriminative region proposal adversarial networks for high-quality image-to-image translation," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11205. Springer, 2018, pp. 796–812. [Online]. Available: https://doi.org/10.1007/978-3-030-01246-5_47

[98] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "Cvae-gan: Fine-grained image generation through asymmetric training," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[99] M. Gorijala and A. Dukkipati, "Image generation and editing with variational info generative adversarialnetworks," *CoRR*, vol. abs/1701.04568, 2017. [Online]. Available: http://arxiv.org/abs/1701.04568

[100] Y. Zhao, B. Deng, J. Huang, H. Lu, and X.-S. Hua, "Stylized adversarial autoencoder for image generation," in *Proceedings of the 25th ACM International Conference on Multimedia*, ser. MM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 244–251. [Online]. Available: https://doi-org/10.1145/3123266.3123450

[101] B. Zhang, S. Gu, B. Zhang, J. Bao, D. Chen, F. Wen, Y. Wang, and B. Guo, "Styleswin: Transformer-based gan for high-resolution image generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 11 304–11 314.

[102] C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama, "Gan-based synthetic brain mr image generation," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 734–738.

[103] S. Niu, B. Li, X. Wang, and H. Lin, "Defect image sample generation with gan for improving defect recognition," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1611–1622, 2020.

[104] A. Bougaham, V. Delchevalerie, M. E. Adoui, and B. Frénay, "Industrial and medical anomaly detection through cycle-consistent adversarial networks," *CoRR*, vol. abs/2302.05154, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.05154

[105] Y. Sagawa and M. Hagiwara, "Face image generation system using attribute information with dcgans," in *Proceedings of the 2nd International Conference on Machine Learning and Soft Computing*, ser. ICMLSC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 109–113. [Online]. Available: https://doi.org/10.1145/3184066.3184071

[106] B. H. Zeno, I. A. Kalinovskiy, and Y. N. Matveev, "Identity preserving face synthesis using generative adversarial networks," in *Proceedings of the 5th International Conference on Engineering and MIS*, ser. ICEMIS '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3330431.3330435

[107] S. Minaee and A. Abdolrashidi, "Finger-gan: Generating realistic fingerprint images using connectivity imposed GAN," *CoRR*, vol. abs/1812.10482, 2018. [Online]. Available: http://arxiv.org/abs/1812.10482

[108] S. K. Mustikovela, S. De Mello, A. Prakash, U. Iqbal, S. Liu, T. Nguyen-Phuoc, C. Rother, and J. Kautz, "Self-supervised object detection via generative image synthesis," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 8589–8598.

[109] Z. Zhao, Z. Zhang, T. Chen, S. Singh, and H. Zhang, "Image augmentations for GAN training," *CoRR*, vol. abs/2006.02595, 2020. [Online]. Available: https://arxiv.org/abs/2006.02595

[110] M.-Y. Liu, X. Huang, J. Yu, T.-C. Wang, and A. Mallya, "Generative adversarial networks for image and video synthesis: Algorithms and applications," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 839–862, 2021.

[111] P. Shamsolmoali, M. Zareapoor, E. Granger, H. Zhou, R. Wang, M. E. Celebi, and J. Yang, "Image synthesis with adversarial networks: A comprehensive survey and case studies," *Inf. Fusion*, vol. 72, pp. 126–146, 2021. [Online]. Available: https://doi.org/10.1016/j.inffus.2021.02.014

[112] M. Tahmid, S. Alam, and M. k. Akram, "Comparative analysis of generative adversarial networks and their variants," in *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, 2020, pp. 1–6.

[113] H. Shibata, S. Hanaoka, Y. Cao, M. Yoshikawa, T. Takenaga, Y. Nomura, N. Hayashi, and O. Abe, "Local differential privacy image generation using flow-based deep generative models," *CoRR*, vol. abs/2212.10688, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.10688

[114] R. Liu, Y. Liu, X. Gong, X. Wang, and H. Li, "Conditional adversarial generative flow for controllable image synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[115] X. Ma, X. Kong, S. Zhang, and E. Hovy, "Macow: Masked convolutional generative flow," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/20c86a628232a67e7bd46f76fba7ce12-Paper.pdf

[116] A. Mukherjee, B. N. Patro, S. Sidheekh, M. Singh, and V. P. Namboodiri, "Attentive contractive flow: Improved contractive flows with lipschitz-constrained self-attention," *CoRR*, vol. abs/2109.12135, 2021. [Online]. Available: https://arxiv.org/abs/2109.12135

[117] A. Bhattacharyya, S. Mahajan, M. Fritz, B. Schiele, and S. Roth, "Normalizing flows with multi-scale autoregressive priors," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8412–8421.

[118] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, "Flow++: Improving flow-based generative models with variational dequantization and architecture design," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2722–2730. [Online]. Available: https://proceedings.mlr.press/v97/ho19a.html

[119] Z. Chen, Y. Luo, S. Wang, J. Li, and Z. Huang, "Gsmflow: Generation shifts mitigating flow for generalized zero-shot learning," *CoRR*, vol. abs/2207.01798, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2207.01798

[120] M. Hajij, G. Zamzmi, R. Paul, and L. Thukar, "Normalizing flow for synthetic medical images generation," in *2022 IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT)*, 2022, pp. 46–49.

[121] P. Pope, Y. Balaji, and S. Feizi, "Adversarial robustness of flow-based generative models," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 3795–3805. [Online]. Available: https://proceedings.mlr.press/v108/pope20a.html

[122] G. G. P. F. Pires and M. A. T. Figueiredo, "Variational mixture of normalizing flows," in *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020*, 2020, pp. 205–210. [Online]. Available: https://www.esann.org/sites/default/files/proceedings/2020/ES2020-188.pdf

[123] M. Tailanián, Á. Pardo, and P. Musé, "U-flow: A u-shaped normalizing flow for anomaly detection with unsupervised threshold," *CoRR*, vol. abs/2211.12353, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.12353

[124] S. Dong, G. Hangel, E. Z. Chen, S. Sun, W. Bogner, G. Widhalm, C. You, J. A. Onofrey, R. de Graaf, and J. S. Duncan, "Flow-based visual quality enhancer for super-resolution magnetic resonance spectroscopic imaging," in *Deep Generative Models*, A. Mukhopadhyay, I. Oksuz, S. Engelhardt, D. Zhu, and Y. Yuan, Eds. Cham: Springer Nature Switzerland, 2022, pp. 3–13.

[125] X. Han, X. Hu, W. Huang, and M. R. Scott, "Clothflow: A flow-based model for clothed person generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[126] T. Wang, X. Gu, and J. Zhu, "A flow-based generative network for photo-realistic virtual try-on," *IEEE Access*, vol. 10, pp. 40 899–40 909, 2022.

[127] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html

[128] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2813–2821. [Online]. Available: https://doi.org/10.1109/ICCV.2017.304

[129] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 8107–8116. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Karras_Analyzing_and_Improving_the_Image_Quality_of_StyleGAN_CVPR_2020_paper.html

[130] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 852–863. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/076ccd93ad68be51f23707988e934906-Abstract.html

[131] J. Prost, A. Houdard, N. Papadakis, and A. Almansa, "Diverse super-resolution with pretrained deep hiererarchical vaes," *CoRR*, vol. abs/2205.10347, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.10347

[132] J. Wang, W. Zhou, G.-J. Qi, Z. Fu, Q. Tian, and H. Li, "Transformation gan for unsupervised image synthesis and representation learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 469–478.

[133] H. Sun, R. Mehta, H. H. Zhou, Z. Huang, S. C. Johnson, V. Prabhakaran, and V. Singh, "Dual-glow: Conditional flow-based generative model for modality transfer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

# A Appendix 1: Extension on GAN and Flow-based Models

## A.1 Generative Adversarial Networks

In this Appendix sub-section alternate GAN loss functions and variant approaches are presented.

