DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Implementation and Evaluation of Variational Autoencoders for High Resolution Medical Imaging

Stefan Dünhuber

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Implementation and Evaluation of Variational Autoencoders for High Resolution Medical Imaging

Implementierung und Evaluierung von Variational Autoencodern für hochauflösende medizinische Bildgebung

Author:	Stefan Dünhuber
Supervisor:	Prof. Dr. med. Rainer Burgkart
Advisor:	Nikolas Wilhelm, M.Sc.
Submission Date:	December 13, 2019

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbstständig durchgeführt zu haben und keine weiteren Hilfsmittel und Quellen als die angegebenen genutzt zu haben. Mit ihrer unbefristeten Aufbewahrung und Bereitstellung in der Lehrstuhlbibliothek erkläre ich mich einverstanden.

München, den 13. Dezember 2019 _____ (Stefan Dünhuber)

☐ Ich stelle die Software, die ich im Rahmen dieser Arbeit entwickelt habe, dem Lehrstuhl für Orthopädie und Sportorthopädie unter den Bedingungen der 3-Klausel-BSD-Lizenz zur Verfügung. Bei der Nutzung meiner Software werde Ich als Autor namentlich genannt.

München, den 13. Dezember 2019 _____ (Stefan Dünhuber)

Head of Orthopaedic Research (Prof. Dr. med. Rainer Burgkart)
Technical University of Munich
Einsteinstraße 65
81675 München
Germany

Leiter der Orthopädischen Forschung (Prof. Dr. med. Rainer Burgkart)
Technische Universität München
Einsteinstraße 65
81675 München
Deutschland

Abstract

Over the last few years deep learning models have achieved great success to improve computer aided diagnosis (CAD). Most of the deep learning models are using supervised learning methods, which have the disadvantage of elaborate preprocessing. Therefore, this work is focusing on an unsupervised learning approach, namely, on variational autoencoders (VAEs) proposed by Kingma et al. [34]. VAEs are powerful generative models, which allow analyses and disentanglements of the latent space for a given input. In this work the standard VAE model and four enhanced models are derived, applied and discussed on a high resolution X-ray knee dataset.

Kurzfassung

In den letzten Jahren konnten Deep Learning Modelle große Erfolge in dem Bereich der computergestützten Diagnostik verzeichnen. Die meisten Deep Learning Modelle verwenden dabei Ansätze des überwachten Lernens, welche den Nachteil der aufwendigen Datenvorverarbeitung haben. In dieser Arbeit liegt deshalb der Fokus auf Ansätzen des unüberwachten Lernens, im speziellen dem variational Autoencoder (VAE) Ansatz von Kingma et al. veröffentlicht in [34]. Varational Autoencoder sind mächtige generative Modelle, welche es ermöglichen den latenten Raum für einen bestimmten Eingang zu analysieren und zu entkoppeln. In dieser Arbeit wird der normale Variational Autoencoder und vier erweiterte Modelle anhand eines hochauflösenden Röntgenkniedatensatzes hergeleitet, angewendet und diskutiert.

Contents

Chapter 1

Introduction

In ancient Greece, philosophers have already dreamed of machines that can think. However, it took humanity over 2000 years until the year 1956 in which the research field of artificial intelligence (AI) was established. AI currently encompasses a huge variety of subfields with many different tasks, such as playing chess [49], proving mathematical theorems [3], natural language processing (NLP) [62], autonomous driving [48] or diagnosing diseases [21]. In short, AI is relevant to any intellectual task [46].

One subfield of AI is machine learning (ML), which is defined as a set of methods that can automatically detect patterns in data to perform different kinds of decision making. In recent years the amount of electronic data and computing power have rapidly increased, whereby deep learning approaches become very successful and popular. Deep learning is a research field of ML [41], which is based on artificial neural network architectures. The approach of deep learning provides a solution to problems that are easy for people to perform but hard to describe formally. Simply speaking, problems that humans solve intuitively such as recognizing spoken words or recognizing objects in images [26].

1.1 Motivation

The motivation of this work is to improve medical imaging analysis by enhancing diagnosis and treatments for patients, meanwhile supporting radiologist and other medical doctors. Because of the recent success of deep learning in different areas of CAD like predicting Alzheimer's disease [38], cancer detection [6], pathology detection [1] surgical assistance [42] and the rapidly increasing amount of data in medical imaging, the approach of deep learning is chosen. Furthermore, real word use cases are an important driver of this work. One of these use cases is to discover latent factors of a CT lung scan, which are responsible for the representation of a lesion. These latent variables can be further analyzed to gain knowledge about the growth of the lesion. Another example is to analyze X-Ray images of knees and find latent factors which are indicating the state of osteoarthritis for a patient.

1.2 Problem

Many deep learning approaches learn on labeled data, which is called supervised learning. Supervised learning consist of two process steps. First, the data has to be labeled, which means assigning properties to each data sample manually. Second, the model is trained with the preprocessed data. The first step can be monotonous like assigning different classes to trivial images, time consuming or difficile whereby expert knowledge is required. To avoid those problems unsupervised or semi-supervised approaches, which require no labeling are appropriate. The family of deep generative models provide the ability of unsupervised learning. A promising deep generative model approach are variational autoencoders (VAEs) proposed by Kingma et al. in [34]. VAEs are powerful generative models, which allow analysis and disentanglements of the latent space for a given input.

1.3 Objective

The overall goal is to gain knowledge about the latent space of high resolution medical images and be able to manipulate the latent space for a specific output by the use of VAE models. All carried out in an unsupervised manner. In the first step the functionality of VAEs are described. The strengths and weaknesses of this model will be discussed. Subsequently, approaches will be shown, which are improving the standard VAE model. Finally, different VAE models will be evaluated and discussed on a high resolution medical imaging dataset. In addition, a framework is provided for evaluating different generative models.

Chapter 2

Fundamentals

Deep learning is a subcategory of machine learning. A machine learning algorithm is an algorithm that is able to learn from data via experience. To get a basic understanding of the principles of ML the textbooks [41] and [7] are recommended. In general, a machine learning algorithm contains four parts. The objective, cost or loss function, which is minimized. An optimization algorithm to find the minimum of the loss function. A model, which describes the underlying data and finally a dataset, where the model will be applied on.

Most ML tasks can be divided into the categories of supervised learning or unsupervised learning. Unsupervised learning algorithms discover datasets, which contain many features by learning useful properties of the dataset structure by themselves. Supervised learning algorithms, in contrast, use a dataset with different features and the associated class, label or target for each sample. More mathematically spoken, unsupervised learning involves observing several examples of a random vector $\boldsymbol{x}$ and attempting to implicitly or explicitly learn the probability distribution $p(\boldsymbol{x})$, or some interesting properties of that distribution. Supervised learning instead involves observing several examples of a random vector $\boldsymbol{x}$ and an associated value or vector $\boldsymbol{y}$. Then learning to predict $\boldsymbol{y}$ from $\boldsymbol{x}$, usually by estimating the probability $p(\boldsymbol{y}|\boldsymbol{x})$. Unsupervised and supervised learning can also be mixed together which leads to semi-supervised learning [26].

To illustrate the principals of a machine learning algorithm, the linear regression approach will be introduced in chapter 2.1. From this approach the weaknesses will be uncovered and a solution will be provided, which leads directly to the approach of deep learning.

2.1 Linear Regression

In this work a dataset is defined as $D = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{n}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is an input vector and $\boldsymbol{y}_i \in \mathbb{R}^k$ is the corresponding target. Thereby, n is the number of samples, d the number of features and k the number of classes of one sample i. It can also be written as a data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ for the input and $\boldsymbol{Y} \in \mathbb{R}^{n \times k}$ for the target. In the following $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{y} \in \mathbb{R}^k$ corresponds to one sample taken from D.

The goal of regression is to build a system which can take an input vector $\boldsymbol{x} \in \mathbb{R}^d$ and predict the output $\boldsymbol{y} \in \mathbb{R}^k$. The linear regression model is derived for a single target donated as y and afterwards adapted to a multiple class formulation. For linear

regression the output is a linear function of the input. The model $f(\cdot)$ can be defined as

$$\hat{y} = f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}, \tag{2.1}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is a vector of parameters. The parameters are weights that determine how much each feature of $\boldsymbol{x}$ affects the prediction. Further, the predicted value is denoted as $\hat{y}$. After introducing the linear regression model, a loss function L for performance measuring is needed. One trivial possibility is to compute the Mean Squared Error (MSE) between input and output of the model. The loss function for this linear regression model is given by

$$L = \text{ MSE}(\boldsymbol{X}, \boldsymbol{y}) = \frac{1}{n}\sum_{i}^{n}(f(\boldsymbol{x}_i, \boldsymbol{w}_i) - y_i)^2 = \frac{1}{n}||\hat{\boldsymbol{y}} - \boldsymbol{y}||_2^2 = \frac{1}{n}||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||_2^2, \tag{2.2}$$

where $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, $\boldsymbol{y} \in \mathbb{R}^n$ and n is the number of samples. To gain the optimal weights $\boldsymbol{w}^*$ for the model the cost function has to be minimized with respect to $\boldsymbol{w}$. This can be done by setting the gradient to zero

$$\nabla_{\boldsymbol{w}} L = \frac{1}{n}\nabla_{\boldsymbol{w}}||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 = 0. \tag{2.3}$$

Hence, the analytical solution for the optimal weights is

$$\rightarrow \boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{2.4}$$

Normally the linear regression model has one additional parameter, called the bias b. So the equation 2.1 can be adapted to

$$f(\boldsymbol{x}, \boldsymbol{w}, b) = \boldsymbol{w}^T\boldsymbol{x} + b. \tag{2.5}$$

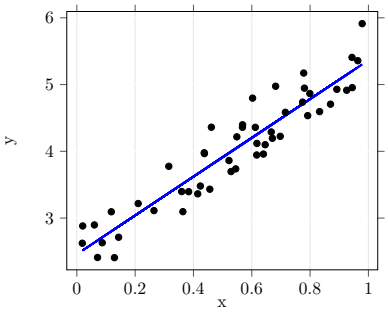
By observing b to $\boldsymbol{w}$ by augmenting $\boldsymbol{x}$ with an extra entry set to 1 equation 2.5 can be written as

$$f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x}, \tag{2.6}$$

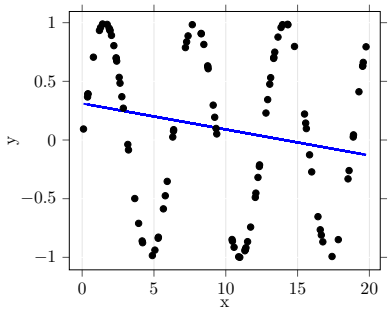
with $\boldsymbol{x} \in \mathbb{R}^{d+1}$ and $\boldsymbol{w} \in \mathbb{R}^{d+1}$. From now on the term linear regression will refer to an affine function [26]. An example for a linear regression problem is shown in figure 2.1a. Linear regression is a very simple, but limited learning algorithm. It already gives an idea how a machine learning algorithm can be developed and how it can learn from data. One big issue of this learning algorithm is, that it can just fit linear data. This problem is illustrated in 2.1b where the linear regression model tries to fit a function sampled from sinus. To overcome this issue equation 2.1 can be adapted by considering linear combinations of fixed non-linear functions $\Phi(\cdot)$ of the input $\boldsymbol{x}$. Resulting to the adapted model

$$\hat{y} = f(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{m-1} \boldsymbol{w}_j\Phi_j(\boldsymbol{x}) = \boldsymbol{w}^T\Phi(x), \tag{2.7}$$

where m is the total number of parameters. The bias term is absorbed by $\Phi_0(\boldsymbol{x}) = 1$. With this trick, the function $f(\cdot)$ is a non linear function of the input vector $\boldsymbol{x}$. Still the model is linear in $\boldsymbol{w}$. The choice of the basis function depends on the data. One possibility would be to use a polynomial basis function, Gaussian basis function or a



(a) Data are sampled from a linear function. Sampled data points are the black dots. The prediction of the model is shown via the blue line. The model estimates the underlying data.



(b) Data are sampled from $sin(x)$. Sampled data points are the black dots. The prediction of the model is shown via the blue line. The model is not able to estimate the underlying data.

Figure 2.1: Fitting data via the simple linear regression problem $y = w_1 x + b$.

sigmoidal basis function [7]. Figure 2.2 shows how a linear regression model with the use of a polynomial basis function $\Phi_j(x) = x^j$ can describe the true data sampled from sinus.

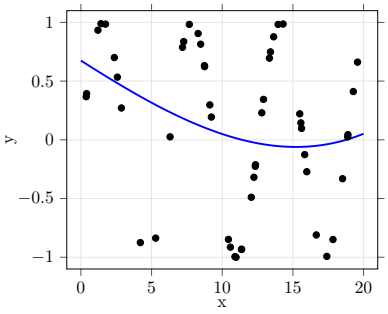
For multiple outputs with $k > 1$, the model from equation 2.7 can be reformulated to

$$\hat{\boldsymbol{y}} = \boldsymbol{W}^T \Phi(\boldsymbol{x}), \tag{2.8}$$

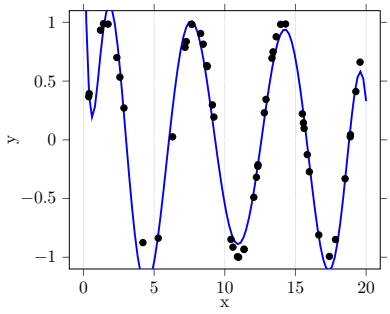
where $\boldsymbol{y}$ is a k-dimensional column vector, $\boldsymbol{W} \in \mathbb{R}^{m \times k}$ and $\Phi(\boldsymbol{x})$ is a m-dimensional column vector. The optimal solution can then be derived as

$$\boldsymbol{W}^* = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{Y} \tag{2.9}$$

with $\boldsymbol{Y} \in \mathbb{R}^{n \times k}$.



(a) Data are sampled from $sin(x)$. Sampled data points are the black dots. The prediction of the model is shown via the blue line. The model uses a polynomial basis function of degree three. The model is not able to capture the underlying data.



(b) Data are sampled from $sin(x)$. Sampled data points are the black dots. The prediction of the model is shown via the blue line. The model uses a polynomial basis function of degree ten. The model is able to capture the underlying data.

Figure 2.2: Fitting data via linear regression model $\hat{y} = \sum_{j=0}^{M-1} w_j \Phi_j(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\Phi}(\boldsymbol{x})$ using a polynomial basis function $\Phi_j(x) = x^j$.

2.1.1 Limitation of Linear Regression

A big advantage of the linear regression model is the linearity in the parameters, which leads to closed-form solution for the least-square problem. Unfortunately, there are some disadvantages which can be overcome by the use of neural networks. One limiting factor of linear regression is that the basis functions $\Phi(\cdot)$ are fixed before training. Thereby, the data manifold should be known before defining the model, which is difficult for complex multidimensional data. Furthermore, the needed parameters are rapidly growing with the number of features d from the input space. Finally, the linear regression model cannot handle data, where the target variables have a significant dependence on a small number of features within the data manifold.

To over come these limitations, deep learning models, which are explained in more detail in the next chapter, use adaptive basis functions to learn parameters so that the regions of input space correspond to the data manifold. Through the use of neural networks the manifold of the data can be exploited by choosing the directions in input space to the corresponding basis functions [7].

2.2 Deep Learning Basics

As shown in section 2.1 a basis function $\Phi(\cdot)$ can be applied to transform the input $\boldsymbol{x}$ to a linear space. The strategy of deep learning is to learn the basis function $\Phi(\cdot)$. Deep learning is also known as deep feed-forward network, feed-forward neural network, multilayer perceptron (MLP) or artificial neural network (ANN). In fact parametric forms for the basis functions are used and the parameters are optimized during training. This results in a significantly more compact model. The disadvantage of this model is that the function of the model parameters are no longer convex, whereby a minimum must not be the global minimum. Furthermore, the optimal solution of the model function can no longer be solved analytically as for the linear regression problem. In order to still get a solution, the gradient descent algorithm is introduced.

2.2.1 Multilayer Perceptron (MLP)

Deep learning models are inspired from biological neural network models of animals, therefore they are often referred as ANN. An ANN is a network of units, known as artificial neurons. These neurons receive weighted inputs and produce an output, depending on the used activation function $h(\cdot)$. Figure 2.3 shows such a single artificial neuron, which produce an output z of the input $\boldsymbol{x}$. Mathematically speaking, a deep learning model is a series of composed functions in a chain $f(\boldsymbol{x}) = f^{(L)}(f^{(L-1)}(\dots f^{(2)}(f^{(1)}(f^{(0)}(\boldsymbol{x})))))$, where each function can be seen as a layer consisting of artificial neurons, producing an output. Thereby, the number of layers are defined by the number of composed functions L. For $L > 1$ the model is called deep. The functions can be defined as linear combinations of the input variable $\boldsymbol{x}$ which will be transformed by using a differentiable, non-linear activation function $h(\cdot)$. The input of the activation function is defined as

$$a_j^{(l)} = \sum_{i=1}^{d} w_{ji}^{(l)} x_i + w_{j0}^{(l)}, \tag{2.10}$$

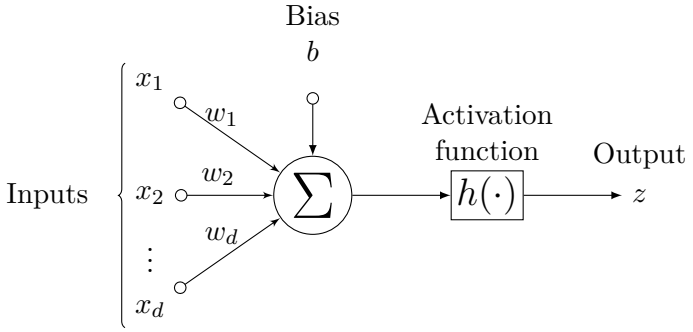


Figure 2.3: Model of an artificial neuron.

where d is the dimension of the input $\boldsymbol{x}$, w_{ji} the weights and w_{j0} the biases. The layer is indicated by $l \in \mathbb{N}^L$ and $j \in \mathbb{N}^J$ defines the neuron in this layer. The number of neurons of each layer is defined as J. Applying the activation function $h(\cdot)$ to equation 2.10 leads to the output

$$z_j^{(l)} = h(a_j^{(l)}) \tag{2.11}$$

of an artificial neuron. Putting the neurons of a layer l together resulting in a function for each layer, defined as

$$f^{(l)}(\boldsymbol{x}_l, \boldsymbol{w}_l) = h(\boldsymbol{a}^{(l)}) = h(\boldsymbol{w}_l \boldsymbol{x}_l). \tag{2.12}$$

The output of $f^{(l)}(\cdot)$ is the input of the next layer $f^{(l+1)}(f^{(l)}(\cdot))$. The complete MLP model can be written as

$$f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}_l h_l(\boldsymbol{w}_{l-1} h_{l-1}(\dots h_0(\boldsymbol{w}_0 \boldsymbol{x}) \dots)). \tag{2.13}$$

In figure 2.4 such a MLP model is depicted.

The first layer is called input layer, the last one is called output layer, all layers between are called hidden layers. The choice of activation functions depends on the nature of data and the desired target. In general, the activation function of the last layer differs from the others. Furthermore, non linear activation functions are used. With the use of linear activation functions the neural network would not be able to separate data with a non linear manifold. If only linear activation functions are selected, then there always exist an equivalent network without hidden units

$$f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}_j h_j(\boldsymbol{w}_{j-1} h_{j-1}(\dots h_0(\boldsymbol{w}_0 \boldsymbol{x}) \dots)) = (\boldsymbol{w}_j \boldsymbol{w}_{j-1} \dots \boldsymbol{w}_0)\boldsymbol{x} = \boldsymbol{w'}\boldsymbol{x}. \tag{2.14}$$

After defining the machine learning model, here our MLP, the cost function and finally the optimization algorithm has to be defined. The cost function depends on the activation function of the last layer, which depends on the dataset being used and the distribution of the target variables. For a binary classification problem $y_i \in 0, 1$ the binary cross-entropy with the sigmoid function for the last layer could be used. The loss function of the binary cross entropy is defined as

$$E(\boldsymbol{w}) = -\sum_{i=1}^{N} (y_i \log f(\boldsymbol{x}_i, \boldsymbol{w}) + (1 - y_i) \log[1 - f(\boldsymbol{x}_i, \boldsymbol{w})]). \tag{2.15}$$

The intuition behind the binary cross entropy is to force the model to return high probabilities for samples that are certainly from class 1 and low probabilities if they are

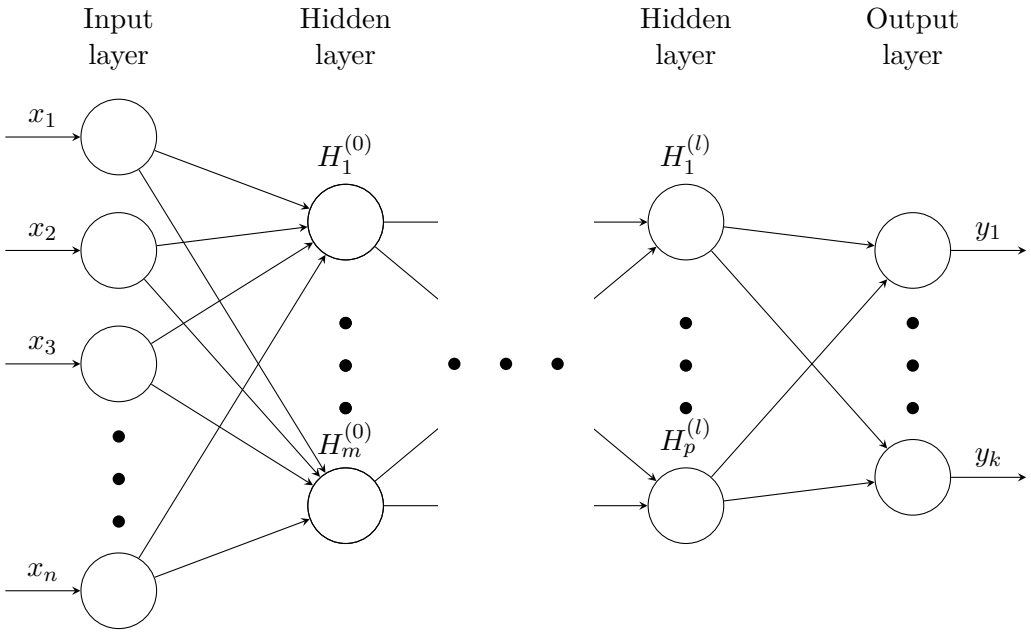


Figure 2.4: Schematic drawing of a MLP also known as a deep neural net model with input $\boldsymbol{x}$, hidden layers H and output $\boldsymbol{y}$.

not from class 1. For a multi class output the cross-entropy between the target $y_i \in 0, 1^K$ and the softmax for the last layer can be used. The cross-entropy loss is defined as

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K}(y_{nk} \log f_k(\boldsymbol{x}_n, \boldsymbol{w})). \tag{2.16}$$

It is the general case of the binary cross-entropy for multi class problems. Normally, the cost function of a deep learning model is non-convex and an analytical solution does not exist. Therefore, optimization is more tricky than for the linear regression model. A common choice is to use the gradient descent algorithm to find a local minimum

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \alpha \nabla_{\boldsymbol{w}} E(\boldsymbol{w}^{(t)}), \tag{2.17}$$

with $\alpha > 0$ known as the learning rate. The gradient descent algorithm is a first-order iterative optimization algorithm, which means optimization is done along the negative of the first deviation of the function. In the deep learning community the Adam optimization algorithm [33] has been established in recent years. It is based on the gradient descent algorithm and is using the first and second momentum, to overcome the fixed learning rate in the gradient descent algorithm. To compute the gradient $\nabla_{\boldsymbol{w}} E(\boldsymbol{w}^{(t)})$ an efficient technique known as error back-propagation or short backprop [13] is used. The back-propagation algorithm is applying the chain rule and is calculating the gradients of the network from the last to the first layer by using the gradients from the previous layers.

2.2.2 Convolutional Neural Network (CNN)

In the MLP model each neuron of a layer l is connected to all neurons of the next layer $l + 1$, therefore it is also known as fully connected layer (FC). This leads to many

trainable parameters especially for a high dimensional input space. Furthermore, no shared weights exist in the MLP model. One solution, to tackle this problem are CNNs.

CNNs are a specialized kind of ANNs. They are used for data which has a known grid-topology like images. CNNs are neural networks that use a convolutional layer (Conv) in place of a FC in at least one of their layers. CNNs have two useful properties.