### A.1.1 Losses

**Non-Saturating GAN Loss**
In the GAN introduction paper [9] an adjusted generator loss is suggested for mitigating the gradient saturation problem. The gradient saturation problem appears when the gradient values are very small, ending up in very slow steps on the gradient and thus slow learning. The adjusted non-saturating GAN loss (Equation 20) provides much stronger gradients early in training, ending up in better training and faster convergence in practice. Instead of the generator minimising the log inverse discriminator loss on generated images, the non-saturating loss maximises the log discriminator loss on generated images.

$$L(G) = \mathbb{E}_{z \sim p_z(z)}[\log(D(G(z))] \tag{20}$$

**Least-Squares GAN Loss**
Mao et al. [128] proposed the Least-Squares GAN to mitigate limitations with binary cross-entropy losses leading to the problem of vanishing gradients. The Least-Squares GAN proposes different loss functions for the discriminator (Equation 21) and the generator (Equation 22), where $a$, $b$, and $c$ are hyperparameters. In the introduction paper, the set parameters of [$a = -1$, $b = 1$, $c = 0$] and [$a = 0$, $b = 1$, $c = 1$] are recommended.

$$L(D) = \frac{1}{2}\mathbb{E}_{x \sim p_{data}(x)}[(D(x) - b)^2] + \frac{1}{2}\mathbb{E}_{z \sim p_z(z)}[(D(G(z)) - a)^2] \tag{21}$$

$$L(G) = \frac{1}{2}\mathbb{E}_{z \sim p_z(z)}[(D(G(z)) - c)^2] \tag{22}$$

**Wasserstein GAN Loss**
Arjovsky et al. [88] proposed the Wasserstein GAN, a variant to ensure more stable training, better generation quality and mitigating mode collapse. It is a loss function (Equation 23) that has proven to be very good, commonly used and improve on the base model. The difference with the Wasserstein loss is that the objective of training is to minimise the distance between the real and generated distributions, instead of the regular GAN objective.

$$L(\mu_G, D) = \mathbb{E}_{x \sim \mu_G}[D(x)] - \mathbb{E}_{x \sim \mu_{ref}}[D(x)] \tag{23}$$

### A.1.2 Variants

**Conditional GAN**
The conditional GAN [14], or also named cGAN, is a GAN variant that extends the base architecture with the capability of conditional properties and thus an extra input. The base GAN architecture does not have any conditional generation, while this variant does. In the model, both the generator and the discriminator get extended with an extra conditional input $y$ on top of the data input $x$. The conditional

GAN has proven to work very well for conditional image generation, where conditional properties such as classes or descriptions can be given to the generator.

**StyleGAN**
StyleGAN [15] is a GAN variant proposed by researchers at Nvidia, where the GAN has an improved generator architecture based on the concept of style transfer. The generator network starts with rather small images and gradually increases the size throughout the style blocks in the architecture. The StyleGAN approaches leads to high-resolution and realistic synthesis, better separation of features, and scale-specific control of the synthesis. There have also been updated approaches on the variant, with the StyleGAN-2 [129] improving the original StyleGAN architecture and leading to better synthesis results and faster training. And the StyleGAN-3 [130] variant solving the texture sticking problem occurring in the previous variant by introducing strict low-pass filters.

**CycleGAN**
CycleGAN [16] is a GAN variant that approaches the problem of paired data collection, by having an approach that can generate between two independent domains. It does this by training two GAN's, one for each domain, so two generators and two discriminators, and the introduction of a cycle-consistency loss that uses the two generators. The CycleGAN has proven to work well on applications of image synthesis, enhancement, and style transfer.

## A.2 Flow-based models

In this Appendix sub-section, some of the flow-based model variant models are presented.

### A.2.1 Variants

**NICE**
NICE, short for Non-linear Independent Component Estimation, was one of the first flow-based models, introduced by Dinh et al. [10]. The NICE framework is a flow-based model within the type of coupling flows. It uses a special type of coupling flow, called the affine coupling flow. The affine coupling layer used in NICE is a function that takes a subset of the input and applies an affine transformation to it. While leaving the remaining subset of the input as is, without any applied transformation. By transforming one subset and leaving the other subset as is, the layer becomes bijective and can there be propagated backwards with a neural network. By sequencing the bijective affine transformations, a more complex distribution can be approximated.

**RealNVP**
RealNVP, short for Real-valued Non-Volume Preserving, introduced by Dinh et al. [17] is a flow-based model continuing on the NICE flow model [10]. RealNVP introduces a scaling parameter on the affine coupling layer that NICE presents. This scaling parameter is learned during the training of the model. The scaling parameter being added to the additive coupling layer of NICE, improves the density estimation. It is also this parameter that makes RealNVP non-volume preserving.

**GLOW**

GLOW, introduced by D. P. Kingma and P. Dhariwal [18], is another flow-based generative model, built further on RealNVP [17]. The GLOW model implements a different architecture consisting out of three components. The first component of the GLOW model is the actnorm layer, a scale and bias layer, that acts similar as batch normalization. The second component is the invertible $1 \times 1$ convolution, which is the main novel component introduced, that also significantly improves the learning. The third and final component is an affine coupling layer. Using this architecture, the GLOW model has improved all of the state of the art benchmarks achieved by RealNVP on various data sets.

# B Appendix 2: Extension on VAE reconstructions

In this Appendix section more training reconstructions per final trained model per data set are given. For every model, 15 reconstructions are given.

## B.1 1K data set Reconstructions

### B.1.1 256D Vanilla VAE

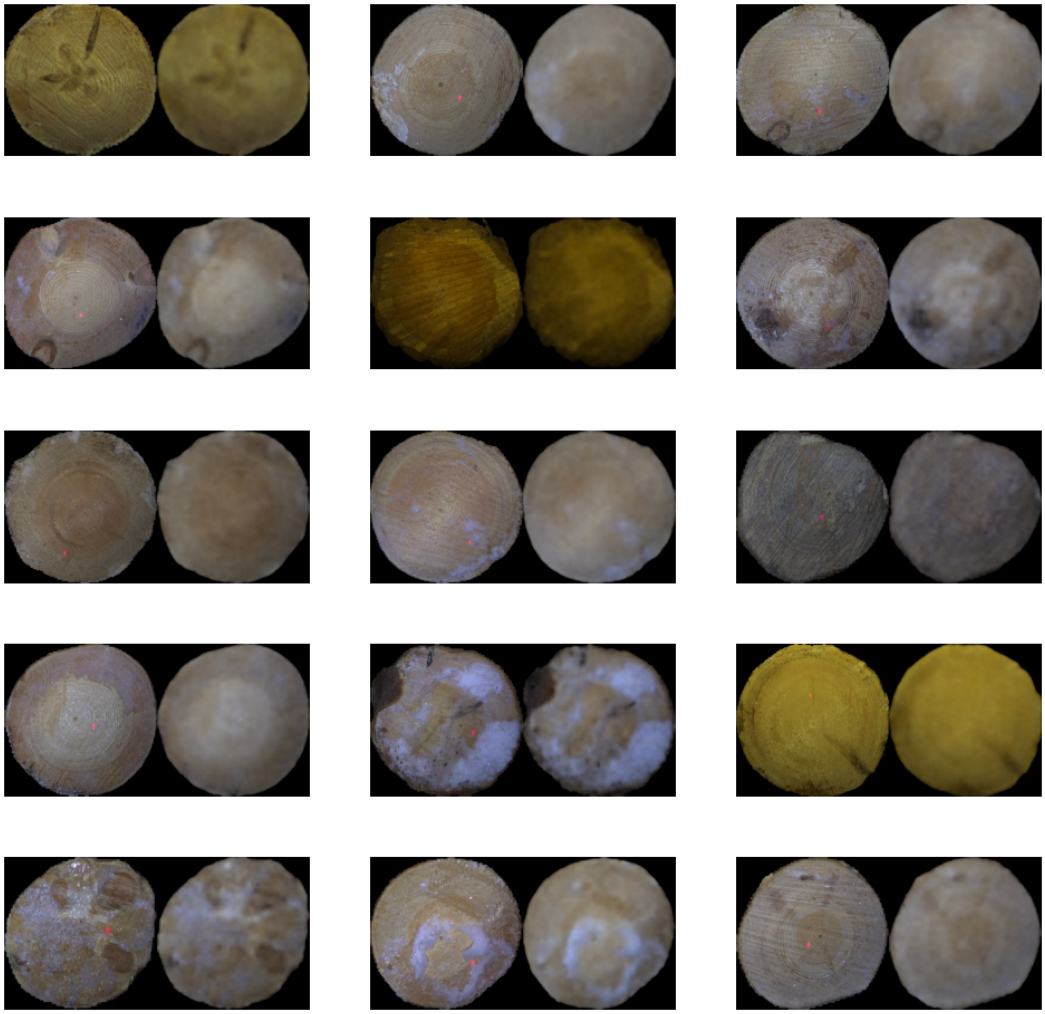

Figure 2.37: Training reconstructions from vanilla model with 256D latent dimensions using the 1K data set.

### B.1.2 512D Vanilla VAE

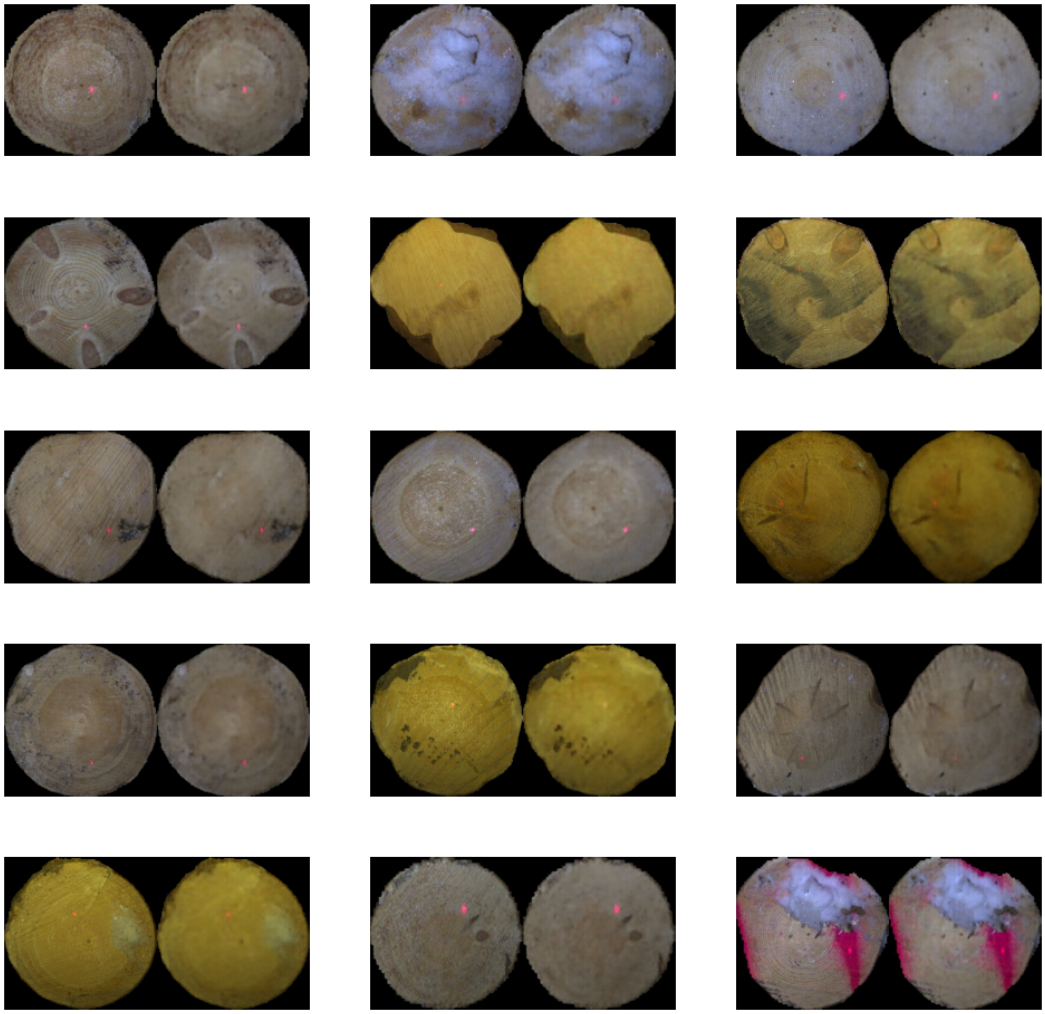

Figure 2.38: Training reconstructions from vanilla model with 512 latent dimensions using the 1K data set.

### B.1.3 512D Variant VAE

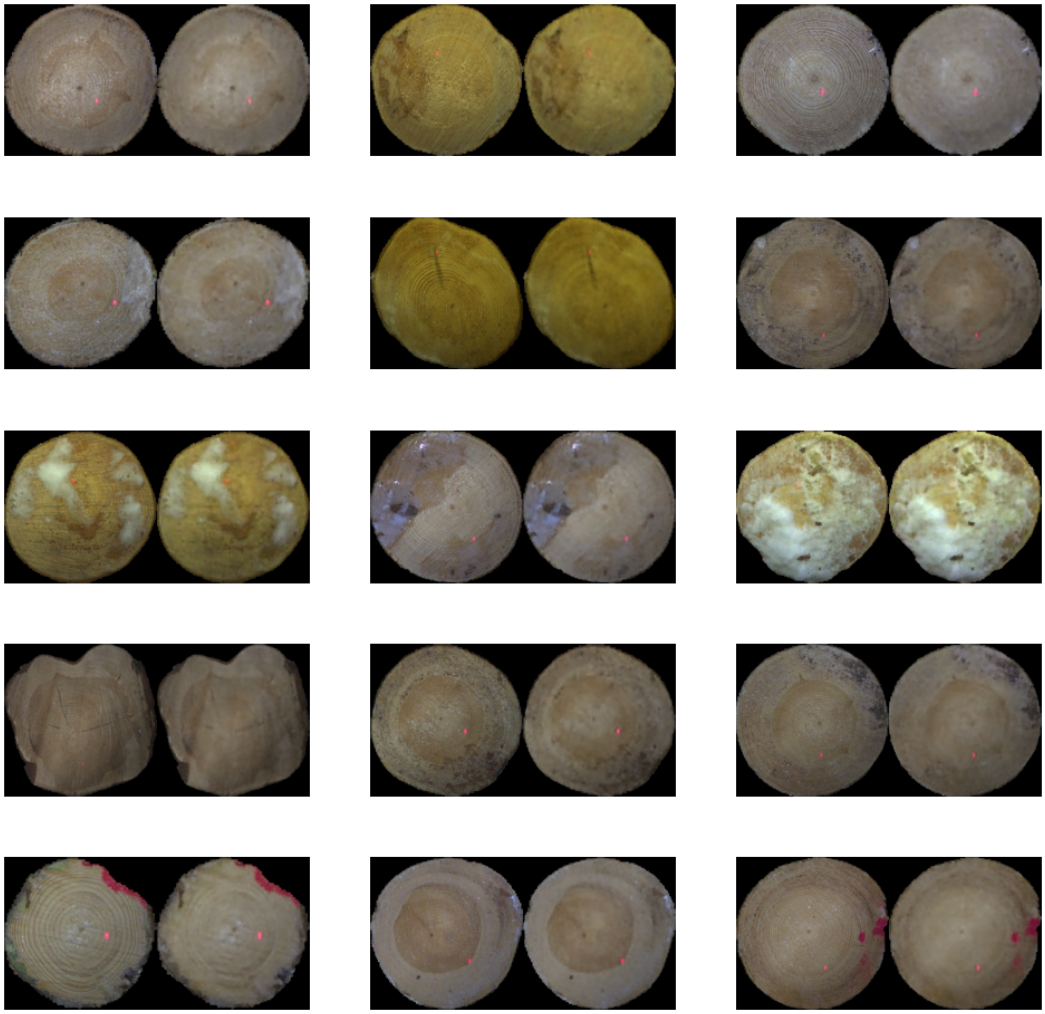

Figure 2.39: Training reconstructions from variant model with 512 latent dimensions using the 1K data set.

## B.2 100K data set Reconstructions

### B.2.1 1024D Vanilla VAE

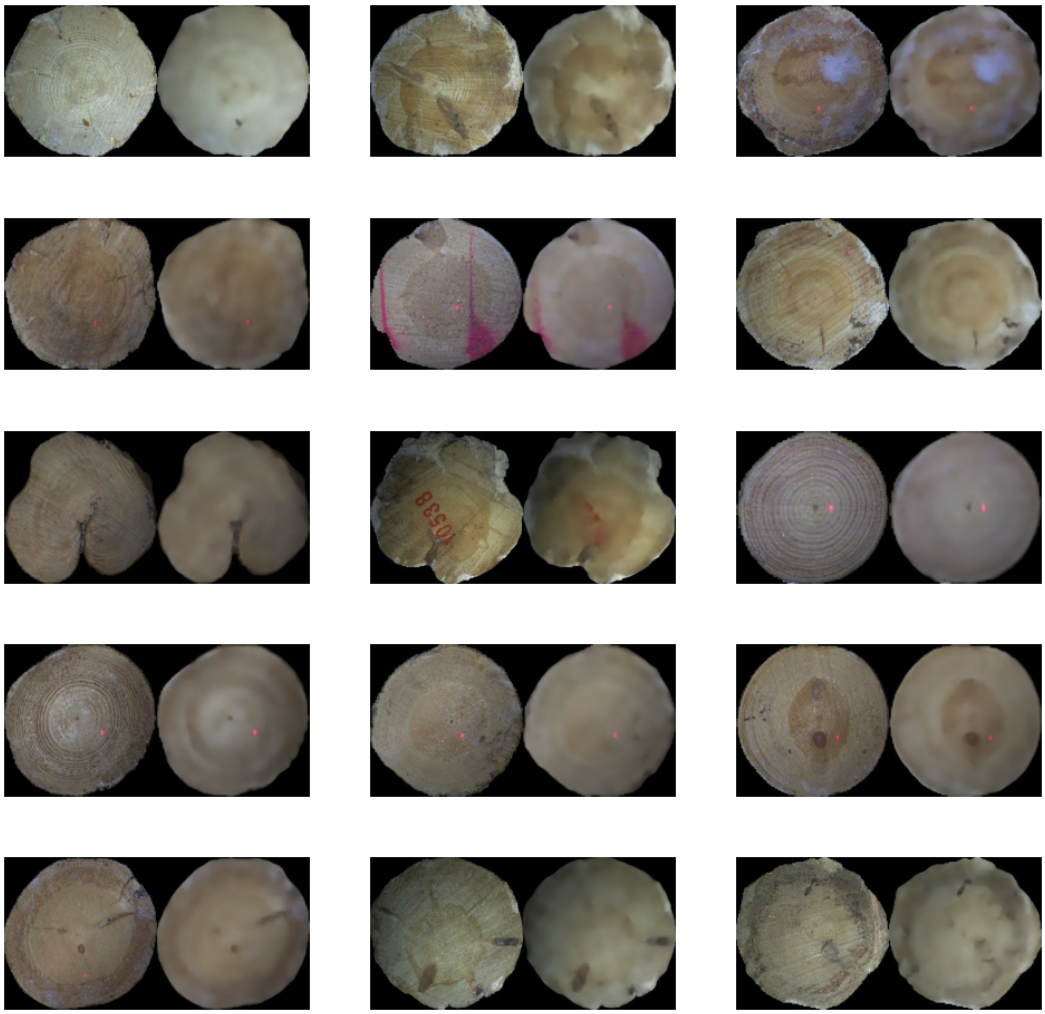

Figure 2.40: Training reconstructions from vanilla model with 1024 latent dimensions using the 100K data set.