1. The number of trainable weights scale well with larger input dimension.
2. The ability to handle spatial information.

A convolutional layer consists of a set of learnable filters (kernels) $\boldsymbol{K}$, which are convolved (slided) with a defined stride across the input matrix $\boldsymbol{I}$. Between the input and the filter dot products $\odot$ are computed. Figure 2.5 illustrates the convolving mechanism. This results in a so called activation map. Due to convolving over the input with a given filter, the dimension of the activation map is smaller than the input dimension. A common technique is to add zeros around the boarder of the activation map, to keep the dimensions the same. This is often referred as zero padding. Depending on the used

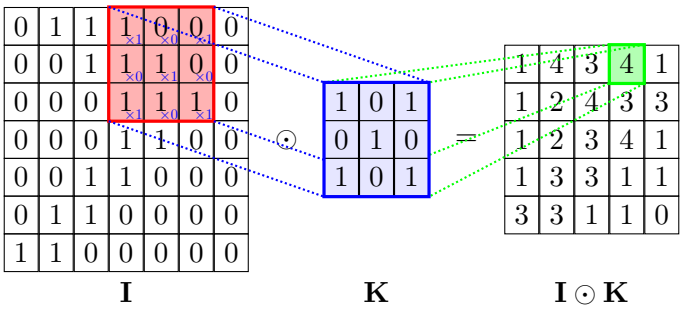


Figure 2.5: Convolving a 3x3 kernel $\boldsymbol{K}$ over the input $\boldsymbol{I}$.

filters, the network will learn some type of visual feature such as edges or clusters of a colour. Finally, the activation maps will be stacked together and the output volume will be produced. To reduce the dimension a stride greater than one can be used. Alternative a pooling layer after the convolution layer can be applied. The pooling layer is normally using the max operator with a defined filter size and stride. Figure 2.6 shows a max pooling layer applied to an input [29].

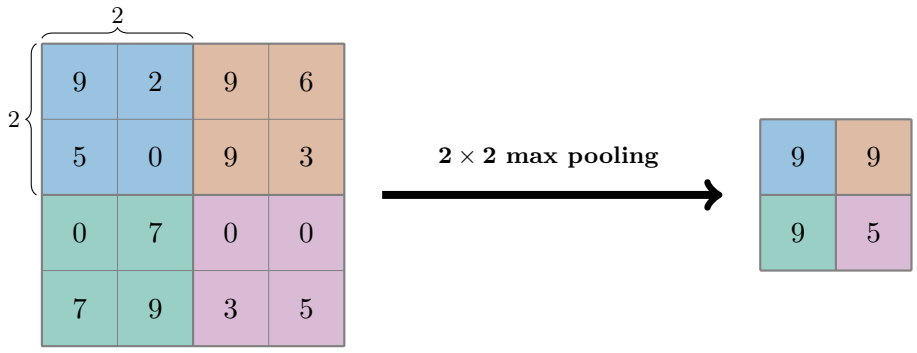


Figure 2.6: Max pooling with a $2x2$ filter and stride 2.

2.2.3 Residual Block (Res-Block)

Increasing the depth of a deep neural network model leads to a more complex one. Usually resulting to better performance, however the depth of a model cannot be increased arbitrary as shown in [22]. With too many layers the model faces the problem of degradation of accuracy caused by vanishing gradients. In [22] He et al. proposed a method, referred as Res-Block, to overcome this problem by adding shortcut connections to a feed forward neural network. The intuition behind the shortcut connections is that the model has the ability to skip some layers. The shortcut connection is realized by an identity mapping from the input $\boldsymbol{x}$ to the output $\boldsymbol{y}$. A Res-Block is defined as

$$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{w}) + \boldsymbol{x}, \tag{2.18}$$

thereby $f(\boldsymbol{x}, \boldsymbol{w})$ represents the residual function with the trainable weights. Usually the residual function $f(\boldsymbol{x}, \boldsymbol{w})$ can be any ANN. Only the use of a single layer is leading to an output similar to a linear layer $\boldsymbol{y} = \boldsymbol{w}\boldsymbol{x} + \boldsymbol{x}$, where the vanishing gradient problem would still arise. In figure 2.7 a residual block with two layers for the residual function is shown.

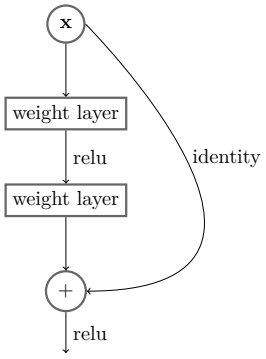


Figure 2.7: A residual block, which is defined by $\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{w}) + \boldsymbol{x}$, where $\boldsymbol{x}$ is the input and $\boldsymbol{y}$ the output. The residual function contains two layers.

2.3 Advanced Deep Learning Models

In the previous section the basics of deep learning are derived and some useful building blocks to improve the performance of a deep learning model are depicted. In this section an intelligent and simple way of how to apply a deep learning model for unsupervised learning task, namely the autoencoder model, is introduced. Further this model is then enhanced to transform it into a deep generative model. Additionally two other deep generative model approaches are depicted.

2.3.1 Autoencoder Model

An autoencoder model is an unsupervised neural network, that is trained to predict the input itself. In short, it is an artificial neural network, which tries to reproduce the input to the output without loss of information. An autoencoder can be split into two parts. The first part is an encoder function $\text{Enc}_{\boldsymbol{\Phi}}(\boldsymbol{x}) = \boldsymbol{z}$, which maps the input

$\boldsymbol{x} \in \mathbb{R}^n$ to a lower dimensional space $\boldsymbol{z} \in \mathbb{R}^k$ with $k < n$. The second part is a decoder function $\text{Dec}_{\boldsymbol{\theta}}(\boldsymbol{z}) = \hat{\boldsymbol{x}}$, which is doing the reverse way. The reconstructed output is defined by $\hat{\boldsymbol{x}} \in \mathbb{R}^n$, which has the same dimension as the input $\boldsymbol{x}$. The trainable weights are indicated by $\boldsymbol{\Phi}$ and $\boldsymbol{\theta}$. Combining the encoder and decoder functions is leading to the complete autoencoder model

$$f_{\boldsymbol{\Phi},\boldsymbol{\theta}}(\boldsymbol{x}) = \text{Dec}_{\boldsymbol{\theta}}(\text{Enc}_{\boldsymbol{\Phi}}(\boldsymbol{x})). \tag{2.19}$$

The loss $L_{\text{rec}}(\boldsymbol{x})$, known as reconstruction loss, is defined as the MSE between the input and the reconstructed output

$$L_{\text{rec}}(\boldsymbol{x}) = \frac{1}{n}||\boldsymbol{x} - \hat{\boldsymbol{x}}||^2 = \frac{1}{n}||\boldsymbol{x} - \text{Dec}_{\boldsymbol{\theta}}(\boldsymbol{z})||^2 = \frac{1}{n}||\boldsymbol{x} - \text{Dec}_{\boldsymbol{\theta}}(\text{Enc}_{\boldsymbol{\Phi}}(\boldsymbol{x}))||^2. \tag{2.20}$$

Hence, the optimization problem follows as

$$\boldsymbol{\Phi}^*, \boldsymbol{\theta}^* = \underset{\boldsymbol{\Phi}^*,\boldsymbol{\theta}^*}{\text{argmin}}||\boldsymbol{x} - \hat{\boldsymbol{x}}||^2 = \underset{\boldsymbol{\Phi}^*,\boldsymbol{\theta}^*}{\text{argmin}}||\boldsymbol{x} - \text{Dec}_{\boldsymbol{\theta}}(\boldsymbol{z})||^2 = \underset{\boldsymbol{\Phi}^*,\boldsymbol{\theta}^*}{\text{argmin}}||\boldsymbol{x} - \text{Dec}_{\boldsymbol{\theta}}(\text{Enc}_{\boldsymbol{\Phi}}(\boldsymbol{x}))||^2, \tag{2.21}$$

which can be solved as a minimization problem using an optimization algorithm like the gradient descent method from equation 2.17. In between the encoder and the decoder the bottleneck is defined as $\boldsymbol{z}$, also referred as the latent space. The bottleneck has in general a smaller dimension than the input space. Otherwise the system would learn an identity mapping and many useful properties of the autoencoder would be lost [26]. In 2.8 the model of an autoencoder is depicted.

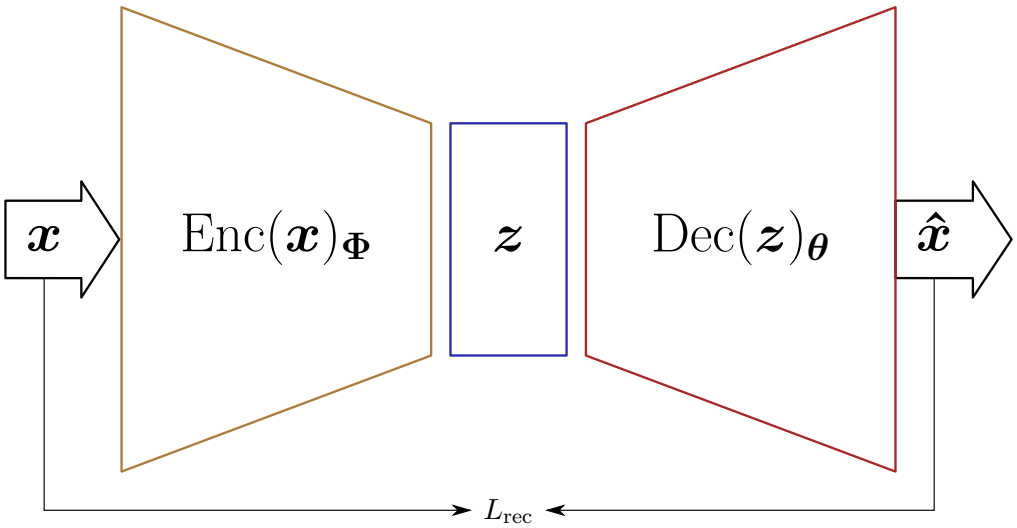


Figure 2.8: An autoencoder model with input $\boldsymbol{x}$, latent space $\boldsymbol{z}$ and reconstructed input $\hat{\boldsymbol{x}}$. The Encoder and Decoder are represented through an ANN with the trainable weights $\boldsymbol{\Phi}$ and $\boldsymbol{\theta}$. L_{rec} is the reconstruction loss calculated between input and output.

Despite its simple principals, the autoencoder model has very useful applications. Historically it was used for dimensionality reduction like the principal component analysis (PCA) method, but with the advantage of capturing non-linearity in the data [17]. Further applications are denoising autoencoders, which can recover sharp images from noisy input images [43]. Recently, enhancements to the normal autoencoder brought a connection between autoencoders and latent variable models. This enables the use of autoencoders as generative models and helps to interpret the latent space - the bottleneck of an autoencoder.

2.3.2 Deep Generative Models

The goal of deep generative models is to generate new samples, which have the same probabilistic distribution $p(\boldsymbol{x})$ of the given data set $\boldsymbol{x}$, using a deep neural network. Three popular approaches of deep generative models exists:

1. Generative Adversarial Networks (GANs) [18]
2. Variational Autoenocders (VAEs) [34]
3. Autoregressive models [57]

The focus of this work is the VAE, therefore, it is described in more detail in the following. Afterwards the principles of the two other generative models are depicted in short.

Variational Autoencoder (VAE)

A VAE is an extension of traditional autoencoders, which allows disentangled, interpretable, automated discovery of the latent space. Instead of encoding an input as a single point, it is encoded as a distribution over the latent space. VAEs are generative models and were first introduced by Kingma and Welling [34] in 2013. A disentangled latent space is a space where single latent units are sensitive to changes of a single generative factor, while being relatively invariant to changes from other factors [5]. This is illustrated in figure 2.9 where the used dataset consists of gray scale images of 2D geometric objects.[1] The generative factors are the shape, horizontal position, vertical position,

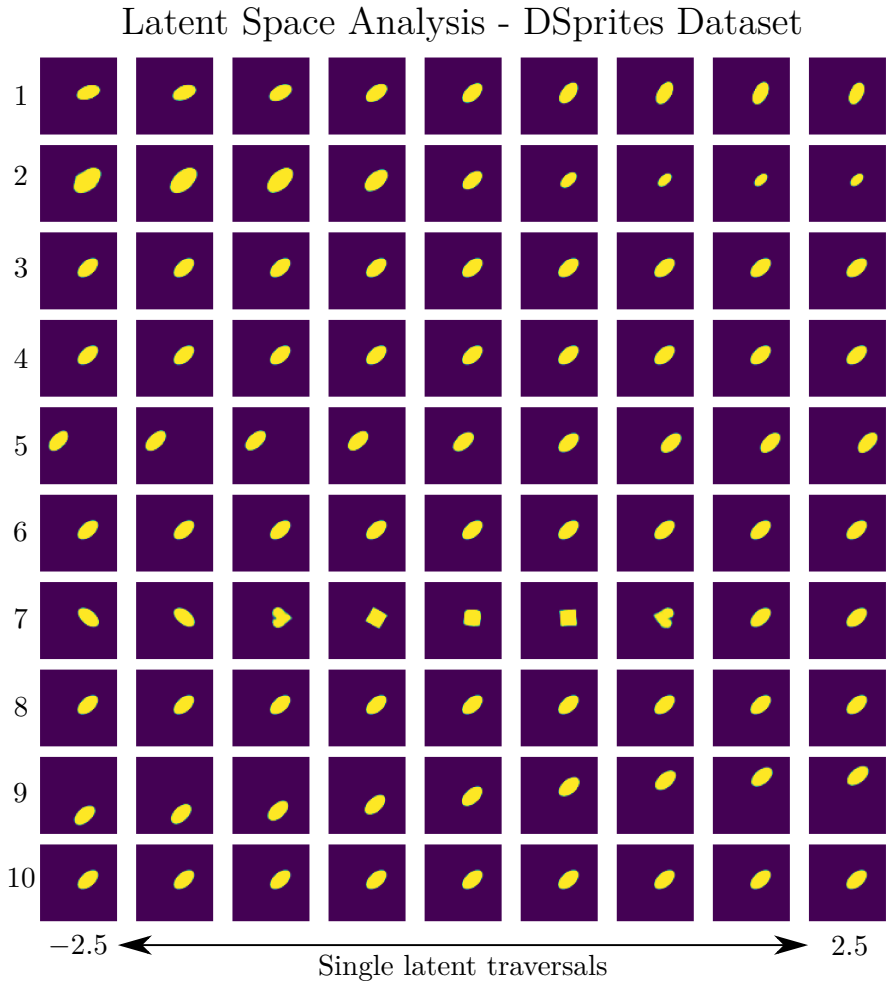


Figure 2.9: Latent traversal plots for the DSprites dataset. The latent space has a dimension of $k = 10$. For each single latent variable the value is traversed from $[-2.5, 2.5]$, while keeping the remaining latent values fixed.

rotation and size of the object. The used VAE model learns each generative factor in a single latent unit, where the change of the value of the latent unit, only effects the property represented through this latent unit. This can be seen for example in row nine, which represents the vertical position indicating that this latent variable has learned this property in an unsupervised manner. The intuition behind VAE models is that random

[1] DSprites is a dataset of 2D shapes procedurally generated from 5 ground truth independent latent factors [39].

variables Z, which can be sampled easily with a given distribution, are used. The goal is to find a non-linear mapping to sample a complicated unknown distribution from the known one. Figure 2.10 illustrates these principals. The mathematical background of a

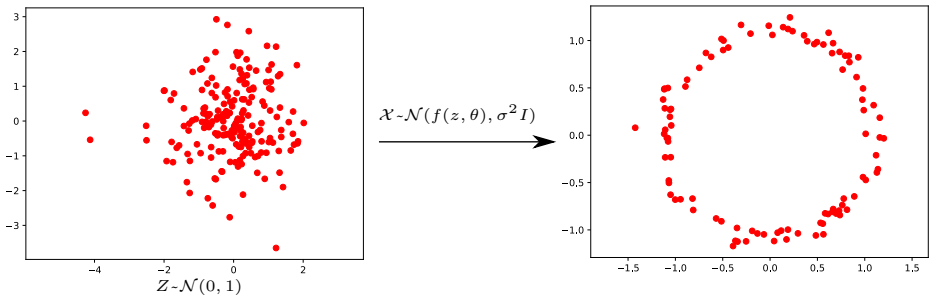


Figure 2.10: The left figure shows samples from a normal distribution. The right figure shows the target distribution. The goal of the VAE is to find a non-linear mapping function through a deep neural net by optimizing the weights θ. In this case the target distribution can be received by applying the mapping function $f(z) = z/10 + z/||z||$.

VAE is further explained. The model of the autoencoder from 2.3.1 can be reformulated into a probabilistic view. The encoder can be defined as $\text{Enc}_{\boldsymbol{\Phi}}(\boldsymbol{x}) = q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})$ and the decoder as $\text{Dec}_{\boldsymbol{\theta}}(\boldsymbol{z}) = q_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, where $\boldsymbol{\Phi}$, $\boldsymbol{\theta}$ are again the trainable weights, $\boldsymbol{x}$ an input sample and $\boldsymbol{z}$ the latent space. To prevent the model from encoding data far apart in the latent space a prior $p(\boldsymbol{z})$ over the latent distribution is introduced. The prior serves as a regularization term. It should be possible to generate data $\boldsymbol{x}$ from the latent variables $\boldsymbol{z}$. In detail, the generative process contains the following two steps for each data point i:

1. Sample latent variables $z_i \sim p(\boldsymbol{z})$
2. Sample data point $x_i \sim p(\boldsymbol{x}|\boldsymbol{z})$

To infer good values for the latent variables $\boldsymbol{z}$, given the observed data $\boldsymbol{x}$, the posterior $p(\boldsymbol{z}|\boldsymbol{x})$ can be inferred by using the Bayes theorem

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})}. \tag{2.22}$$

Because of high dimensional input space, the denominator $p(\boldsymbol{x})$ known as evidence is often intractable. To overcome the problem of intractability, one possible solution is to approximate the posterior distribution by variational inference. With variational inference an unknown distribution can be approximated with a known distribution family. Here, the distribution family is donated as $q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})$. Thereby $\boldsymbol{\lambda}$ indicates the parameter of the distribution family. For example if the family of distribution is selected as Gaussian, it would be the mean and variance for each datapoint $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. To measure how good the approximated distribution $q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})$ fits the original one $p(\boldsymbol{z}|\boldsymbol{x})$, the Kullback-Leibler (KL) divergence (appendix A.1) can be used

$$\mathbb{KL}(q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x})) = \int q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})} dx = E_{z\sim q}[\log q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z}|\boldsymbol{x})]. \tag{2.23}$$

Applying $p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x},\boldsymbol{z})}{p(\boldsymbol{x})}$ to equation 2.23 results to

$$\mathbb{KL}(q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x})) = E_{z\sim q}[\log q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{x},\boldsymbol{z})] + \log p(\boldsymbol{x}). \tag{2.24}$$

Thereby, $\log p(\boldsymbol{x})$ comes out of the expectation because it does not depend on $\boldsymbol{z}$. To find the best variational parameters $\boldsymbol{\lambda}^*$ equation 2.24 can be formulated into an optimization problem

$$\begin{aligned} q^*_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x}) = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}} \mathbb{KL}(q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x})) = \\ \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} E_{z\sim q}[\log p(\boldsymbol{x},\boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})] - \log p(\boldsymbol{x}) = \\ \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \underbrace{E_{z\sim q}[\log p(\boldsymbol{x},\boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})]}_{ELBO}. \end{aligned} \tag{2.25}$$

Since $p(\boldsymbol{x})$ is not depending on $\boldsymbol{\lambda}$, the KL divergence can be minimized by maximizing $E_{z\sim q}[\log p(\boldsymbol{x},\boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})]$, which is called the evidence lower bound (ELBO). The ELBO can be reformulated to

$$\begin{aligned} \text{ELBO} = E_{z\sim q}[\log p(\boldsymbol{x}|\boldsymbol{z}) + \log p(\boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})] = \\ E_{z\sim q}[\log p(\boldsymbol{x}|\boldsymbol{z})] - E_{z\sim q}[\log q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x}) - \log(\boldsymbol{z})] = \\ E_{z\sim q}[\log p(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{KL}(q_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})). \end{aligned} \tag{2.26}$$

Finally, the connection back to the neural network model defined as above is made. The encoder $\text{Enc}(\boldsymbol{x}) = q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})$ takes as input $\boldsymbol{x}$ and outputs the parameters $\boldsymbol{\lambda}$. The decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ takes the latent variable $\boldsymbol{z}$, which is sampled from the parameter $\boldsymbol{\lambda}$ as input and outputs the reconstructed data. Hence, the loss function for the VAE model is defined as

$$L_{\text{VAE}}(\boldsymbol{\theta},\boldsymbol{\Phi}) = -ELBO = \underbrace{-\mathbb{E}_{z\sim q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}_{L_{\text{rec}}} + \underbrace{\mathbb{KL}(q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))}_{L_{\text{KL}}}. \tag{2.27}$$

Where the first term is the reconstruction loss L_{rec} and the second term L_{KL} acts as regularization, which forces the VAE model to sample from a given distribution [28]. To increase and control the disentanglement of the latent space Higgins et al. enhanced the VAE model by simply adding a hyper parameter β to the L_{KL} term. Which leads to the so called β-VAE model. The new loss results to

$$L_{\beta-\text{VAE}} = L_{\text{rec}} + \beta L_{KL}. \tag{2.28}$$

Normally, a centered isotropic multivariate Gaussian $\mathcal{N}(0,\boldsymbol{I})$ as prior $p(\boldsymbol{z})$ is used. Hence, the closed form solution for the regularization term L_{KL} is given as

$$L_{\text{KL}}(\boldsymbol{\mu},\boldsymbol{\sigma}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M_z}(1 + \log(\sigma_{ij}^2) - \mu_{ij}^2 - \sigma_{ij}^2), \tag{2.29}$$

where M_z is the dimension of the latent space $\boldsymbol{z}$ and N is the number of data samples (For derivation see Appendix B in [34]). Finally, Kingma et al. apply a so called reparameterization trick to ensure that the ELBO can be straightforwardly differentiated with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\Phi}$. Hence, the latent space $\boldsymbol{z}$ is defined as

$$\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{\Phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\Phi}}(\boldsymbol{x}) * \boldsymbol{\varepsilon}, \tag{2.30}$$

Algorithm 2.1 : Training of a VAE model using the reparameterization trick and sampling from $\mathcal{N}(0, \boldsymbol{I})$

Data : $\boldsymbol{X}$ images

$\boldsymbol{\Phi}, \boldsymbol{\theta} \leftarrow$ Initialize the weights of the encoder and decoder;
while *not converged* **do**
  $\boldsymbol{x} \leftarrow$ Take random mini-batch from $\boldsymbol{X}$;
  $\boldsymbol{\mu}, \boldsymbol{\sigma} \leftarrow \text{Enc}(\boldsymbol{x})$;
  $\boldsymbol{\varepsilon} \leftarrow$ Samples from $\mathcal{N}(0, \boldsymbol{I})$;
  $\boldsymbol{z} \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} * \boldsymbol{\varepsilon}$;
  $\hat{\boldsymbol{x}} \leftarrow \text{Dec}(\boldsymbol{z})$;
  $L_{\text{rec}} \leftarrow L_{\text{rec}}(\hat{\boldsymbol{x}}, \boldsymbol{x})$;
  $L_{\text{KL}} \leftarrow \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M_z} (1 + \log(\sigma_{ij}^2) - \mu_{ij}^2 - \sigma_{ij}^2)$;
  $L \leftarrow L_{\text{rec}} + L_{\text{KL}}$;
  $\Delta\boldsymbol{\Phi}, \Delta\boldsymbol{\theta} \leftarrow \nabla_{\boldsymbol{\Phi}, \boldsymbol{\theta}} L$ ▷ `Get gradients`
  $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \eta \Delta\boldsymbol{\Phi}$ ▷ `Perform gradient step`
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \Delta\boldsymbol{\theta}$ ▷ `Perform gradient step`

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \boldsymbol{I})$. Training flow of a VAE model with the reparameterization trick is depicted in algorithm 2.1. Summarizing the approach of VAE it is an elegant generative model, which is easy to train. The model represents the input in a reduced disentangled latent space and can be used for unsupervised learning processes. The disadvantage is that the models tend to create blurry images and achieve limited disentangling performance on complex datasets [27]. These limitations can be attributed to the limited expressiveness of the inference models [64], the injected noise through the sampling process and the imperfect element-wise similarity metric such as the mean squared error for the reconstruction loss [37]. The VAE model with the reparameterization trick is depicted in figure 2.11.

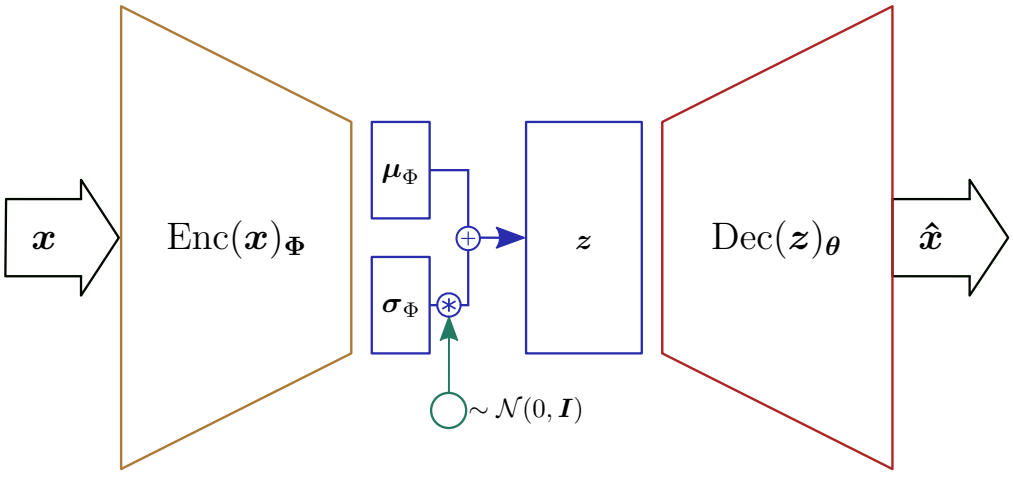


Figure 2.11: Model of an VAE using the reparameterization trick from equation 2.30.