### B.2.2 1024D Variant VAE

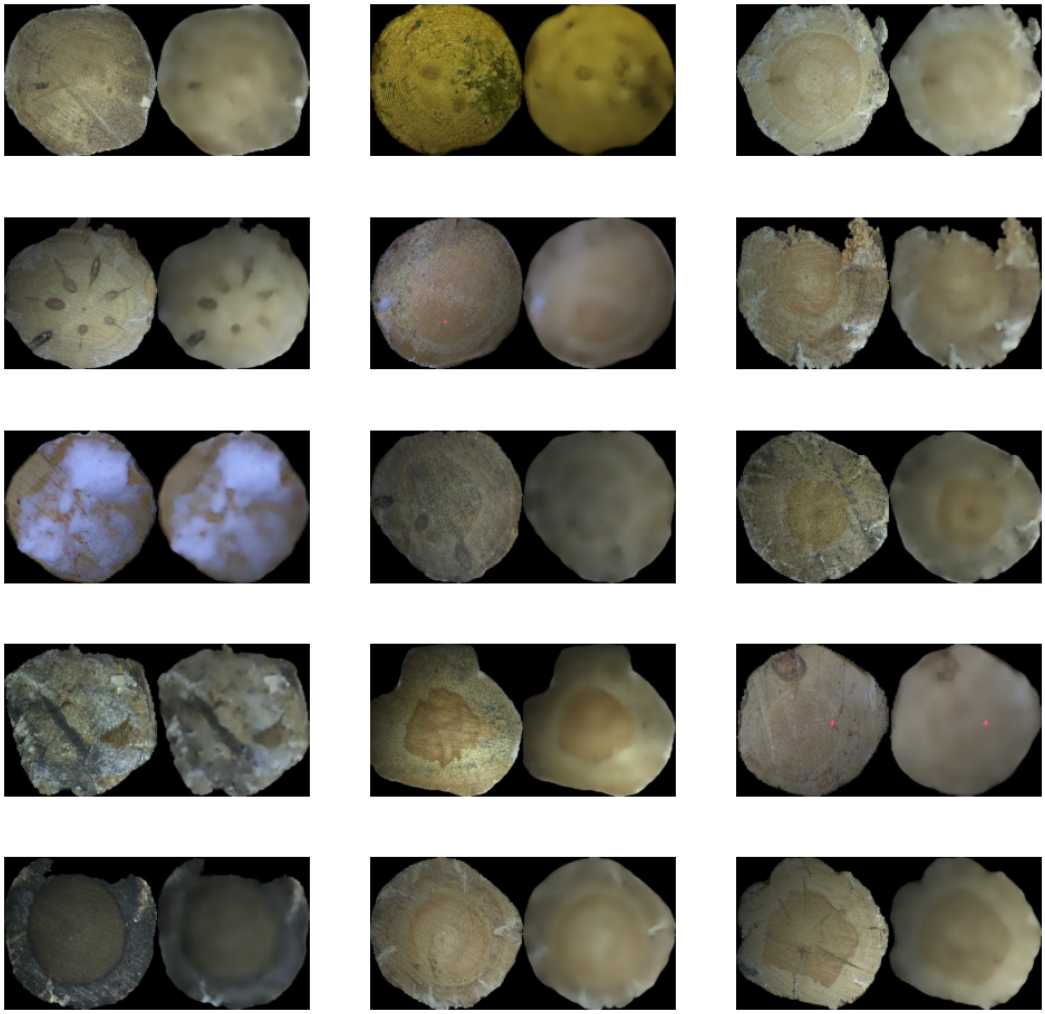

Figure 2.41: Training reconstructions from variant model with 1024 latent dimensions using the 100K data set.

# C Appendix 3: Extension on VAE sampled triplets

In this Appendix section more sampled triplets generated from the vanilla and variant VAE with 1024 latent dimensions using the 100K data set are given.

## C.1 Vanilla VAE

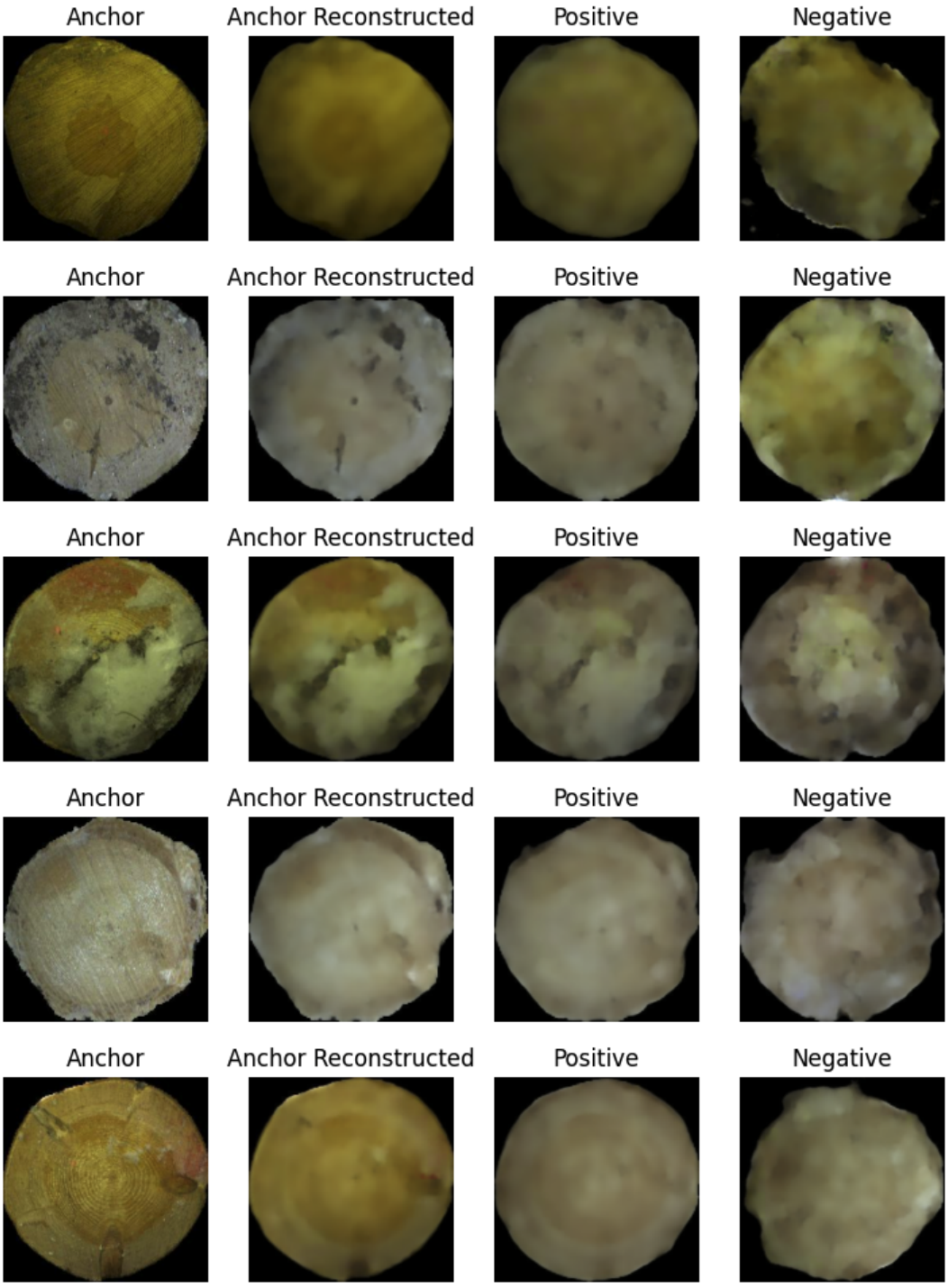


Figure 3.42: Triplets sampled with selected optimal parameters from vanilla model with 1024 latent dimensions using the 100K data set.

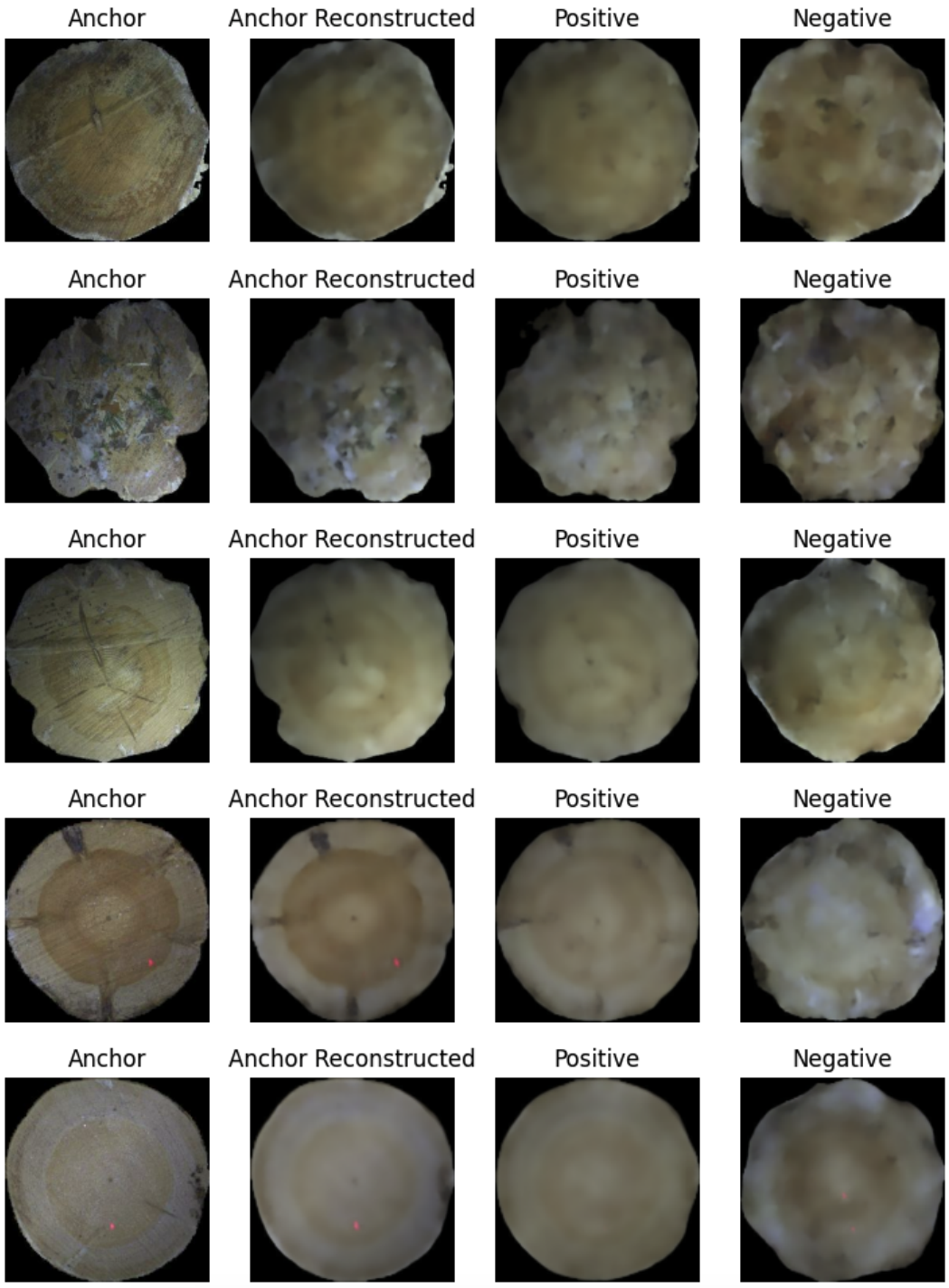


Figure 3.43: Triplets sampled with selected optimal parameters from vanilla model with 1024 latent dimensions using the 100K data set.

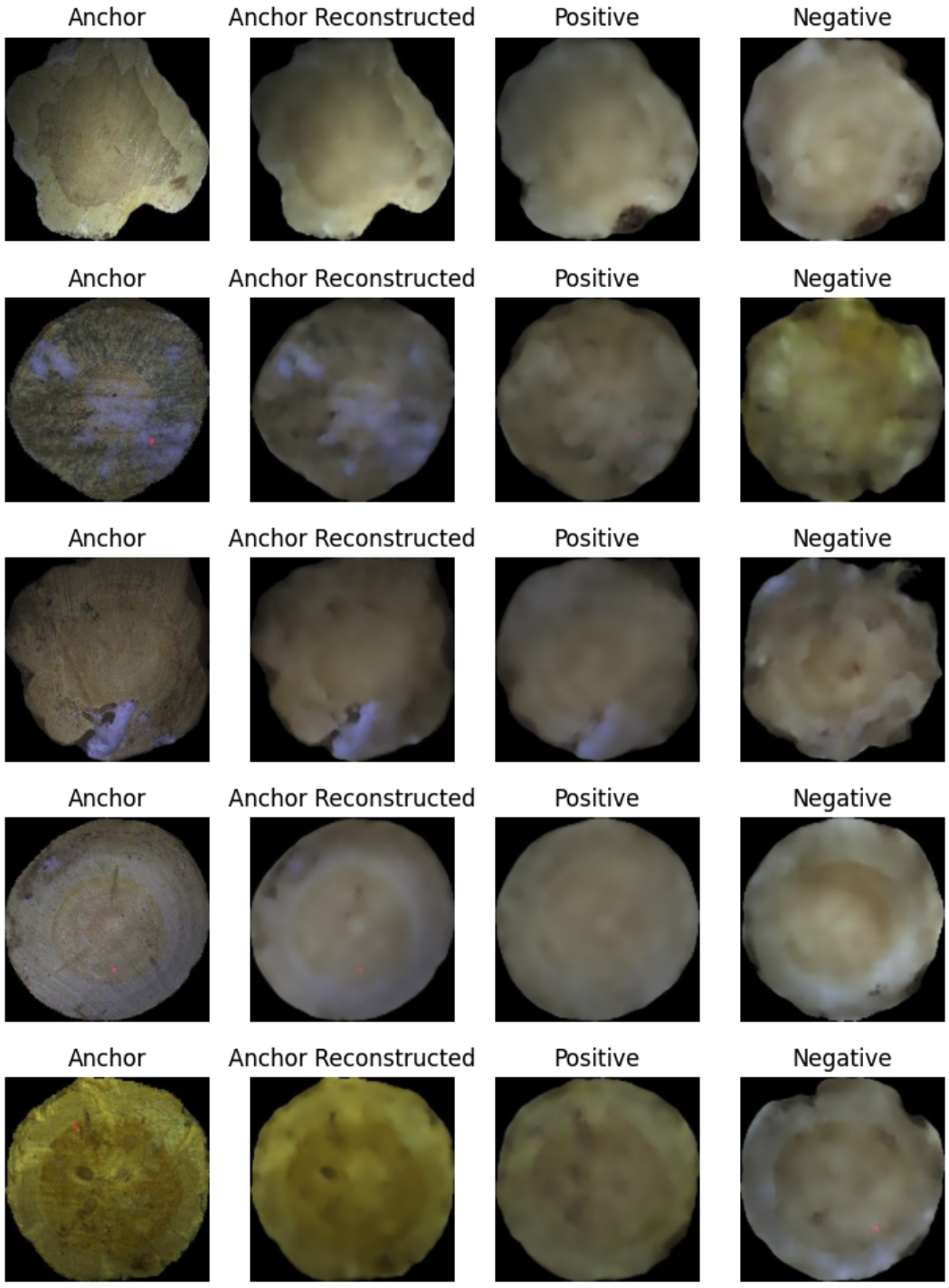


Figure 3.44: Triplets sampled with selected optimal parameters from vanilla model with 1024 latent dimensions using the 100K data set.