Generative Adversarial Net (GAN)

A GAN is a deep generative model consisting of an adversarial process. It was proposed by Goodefloow et al. in the year 2014 [18]. In the adversarial process two models are trained during a two-player min-max game. One model is the generative model G, that captures the underlying data distribution of the input data. The other model is a discriminative model D, which tries to distinguish between a real sample or a fake sample generated from G.

The generator $G_{\boldsymbol{\Phi}}(\boldsymbol{z}) = \boldsymbol{x}_{\text{fake}}$ can be defined as a multilayer perceptron model with the learnable weights $\boldsymbol{\Phi}$ and the input sampled from random noise variables $p_{\boldsymbol{z}}(\boldsymbol{z})$ to learn the generator's distribution $p_g(\boldsymbol{x})$ over the data $\boldsymbol{x}$. The generator produces the fake samples $\boldsymbol{x}_{\text{fake}}$. The discriminator $D_{\boldsymbol{\theta}}(\bar{\boldsymbol{x}})$ is also a multilayer perceptron model with the parameters $\boldsymbol{\theta}$. The discriminator gets either the real data $\boldsymbol{x}$ or the fake data $\boldsymbol{x}_{\text{fake}}$ as input $\bar{\boldsymbol{x}}$. The discriminator tries to distinguish between the true data $\boldsymbol{x}$ and the fake data $\boldsymbol{x}_{\text{fake}}$, while the generator tries to generate samples to fool the discriminator. This results into a min-max two-player game between the generator and the discriminator. Thus the model is forcing itself to generate samples with better quality. The min-max game is defined as

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{2.31}$$

In Figure 2.12 the GAN approach is schematically shown.

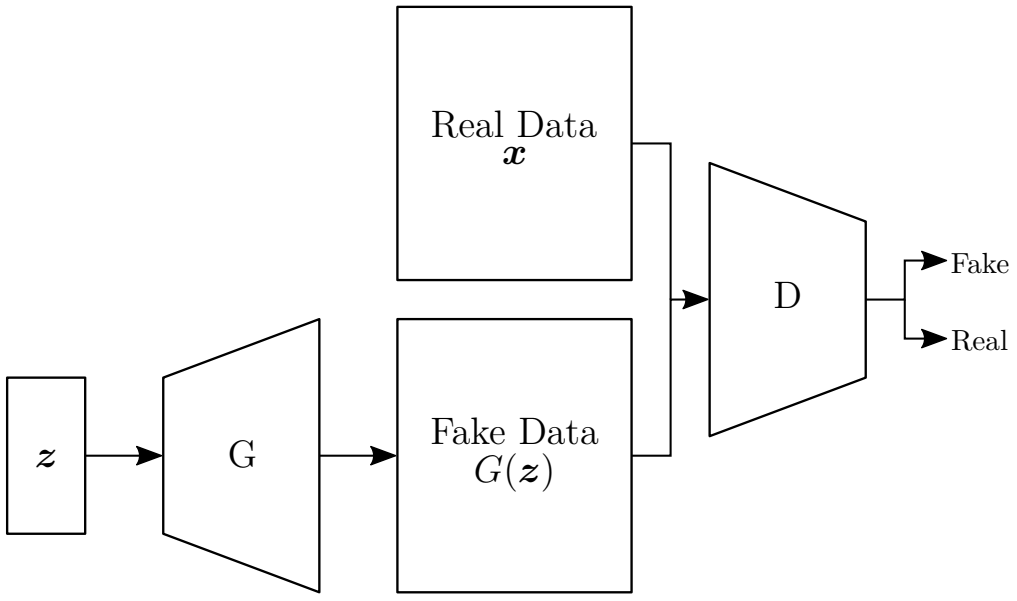


Figure 2.12: GAN model: The discriminator tries to distinguish between the real data $\boldsymbol{x}$ and the fake data $G(\boldsymbol{z})$, whereby $\boldsymbol{z}$ is sampled from an arbitrary distribution.

GANs are generating sharp images, but the models are difficult to train. The training process is usually unstable and is prone to mode collapse [24]. Another disadvantage is that the standard GAN approach can only generate random samples. With a standard GAN it is not possible to manipulate the latent space to get a certain output.

Autoregressive Generative Models

Autoregressive models in deep learning have been invented by Larochelle et al. in the year 2011 [25]. The approach was enhanced by van den Oord et al. in [57], where it was applied to the image generation domain. The basic idea of such a model is to use the conditional distributions over the pixels to estimate the joint distribution $p(\boldsymbol{x})$ of an image. In fact, the i-th pixel x_i can be estimated using the product of the conditional distributions of the previous pixels

$$p(\boldsymbol{x}) = \prod_{i=1}^{n^2} p(x_i|x_1, \ldots, x_{i-1}). \tag{2.32}$$

The generation proceeds pixel by pixel and row by row. For the output a 256 way softmax layer is used. This defines the pixel value between 0 and 255. Figure 2.13 illustrates the autoregressive process. For multi channel images it works analogically. To calculate the i-th pixel for a channel the product of the conditional distributions over all previous pixels for all channels are used. Van den Oord et al. proposed two different architecture named Pixel Recurrent Neural Networks (PixelRNN) and Pixel Convolutional Neural Networks (PixelCNN), which are using convolutional layers to model the autoregressive process. In [58], [47] and [10] further model improvements have been done named Gated PixelCNN, PixelCNN++ and PixelSNAIL.

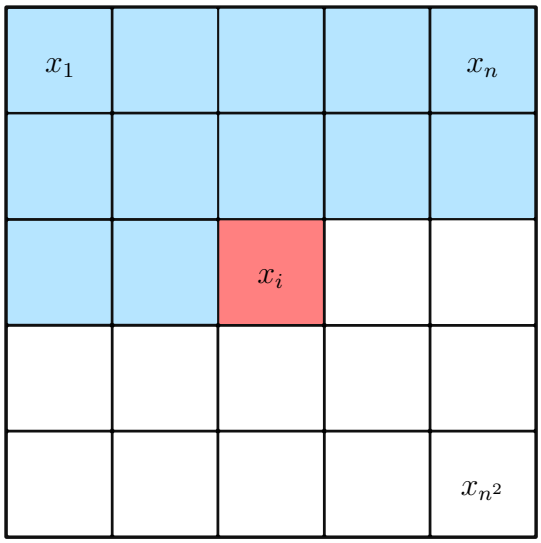


Figure 2.13: Autoregressive model: x_i is calculated with the use of previous pixels x_{i-1} to x_1.

Chapter 3

Related Work

As mentioned in 2.3.2 VAEs tend to reproduce blurry images and are limited on disentangling more complex datasets. Therefore, it is still an open research topic to improve the VAE model proposed by Kingma et al. The number of citations of the original VAE paper from Kingma et al. was round about 2.300 in the year 2018[1]. This indicates the big interest of the deep learning community in generative VAE models. Due to the high number of citations only the most relevant once for this work are discussed. In the first part of this chapter enhancements of the standard VAE are introduced. In the second part applications of VAE models on medical images are presented.

3.1 Variational Autoencoder (VAE) Models

Many research papers deal with the topic of analyzing and improving the VAE model. The different enhancements can be roughly split into:

1. Pure VAE approaches
2. Hybrid approaches combining GAN and VAE models
3. Hybrid approaches combining autoregressive and VAE models

3.1.1 Pure VAE Approaches

Pure VAE approaches can be divided into models with modified loss functions, improved prior distributions or improved model architectures. First of all, the models, which adapt or enhance the loss function are presented. One very simple enhancement was proposed by Higgins et al., the so called β-VAE [27]. The β-VAE add a hyper parameter to the $\mathbb{KL}$ term of equation 2.27 to weight the $\mathbb{KL}$ more in comparison to the reconstruction term. This leads to a better disentangling capability of the latent space, while decreasing the reconstruction quality of the output. Zhao et al. proposed a generalization of the β-VAE, which divides the $\mathbb{KL}$ term further and adds a mutual information maximization term [64]. The approach is called InfoVAE and claims to improve the reconstructed image quality and makes an effective use of the latent space compared to the standard VAE. In [8] the $\mathbb{KL}$ term is divided into three parts and further analyzed, similar to the approach

[1] The number of citation for the year to 2018 was taken from google scholar `https://bit.ly/2pXCBGt`.

of Zhao et al. Also in [36] the loss function is adapted. In this approach the expectation of the approximate posterior over observed data is added to the regularization term to encourage the disentanglement. Compared to the β-VAE this approach claims to preserve the sharpness of the standard VAE.

In the standard VAE an univariate normal distribution with zero mean and variance one for the prior is usually selected. This leads to a computational efficient model, but is often not flexible enough to match the true posterior distribution. Therefore, different models which improve the prior distribution are proposed. Tomczak et al. proposed an improvement of VAE using Householder Flow [55]. Householder Flow is based on normalizing flow developed by [45], which is a framework to building posterior distribution from a random variable. To overcome the limitation of the normal distribution as prior and the expensiveness of optimizing the ELBO by maximizing a Lagrange function, a so called variational mixture of posteriors prior was proposed in [54].

Finally, the architecture of the VAE model can be adapted to improve the performance. The so called Ladder VAE model proposed in [50] corrects the generative distribution by recursively computing the posterior and generative distribution, instead of just doing a bottom-up inference from the latent space to the reconstructed output, as in the standard VAE. Another model is the Wasserstein Autoencoder [53], which is minimizing the optimal transport cost. The optimal transport cost is a way to measure a distance between probability distributions and provides a much weaker topology than the $\mathbb{KL}$ divergence from the standard VAE. This can be an advantage for data represented on a low dimensional manifold, where the $\mathbb{KL}$ divergence is not providing useful gradients for training. The VPGA [63] and the spatial VAE [60] are two further models, which are adapting the architecture of the standard VAE model. They are discussed in more detail in section 4.1 and 4.2.

In addition to the previously described approaches, there exists a semi supervised model proposed in [2], called N-VAE, which is using the class label to split the latent space into general latent factors and specific latent factors. It is a promising approach to generate new samples with certain properties.

3.1.2 Combined GAN and VAE Approaches

One state of the art research field is to combine GAN and VAE models to one approach. The combined models are proposed by Makhzani et al. [40] and Larsen et al. [37]. Makhzani et al. is using a simple autoencoder and a discriminator which tries to distinguish if a sample is from the latent space of the autoencoder or sampled from a given distribution. In contrast Larsen is using a VAE instead of an autoencoder. These two models are still suffering from mode collapse, therefore, Srivastava et al. [51], Khan et al. [31] and Huang et al. [24] are proposing approaches to counteract this problem. Srivastav et al. model is called VEEGAN, which is reversing the action of the generator by mapping from data to noise. The model from Huang et al., called IntroVAE is doing it in an introspective manner, which is described in more detail in section 4.4. To avoid the mode collapse Khan et al. is using an error feedback loop. The approach was also evaluated on high resolution images.

3.1.3 Combined Autoregressive and VAE Approaches

Instead of combining GANs and VAE model. Autoregressive models and VAE models can be combined. Autoregressive models like PixelCNN [58] capture details very well, but cannot be used for latent space analysis. The model of Rezende et al. [45], which is using normalization flow, can be improved by using an inverse autoregressive flow proposed by Kingma et al. [35]. A similar approach was proposed in [9], using an autoregressive flow to learn the prior from some noise. The so called PixelVAE [20] is a model with an autoregressive decoder based on PixelCNN. The work of van den Oord et al. [59] enhances the PixelVAE, which is using a discrete latent space described in more detail in chapter 4.3. One of the latest publications in this direction is the VQ-VAE-2 [44]. It is using different quantized feature sizes and combines them together.

3.2 Medical Applications

Although deep learning for medical imaging is a state of the art research field and has great success in classification tasks [19], the applications of generative models, especially VAE models are still very rare. In [11] different VAE approaches are used for unsupervised anomaly detection in brain magnetic resonance imaging (MRI). For the anomaly detection the model was trained only with healthy data. The inference step is using the same property as denoising autoencoders. An anomaly can be detected by subtracting the reconstructed image from the original one. Baur et al. [4] is proposing a similar approach for anomaly segmentation in Brain MRI by using a spatial VAE. Also Uzunova et al. [56] doing anomaly detection in the similar way with a conditional VAE [32].

Chapter 4

Advanced Generative Models

As discussed in chapter 3, improving the VAE model to generate sharper images is still a state of the art research filed. Many different VAE models have been proposed recently. Therefore only four promising approaches out of the many possibles are depicted. The four selected models are

1. spatial variational autoencoder (SVAE)
2. variational perceptual generative autoencoder (VPGA)
3. vector quantized variational autoencoder (VQ-VAE)
4. introspective variational autoencoder (IntroVAE)

In the following the enhanced models are depicted theoretically and reasons that speak for these models are identified . In chapter 6 practical experiments amend the theoretical discussion.

4.1 Spatial Variational Autoencoder (SVAE)

One disadvantage of the standard VAE is that spatial information can only convey implicitly when coupled with a powerful decoder. This explains the blurry results when handling complex datasets. One trivial solution would be to enlarge the model to handle spatial information. In [60] Wang et al proposed the so called SVAE that uses $d \times d$ ($d > 1$) feature maps as latent representations, which are generated from matrix-variate normal (MVN) distributions, whose parameters are computed from the encoder network similar to the classical VAEs. This approach is selected due to its simplicity. Only the sampling process has to be adapted and the fact that it has already achieved good results in medical applications as proposed in [4].

As introduced in section 2.3.2 the VAE model can be split into an encoder $q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})$ and a decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ part, where $\boldsymbol{x}$ represents the input and $\boldsymbol{z}$ the latent space. In the classical VAE the latent space $\boldsymbol{z}$ can be seen as a vector with a 1×1 feature map. This can lead to some limitations, if spatial information are important as shown in [65]. It exits two methods to overcome this problem. One is to have more complex decoders as proposed in [20], where conditional PixelCNNs [58] are used as decoders. The second

one is to enlarge the latent space $\boldsymbol{z}$ with spatial information by having $d \times d$ ($d > 1$) feature maps instead of 1×1, which is discussed in the following.

The naive way to enlarge the latent space is to reshape the original latent vector $\boldsymbol{z}$ into N feature maps of size $d \times d$. This is problematic since the sampling process does not change the relationship among locations within each feature. Due to this, Wang et al. proposed sampling via MVN distributions instead of a simple normal distribution. The MVN distribution is a generalization of the multivariate normal distribution to matrix-valued random variables. The probability density function of the MVN is depicted in the appendix A.2. The MVN can be related to the multivariate normal distribution for a random matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ in the following way

$$\boldsymbol{X} \sim \mathcal{N}_{m,n}(\boldsymbol{M}, \boldsymbol{\Omega}, \boldsymbol{\Psi}), \tag{4.1}$$

if and only if $\text{vec}(\boldsymbol{X}) \sim \mathcal{N}(\text{vec}(\boldsymbol{M}), \boldsymbol{\Omega} \otimes \boldsymbol{\Psi})$. Here, $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ is the mean matrix, $\boldsymbol{\Omega} \in \mathbb{R}^{m \times m}$, $\boldsymbol{\Psi} \in \mathbb{R}^{n \times n}$ are diagonal matrices indicating the variance, $\otimes$ denotes the Kronecker product and $\text{vec}(\cdot)$ denotes transforming a $\mathbb{R}^{m \times n}$ matrix into an $\mathbb{R}^{(m*n) \times 1}$ vector by concatenating the columns. In spatial VAEs N feature maps of size $d \times d$ can be sampled from N independent MVN distributions:

$$F_k \sim \mathcal{N}_{d,d}(M_{k\Phi}(\boldsymbol{x}), \Omega_{k\Phi}(\boldsymbol{x}) \otimes \Psi_{k\Phi}(\boldsymbol{x})) \tag{4.2}$$

with $k = 0, \ldots, N$ and $M_{k\Phi}(\boldsymbol{x})$, $\Omega_{k\Phi}(\boldsymbol{x})$ and $\Psi_{k\Phi}(\boldsymbol{x})$ are computed through the encoder. Additionally $\boldsymbol{x}$ is the input and $\boldsymbol{\Phi}$ are the learned weights of the encoder. The sampling process with the reparameterization trick from equation 2.30 changes to

$$\boldsymbol{z}_k = M_{k\Phi}(\boldsymbol{x}) + \text{diag}(\Omega_{k\Phi}(\boldsymbol{x}) \otimes \Psi_{k\Phi}(\boldsymbol{x}))^{\frac{1}{2}} * \boldsymbol{\varepsilon}_k, \tag{4.3}$$

where $\boldsymbol{\varepsilon}_k \sim \mathcal{N}(0, \boldsymbol{I})$. The shape of $\boldsymbol{z}_k$ is $\boldsymbol{z}_k \in \mathbb{R}^{d \times d}$. The required number of outputs from the encoder are reduced compared to the naive approach to $(d^2 + 2d)N$. The required number of outputs for the naive approach are $2d^2N$. Therefore, spatial VAEs via MVN distributions leads to a simpler model while adding structural ties among locations. To further reduce the number of needed parameters the low-rank formulation of the MVN distributions is introduced. The low-rank formulation of a MVN distribution is defined as $\mathcal{N}_{d,d}(\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\Omega} \otimes \boldsymbol{\Psi})$ where the mean matrix M is computed by the out-product $\boldsymbol{\mu}\boldsymbol{\nu}^T$ with $\boldsymbol{\mu} \in \mathbb{R}^m$ and $\boldsymbol{\nu} \in \mathbb{R}^n$. Resulting in the final spatial VAE model with low-rank MVN distributions. Equation 4.2 can be reformulated to

$$F_k \sim \mathcal{N}_{d,d}(\underbrace{\boldsymbol{\mu}_{k\Phi}(\boldsymbol{x}), \boldsymbol{\nu}_{k\Phi}(\boldsymbol{x})}_{\boldsymbol{M}_{k\Phi}(\boldsymbol{x})}, \boldsymbol{\Omega}_{k\Phi}(\boldsymbol{x}) \otimes \boldsymbol{\Psi}_{k\Phi}(\boldsymbol{x})). \tag{4.4}$$

where $\boldsymbol{\mu}_{k\Phi}(\boldsymbol{x})$ and $\boldsymbol{\nu}_{k\Phi}(\boldsymbol{x})$ are d-dimensional vectors. The number of outputs for the encoder is further reduced to $4dN$. The sampling process from equation 4.3 is adapted to

$$\boldsymbol{z}_k = \boldsymbol{\mu}_{k\Phi}(\boldsymbol{x})\boldsymbol{\nu}_{k\Phi}^T(\boldsymbol{x}) + \text{diag}(\boldsymbol{\Omega}_{k\Phi}(\boldsymbol{x}) \otimes \boldsymbol{\Psi}_{k\Phi}(\boldsymbol{x}))^{\frac{1}{2}} * \boldsymbol{\varepsilon}_k. \tag{4.5}$$

In figure 4.1 the model of spatial VAE with low rank formulation is depicted. As shown in [60] the spatial VAE produce sharper images on the CelebA, CIFAR-10 and MNIST dataset compared to the classical VAE. This will be evaluated on the used knee osteoarthritis data set in the experimental chapter 6.2.

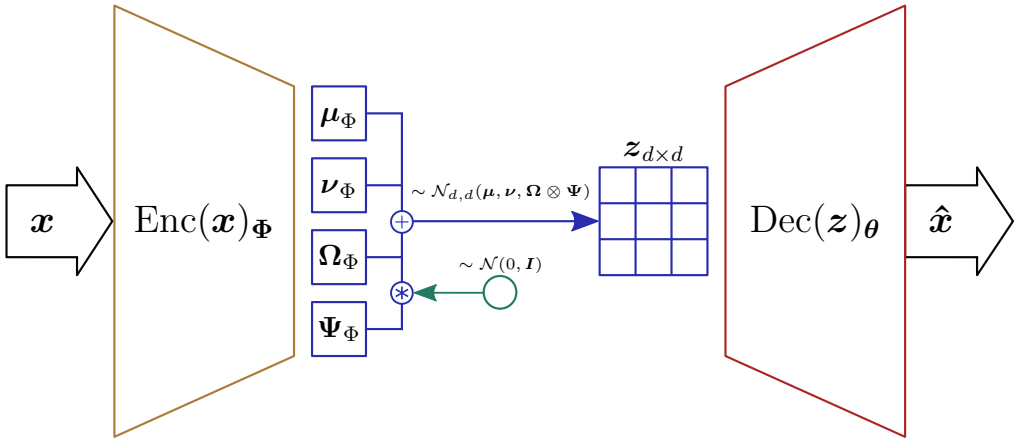


Figure 4.1: Model of the SVAE with low rank formulation of the MVN distribution.

4.2 Variational Perceptual Generative Autoencoder (VPGA)

In [63] Zhang et al. proposed a model where instead of matching the target distribution in data space, the target distribution is matched in latent space. The intuition behind this approach is to train the model reversible, because the data usually lie on a low dimensional manifold. In [12] is shown that the discrepancy between the latent space dimension and the input space dimension can lead to difficulties in training a VAE model. Furthermore, Zhang et al. proves, under mild assumptions, minimizing a latent reconstruction error, matching the target distribution in latent space implies matching the target distribution in data space.

Again let $\text{Enc}_{\boldsymbol{\Phi}}(\boldsymbol{x}) = q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})$ be the encoder, which maps the data distribution $\mathcal{D}$ to the latent space $\mathcal{H}$: $\mathbb{R}^D \to \mathbb{R}^H$ and $\text{Dec}_{\boldsymbol{\theta}}(\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ the decoder, which maps $\mathbb{R}^H \to \mathbb{R}^D$. Also the reconstruction loss is defined as in equation 2.27:

$$L_{\text{rec}} = \mathbb{E}_{z\sim q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[||\hat{\boldsymbol{x}} - \boldsymbol{x}||_2^2] \tag{4.6}$$

where $\hat{\boldsymbol{x}} = \text{Dec}(\text{Enc}(\boldsymbol{x}))$. Due to the difficulty to directly match $\hat{\mathcal{D}}$, which denotes the distribution of $\hat{\boldsymbol{x}}$, Zhang et al. propose to map $\hat{\mathcal{D}}$ to the latent space and match the mapped distribution $\hat{\mathcal{H}}$ in latent space. Therefore, the encoder is used to map $\hat{\mathcal{D}}$ to $\hat{\mathcal{H}}$, whereby $\hat{\mathcal{H}}$ should still be mapped to $\mathcal{N}(0, \boldsymbol{I})$. This is realized with the tow losses

$$L_{\text{lr},\mathcal{N}} = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\boldsymbol{I})}[||h(\boldsymbol{z}) - \boldsymbol{z}||_2^2], \tag{4.7}$$

with $h(\cdot) = \text{Enc}(\text{Dec}(\cdot))$ and

$$L_{\text{lr},\mathcal{H}} = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim\mathcal{H}(0,\boldsymbol{I})}[||h(\boldsymbol{z}) - \boldsymbol{z}||_2^2], \tag{4.8}$$

where $h(\boldsymbol{z}) \sim \hat{\mathcal{H}}$. Finally the reconstruction loss of the VAE loss from 2.27 is adapted to map the latent space, here denoted as L_{vrec}. The VAE loss of the VPGA model results to

$$L_{\text{pvae}} = L_{\text{vrec}} + L_{\text{kl}} = \underbrace{-\mathbb{E}_{z\sim q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{z}}|\boldsymbol{z})]}_{L_{\text{vrec}}} + \underbrace{\mathbb{KL}(q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))}_{L_{\text{kl}}} \tag{4.9}$$

Hence, the total loss of the VPGA model is defined as

$$L_{\text{vpga}} = L_{\text{rec}} + \alpha L_{\text{lr},\mathcal{N}} + \beta L_{\text{lr},\mathcal{H}} + \eta[L_{\text{vrec}} + L_{\text{kl}}], \tag{4.10}$$

whereby α, β and η are hyperparameters. The complete VPGA model with its different losses is drawn in figure 4.2. It can be observed that $L_{\text{lr,H}}$ and L_{vrec} are very similar. For L_{vr} some random noise is added. For simplification $L_{\text{lr,H}}$ can be negligible.

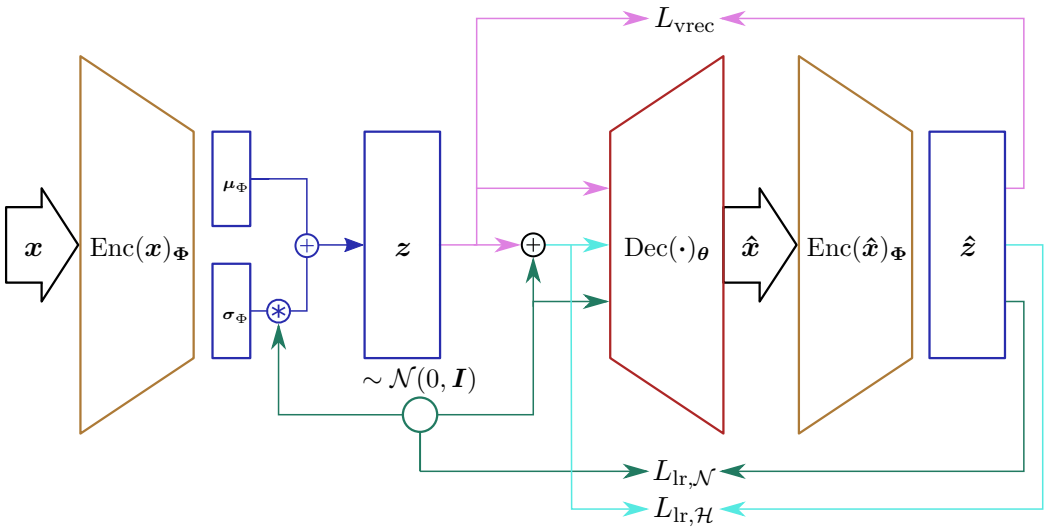


Figure 4.2: The VPGA model with the additional losses from equation 4.9. For simplicity, the losses L_{kl} and L_{rec} are not depicted.