## C.2 Variant VAE

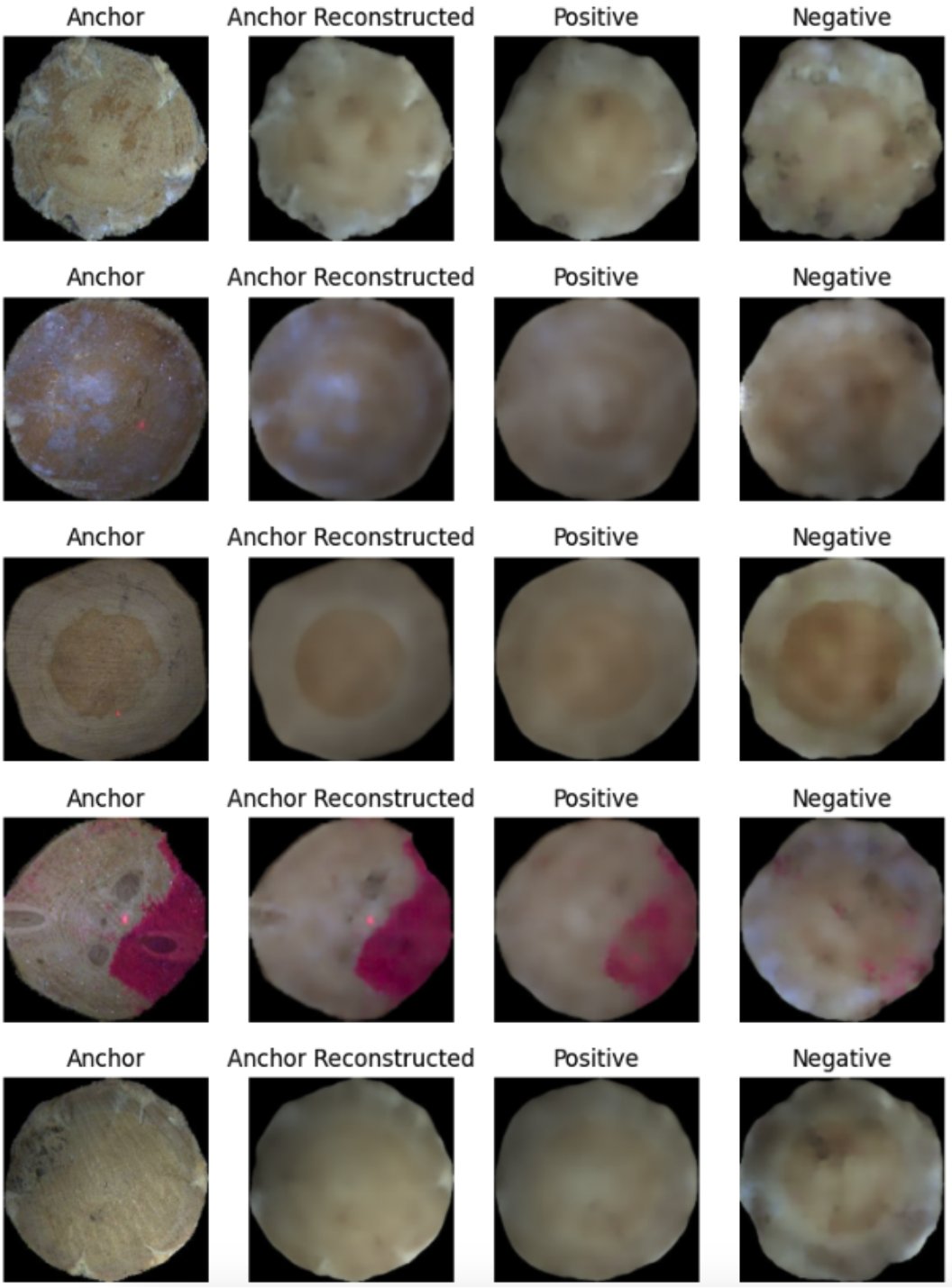


Figure 3.45: Triplets sampled with selected optimal parameters from variant model with 1024 latent dimensions using the 100K data set.

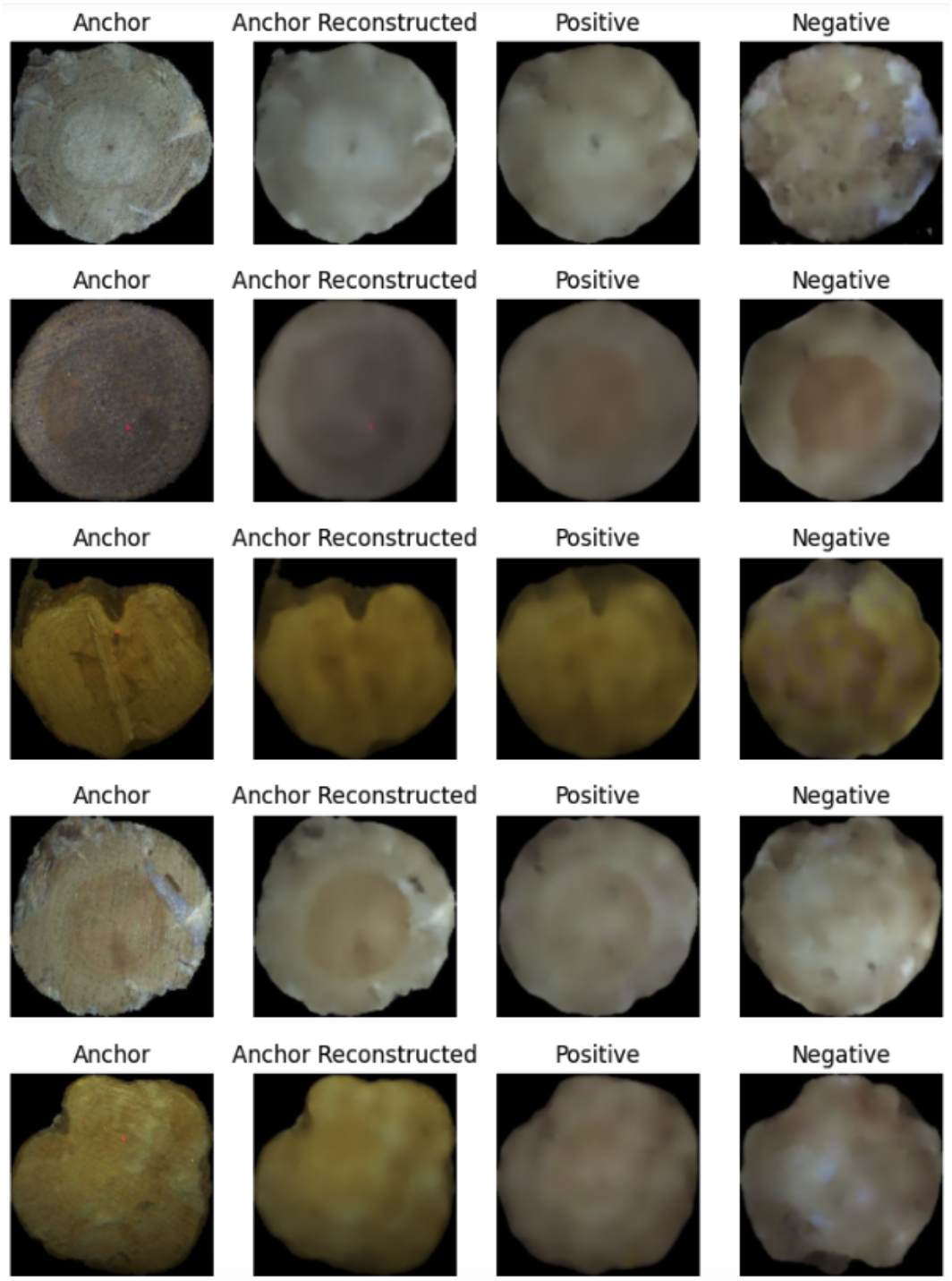


Figure 3.46: Triplets sampled with selected optimal parameters from variant model with 1024 latent dimensions using the 100K data set.

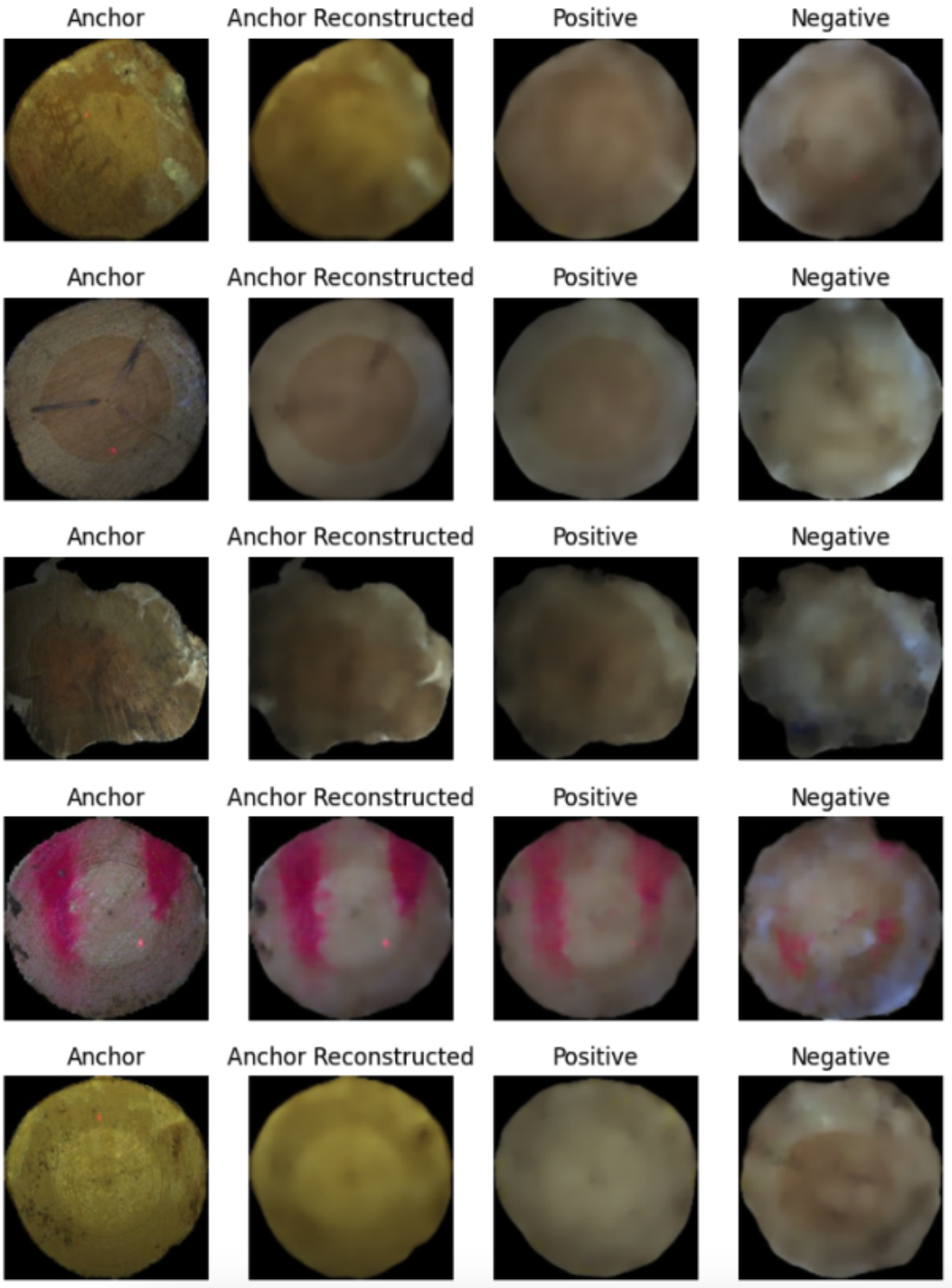


Figure 3.47: Triplets sampled with selected optimal parameters from variant model with 1024 latent dimensions using the 100K data set.

# D Appendix 4: Source code for the models

The source code for the models and algorithms have been published on GitHub: `https://github.com/TiboBruneel/Thesis-VAE-Models`.

The vanilla VAE published on: `https://github.com/TiboBruneel/Thesis-VAE-Models/blob/main/VanillaVAE.py`.

The variant VAE published on: `https://github.com/TiboBruneel/Thesis-VAE-Models/blob/main/VariantVAE.py`.

The triplet VAE sampling algorithm published on: `https://github.com/TiboBruneel/Thesis-VAE-Models/blob/main/TripletVAESampling.py`.

The recognition model published on: `https://github.com/TiboBruneel/Thesis-VAE-Models/blob/main/RecognitionModel.py`.

## E Appendix 5: Literature Review Data Extraction

| VAE Data Extraction Studies | | | | |
|---|---|---|---|---|
| **Ref** | **Source and date extraction** | **Search term** | **Type** | **data sets** |
| [52] | Google Scholar, 6 Feb. 2023 | "Assessment of variational autoencoders image generation" | VAE | 1 (Jet data set) |
| [61] | Google Scholar, 6 Feb. 2023 | "Applications of VAE Image generation" | VAE, GAN | 1 (Probe Data) |
| [62] | IEEE Explore, 8 Feb. 2023 | "VAE Image generation" | VAE, GAN | 3 (Street View House Numbers, CIFAR-10, Chest X-Ray) |
| [66] | arXiv, 7 Feb. 2023 | "VAE Image generation" | VAE, GAN | 1 (dSprite) |
| [65] | NeurIPS, 7 Feb. 2023 | "VAE" | VAE | 2 (DSprites, Shapes3D) |
| [69] | ICCV, 7 Feb. 2023 | "VAE" | VAE | 4 (Multi-MNIST, Multi-Fashion, Multi-COIL-10, Multi-COIL-20) |
| [68] | Google Scholar, 6 Feb. 2023 | "Applications of VAE Image generation" | VAE | 3 (MNIST, Galaxy Zoon, EM data sets) |
| [64] | arXiv, 6 Feb. 2023 | "VAE Variant" | VAE | 2 (DSprites, MNIST) |
| [31] | arXiv, 6 Feb. 2023 | "VAE Generation" | VAE | 3 (Spatial blobs, DSprites, 3DChairs) |
| [55] | Google Scholar, 6 Feb. 2023 | "Assessment of variational autoencoders image generation" | VAE | 1 (MNIST) |
| [53] | IEEE Explore, 8 Feb. 2023 | "VAE Image generation" | VAE | 1 (AR data set) |
| [46] | ACM Digital Library, 7 Feb. 2023 | "Assessment of variational autoencoders image generation" | VAE | 2 (CelebA, MNIST) |
| [49] | arXiv, 7 Feb. 2023 | "VAE Image generation" | VAE | 1 (CelebA) |
| [59] | arXiv, 7 Feb. 2023 | "VAE Image generation" | VAE | 3 (CelebA-HQ-128, CIFAR-10, CelebA-64) |
| [54] | arXiv, 7 Feb. 2023 | "VAE Variant" | VAE | 1 (MNIST) |
| [50] | NeurIPS, 7 Feb. 2023 | "VAE" | VAE | 4 (MNIST, Fashion-MNIST, NotMNIST, CelebA) |

| VAE Data Extraction Studies | | | | |
|---|---|---|---|---|
| **Ref** | **Source and date extraction** | **Search term** | **Type** | **data sets** |
| [67] | NeurIPS, 7 Feb. 2023 | "VAE" | VAE | 1 (MNIST) |
| [51] | ICCV, 7 Feb. 2023 | "VAE" | VAE | 7 (Describable Textures data set, CelebA, CelebA-HQ, CMP Facades, shoes, Cityscapes, ADE20K) |
| [60] | CVPR, 7 Feb. 2023 | "VAE" | VAE | 2 (LSUN, FF-HQ) |
| [43] | arXiv, 7 Feb. 2023 | "VAE Image generation" | VAE | 3 (CIFAR-10, ImageNet-64, FFHQ-256) |
| [48] | NeurIPS, 7 Feb. 2023 | "VAE" | VAE | 4 (MNIST, Fashion-MNIST, CIFAR-10, CelebA) |
| [47] | NeurIPS, 7 Feb. 2023 | "VAE" | VAE | 4 ( MNIST, Fashion-MNIST, Omniglot, CelebA) |
| [131] | arXiv, 7 Feb. 2023 | "VAE Image generation" | VAE | 1 (FFHQ-256) |
| [56] | arXiv, 7 Feb. 2023 | "Variational autoencoder image generation" | VAE | 3 (Celeb-A, CelebA-HQ, LSUN BED-ROOM) |
| [57] | Google Scholar, 7 Feb. 2023 | "Assessment of variational autoencoders image generation" | VAE | 3 (CIFAR-10, CelebA-HQ, FFQQ) |
| [58] | arXiv, 7 Feb. 2023 | "VAE Image generation" | VAE | 4 (MNIST, CIFAR-10, CelebA-128, CelebA-256). |
| [12] | Google Scholar, 6 Feb. 2023 | "Applications of VAE Image generation" | VAE | 2 (ImageNet, FFHQ) |
| [44] | arXiv, 7 Feb. 2023 | "VAE Image generation" | VAE | 2 (FaceMask, RoomCrop) |
| [63] | arXiv, 7 Feb. 2023 | "VAE Variant" | VAE | 2 (2D Shapes, CelebA) |
| [70] | arXiv, 7 Feb. 2023 | "VAE Image generation" | VAE | 1 (MNIST) |