4.3 Vector Quantized Variational Autoencoder (VQ-VAE)

Aaron et al. propose an enhancement to the standard VAE model, which is named VQ-VAE [59]. The novel model is using discrete latent space representations instead of continuous representation like the standard VAE. The VQ-VAE has no variance issue and prevents the model from posterior collapse, which leads to sharper reconstruction images. Furthermore, Aaron et al. is training the prior distribution over the discrete latent space $p(\boldsymbol{z})$ with an autoregressive model to generate high resolution images. Using a discrete representation seems reasonable, because many natural problems can be described as discrete. An example is language which is normally represented as a sequence of discrete symbols or images which can be described by language. This approach seems promising because it shows already very sharp reconstruction results on high resolution images in a different domain. For the VQ-VAE model the posterior $p(\boldsymbol{x}|\boldsymbol{z})$ is categorical and the prior $p(\boldsymbol{z})$ is uniform distributed. Therefore, the encoder is modeling a categorical distribution, which results in integral values. The integral numbers are used to index a dictionary of embeddings. Finally, the indexed values are passed through the decoder in order to reproduce the input. The indexing is done via vector quantization (VQ), which has its origin in signal processing and was originally used for data compression. The quantization output is an index value, which indicates the closest vector form a finite set of vectors, which are called embeddings or codebook. In figure 4.3 a simple example illustrates how VQ is working. It should be emphasized that the values of the codebook are learned during the training process. VQ can be seen as a clustering algorithm like k-means.

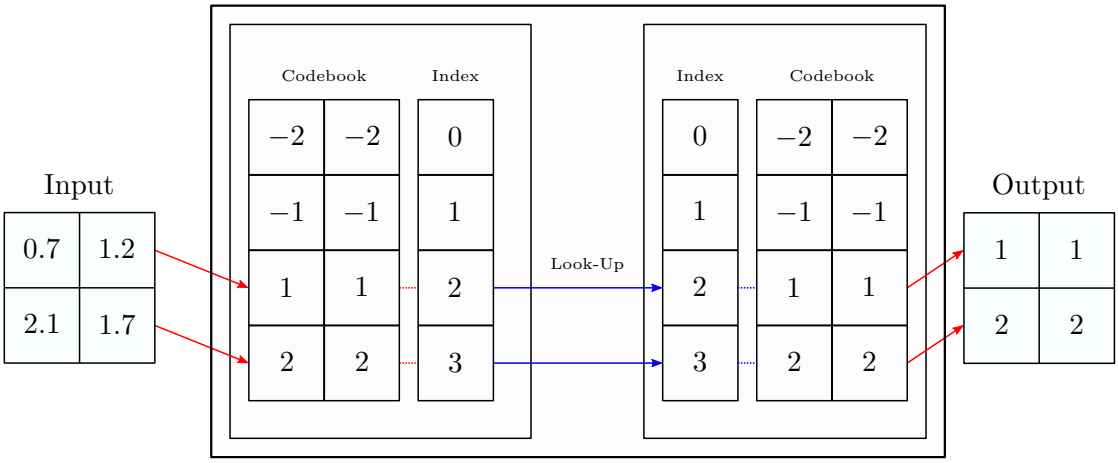


Figure 4.3: VQ example: The Input is quantized by finding the nearest neighbor in codebook.

In Figure 4.4 the model of a VQ-VAE is depicted. The embedding space is defined as $e \in \mathbb{R}^{K \times D}$ where K is the dimension of the discrete latent space and D is the dimensionality of each latent embedding vector. The encoder takes an input image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ where H is the height, W is the width, C the number of channels of the image and produce the output $z_e \in \mathbb{R}^{H^* \times W^* \times D}$. Note that the last dimension of z_e must be equal to the dimensionality of the latent embedding vector $e_i \in \mathbb{R}^D$ with $i \in 1, 2, \cdots, K$. By doing a nearest neighbor look-up in the VQ layer using the shared embedding space $\boldsymbol{e}$, $\boldsymbol{z}_e$ is transformed to $\boldsymbol{z}_q$. For this $\boldsymbol{z}_e$ is reshaped to (N, K) with $N = H^* * W^*$ and the nearest neighbor look-up is performed for each of the N vectors. Before passing the values to the decoder it is reshaped back again to $\boldsymbol{z}_q \in \mathbb{R}^{H^* \times W^* \times D}$. The nearest neighbor look-up is defined as

$$\boldsymbol{z}_q(\boldsymbol{x}) = \boldsymbol{e}_k, \text{ where } k = \arg\min_j ||\boldsymbol{z}_e(\boldsymbol{x}) - \boldsymbol{e}_j||_2, \tag{4.11}$$

which means k is the index of the nearest embedding vector $\boldsymbol{e}_k$ to $\boldsymbol{z}_e$. Due to the fact, that there is no real gradient for equation 4.11 the gradients from the decoder input $\boldsymbol{z}_q(\boldsymbol{x})$ are copied to the encoder output $\boldsymbol{z}_e(\boldsymbol{x})$. The total loss of the VQ-VAE model is defined in equation 4.12. It is composed of three terms. The first term is the reconstruction loss. The second term is used to move the embedding vectors $\boldsymbol{e}_i$ towards the encoder outputs $\boldsymbol{z}_e(\boldsymbol{x})$. To regularize the growing of the embedding space the third term is introduced. For the no gradient operation the symbol ng$[\cdot]$ is used.

$$L_{\text{VQ-VAE}} = \mathbb{E}_{z \sim q_{\boldsymbol{\Phi}(\boldsymbol{z}|\boldsymbol{x})}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z_q})] + ||\text{ng}[\boldsymbol{z}_e(\boldsymbol{x})] - \boldsymbol{e}||_2^2 + \beta||\boldsymbol{z}_e(\boldsymbol{x}) - \text{ng}[\boldsymbol{e}]||_2^2 \tag{4.12}$$

It should be noted that there is no $\mathbb{KL}$ term in the loss function. This is because of the fact that the model is deterministic with a uniform prior, which leads to a constant $\mathbb{KL}$ divergence equal to $\log(K)$. In the proposed work from Aaron et al. an autoregressive model is trained to further optimize the prior for the generative task. This is not done in here, because in this work the focus is in the latent space and the reconstruction quality of the model.

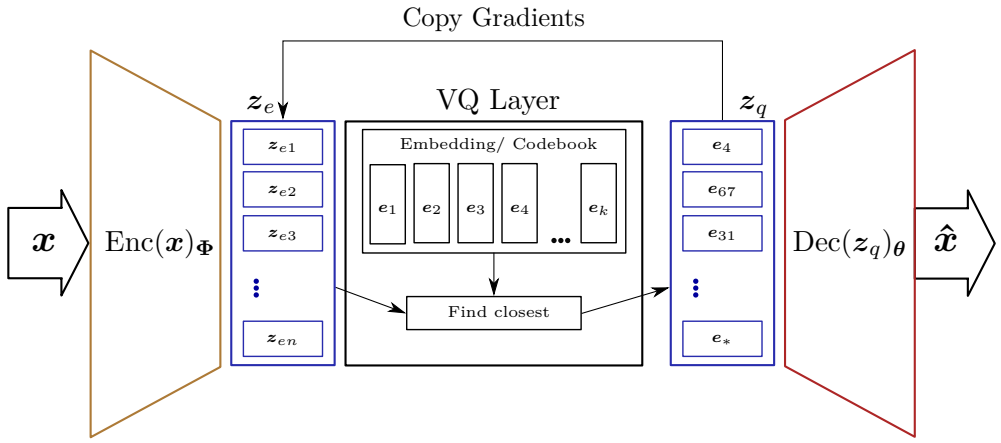


Figure 4.4: The VQ-VAE model with the use of the VQ Layer for quantization.

4.4 Introspective Variational Autoencoder (IntroVAE)

Huang et al. propose a novel generative model for high resolution images, where the advantages of GANs and the advantages of VAEs are combined with an introspective manner. The model is called introspective variational autoencoder (IntroVAE) [24]. Compared to other hybrid models of VAEs and GANs [37],[40], [16], [15] or [51] the IntroVAE can be trained in a single stage, because it doesn't require an extra discriminator. Therefore, the model is prevented from mode collapse and more stable to train. Furthermore, due to the use of an VAE the latent space of the input image can be analyzed to a certain degree as it will be shown in the experiment chapter 6.5. Huang et al. shows that the IntroVAE model is able to generate high-resolution images e.g. for the CelebA-HQ dataset with a resolution of 1024×1024.

The IntroVAE model can distinguish between generated samples and real data by splitting the model into two parts. A discriminator, the encoder $\text{Enc}_{\boldsymbol{\Phi}}(\boldsymbol{x}) = q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})$ of the VAE model and a generator, the decoder $\text{Dec}_{\boldsymbol{\theta}}(\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. These two parts play a min-max game analogues a normal GAN model. The approach use the KL divergence $L_{\text{KL}}(\boldsymbol{x}) = \mathbb{KL}(q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$ to match the distribution of the generated samples with the true distribution. The encoder is trained to minimize the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ of the real data $\boldsymbol{x}$ with the prior $p(\boldsymbol{z})$ and simultaneously to maximize the deviation from the generated samples $G(\boldsymbol{z}')$ of the posterior $q_{\boldsymbol{\Phi}}(\boldsymbol{z}|G(\boldsymbol{z}'))$ from the prior $p(\boldsymbol{z})$. Whereby $\boldsymbol{z}'$ is sampled from $p(\boldsymbol{z})$. The generator L_G is trained reverse, it produce samples $G(\boldsymbol{z}')$ that minimize the posterior $q_{\boldsymbol{\Phi}}(\boldsymbol{z}|G(\boldsymbol{z}'))$ from the prior $p(\boldsymbol{z})$. The losses for the encoder model E and for the Generator model G can be defined as following:

$$L_E(\boldsymbol{x}, \boldsymbol{z}) = L_{\text{KL}}(\boldsymbol{x}) + [m - L_{\text{KL}}(G(\boldsymbol{z}'))]^+ \tag{4.13a}$$

$$L_G(\boldsymbol{z}) = L_{\text{KL}}(G(\boldsymbol{z}')) \tag{4.13b}$$

where $[\cdot]^+ = \max(0, \cdot)$ and m is a positive margin. Huang et al. is further proposing, for an introspective training, to add the reconstruction loss $L_{\text{rec}}(\boldsymbol{x}) = -\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$ from equation 2.27 to the Encoder and the Generator to prevent mode collapse. Hence, the two losses from before can be enhanced to:

$$L_E(\boldsymbol{x}, \boldsymbol{z}) = L_{\text{KL}}(\boldsymbol{x}) + [m - L_{\text{KL}}(G(\boldsymbol{z}'))]^+ + L_{\text{rec}}(\boldsymbol{x}) \tag{4.14a}$$

$$L_G(\boldsymbol{z}) = L_{\text{KL}}(G(\boldsymbol{z}')) + L_{\text{rec}}(\boldsymbol{x}) \tag{4.14b}$$

Now all terms can be plugged in. For the prior $p(\boldsymbol{z})$ a centered isotropic multivariate Gaussian as for the standard VAE is used, therefore, the loss function $L_{\mathrm{KL}}(\cdot)$ as defined in equation 2.29 is used. For the samples $\boldsymbol{z}'$, reconstruction samples $\boldsymbol{z}_r$ and new random generated samples $\boldsymbol{z}_p$ are used. Further, the hyperparameters α and β are introduced. The total loss functions L_E and L_G redefined to:

$$L_E(\boldsymbol{x}) = L_{\mathrm{KL}}(\mathrm{Enc}(\boldsymbol{x})) + \alpha \sum_{s=r,p} [m - L_{\mathrm{KL}}(\mathrm{Enc}(\mathrm{ng}(\boldsymbol{x}_s)))]^+ + \beta L_{\mathrm{rec}}(\boldsymbol{x}) \tag{4.15a}$$

$$L_G(\boldsymbol{x}) = \alpha \sum_{s=r,p} L_{\mathrm{KL}}(\mathrm{Enc}(\boldsymbol{x}_s)) + \beta L_{\mathrm{rec}}(\boldsymbol{x}) \tag{4.15b}$$

where $\mathrm{ng}(\cdot)$ indicates no gradient calculated for the back propagation. The optimization algorithm for the IntroVAE is depicted in algorithm 4.1. Additionally the model of the IntroVAE is represented in figure 4.5.

Algorithm 4.1 : Training IntroVAE model. Adapted from [24]

Data : $\boldsymbol{X}$ images
$\boldsymbol{\Phi}, \boldsymbol{\theta} \leftarrow$ Initialize the weights of the encoder and decoder;
while *not converged* **do**
  $\boldsymbol{x} \leftarrow$Take random mini-batch from $\boldsymbol{X}$;
  $\boldsymbol{z} \leftarrow \mathrm{Enc}(\boldsymbol{x})$;
  $\boldsymbol{z_p} \leftarrow$ Samples from prior $\mathcal{N}(0, \boldsymbol{I})$;
  $\boldsymbol{x_r} \leftarrow \mathrm{Dec}(\boldsymbol{z})$;
  $\boldsymbol{x_p} \leftarrow \mathrm{Dec}(\boldsymbol{z_p})$;
  $L_{\mathrm{rec}} \leftarrow L_{\mathrm{rec}}(\boldsymbol{x_r}, \boldsymbol{x})$;
  $L_{E(\mathrm{reg})} \leftarrow L_{\mathrm{KL}}(\mathrm{Enc}(\boldsymbol{x})) + \alpha[m - L_{\mathrm{KL}}(\mathrm{Enc}(\mathrm{ng}(\boldsymbol{x}_r)))]^+ + [m - L_{\mathrm{KL}}(\mathrm{Enc}(\mathrm{ng}(\boldsymbol{x}_p)))]^+$;
  $L_E \leftarrow L_{E(\mathrm{reg})} + \beta L_{\mathrm{rec}}$;
  $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \eta \nabla_{\boldsymbol{\Phi}} L_E$ ▷ `Perform gradient step for` Φ
  $L_{G(\mathrm{reg})} \leftarrow \alpha[L_{\mathrm{KL}}(\mathrm{Enc}(\boldsymbol{x}_r)) + L_{\mathrm{KL}}(\mathrm{Enc}(\boldsymbol{x}_p))]$;
  $L_G \leftarrow L_{G(\mathrm{reg})} + \beta L_{\mathrm{rec}}$;
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} L_G$ ▷ `Perform gradient step for` θ

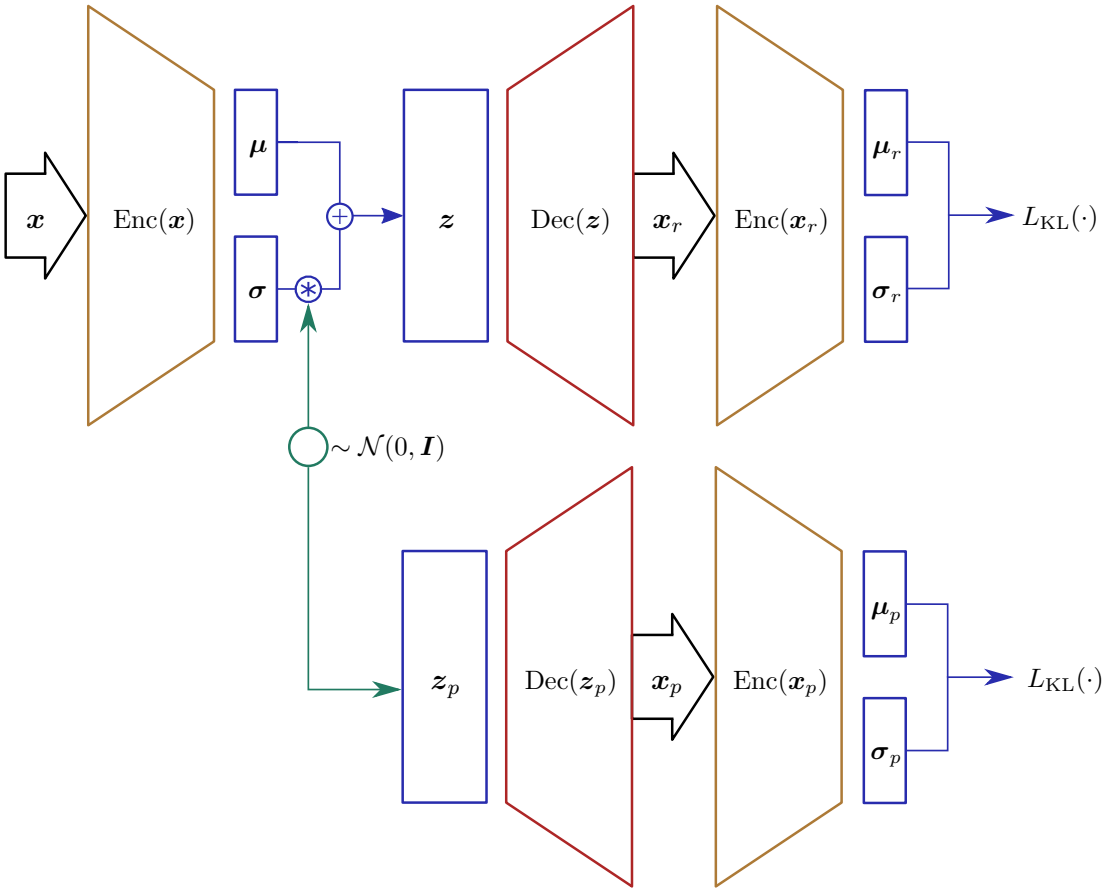


Figure 4.5: The IntroVAE model with the additional losses indicated.

Chapter 5

Method

5.1 Hardware and Software

The models are trained on a GeForce RTX 2080 Ti with 11 GB video ram. The models are implemented in python using the machine learning frameworks TensorFlow and PyTorch.

5.2 Datasets

At the beginning the CT dataset DeepLesion was used [61]. It turned out that due to the high complexity of the dataset and the large scaled images, it was hard to evaluate the models with different parameters efficiently and fast. Therefore the more handy X-Ray knee osteoarthritis severity grading dataset was used [1].

5.2.1 Knee Osteoarthritis Severity Grading Dataset

The dataset consists of knee X-ray data with the corresponding semi-quantitative knee KL grading [30]. The KL grading has five categories: KL-0 (no OA changes), KL-1 (Doubtful OA), KL-2 (Early OA changes), KL-3 (Moderate OA) and KL-4 (End-stage OA). The dataset has 9,786 images which are divided into train and test set with the corresponding KL class as shown in table 5.1. It is noticeable that the five KL classes are not unifrom distributed. Only three percent of all samples are from KL class 4, whereas 39 percent of the samples are labeled as KL class 0.

Group	Images	KL-0	KL-1	KL-2	KL-3	KL-4
Train	8,960	3,529	1,617	2,366	1,180	268
Test	826	328	153	212	106	27

Table 5.1: The number provided in the table indicate the number of knees used in each group

[1] `https://oai.epi-ucsf.org/datarelease/`

5.3 Standard Architecture

To make the different models comparable, the same encoder and decoder architecture as described in table 5.2 for the standard VAE model are used. Only the latent space is adapted depending on the model. This is highlighted in the corresponding experimental section.

Encoder		
Type	**Parameters** [#filters, kernel, stride]	**Shape**
Input	-	$256 \times 256 \times 1$
Conv.	$[8, 4 \times 4, 2]$	$128 \times 128 \times 8$
Res-block	$\begin{bmatrix} 16, & 1 \times 1, & 2 \\ 16, & 4 \times 4, & 1 \\ 16, & 4 \times 4, & 2 \end{bmatrix}$	$64 \times 64 \times 16$
Res-block	$\begin{bmatrix} 32, & 1 \times 1, & 2 \\ 32, & 4 \times 4, & 1 \\ 32, & 4 \times 4, & 2 \end{bmatrix}$	$32 \times 32 \times 32$
Res-block	$\begin{bmatrix} 64, & 1 \times 1, & 2 \\ 64, & 4 \times 4, & 1 \\ 64, & 4 \times 4, & 2 \end{bmatrix}$	$16 \times 16 \times 64$
Res-block	$\begin{bmatrix} 128, & 1 \times 1, & 2 \\ 128, & 4 \times 4, & 1 \\ 128, & 4 \times 4, & 2 \end{bmatrix}$	$8 \times 8 \times 128$
Res-block	$\begin{bmatrix} 256, & 1 \times 1, & 2 \\ 256, & 4 \times 4, & 1 \\ 256, & 4 \times 4, & 2 \end{bmatrix}$	$4 \times 4 \times 256$
FC-100	units = 100	$100 \times 1 \times 1$
Split	-	50,50

Decoder		
Type	**Parameters** [#filters, kernel, stride]	**Output**
Input	-	$50 \times 1 \times 1$
FC-$(4 * 4 * 256)$	Reshape	$4 \times 4 \times 256$
Res-block Up	$\begin{bmatrix} 128, & 1 \times 1, & 2 \\ 128, & 4 \times 4, & 1 \\ 128, & 4 \times 4, & 2 \end{bmatrix}$	$8 \times 8 \times 128$
Res-block Up	$\begin{bmatrix} 64, & 1 \times 1, & 2 \\ 64, & 4 \times 4, & 1 \\ 64, & 4 \times 4, & 2 \end{bmatrix}$	$16 \times 16 \times 64$
Res-block Up	$\begin{bmatrix} 32, & 1 \times 1, & 2 \\ 32, & 4 \times 4, & 1 \\ 32, & 4 \times 4, & 2 \end{bmatrix}$	$32 \times 32 \times 32$
Res-block Up	$\begin{bmatrix} 16, & 1 \times 1, & 2 \\ 16, & 4 \times 4, & 1 \\ 16, & 4 \times 4, & 2 \end{bmatrix}$	$64 \times 64 \times 16$
Res-block Up	$\begin{bmatrix} 8, & 1 \times 1, & 2 \\ 8, & 4 \times 4, & 1 \\ 8, & 4 \times 4, & 2 \end{bmatrix}$	$128 \times 128 \times 8$
ConvUp	$[8, 4 \times 4, 2]$	$256 \times 256 \times 4$
Conv	$[1, 1 \times 1, 1]$	$256 \times 256 \times 1$

Table 5.2: Architecture of the encoder (left table) and decoder(right table) for the standard VAE model.

5.4 Evaluation Scores

The dataset is split into training and validation set. The models are only trained on the training set. For qualitatively analyses, reconstructed images are plotted from the training and validation set. Furthermore random image samples will be generated from each model. To analyse the different models quantitative the Mean Square Error (MSE), the MS-SSIM and the FID are used.

5.4.1 Mean Squared Error (MSE)

The MSE is defined as

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)^2 = \frac{1}{n}||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2, \tag{5.1}$$

where $\boldsymbol{x}$ is the input image and $\hat{\boldsymbol{x}}$ is the reconstructed image. N defines the number of pixels. The MSE is a standard metric which measure the average of the squared errors. It is simple and fast to calculate and has a clear physical meaning. One disadvantage of the MSE is that it does not include the human visual perception of images [66]. Therefore Wang et al. proposed a measurement metric called multi-scale structural similarity [67].

5.4.2 Multi-Scale Structural Similarity (MS-SSIM)

The perception of humans is highly adapted for extracting structural information from an image. Therefore Wang et al. established the multi-scale structural similarity (MS-SSIM) method, which measures the structural similarity. It is based on the single-scale approach structural Similarity (SSIM) [66]. Let $\boldsymbol{x}$ be the input image, $\hat{\boldsymbol{x}}$ the reconstructed image, μ the mean, σ^2 the variance of $\boldsymbol{x}$, $\hat{\boldsymbol{x}}$) respectively and $\sigma_{x\hat{x}}$ the covariance between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$. The SSIM is defined as following:

$$\text{SSIM}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = l(\boldsymbol{x}, \hat{\boldsymbol{x}}) \cdot c(\boldsymbol{x}, \hat{\boldsymbol{x}}) \cdot s(\boldsymbol{x}, \hat{\boldsymbol{x}}), \tag{5.2}$$

where the luminance l, the contrast c and structure comparison measures s are given in equation 5.3.

$$\begin{aligned} l(\boldsymbol{x}, \hat{\boldsymbol{x}}) &= \frac{2\mu_x\mu_{\hat{x}} + C_1}{\mu_x^2 + \mu_{\hat{x}}^2 + C_1} \\ c(\boldsymbol{x}, \hat{\boldsymbol{x}}) &= \frac{2\sigma_x\sigma_{\hat{x}} + C_2}{\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2} \\ s(\boldsymbol{x}, \hat{\boldsymbol{x}}) &= \frac{\sigma_{x\hat{x}} + C_3}{\sigma_x\sigma_{\hat{x}} + C_3} \end{aligned} \tag{5.3}$$

C_1, C_2 and C_3 small constants. The SSIM is symmetric $\text{SSIM}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \text{SSIM}(\hat{\boldsymbol{x}}, \boldsymbol{x})$, bounded $\hat{\boldsymbol{x}} \leq 1$ and has a unique maximum at 1 if and only if $\boldsymbol{x} = \hat{\boldsymbol{x}}$. The final SSIM score is calculated by convolving a window over the images and calculating the SSIM for each local window. Finally, the mean of the local SSIMs is used. The MS-SSIM extends the SSIM by calculating the SSIM at multiply resolution. Therefore, the images is down sampled by a scaling factor of 2. The original image has scaling index 1 and the highest scaling index is defined as M. At each scale the contrast $c(\cdot)$ and the structure comparison $s(\cdot)$ are calculated. The luminance $l(\cdot)$ is only computed at scale level M. Hence, the MS-SSIM is

$$\text{MS-SSIM}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = [l_M(\boldsymbol{x}, \hat{\boldsymbol{x}})]^{\alpha_M} \cdot \prod_{j=i}^{M} [c_j(\boldsymbol{x}, \hat{\boldsymbol{x}})]^{\beta_j} [s_j(\boldsymbol{x}, \hat{\boldsymbol{x}})]^{\gamma_j}. \tag{5.4}$$

The exponents α_M, β_j and γ_j are hyper parameters which are used to adjust the relative importance for each component. Again the MS-SSIM is symmetric, bounded and has a unique maximum as defined for the SSIM before.

5.4.3 Fréchet Inception Distance (FID)

To measure the similarity of random generated images compared to the original images Heusel et al. proposed the Fréchet Inception Distance (FID) [23]. It is calculated by

measuring the difference of two multivariate Gaussians via the Fréchet distance [14]. The FID is calculated as:

$$\mathrm{FID}^2((\boldsymbol{\mu}_{org}, \boldsymbol{\Sigma}_{org}), (\boldsymbol{\mu}_{gen}, \boldsymbol{\Sigma}_{gen})) = ||\boldsymbol{\mu}_{org} - \boldsymbol{\mu}_{gen}||_2^2 + \mathrm{Tr}(\boldsymbol{\Sigma}_{org} + \boldsymbol{\Sigma}_{gen} - 2\sqrt{\boldsymbol{\Sigma}_{org}\boldsymbol{\Sigma}_{gen}}), \quad (5.5)$$

where $(\boldsymbol{\mu}_{org}, \boldsymbol{\Sigma}_{org})$ are the mean and covariance matrix produced from the original images. The $(\boldsymbol{\mu}_{gen}, \boldsymbol{\Sigma}_{gen})$ are the mean and covariance matrix produced from the generated images by using the pre-trained Inception model [52]. The mean and covariance matrix is calculated from the output of the last pooling layer of the Inception model[2]. The FID score for the test data set compared to the training data set is $\mathrm{FID}_{Testset} = 8.52$.

[2]The pre trained model can be downloaded from `http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz`.

Chapter 6

Experiments and Results

Experiments have been implemented and evaluated on the following five different models using the architecture described in chapter 5.3 :

- Variational autoencoder (VAE)
- spatial variational autoencoder (SVAE)
- variational perceptual generative autoencoder (VPGA)
- vector quantized variational autoencoder (VQ-VAE)
- introspective variational autoencoder (IntroVAE)

Furthermore, different training settings are used to show limitations or strengths of a specific model. All experiments have been performed four times to get an estimation of the variance and stability of the training process. To generate random samples for each model, the values for the latent space $\boldsymbol{z}$ have been sampled from a normal distribution $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$. Only the integral values for the latent space of the VQ-VAE are sampled from a uniform distribution $\boldsymbol{z} \sim \mathcal{U}(0, k)$, where k is the number of embedding vectors. Furthermore, the mean of the average learning curves of all experiments are summarized in one plot for each score at the end of this chapter. The detailed learning curves with the standard deviation for every experiment are in the appendix A.3. Reconstruction results of the training data set are also depicted in the appendix for each experiment. The best quantitative values of the scores are summarized at the end of this chapter in table 6.1.

6.1 Variational Autoencoder (VAE)

6.1.1 Standard Architecture VAE_{50}

For this experiment, the standard VAE architecture from table 5.2 is used with the latent space of 50. The experiment is referred as VAE_{50}. The average learning curves of the MSE, MS-SSIM and FID are depicted in figure A.2. It can be seen that over-fitting starts around epoch 120. The standard deviation for the MSE and the MS-SSIM is very small, which indicates a stable trainings process. The FID is decreasing a little bit, but

with the lowest mean score of 317 it is still very high. Figure 6.1 clarifies, that the standard VAE with a latent space of 50 can already reconstruct the general structure of the knee from unseen data. Still the reproduced images are to blurry for medical

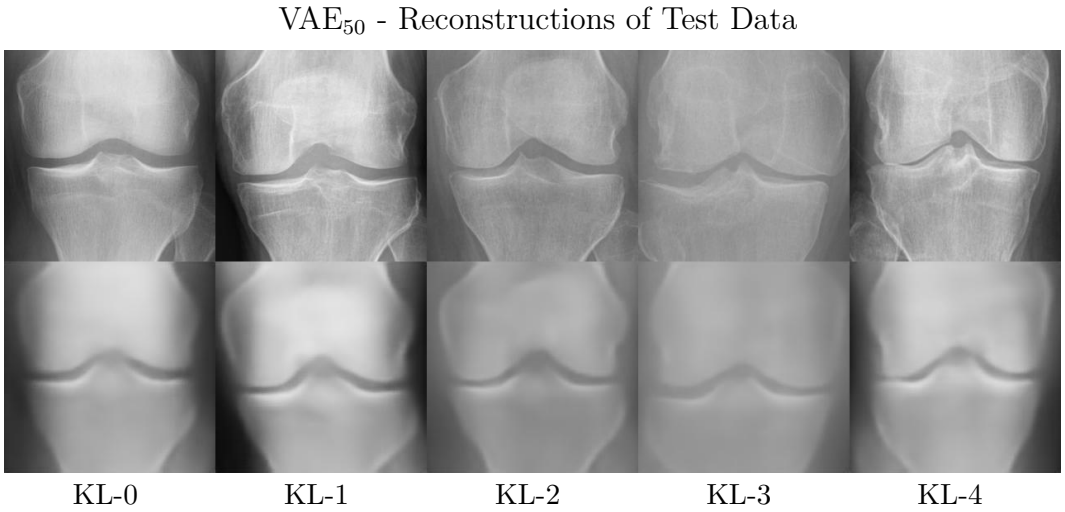


Figure 6.1: Reconstructed images from test dataset with the VAE_{50} model. The first row are the original images with one sample from each KL class (0-4). The second row are the reconstructed images.

diagnosis. The details of the structure are not reproduced. This outcome was expected as already discussed before due to the fact that the VAE model tends to produce blurry images. In the appendix A.1 reconstructed images from the training data set are shown, which demonstrate similar results with small indications of the inner structure of the knee. A trivial solution to increase the sharpness of the image would be to increase the dimension of the latent space. Therefore, an additional experiment with a bigger latent space of 8192 is conducted and described in the next section. The sample quality is very blurry only rough outlines of the knee can bee imagined. This is in line with the high FID score. The random generated samples are shown in figure 6.2. Moreover, the latent

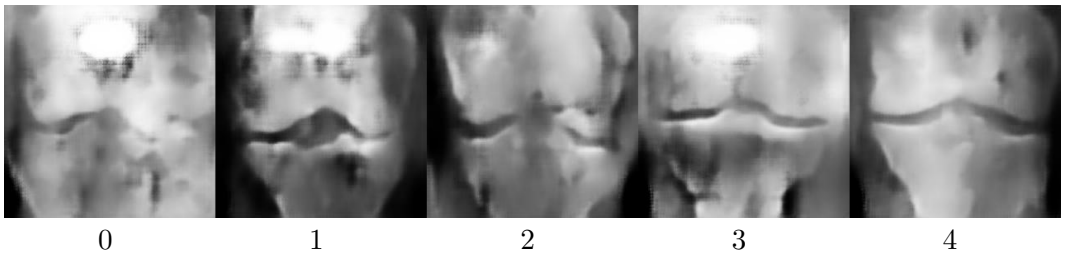


Figure 6.2: Random generated samples with the VAE_{50} model.

space of the VAE_{50} is analyzed. Therefore, a sample from the training set is chosen and each latent variable is traversed from $[-1, 1]$, while leaving the remaining latents constant to its original value. In figure 6.3 four noticeable traversed latent variables are plotted. Regions where some changes indicated are marked with red geometric shapes. It shows, although the changes are small, the model learns some features of the knee and the changes from one traversed value to the next are smooth.

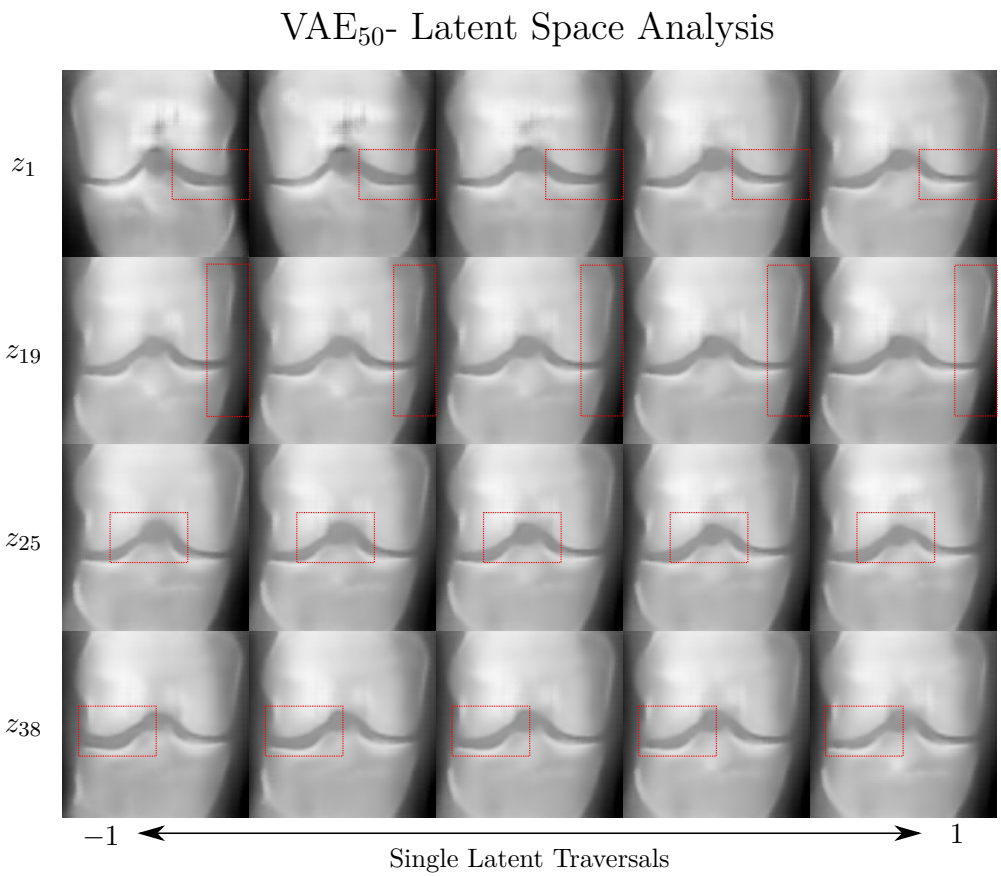
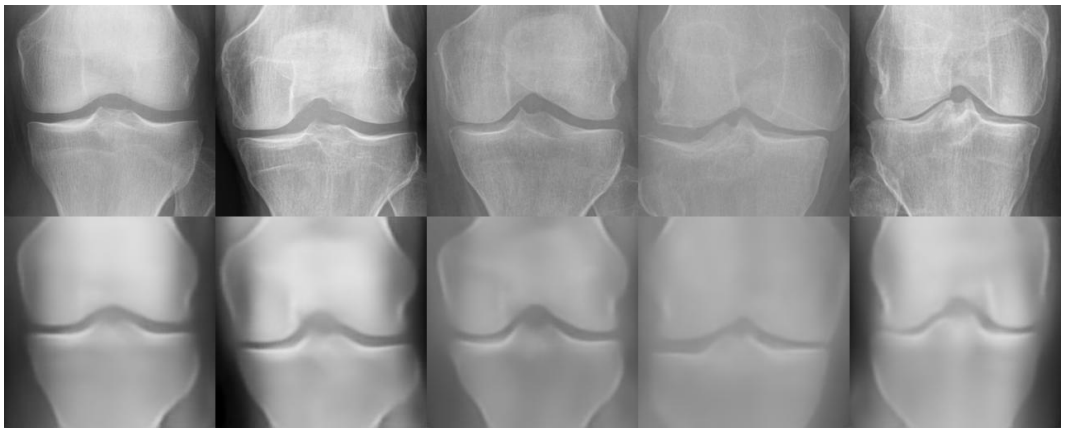


Figure 6.3: Four picked latents from the VAE_{50} model and traversed between $[-1, 1]$. The red geometric shapes indicate regions with changes during the traversals. Each row is a different latent with its value changing, while the other kept constant.

6.1.2 Adapted Architecture VAE_{8192}

The architecture from table 5.2 is adapted by removing the first residual block of the encoder and the last residual block of the decoder. The dimension of the latent space was set to 8192. The MSE decreases and the MS-SSIM increases compared to the VAE_{50} experiment. However, the improvement is not significant. Compared to the much bigger latent space of the VAE_{8192} to the VAE_{50} model the improvement is small. The reproduced images are still blurry as it can be seen from figure 6.4 for the test set and in appendix A.3 for the training set. With the bigger latent space, the quality of random generated images is decreasing strongly. The samples are only a noisy cloud (figure 6.5). Also the high FID from table 6.1 indicates bad sampling quality. The average learning rate doesn't show any conspicuousness compared to the standard VAE_{50} model. For completeness it is depicted in the appendix A.4.

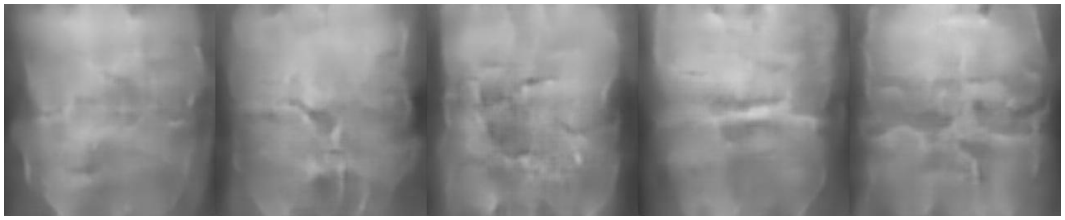
VAE_{8192} - Reconstructions of Test Data



KL-0 KL-1 KL-2 KL-3 KL-4

Figure 6.4: Reconstructed images from test dataset with the VAE_{8192} model. The first row shows original images with two images from each KL class (0-4). The second row shows the corresponding reconstructed images.

VAE_{8192} - Random Generated Samples



0 1 2 3 4

Figure 6.5: Random generated samples with the VAE_{8192} model.

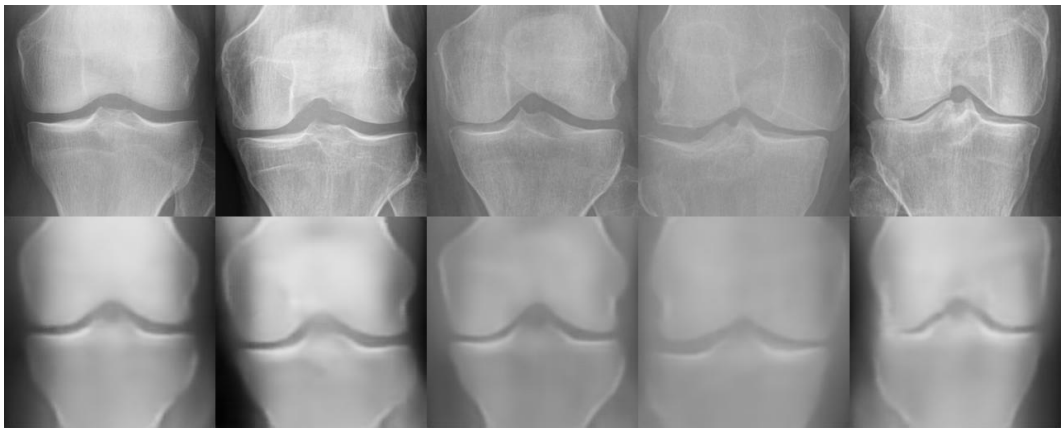
6.2 Spatial Variational Autoencoder (SVAE)

6.2.1 Standard Architecture $SVAE_{3\times3\times9}$

Due to the fact that the spatial VAE is using a feature map $d \times d$ with $d > 1$ the latent space from table 5.2 is adapted. The following experiment is referred to $SVAE_{3\times3\times9}$. For the $SVAE_{3\times3\times9}$ experiment the latent space is set to $N = 9$ feature maps and $d \times d = 3 \times 3$ for the feature size. This results to 108 output parameters for the encoder with the low rank formulation, which are similar to the 100 output parameters from standard architecture. The quantitative results for the MSE, MS-SSIM and FID are similar to the scores from the VAE_{50} model as depicted in table 6.1. Only the MSE of the $SVAE_{3\times3\times9}$ model for the test data set is insignificant better than for the VAE_{50} experiment. The FID score is higher compared to the VAE_{50} experiment. This can be explained by the more complex model architecture. The qualitative reconstruction images show no significant difference to the VAE_{50} model. The reconstruction images for the test dataset are depicted in figure 6.6.

The quality of the random generated samples for this experiment as shown in figure 6.7 is worse than the samples from the VAE_{50} experiment. The knee structure is hard to recognize. The bad sampling property correlates with the higher FID score of 333. The average learning curves are also similar to the learning curves of the VAE_{50} model, which

SVAE$_{3\times3\times9}$ - Reconstructions of Test Data



KL-0 KL-1 KL-2 KL-3 KL-4

Figure 6.6: Reconstructed images from test dataset with the SVAE$_{3\times3\times9}$ model. The first row shows original images with two images from each KL class (0-4). The second row shows the corresponding reconstructed images.

SVAE$_{3\times3\times9}$ - Random Generated Samples

0 1 2 3 4

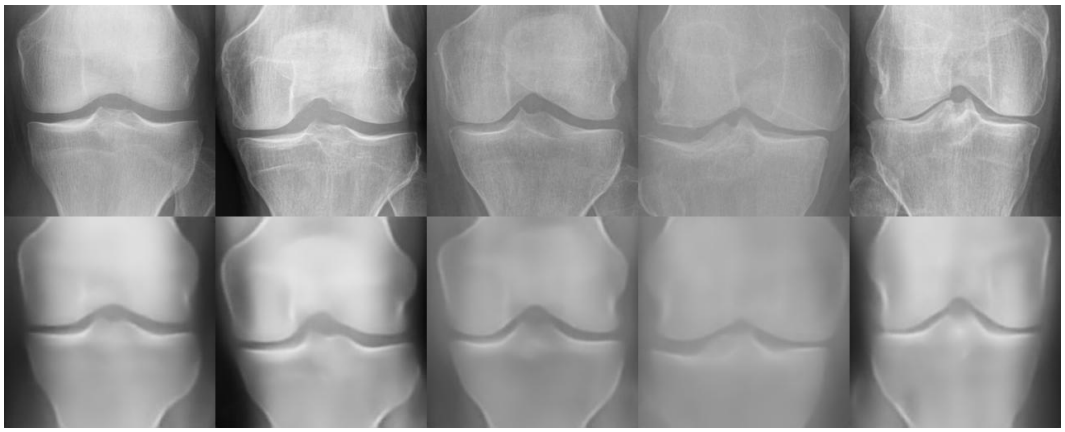
Figure 6.7: Random generated samples with the VAE$_{8192}$ model.

are depicted in the appendix A.6. Also reconstruction samples on the training data set are shown in the appendix A.5.

6.2.2 Adapted Architecture SVAE$_{16\times16\times32}$

In [4] a spatial VAE with a higher dimensional latent space leads to significant improvements of the reconstruction quality. Therefore, an experiment with an adapted architecture is performed. As for the VAE$_{8192}$ the first residual block of the encoder and the last residual block of the decoder are removed. The latent space is set to $N = 32$ feature maps and $d = 16$ feature size. Indeed the reconstruction quality improves. The lowest MSE and the highest MS-SSIM perform better than the scores from the VAE experiments and the SVAE$_{3\times3\times9}$ experiment (6.1). The reconstructed images are a little bit sharper than the images from the previous experiments as shown in figure 6.8. Nevertheless, the reconstruction images still look blurry and details of the knee structure are missing. The generated samples are a noisy cloud depicted in figure 6.9, no knee structure can be detected at all. The learning curves show no abnormalities compared to the other experiments. The curves are shown in the appendix A.8.

$SVAE_{16\times16\times32}$ - Reconstructions of Test Data



KL-0 KL-1 KL-2 KL-3 KL-4

Figure 6.8: Reconstructed images from test dataset with the $SVAE_{16\times16\times32}$ model. The first row shows original images with two images from each KL class (0-4). The second row shows the corresponding reconstructed images.

$SVAE_{16\times16\times32}$ - Random Generated Samples

0 1 2 3 4

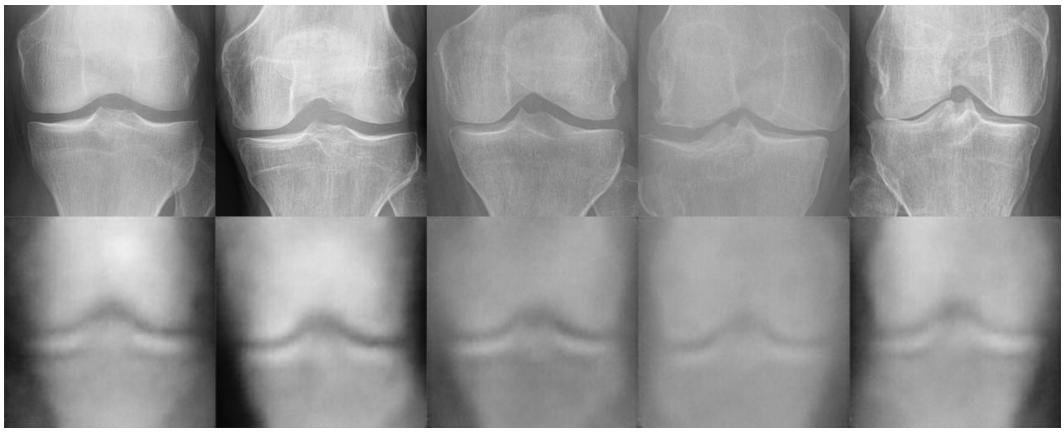
Figure 6.9: Random generated samples with the $SVAE_{16\times16\times32}$ model.

6.3 Variational Perceptual Generative Autoencoder (VPGA)

6.3.1 Standard Architecture $VPGA_{50}$

The VPGA model is trained with a latent dimension of 50 and referred as $VPGA_{50}$. The results for reconstruction and generation show poor outcomes. The reconstructed images are the most blurry ones and the generated samples are just random noise. The quantitative scores from table 6.1 and the qualitative image observation can substantiate the out comes. Reconstruction images are depicted in 6.10 for the test data set and generated images are shown in 6.11. The average learning curves show no special behavior A.10. The model converges after approximately 100 epochs as observed in the other experiments. Experiments with different architectures and configurations for the VPGA model are performed, but no settings,which lead to better performance of the VPGA model can be found.

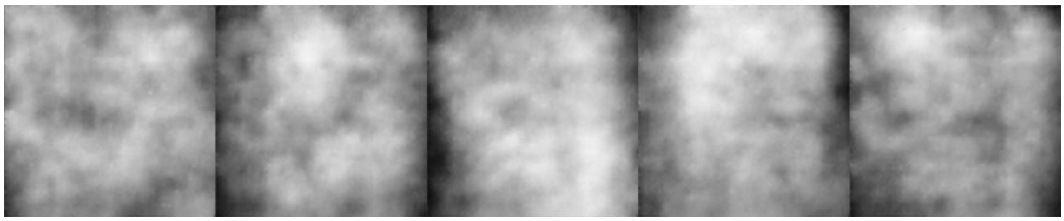
VPGA$_{50}$ - Reconstructions of Test Data



KL-0 KL-1 KL-2 KL-3 KL-4

Figure 6.10: Reconstructed images from test dataset with the VPGA$_{50}$ model. The first row shows original images with two images from each KL class (0-4). The second row shows the corresponding reconstructed images.

VPGA$_{50}$ - Random Generated Samples



0 1 2 3 4

Figure 6.11: Random generated samples with the VPGA$_{50}$ model.