| GAN Data Extraction Studies | | | | |
|---|---|---|---|---|
| **Ref** | **Source and date extraction** | **Search term** | **Type** | **data sets** |
| [98] | Google Scholar, 13 Feb. 2023 | "GAN Image Generation" | GAN, VAE | 3 (FaceScrub, 102 Category Flower, CUB-200) |
| [102] | Google Scholar, 13 Feb. 2023 | "GAN Image Generation" | GAN | 1 (BRATS 2016) |
| [75] | Google Scholar, 13 Feb. 2023 | "GAN Image Generation" | GAN | 3 (Large-scale CelebFaces Attributes (CelebA), Caltech-UCSD Birds-200-2011, Stanford's Cars) |
| [101] | Google Scholar, 13 Feb. 2023 | "GAN Image Generation" | GAN, Trans-former | 3 (CelebA-HQ, LSUN Church, FFHQ) |
| [91] | Google Scholar, 15 Feb. 2023 | "GAN Image Generation" | GAN | Inclusion |
| [78] | Google Scholar, 15 Feb. 2023 | "GAN Image Generation" | GAN | 3 (MNIST, CIFAR-10, CUB-200) |
| [93] | Google Scholar, 15 Feb. 2023 | "GAN Image Generation" | GAN | 5 (Omniglot, EMNIST, VGGFace, Flow-ers, Animal Faces) |
| [89] | Google Scholar, 15 Feb. 2023 | "GAN Image Generation" | GAN | 5 (ImageNet, CelebA, CIFAR-10, STL-10, CIFAR-100) |
| [109] | Google Scholar, 15 Feb. 2023 | "GAN Image Generation" | GAN | 1 (CIFAR-10) |
| [87] | Google Scholar, 15 Feb. 2023 | "GAN Image Generation" | GAN | 2 (MNIST, Fashion-MNIST) |
| [103] | Google Scholar, 15 Feb. 2023 | "GAN Image Generation" | GAN | 1 (Commutator cylinder surface defect imagery) |
| [110] | Google Scholar, 15 Feb. 2023 | "Generative Adversarial Networks Applications" | GAN | Inclusion |
| [76] | Google Scholar, 15 Feb. 2023 | "Generative Adversarial Networks" | GAN | 3 (Large-scale Scene Understanding, Imagenet-1k, newly assembled Faces) |
| [15] | Google Scholar, 15 Feb. 2023 | "Generative adversarial networks image generation" | GAN | 1 (FFHQ) |
| [90] | Google Scholar, 15 Feb. 2023 | "Generative adversarial networks image generation" | GAN | 1 (CIFAR-10) |

R

| GAN Data Extraction Studies | | | | |
|---|---|---|---|---|
| **Ref** | **Source and date extraction** | **Search term** | **Type** | **data sets** |
| [79] | Google Scholar, 15 Feb. 2023 | "Generative adversarial networks image generation" | GAN | 1 (LSVRC2012 (ImageNet)) |
| [94] | Google Scholar, 15 Feb. 2023 | GAN | 2 (CelebA, RaFD) | Inclusion |
| [71] | arXiv, 15 Feb. 2023 | "GAN Image Generation" | GAN | 1 (FFHQ) |
| [104] | arXiv, 15 Feb. 2023 | "GAN Image Generation" | GAN | 2 (MCTEC-AD, PCAM, OCT) |
| [99] | arXiv, 15 Feb. 2023 | "GAN Image Generation" | GAN | 3 (Labeled Faces in the Wild (LFW), celebA and a modified version of MNIST data sets) |
| [82] | arXiv, 15 Feb. 2023 | "GAN Image Generation" | GAN | 3 (MNIST, CIFAR-10, CelebA) |
| [81] | arXiv, 15 Feb. 2023 | "GAN Image Generation" | GAN | 3 (MNIST, SVHN, CIFAR-10) |
| [107] | arXiv, 15 Feb. 2023 | "GAN Image Generation" | GAN | 1 (PolyU Fingerprint Databases) |
| [111] | arXiv, 15 Feb. 2023 | "Survey on GAN Image generation" | GAN | Inclusion |
| [95] | arXiv, 15 Feb. 2023 | "Generative adversarial networks image generation" | GAN | 3 (Set5, Set14, BSD100) |
| [105] | ACM Digital Library, 15 Feb. 2023 | "GAN Image Generation" | GAN | 1 (CelebA) |
| [100] | ACM Digital Library, 15 Feb. 2023 | "GAN Image Generation" | GAN | 4 (MNIST, Labeled Faces in the Wild(LFW), IIIT-5K, Plate) |
| [96] | ACM Digital Library, 15 Feb. 2023 | "GAN Image Generation" | GAN | 3 (Set5, Set14, BSD100) |
| [106] | ACM Digital Library, 15 Feb. 2023 | "Generative adversarial networks image generation" | GAN | 1 (CelebA) |
| [112] | IEEE Explore, 15 Feb. 2023 | "GAN Variants" | GAN | 1 (CIFAR-10) |
| [84] | IEEE Explore, 15 Feb. 2023 | "GAN Image Generation" | GAN | 3 (church_outdoor from LSUN, CelebA-HQ, Car) |

| GAN Data Extraction Studies | | | | |
|---|---|---|---|---|
| **Ref** | **Source and date extraction** | **Search term** | **Type** | **data sets** |
| [77] | IEEE Explore, 15 Feb. 2023 | "GAN Image Generation" | GAN | 1 (Cityscapes) |
| [80] | IEEE Explore, 15 Feb. 2023 | "Generative adversarial networks image generation" | GAN | 3 (Anime face, LSUN bedroom, CelebA) |
| [72] | ICCV, 16 Feb. 2023 | "GAN" | GAN | 5 (AFHQ, FFHQ, Plant-Village, Retinal Fundus, CUB-2011) |
| [73] | ICCV, 16 Feb. 2023 | "GAN" | GAN | 6 (FFHQ, CelebA-HQ, Stanford Cars, LSUN Horse, LSUN Church, AFHQ Wild) |
| [108] | ICCV, 16 Feb. 2023 | "GAN" | GAN | 3 (Compcars, KITTI, Cityscapes) |
| [74] | ICCV, 16 Feb. 2023 | "GAN" | GAN | 2 (CelebA-HQ, FFHQ) |
| [85] | CVPR, 16 Feb. 2023 | "GAN" | GAN | 3 (CelebA, Cats, CARLA) |
| [132] | CVPR, 16 Feb. 2023 | "GAN" | GAN | 5 (CIFAR-10, ImageNet, CELEBA-HQ, LSUN-BEDROOM, CIFAR-10) |
| [86] | CVPR, 16 Feb. 2023 | "GAN" | GAN | 1 (In-shop Clothes Retrieval Benchmark DeepFashion) |
| [83] | NeurIPS, 16 Feb. 2023 | "GAN" | GAN | 2 (FFHQ, AAHQ) |
| [92] | NeurIPS, 16 Feb. 2023 | "GAN" | GAN | 2 (DeepFashion, Market-1501) |
| [97] | International Journal of Computer Vision, 16 Feb. 2023 | "GAN" | GAN | 4 (ImageNet,CityScapes, Facades, Aerial data) |

| Flow-based Model Data Extraction Studies | | | | |
|---|---|---|---|---|
| **Ref** | **Source and date extraction** | **Search term** | **Type** | **data sets** |
| [133] | Google Scholar, 1. Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 1 (The Alzheimer's Disease Neuroimaging Initia- tive (ADNI)) |
| [125] | Google Scholar, 1. Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 2 (DeepFashion, VITON) |
| [121] | Google Scholar, 1. Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 3 (CIFAR-10, LSUN Bedroom, CelebA) |
| [118] | Google Scholar, 1. Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 4 (CIFAR10, 32x32 ImageNet, 64x64 ImageNet, 5-bit CelebA) |
| [114] | Google Scholar, 1. Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 2 (MNIST, CelebA) |
| [18] | Google Scholar, 1. Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 4 (CIFAR, LSUN, ImageNet, CelebA) |
| [115] | Google Scholar, 1. Mar. 2023 | "Flow-based Generative Model Image Variant" | Flow-Based Model | 4 (CIFAR-10, ImageNet-64, CelebA-HQ, LSUN) |
| [113] | arXiv, 3 Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 1 (RSNA Pneumonia Detection Challenge data set) |
| [123] | arXiv, 3 Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 1 (MvTec-AD) |
| [124] | arXiv, 3 Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 1 (H-MRSI) |
| [119] | arXiv, 3 Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 4 (Caltech-UCSD Birds-200-2011, Oxford Flowers, Attribute Pascal and Yahoo, Animals with Attributes 2) |
| [116] | arXiv, 3 Mar. 2023 | "Flow-based Generative Model Variant" | Flow-Based Model | 4 (MNIST, CIFAR-10, ImageNet-32, ImageNet-64) |

| Flow-based Model Data Extraction Studies | | | | |
|---|---|---|---|---|
| **Ref** | **Source and date extraction** | **Search term** | **Type** | **data sets** |
| [122] | arXiv, 3 Mar. 2023 | "Flow-based Generative Model Variant" | Flow-Based Model | 3 (PinWheel, Two-Circle, MNIST) |
| [120] | IEEE Explore, 3 Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 2 (Chest x-ray, Skin cancer) |
| [117] | IEEE Explore, 3 Mar. 2023 | "Flow-based Generative Model Image Generation" | Flow-Based Model | 3 (MNIST, CIFAR-10, ImageNet) |
| [126] | IEEE Explore, 3 Mar. 2023 | "Flow-based Generative Model Network" | Flow-Based Model | 1 (VITON) |