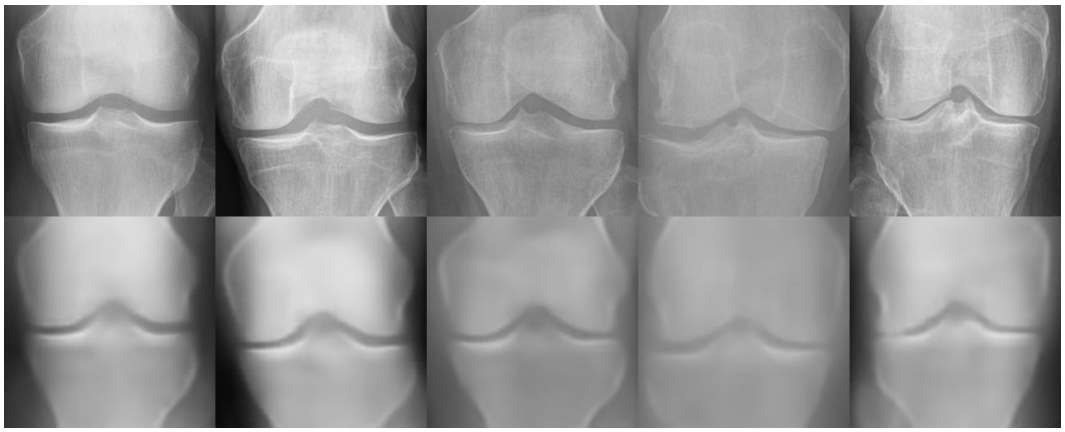
6.4 Vector Quantized Variational Autoencoder (VQ-VAE)

6.4.1 Standard Architecture VQ-VAE$_{std}$

The experiment VQ-VAE$_{std}$ with the standard architecture from 5.2 was done with the VQ-VAE model described in section 4.3. The dimension of the discrete latent space is $K = 512$ and the dimensionality of each latent embedding vector is set to $D = 32$. Surprisingly, no improvements to the standard VAE$_{50}$ model for the reconstruction ability of the VQ-VAE model was established. As shown in the overall result table 6.1 the MSE and MS-SSIM both are slightly worse than the scores from the standard VAE$_{50}$ experiment. This observations can also be made from the blurry reconstruction images on the training and test data set as shown in figure 6.12. The random generated samples outperform the VAE$_{50}$ model as it can be seen from the lower FID score and sharper generated images in figure 6.13. The average learning rates for the MSE, MS-SSIM and FID show no specificities compared to the VAE$_{50}$ model. As regards the completeness, the plots of the learning rates for the VQ-VAE$_{std}$ are depicted in the appendix A.12.

By changing the model's architecture a little bit, the reconstruction quality can be increased dramatically as shown in the next section.

VQ-VAE$_{std}$ - Reconstructions of Test Data



KL-0 KL-1 KL-2 KL-3 KL-4

Figure 6.12: Reconstructed images from test dataset with the VQ-VAE$_{std}$ model. The first row shows original images with one image from each KL class (0-4). The second row shows reconstructed images.

VQ-VAE$_{std}$ - Random Generated Samples

0 1 2 3 4

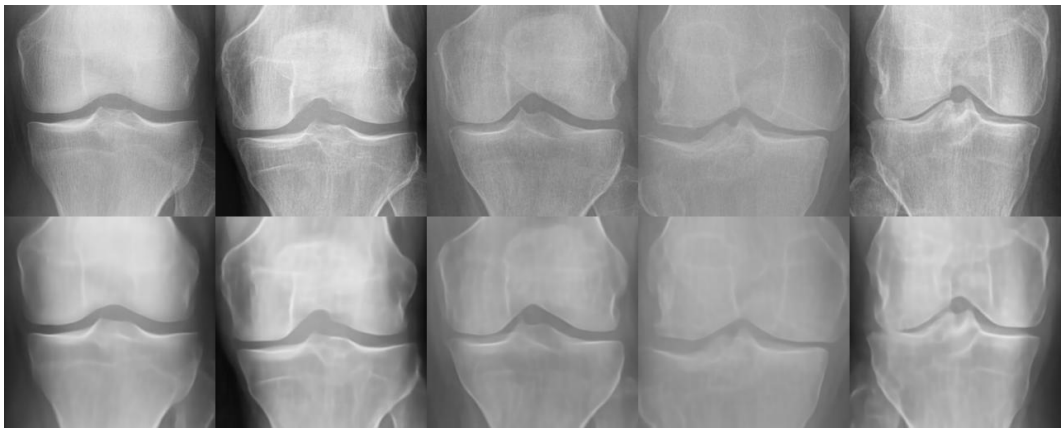
Figure 6.13: Random generated samples with the VQ-VAE$_{std}$ model.

6.4.2 Adapted Architecture VQ-VAE$_{adpt}$

First, experiments with the same adapted architecture as for the VAE and SVAE experiments are performed for the VQ-VAE model, however the reconstruction quality of the VQ-VAE model does not increase as dramatically as for the following adaptations of the architecture. The standard architecture from table 5.2 is simplified by omitting the first three residual blocks of the encoder and the last three residual blocks of the decoder. The number of filters for the convolutional layer is enhanced to 64. The output of the last residual block of the encoder results to $32 \times 32 \times 256$. The dimension of the embedding space stays the same ($K = 512, D = 32$). This experiment is referred to VQ-VAE$_{adpt}$. From the MSE and MS-SSIM curves for the VQ-VAE$_{adpt}$ in figure 6.19 it can be seen that the model is converging very fast. Furthermore, the standard variance for these two curves is very minimal, as it is shown in the detailed learning curve in figure A.14.

The reconstruction quality of the images are very sharp and contains almost all details as shown for the test data in figure 6.14 and for the train data in the appendix A.13. Furthermore, the MSE and MS-SSIM scores outperform all the other models as shown in table 6.1. In contrast to the good reconstruction capabilities of the VQ-VAE model the generated samples are just noisy points as it is depicted in figure 6.15.

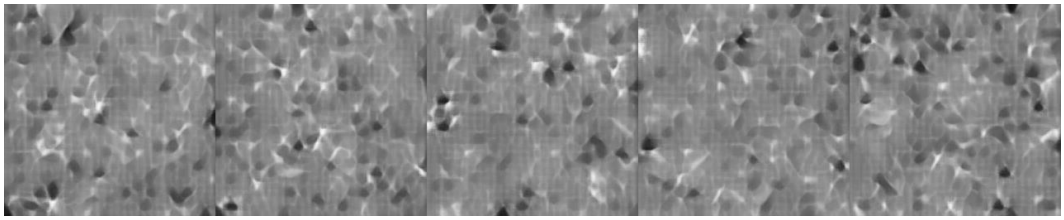
VQ-VAE$_{adpt}$ - Reconstructions of Test Data



KL-0 KL-1 KL-2 KL-3 KL-4

Figure 6.14: Reconstructed images from test dataset with the VQ-VAE$_{adpt}$ model. The first row shows original images with one image from each KL class (0-4). The second row shows reconstructed images.

VQ-VAE$_{adpt}$ - Random Generated Samples



0 1 2 3 4

Figure 6.15: Random generated samples with the VQ-VAE$_{adpt}$ model.

6.5 Introspective Variational Autoencoder (IntroVAE)

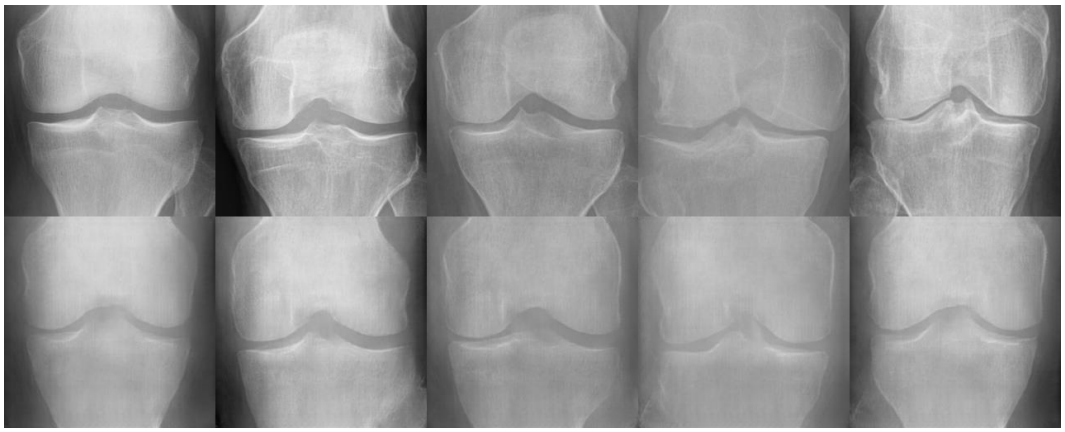
6.5.1 IntroVAE$_{50}$

The IntroVAE$_{50}$ refers to the IntroVAE model with a latent dimension of 50 using the architecture from table 5.2. The first epochs, the model is trained with a standard VAE model to initialize the weights, then the IntroVAE model is optimized.

The learning curves in the plots of figure 6.19 of the IntroVAE$_{50}$ experiment demonstrate that are sharp cut exits after epoch 100 in all three figures. The cut can be explained with the transition from the VAE training to the GAN training. Furthermore, it is conspicuous that the MSE slightly increases and the MS-SSIM indicator decreases after epoch 100, while the FID score decreases strongly. This is also an indicator that the GAN part overwhelm the VAE part. The quantitative values show a higher MSE and a lower MS-SSIM error compared to the other models. However, the FID is the lowest value of all experiments, which indicates good sampling properties. The highest and lowest values are shown in table 6.1. The reconstruction images from the training (figure A.15) and test data set (figure 6.16) show also an interesting behavior. The reconstructed images are sharp, but sometimes differ from the original input image. Especially for the rare KL class 4. There, the reconstructed image looks more like an image from a lower class. On one hand, this effect can be explained by the given sample distribution, on the other hand

it can be explained with the adapted loss function by the introspective GAN approach. The assumptions of the good sampling properties of the IntroVAE can be approved by

IntroVAE$_{50}$ - Reconstructions of Test Data

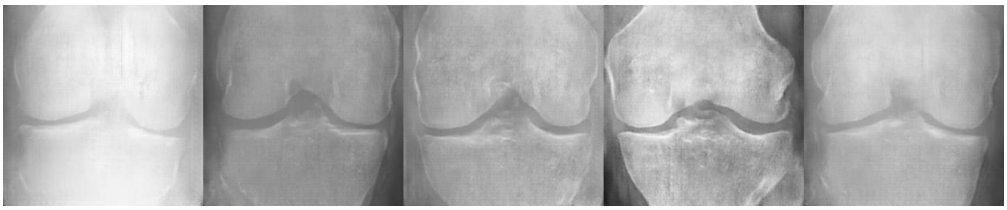


KL-0 KL-1 KL-2 KL-3 KL-4

Figure 6.16: Reconstructed images from test dataset with the IntroVAE$_{50}$ model. The first row shows original images with one image from each KL class (0-4). The second row shows reconstructed images.

the qualitative observations from random sampled images as depicted in figure 6.17. The random generated images look very realistic. It is hard to distinguish between the original sample and the random generated images. As for the VAE$_{50}$ experiment, the

IntroVAE$_{50}$ - Random Generated Samples



0 1 2 3 4

Figure 6.17: Random generated samples with the IntroVAE$_{50}$ model.

latent space of the IntroVAE$_{50}$ experiment is analyzed in the following. Therefore, a sample from the training set is chosen and each latent variable is traversed from $[-1, 1]$, while leaving the remaining latents constant to its original value. In figure 6.18 four noticeable traversed latent variables are plotted. Regions with remarkable changes are marked with red geometric shapes. Although the changes are not significant the model learns some features of the knee. Further, the changes from one traversed value to the next are smooth. Compared to the latent space analysis of the VAE$_{50}$ experiment, the images of the latent space from IntroVAE$_{50}$ are significant sharper.

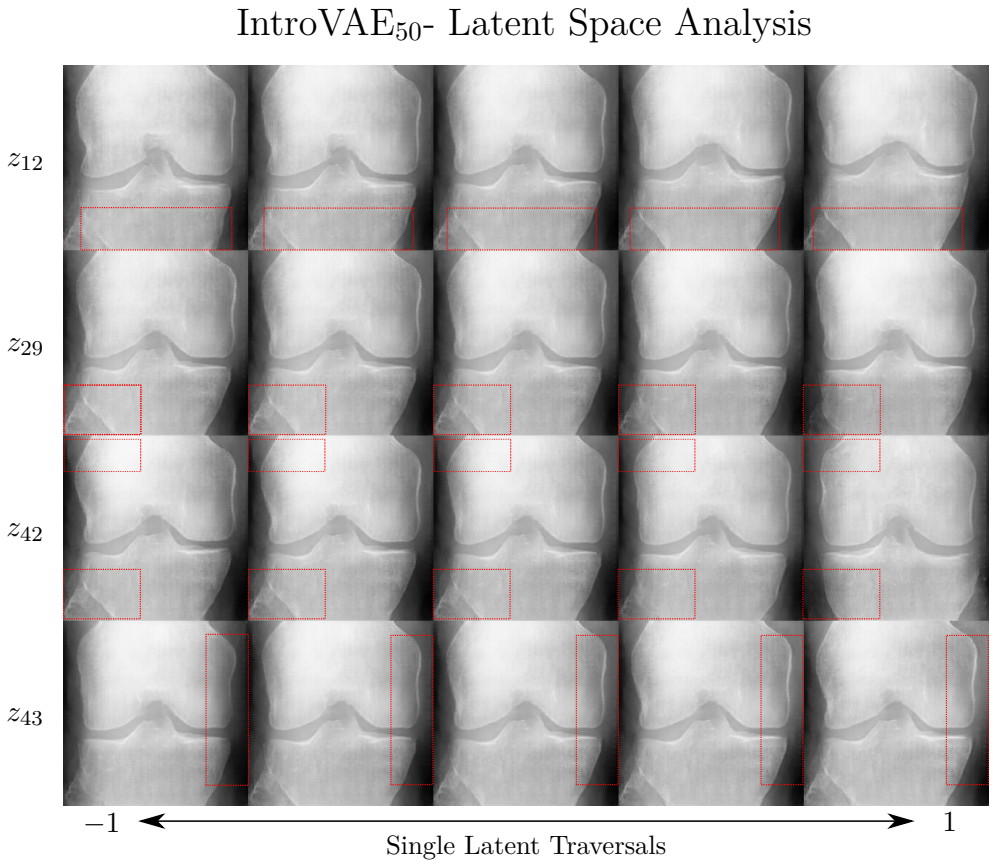


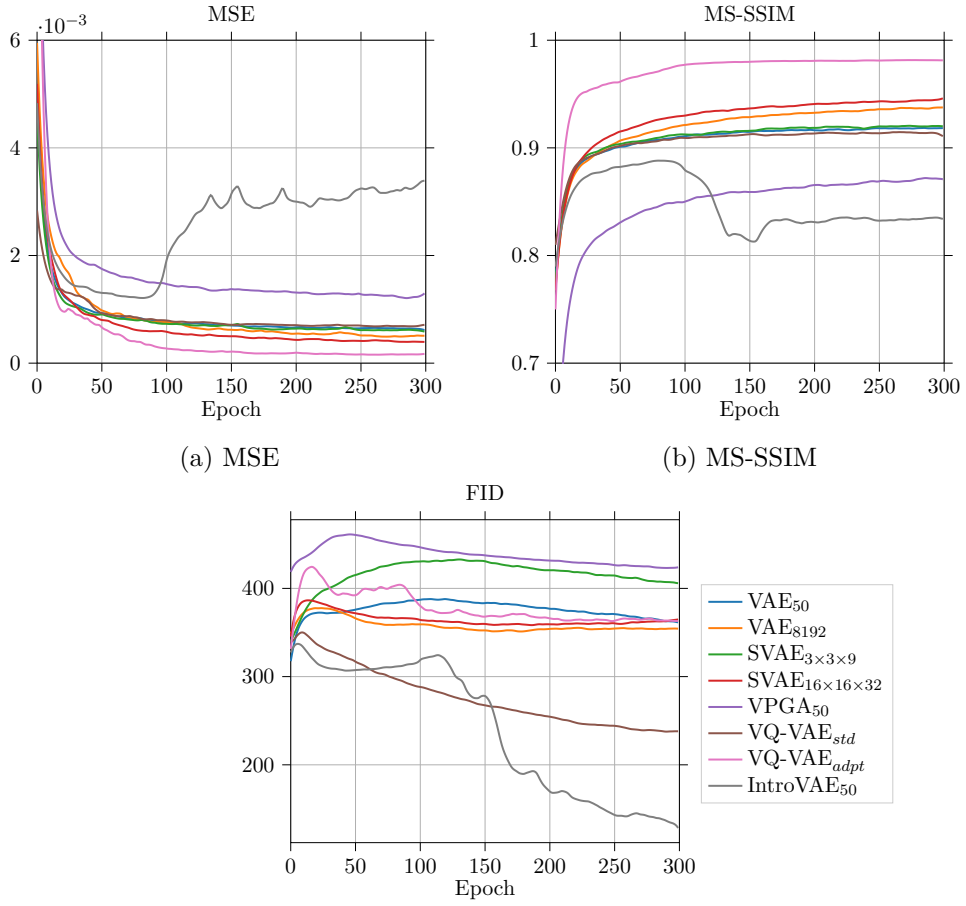
Figure 6.18: Four picked latents from the IntroVAE$_{50}$ model and traversed between $[-1, 1]$. The red geometric shapes indicate regions with changes during the traversals. Each row is a different latent, which value is changing, while the other kept constant.

6.6 Overall Results

In table 6.1 the mean and standard deviation of the MSE, MS-SSIM and FID for all performed experiments are depicted. The VQ-VAE model has the best MSE and MS-SSIM scores, which indicates good reconstruction properties of the model. The IntroVAE model has the best FID score. The VPGA experiment shows the worst performance comparing the quantitative values. In figure 6.19 the learning curves for all experiments are depicted. It can be seen that the curves of the models VAE$_{50}$, VAE$_{8192}$, SVAE$_{3\times3\times9}$, SVAE$_{16\times16\times32}$, VQ-VAE$_{std}$ proceed in a similar manner. The curves of the VPGA$_{50}$ performs below average. The MSE and MS-SSIM curves of the VQ-VAE$_{adpt}$ experiments exceed the others and converge much faster. The curves of the IntroVAE$_{50}$ experiment are showing the influence of the GAN model. In figure 6.20 the reconstructed images of the test dataset for all experiments are summarized. It clearly shows that the VQ-VAE$_{adpt}$ model outperforms the others and the VPGA$_{50}$ performs under average. The random generated images for all experiments are summarized into figure 6.21. It shows that the experiments for the VAE$_{8192}$, SVAE$_{16\times16\times32}$, VPGA$_{50}$ and VQ-VAE$_{adpt}$ only produce random noise. The IntroVAE$_{50}$ outperforms the others.

	MSE Train $[10^{-5}]$	MSE Test $[10^{-5}]$	MS-SSIM Train $[10^{-2}]$	MS-SSIM Test $[10^{-2}]$	FID
VAE$_{50}$	40 ±1.2	54±0.98	94 ±0.22	93 ±0.098	317 ±6.4
VAE$_{8192}$	34±1.1	40±1.6	95±0.094	94±0.099	332±3.0
SVAE$_{3\times3\times9}$	40±1.0	51±1.2	94±0.065	93 ±0.097	333±5.1
SVAE$_{16\times16\times32}$	27±1.1	32±0.88	96±0.054	95±0.050	336±5.2
VPGA$_{50}$	99±2.7	110±1.9	89 ±0.099	88 ±0.15	409±18
VQ-VAE$_{std}$	47±1.9	57±3.6	93 ±0.21	92±0.11	229±16
VQ-VAE$_{adpt}$	11±0.11	11±0.40	98±0.049	98±0.024	319±7.6
IntroVAE$_{50}$	99±1.7	110±2.7	90±0.088	90±0.30	117±8.3

Table 6.1: The table shows the mean of the lowest MSE, the highest MS-SSIM and lowest FID scores for each experiment over four trial runs. The scores are calculated for the training- and test dataset. Furthermore, the standard deviation of each score is depicted. The best results are marked in green.



(a) MSE

(b) MS-SSIM

(c) FID

Figure 6.19: Average learning curves for all experiments. For a better visualization a low pass filter is applied.

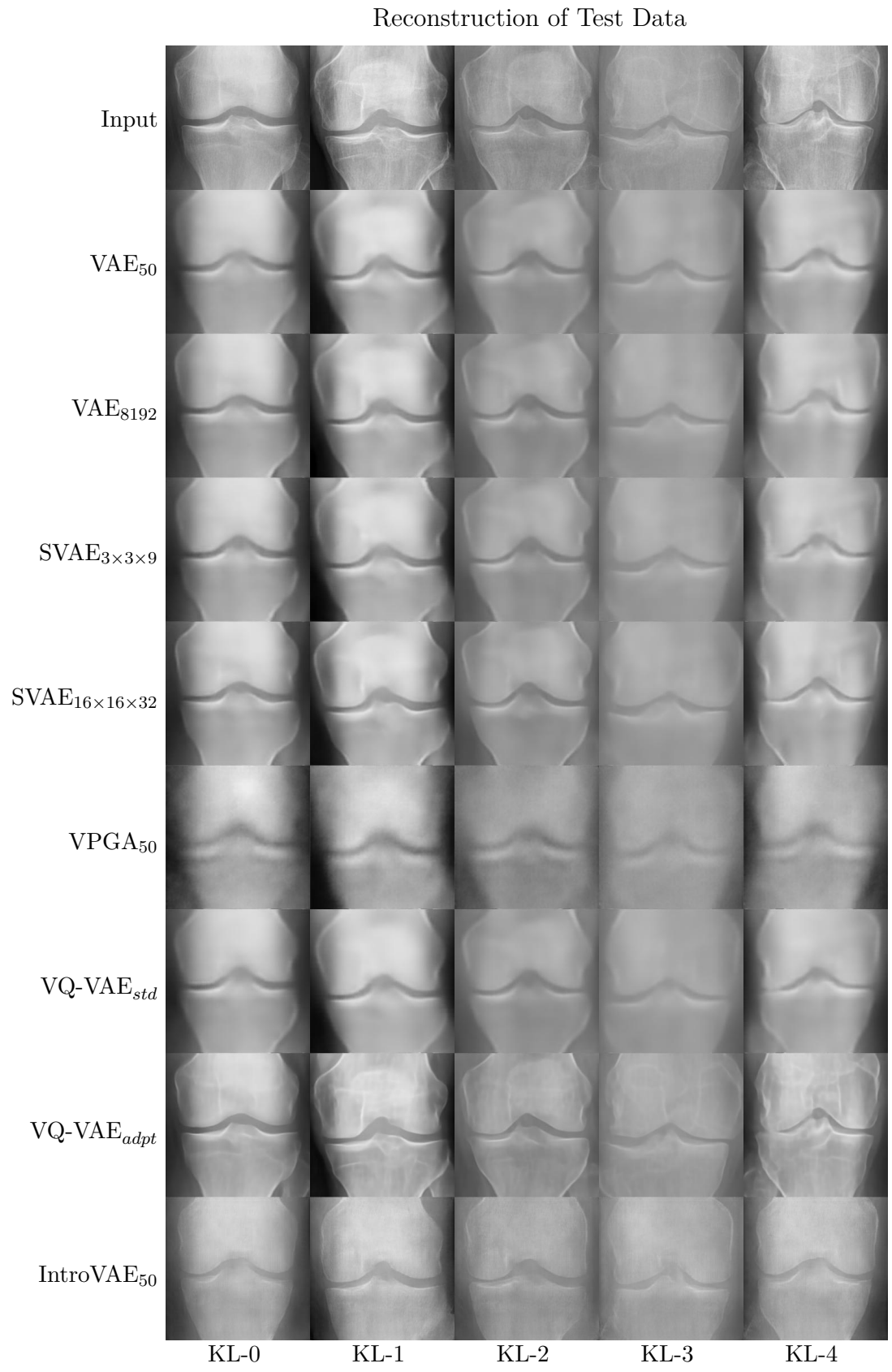


Figure 6.20: Comparison of the reconstruction capability of all experiments on the test data set.

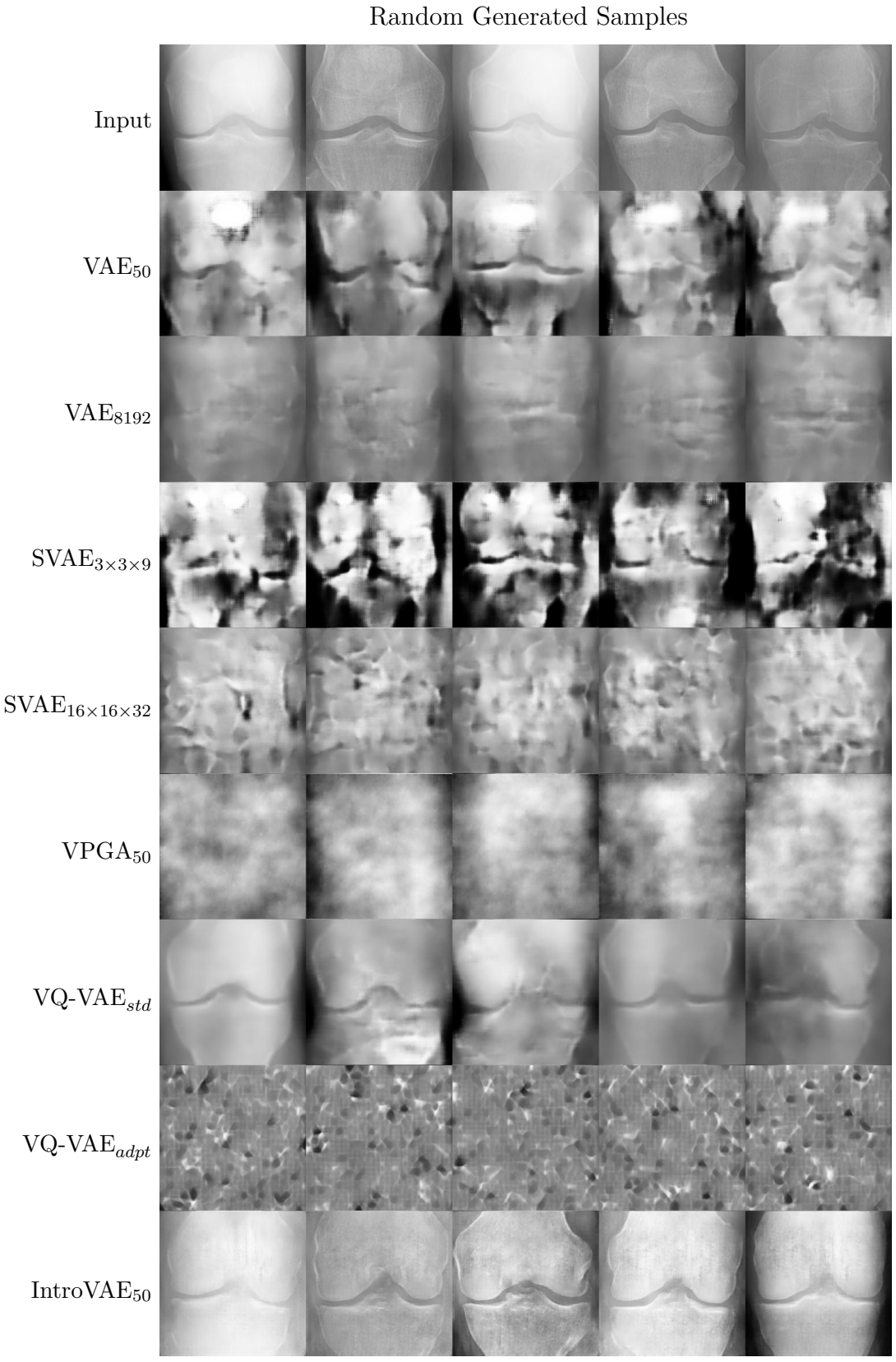


Figure 6.21: Comparison of the random generation capability of all experiments.

Chapter 7

Conclusion and Outlook

7.1 Conclusion

Within this thesis different VAE models were applied and evaluated on X-Ray knee images with a resolution of 256×256 pixels. It was shown that the standard VAE has the capability to disentangle the latent space for a complex data distribution. However, the reconstructed images are too blurry for medical applications. Therefore, enhancements of the standard VAE have been implemented and evaluated. One problem, which can be observed during the evaluation of some enhanced models was that no improvements can be determined on the used dataset, although the model leads to better results on simple datasets like MNIST. An example of this is the VPGA model which performed significantly worse on the knee dataset than the standard VAE approach. The result can be explained by the fact that deep learning approaches lead to a non convex optimization problem, which does not guarantee to find the optimal solution for a problem. Through the complexity of the loss function for the VPGA model no suitable hyper-parameters can be found. Further it can be shown that increasing the latent space leads in general to better reconstruction quality, but also to worse sampling properties of the model. No model can be determined, which had excellent performance in reconstructing the original input, generating random realistic samples and disentangling the latent space. However, two very promising approaches can be determined which performed fine in at least one of the required tasks. The VQ-VAE model is able to reconstruct the knee with almost no loss of information with the disadvantage of the interpretability of the multi dimensional discrete latent space. The hybrid approach of the IntroVAE model between a GAN and a VAE leads to a model with very good sampling properties. The random generated samples of the IntroVAE are very realistic and due to the fact that the latent space is one dimensional it is easy to manipulate. The diversity between the reconstructed images and the input images for classes with low proportion of samples to the total samples is the biggest disadvantage of this approach. This effect can be explained through the GAN part of the model. Overall this work builds a sold basis for further investigations to apply enhanced VAE models on complex and high resolution medical images.

7.2 Outlook

The latent space of the VQ-VAE model can be analyzed in more detail to find a way which allows interpretability of the latent space. One possibility of doing this would be the use of a second deep neural network in some way. Furthermore, the experiments can be performed on the larger Multicenter Osteoarthritis Study (MOST)[1] dataset, which was not available during the time of this thesis. This could lead to better performance of the reconstruction quality, especially for the IntroVAE. This approach can also be enhanced to force the model to better reconstruct the original input. Another possibility is to use a semi supervised learning approach like in [2], the so called N-VAE.

[1] `http://most.ucsf.edu/default.asp`

Appendix A

Appendix

A.1 Kullback leibler divergence

Simply speaking the KL divergence tells us how well a probability distribution $p(x)$ approximates a probability distribution $q(x)$. Formally the KL divergence for two probability density functions PDFs is defined as

$$\mathbb{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx = \int p(x) \log p(x) dx - \int p(x) \log q(x) dx. \tag{A.1}$$

This can be rewrite to

$$\mathbb{KL}(p||q) = \mathbb{E}_{x\sim p}[\log p(x)] - \mathbb{E}_{x\sim p}[\log q(x)] = -\mathbb{H}(p(x)) + \mathbb{H}(p(x), q(x)), \tag{A.2}$$

where the cross entropy $\mathbb{H}(\cdot)$ is defined as

$$\mathbb{H}(p, q) = -\int p(x) \log q(x) dx. \tag{A.3}$$

The KL divergence is usually **not** symmetric $\mathbb{KL}(p||q) \neq \mathbb{KL}(q||p)$, therefore it is not a distance. Furthermore the KL divergence is non negative $\mathbb{KL}(p||q) \geq 0$ with $\mathbb{KL}(p||q) = 0$ if and only if $p = q$ [41].

A.2 Matrix-Variate Normal (MVN)

The PDF of the MVN is defined for a random matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ as

$$\boldsymbol{X} \sim p(\boldsymbol{M}, \boldsymbol{\Omega}, \boldsymbol{\Psi}) = \frac{\exp(-\frac{1}{2} \operatorname{tr}[\boldsymbol{\Psi}^{-1}(\boldsymbol{X} - \boldsymbol{M})^T \boldsymbol{\Omega}^{-1}(\boldsymbol{X} - \boldsymbol{M})])}{(2\pi)^{\frac{mn}{2}} |\boldsymbol{\Psi}|^{\frac{m}{2}} |\boldsymbol{\Omega}|^{\frac{n}{2}}}, \tag{A.4}$$

where $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ is the mean matrix, $\boldsymbol{\Omega} \in \mathbb{R}^{m\times m}$, $\boldsymbol{\Psi} \in \mathbb{R}^{n\times n}$ are diagonal matrices and $\operatorname{tr}(\cdot)$ denotes the trace of a matrix.

A.3 Experiment Results

A.3.1 VAE_{50}

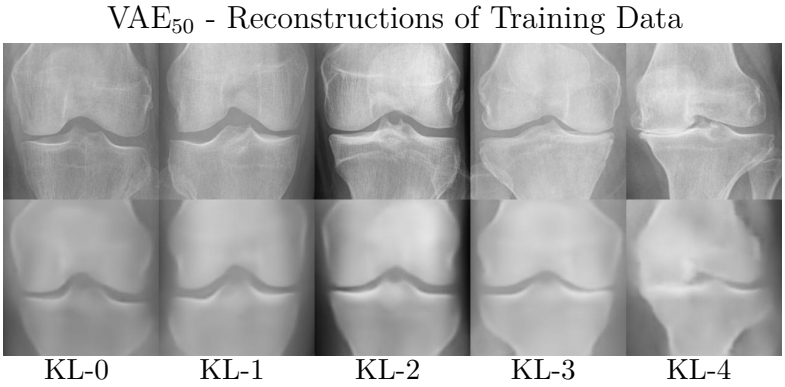


Figure A.1: Reconstructed images from the training data set with the VAE_{50} model. The first row shows the original images with one sample from each KL class (0-4). The second row shows the reconstructed images.

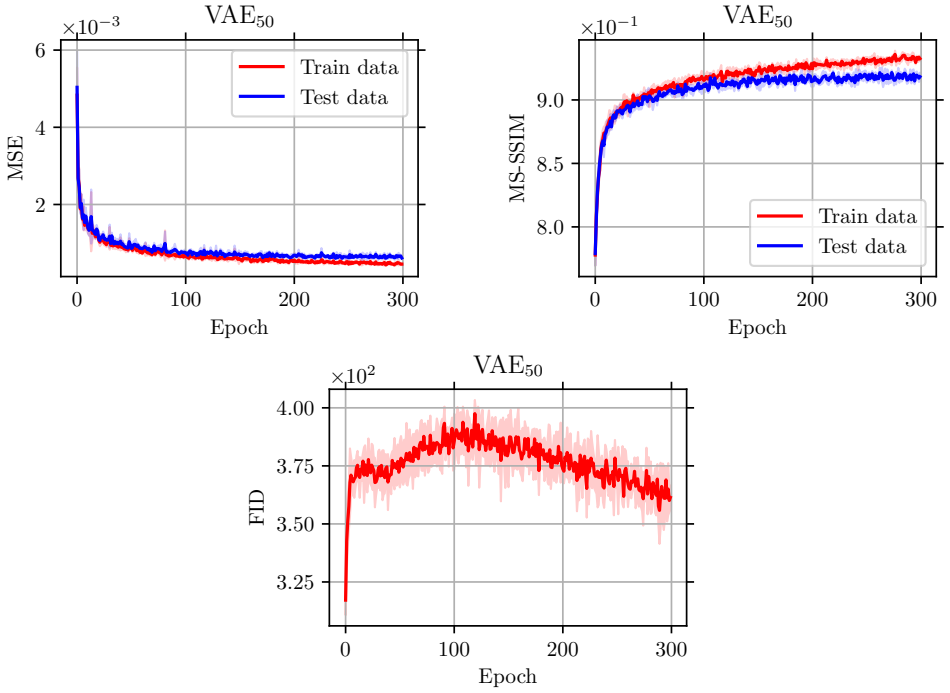


Figure A.2: Average learning curve for the VAE_{50} with a latent space of 50 over 4 trials. The line represents the mean, the shaded region marks the standard deviation.

A.3.2 VAE_{8192}

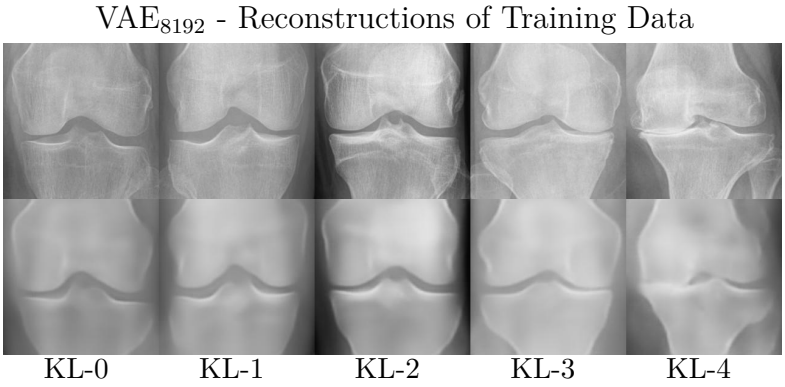


Figure A.3: Reconstructed images from the training data set with the VAE_{8192} model. The first row shows the original images with one sample from each KL class (0-4). The second row shows the reconstructed images.

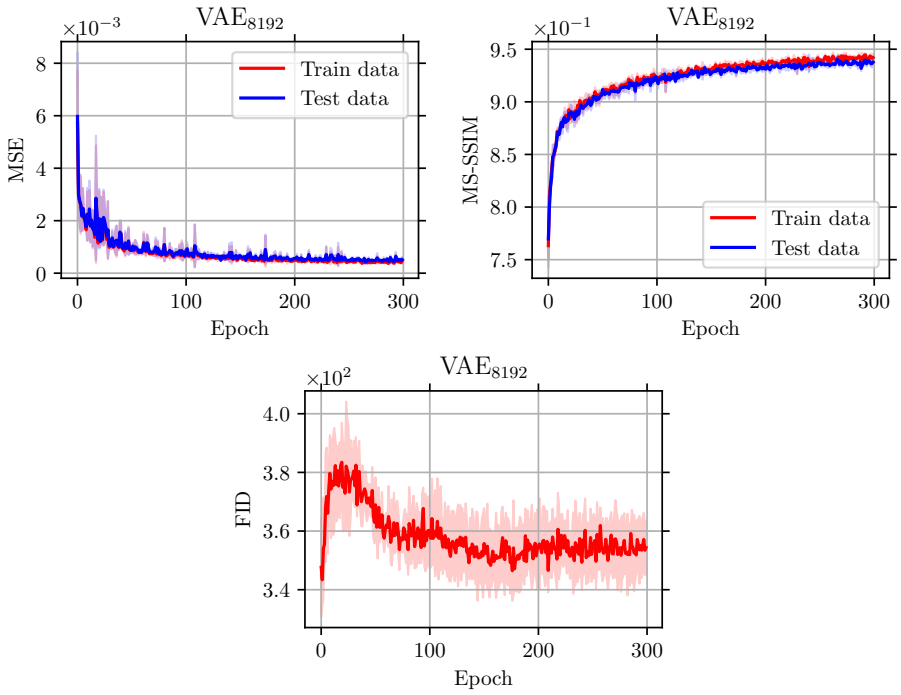


Figure A.4: Average learning curves for the VAE_{8192} model over 4 trials. The lines represents the mean, the shaded regions mark the standard deviation.

A.3.3 $\text{SVAE}_{3\times3\times9}$

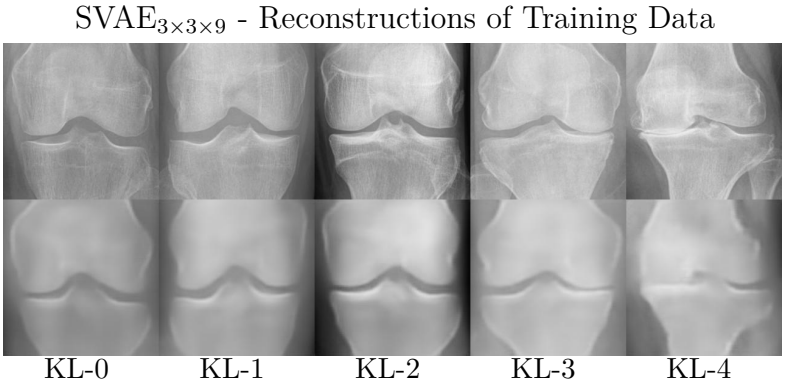


Figure A.5: Reconstructed images from training dataset with the $\text{SpatialVAE}_{3\times3\times9}$ model. The first row shows the original images with two images from each KL class (0-4). The second row shows the reconstructed images.

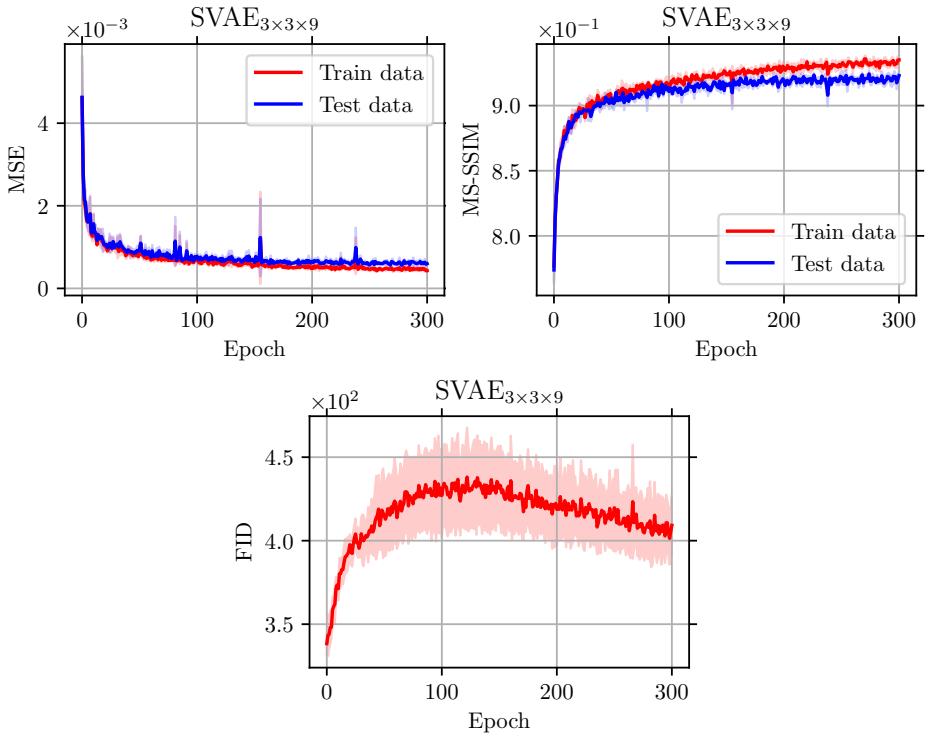


Figure A.6: Average learning curves for the $\text{SpatialVAE}_{3\times3\times9}$ model over 4 trials. The lines represents the mean, the shaded regions mark the standard deviation.

A.3.4 $\text{SVAE}_{16\times16\times32}$

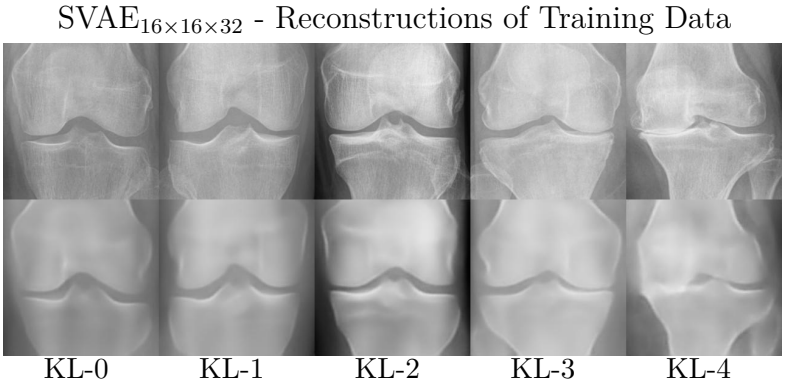


Figure A.7: Reconstructed images from training dataset with the $\text{SVAE}_{16\times16\times32}$ model. The first row shows the original images with two images from each KL class (0-4). The second row shows the reconstructed images.

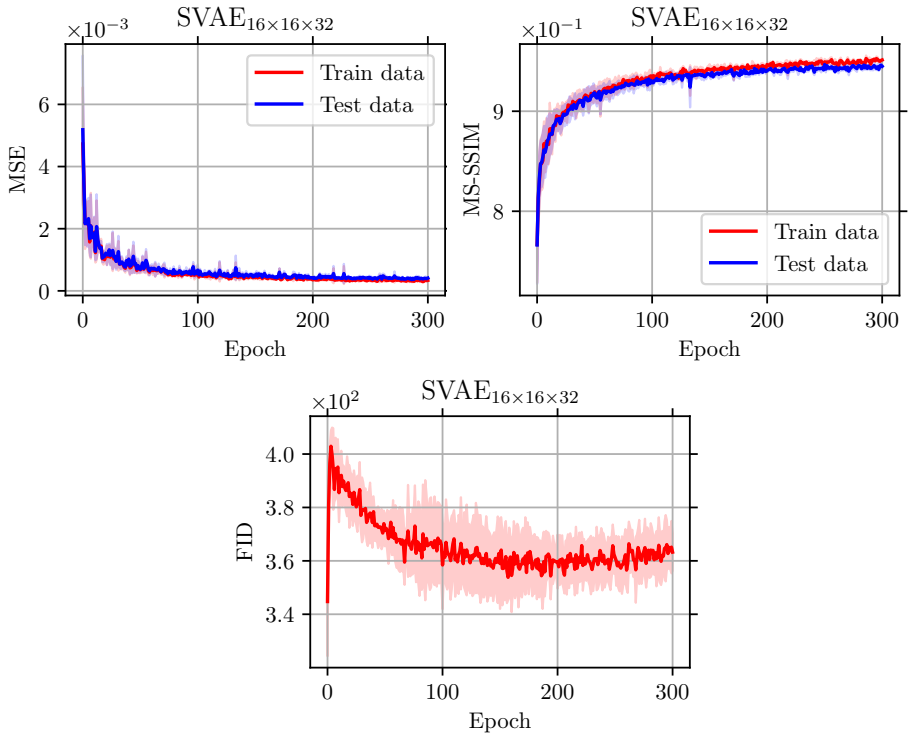


Figure A.8: Average learning curves for the $\text{SVAE}_{16\times16\times32}$ model over 4 trials. The lines represents the mean, the shaded regions mark the standard deviation.

A.3.5 VPGA$_{50}$

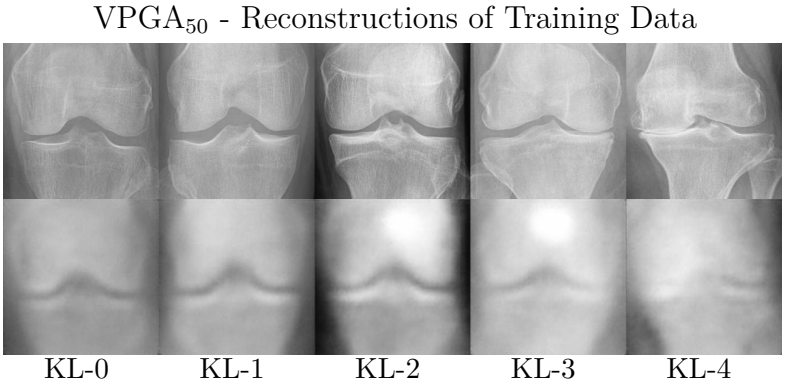


Figure A.9: Reconstructed images from train dataset with the VPGA$_{50}$ model. The first row shows the original images with two images from each KL class (0-4). The second row shows the reconstructed images.

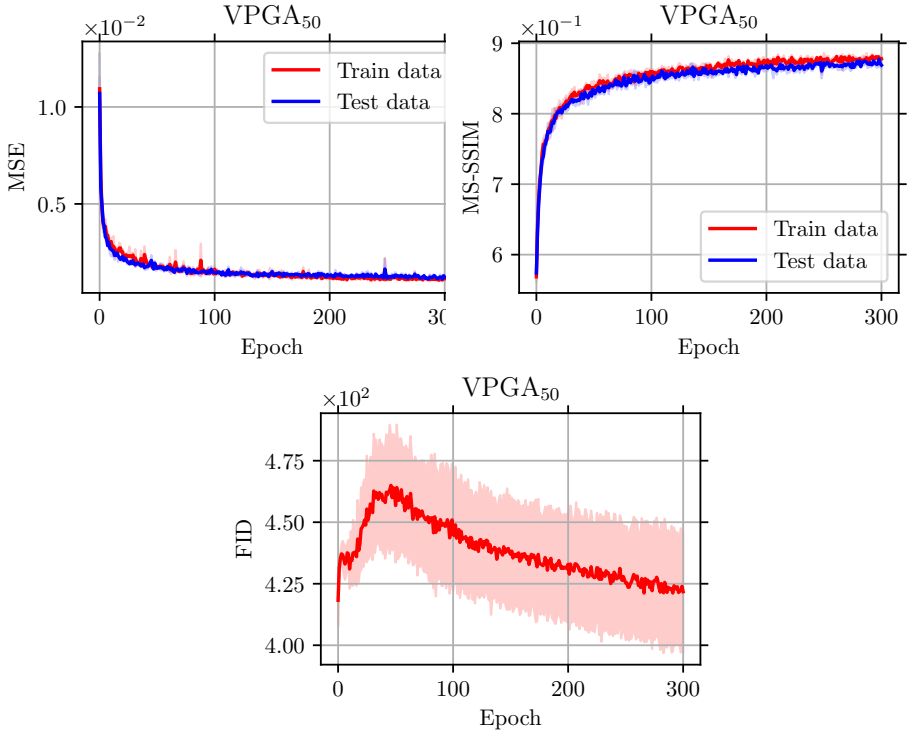


Figure A.10: Average learning curve for the VPGA$_{50}$ experiment over 4 trials. The line represents the mean, the shaded region marks the standard deviation.

A.3.6 VQ-VAE$_{std}$

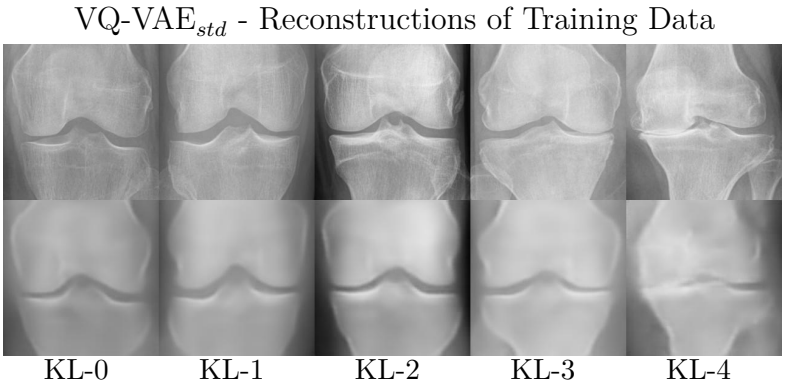


Figure A.11: Reconstructed images from train dataset with the VQ-VAE$_{std}$ model. The first row shows original images with one images from each KL class (0-4). The second row shows reconstructed images.

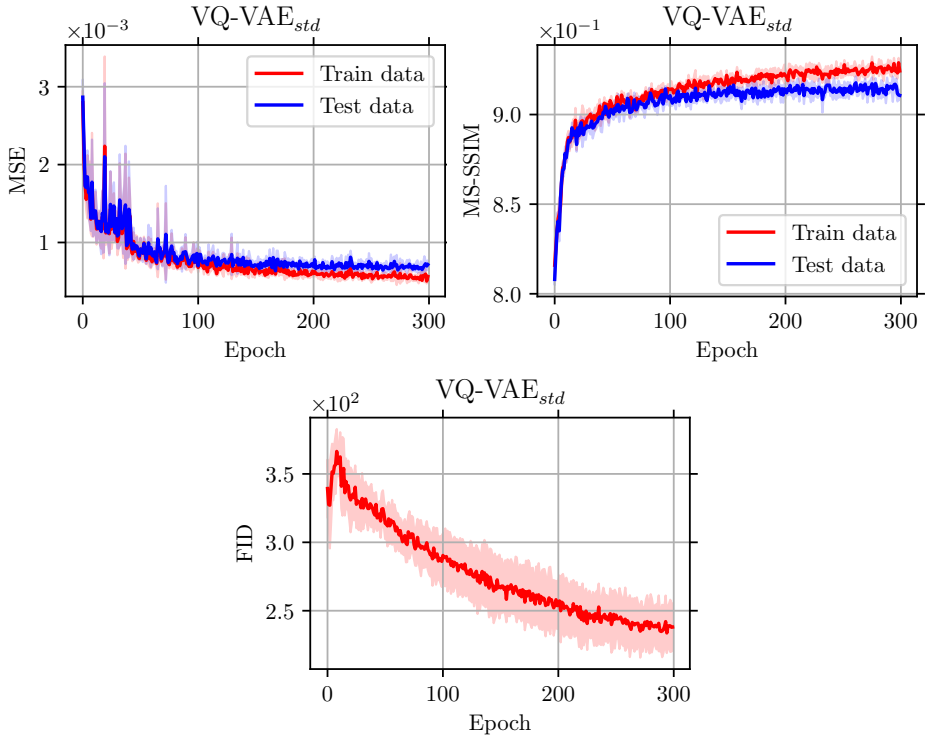


Figure A.12: Average learning curve for the VQ-VAE$_{std}$ experiment over 4 trials. The line represents the mean, the shaded region marks the standard deviation.

A.3.7 VQ-VAE$_{adpt}$

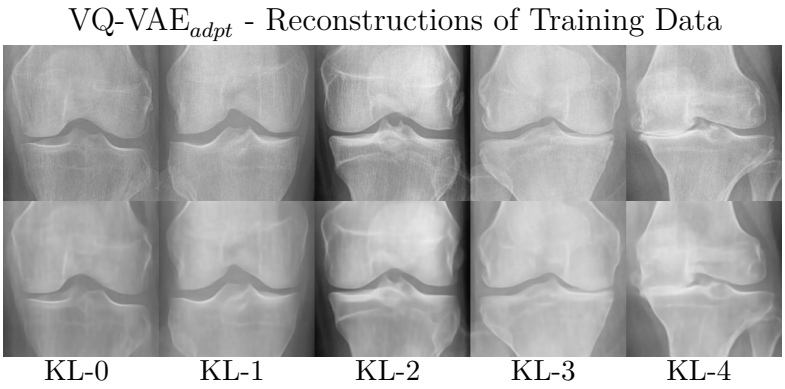


Figure A.13: Reconstructed images from training dataset with the VQ-VAE$_{adpt}$ model. The first row shows original images with one image from each KL class (0-4). The second row shows reconstructed images.

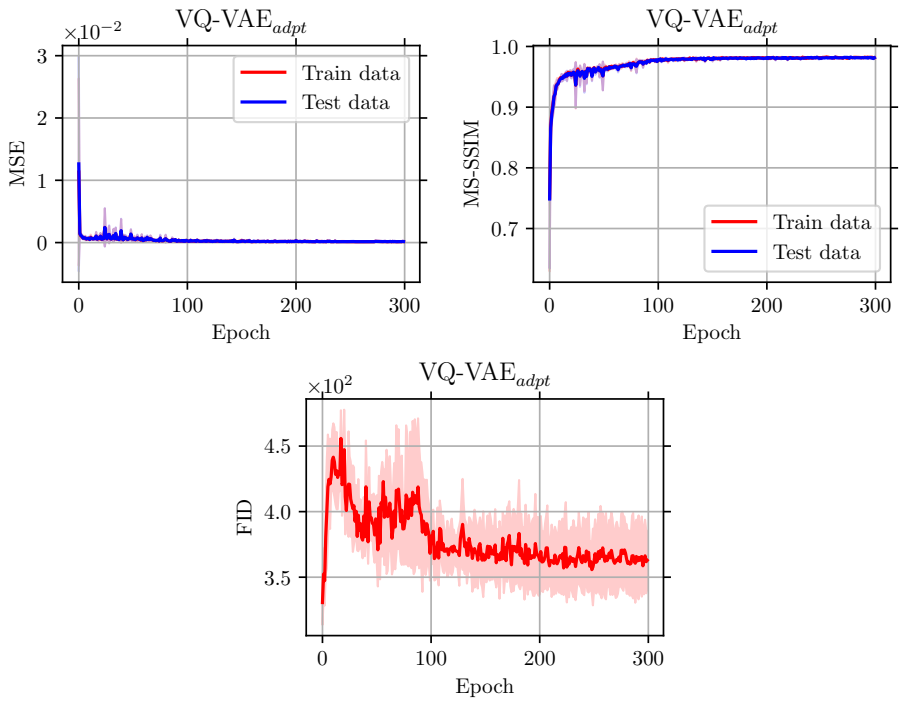


Figure A.14: Average learning curve for the VQ-VAE$_{adpt}$ experiment over 4 trials. The line represents the mean, the shaded region marks the standard deviation.

A.3.8 IntroVAE$_{50}$

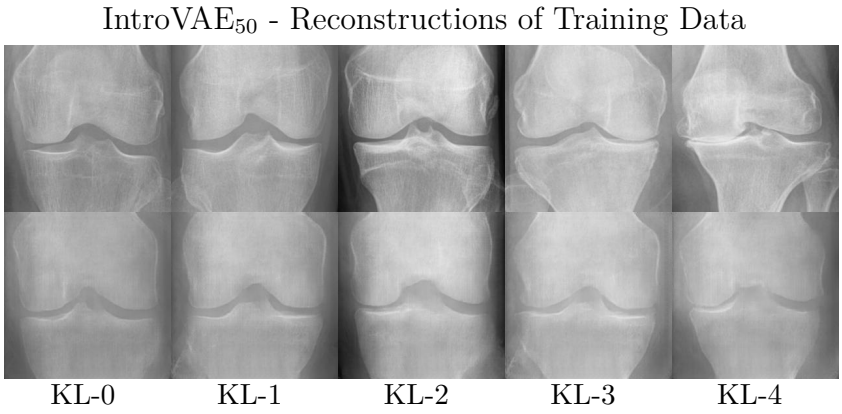


Figure A.15: Reconstructed images from training dataset with the IntroVAE$_{50}$ model. The first row shows original images with one image from each KL class (0-4). The second row shows reconstructed images.

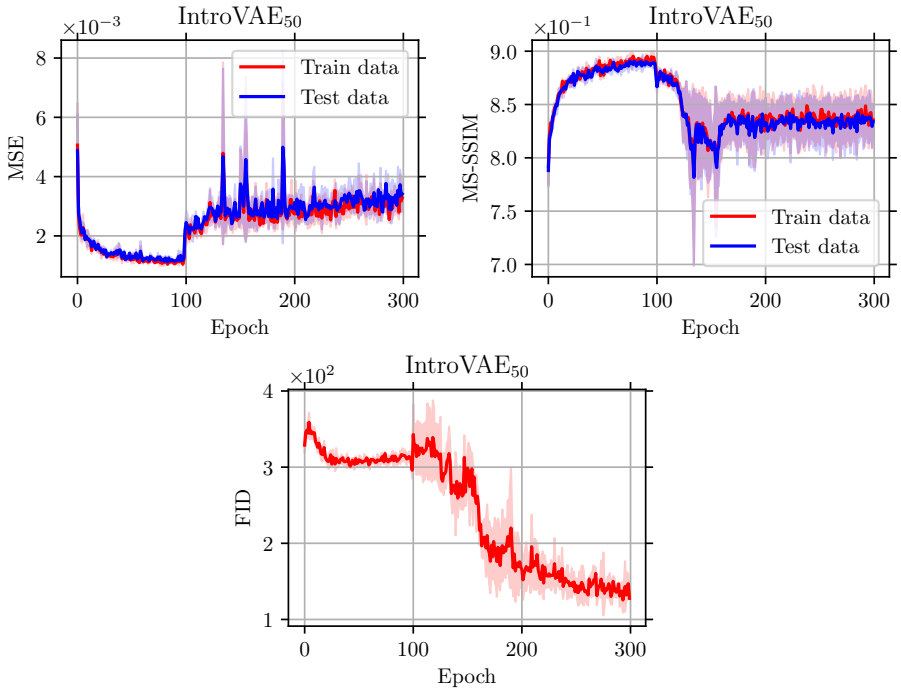


Figure A.16: Average learning curve for IntroVAE$_{50}$ experiment over 4 trials. The line represents the mean, the shaded region marks the standard deviation.

A.3.9 Reconstructions on Training Dataset

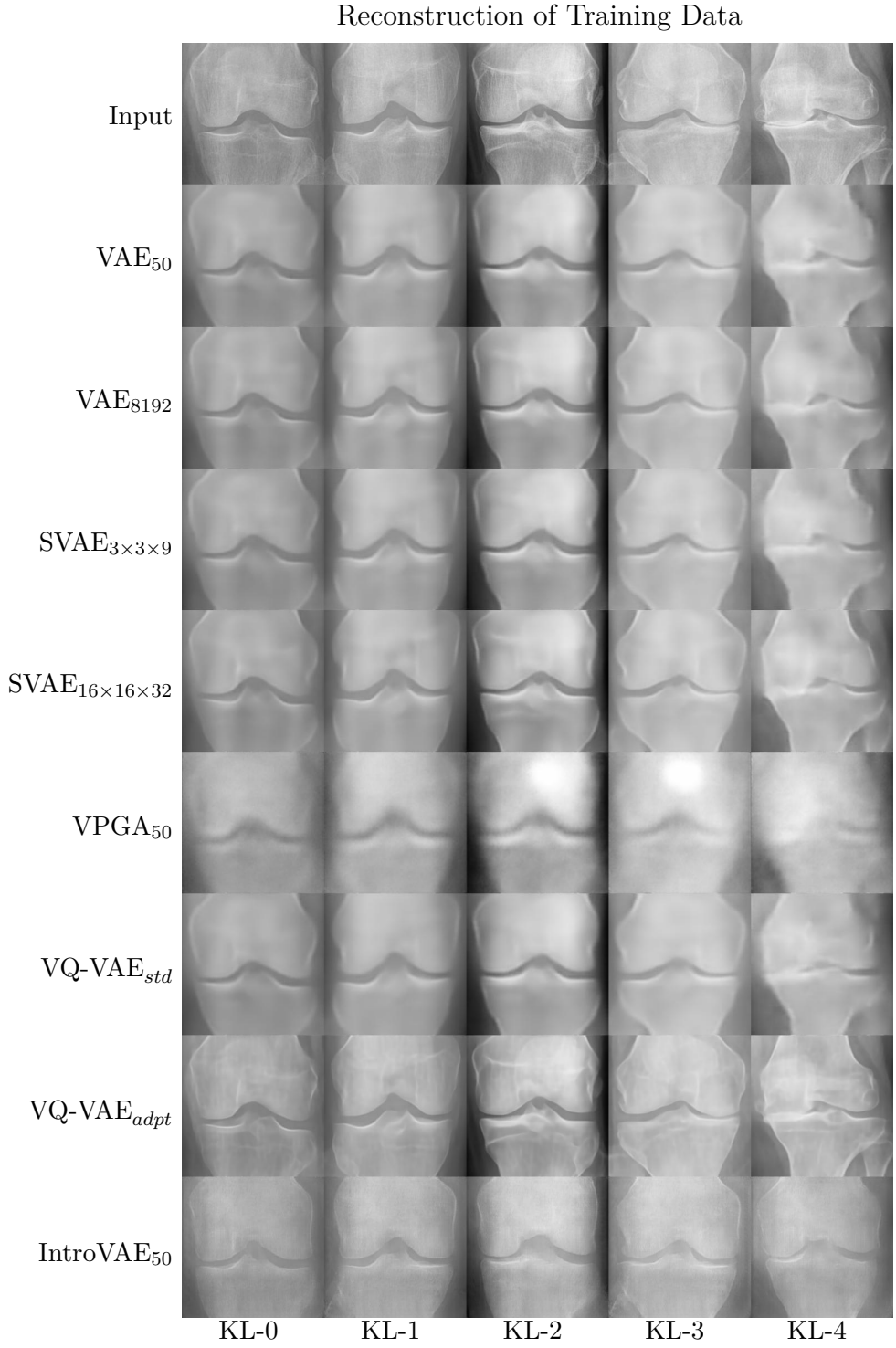


Figure A.17: Comparison of the models on the training data set.

A.4 Notation

Symbol	Meaning
y	Scalar
$\boldsymbol{x}$	Vector or matrix
$\boldsymbol{A}$	Matrix
$\boldsymbol{A} \otimes \boldsymbol{B}$	Kronecker product
∇	Vector of first derivatives
$\mathbb{R}$	The real numbers
$\underset{\boldsymbol{x}^*}{\text{argmin}} f(\boldsymbol{x})$	Argmin: the values $\boldsymbol{x}^*$ that minimize f
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
$\text{tr}(\boldsymbol{A})$	Trace of a matrix
$\boldsymbol{A}^{-1}$	Inverse of a matrix
$\boldsymbol{A}^T$	Transpose of a matrix
$\boldsymbol{I}$	Identity matrix
$\|\|\boldsymbol{x}\|\|_2$	Euclidean norm
$p(\boldsymbol{x})$	Probability density or mass function
$p(\boldsymbol{x}\|\boldsymbol{y})$	Conditional probability density of $\boldsymbol{x}$ given $\boldsymbol{y}$
$X \sim p$	X is distributed according to distribution p
$\mathbb{E}[X]$	Expected value of X
$\mathbb{E}_q[X]$	Expected value of X wrt. distribution q
$\mathbb{KL}(p\|\|q)$	KL divergence from distribution p and q
$\mathbb{H}(X)$	Entropy of distribution p(X)
$\boldsymbol{\mu}$	Mean of a multivariate distribution
$\boldsymbol{\Sigma}$	Covariance matrix
$\text{Enc}_{\boldsymbol{\Phi}}(\boldsymbol{x})$	Encoder network with the weights $\boldsymbol{\Phi}$
$\text{Dec}_{\boldsymbol{\theta}}(\boldsymbol{z})$	Decoder network with the weights $\boldsymbol{\theta}$
L	Loss term
$\boldsymbol{\varepsilon}$	Follows a centered isotropic multivariate Gaussian $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
$\text{ng}(\cdot)$	No gradient calculated for the back propagation
$[\cdot]^+$	$\max(0, \cdot)$
$\odot$	Dot product

A.5 Abbreviations and Acronyms

AI artificial intelligence

ANN artificial neural network

CAD computer aided diagnosis

CNN convolutional neural network

Conv convolutional layer

CT computed tomography

ELBO evidence lower bound

FC fully connected layer

FID Fréchet Inception Distance

GAN generative adversarial net

IntroVAE introspective variational autoencoder

KL Kullback-Leibler

ML machine learning

MLP multilayer perceptron

MOST Multicenter Osteoarthritis Study

MRI magnetic resonance imaging

MS-SSIM multi-scale structural similarity

MSE Mean Squared Error

MVN matrix-variate normal

NLP natural language processing

OAI Osteo Arthritis Initiative

PCA principal component analysis

PDF probability density function

PixelCNN Pixel Convolutional Neural Networks

PixelRNN Pixel Recurrent Neural Networks

Res-Block residual block

SSIM structural Similarity

SVAE spatial variational autoencoder

VAE variational autoencoder

VPGA variational perceptual generative autoencoder

VQ vector quantization

VQ-VAE vector quantized variational autoencoder

List of Algorithms

List of Figures

List of Tables

References

[1] Musaed Alhussein, Ghulam Muhammad, and M. Shamim Hossain. Eeg pathology detection based on deep learning. *IEEE Access*, 7:27781–27788, 2019. doi: 10.1109/ACCESS.2019.2901672.

[2] Javier Antoran and Antonio Miguel. Disentangling in variational autoencoders with natural clustering. URL `http://arxiv.org/pdf/1901.09415v2`.

[3] Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, Christian Szegedy, and Stewart Wilcox. Holist: An environment for machine learning of higher-order theorem proving. URL `http://arxiv.org/pdf/1904.03241v2`.

[4] Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, and Nassir Navab. Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. *0302-9743*, 11383(9):161–169, 2019. ISSN 0302-9743. doi: 10.1007/978-3-030-11723-8{\textunderscore}16. URL `http://arxiv.org/pdf/1804.04488v1`.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. URL `http://arxiv.org/pdf/1206.5538v3`.

[6] Wenya Linda Bi, Ahmed Hosny, Matthew B. Schabath, Maryellen L. Giger, Nicolai J. Birkbak, Alireza Mehrtash, Tavis Allison, Omar Arnaout, Christopher Abbosh, Ian F. Dunn, Raymond H. Mak, Rulla M. Tamimi, Clare M. Tempany, Charles Swanton, Udo Hoffmann, Lawrence H. Schwartz, Robert J. Gillies, Raymond Y. Huang, and Hugo J. W. L. Aerts. Artificial intelligence in cancer imaging: Clinical challenges and applications. *CA: a cancer journal for clinicians*, 69(2):127–157, 2019. doi: 10.3322/caac.21552.

[7] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006. ISBN 9780387310732.

[8] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders, . URL `http://arxiv.org/pdf/1802.04942v5`.

[9] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder, . URL `http://arxiv.org/pdf/1611.02731v2`.

[10] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model, . URL `http://arxiv.org/pdf/1712.09763v1`.

[11] Xiaoran Chen and Ender Konukoglu. Unsupervised detection of lesions in brain mri using constrained adversarial auto-encoders. URL https://arxiv.org/pdf/1806.04972.pdf.

[12] Bin Dai and David Wipf. Diagnosing and enhancing vae models. URL http://arxiv.org/pdf/1903.05789v1.

[13] David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams. Learning representations by back-propagating error. 1968.

[14] B.VLandau D.CDowson. The fréchet distance between multivariate normal distributions. 1979.

[15] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. URL http://arxiv.org/pdf/1605.09782v7.

[16] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. URL http://arxiv.org/pdf/1606.00704v3.

[17] G. E. Hinton, R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science (New York, N.Y.)*, 313(5786):502–504, 2006. doi: 10.1126/science.1129198.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. URL http://arxiv.org/pdf/1406.2661v1.

[19] Hayit Greenspan, Bram van Ginneken, and Ronald M. Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016. ISSN 0278-0062. doi: 10.1109/TMI.2016.2553401.

[20] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. URL http://arxiv.org/pdf/1611.05013v1.

[21] Jianxing He, Sally L. Baxter, Jie Xu, Jiming Xu, Xingtao Zhou, and Kang Zhang. The practical implementation of artificial intelligence technologies in medicine. *Nature medicine*, 25(1):30–36, 2019. doi: 10.1038/s41591-018-0307-0.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. URL http://arxiv.org/pdf/1512.03385v1.

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 2018. URL http://arxiv.org/pdf/1706.08500v6.

[24] Huaibo Huang, Zhihang Li, Ran He, Zhenan Sun, and Tieniu Tan. Introvae: Introspective variational autoencoders for photographic image synthesis. URL http://arxiv.org/pdf/1807.06358v2.

[25] Iain Murray Hugo Larochelle. The neural autoregressive distribution estimator. 2011.

[26] Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep learning. 2016.

[27] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2017.

[28] Jaan Altosaar. what-is-variational-autoencoder-vae-tutorial, 2019. URL `https://jaan.io/what-is-variational-autoencoder-vae-tutorial/?`

[29] Justin Johnson and Andrej Karpathy. Cs231n convolutional neural networks for visual recognition, 2019. URL `http://cs231n.github.io/convolutional-networks/`.

[30] J. H. KELLGREN and J. S. LAWRENCE. Radiological assessment of osteo-arthrosis. *Annals of the rheumatic diseases*, 16(4):494–502, 1957. ISSN 0003-4967. doi: 10.1136/ard.16.4.494.

[31] Salman H. Khan, Munawar Hayat, and Nick Barnes. Adversarial training of variational auto-encoders for high fidelity image generation. URL `http://arxiv.org/pdf/1804.10323v1`.

[32] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. 2015.

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. URL `http://arxiv.org/pdf/1412.6980v9`.

[34] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. URL `https://arxiv.org/pdf/1312.6114.pdf`.

[35] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. URL `http://arxiv.org/pdf/1606.04934v2`.

[36] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. URL `http://arxiv.org/pdf/1711.00848v3`.

[37] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. URL `http://arxiv.org/pdf/1512.09300v2`.

[38] Garam Lee, Kwangsik Nho, Byungkon Kang, Kyung-Ah Sohn, and Dokyoon Kim. Predicting alzheimer's disease progression using multi-modal deep learning approach. *Scientific reports*, 9(1):1952, 2019. doi: 10.1038/s41598-018-37769-z.

[39] Loic Matthey and Irina Higgins and Demis Hassabis and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset, 2017. URL `https://github.com/deepmind/dsprites-dataset/`.

[40] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. URL `http://arxiv.org/pdf/1511.05644v2`.

[41] Kevin P. Murphy. *Machine learning: A probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, Cambridge MA, 2012. ISBN 9780262018029.

[42] Daniil Pakhomov, Vittal Premachandran, Max Allan, Mahdi Azizian, and Nassir Navab. Deep residual learning for instrument segmentation in robotic surgery. URL http://arxiv.org/pdf/1703.08580v1.

[43] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. 2010.

[44] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. URL http://arxiv.org/pdf/1906.00446v1.

[45] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. URL http://arxiv.org/pdf/1505.05770v6.

[46] Stuart Russell. *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, Place of publication not identified, 2016. ISBN 1292153962.

[47] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. URL http://arxiv.org/pdf/1701.05517v1.

[48] AhmadEL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017 (19):70–76, 2017. ISSN 2470-1173. doi: 10.2352/ISSN.2470-1173.2017.19.AVM-023.

[49] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science (New York, N.Y.)*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

[50] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. URL http://arxiv.org/pdf/1602.02282v3.

[51] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. URL http://arxiv.org/pdf/1705.07761v3.

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. URL http://arxiv.org/pdf/1512.00567v3.

[53] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. URL http://arxiv.org/pdf/1711.01558v3.

[54] Jakub M. Tomczak and Max Welling. Vae with a vampprior, . URL http://arxiv.org/pdf/1705.07120v5.

[55] Jakub M. Tomczak and Max Welling. Improving variational auto-encoders using householder flow, . URL http://arxiv.org/pdf/1611.09630v4.

[56] Hristina Uzunova, Sandra Schultz, Heinz Handels, and Jan Ehrhardt. Unsupervised pathology detection in medical images using conditional variational autoencoders. *International journal of computer assisted radiology and surgery*, 14(3):451–461, 2019. doi: 10.1007/s11548-018-1898-0.

[57] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks, . URL `http://arxiv.org/pdf/1601.06759v3`.

[58] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders, . URL `http://arxiv.org/pdf/1606.05328v2`.

[59] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, . URL `http://arxiv.org/pdf/1711.00937v2`.

[60] Zhengyang Wang, Hao Yuan, and Shuiwang Ji. Spatial variational auto-encoding via matrix-variate normal distributions. URL `http://arxiv.org/pdf/1705.06821v2`.

[61] Ke Yan, Xiaosong Wang, Le Lu, and Ronald M. Summers. Deeplesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning. *Journal of medical imaging (Bellingham, Wash.)*, 5(3):036501, 2018. ISSN 2329-4302. doi: 10.1117/1.JMI.5.3.036501.

[62] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018. ISSN 1556-603X. doi: 10.1109/MCI.2018.2840738.

[63] Zijun Zhang, Ruixiang Zhang, Zongpeng Li, Yoshua Bengio, and Liam Paull. Perceptual generative autoencoders. URL `http://arxiv.org/pdf/1906.10335v1`.

[64] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders, . URL `http://arxiv.org/pdf/1706.02262v3`.

[65] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models, . URL `http://arxiv.org/pdf/1702.08658v1`.

[66] Wang Zhou. Image quality assessment: From error visibility to structural similarity. 2004.

[67] Zhou Wang, Eero P. Simoncelli, Alan C. Bovik. Multiscale structural similarity for image quality assessment. 2003.