DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Generative Models and Feature Extraction on Patient Images and Structure Data in Radiation Therapy

HANNA GRUSELIUS

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Generative Models and Feature Extraction on Patient Images and Structure Data in Radiation Therapy

HANNA GRUSELIUS

Degree Projects in Mathematical Statistics (30 ECTS credits)
Degree Programme in Industrial Engineering and Management (120 credits)
KTH Royal Institute of Technology year 2018
Supervisors at RaySearch Laboratories: Fredrik Löfman, Mats Holmström
Supervisor at KTH: Henrik Hult
Examiner at KTH: Henrik Hult

TRITA-SCI-GRU 2018:232
MAT-E 2018:40

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Abstract

This Master thesis focuses on generative models for medical patient data for radiation therapy. The objective with the project is to implement and investigate the characteristics of a Variational Autoencoder applied to this diverse and versatile data. The questions this thesis aims to answer are: (i) whether the VAE can capture salient features of medical image data, and (ii) if these features can be used to compare similarity between patients. Furthermore, (iii) if the VAE network can successfully reconstruct its input and lastly (iv) if the VAE can generate artificial data having a reasonable anatomical appearance. The experiments carried out conveyed that the VAE is a promising method for feature extraction, since it appeared to ascertain similarity between patient images. Moreover, the reconstruction of training inputs demonstrated that the method is capable of identifying and preserving anatomical details. Regarding the generative abilities, the artificial samples generally conveyed fairly realistic anatomical structures. Future work could be to investigate the VAEs ability to generalize, with respect to both the amount of data and probabilistic considerations as well as probabilistic assumptions.

Keywords: Variational Autoencoder, Feature extraction, Deep learning on medical images, Computed tomography, Radiation therapy

Sammanfattning

Fokuset i denna masteruppsats är generativa modeller för patientdata från strålningsbehandling. Syftet med projektet är att implementera och undersöka egenskaperna som en “Variational Autoencoder” (VAE) har på denna typ av mångsidiga och varierade data. Frågorna som ska besvaras är: (i) kan en VAE fånga särdrag hos medicinsk bild-data, och (ii) kan dessa särdrag användas för att jämföra likhet mellan patienter. Därutöver, (iii) kan VAE-nätverket återskapa sin indata väl och slutligen (iv) kan en VAE skapa artificiell data med ett rimligt anatomiskt utseende. De experiment som utfördes pekade på att en VAE kan vara en lovande metod för att extrahera framtydande drag hos patienter, eftersom metoden verkade utröna likheter mellan olika patienters bilder. Dessutom påvisade återskapningen av träningsdata att metoden är kapabel att identifiera och bevara anatomiska detaljer. Vidare uppvisade generellt den artificiellt genererade datan, en realistisk anatomisk struktur. Framtida arbete kan bestå i att undersöka hur väl en VAE kan generalisera, med avseende på både mängd data som krävs och sannolikhetsteorietiska avgränsningar och antaganden.

Acknowledgements

First and foremost I would like to thank my supervisors Henrik, Fredrik and Mats for their invaluable support and input during the work of this thesis. I would also like to express my gratitude to my colleagues Marcus and Jonas at *RaySearch Laboratories*, without your eagerness and commitment to discuss and ponder on questions, this thesis would not have been as giving and exciting as it has been. Furthermore, I extend thanks to my parents and grandparents, for their incessant assistance of my education and for teaching me to have diligence in all matters regarding knowledge. Lastly, I would like to thank my husband for introducing me to the field of engineering and for his inestimable support and cheers in everything I take on, life would not be the same without you.

Stockholm, Juni 2018

Hanna

Contents

Chapter 1

Introduction

The occurrence of cancer is increasing globally. In the *World Cancer Report*, compiled by WHO, it is reported that in industrialized countries more than one in four will pass away due to the disease [54]. And by 2030, *major cancers* [1], are expected to have risen by 68% relative to the levels 2012; given that current trends are sustained globally [8, 17]. Consequently, each and every one of us will most likely encounter pain and other implications following cancer, either by being a patient or knowing a person ill of the disease. This presents increasingly great challenges for healthcare worldwide, since resources must meet a growing need for specialized care. The treatment of cancer can be either *curative*, *adjuvant*, *neo-adjuvant* or *palliative* [38]. The curative treatment aims at curing the disease whilst the palliative aims at relieving the patient from the effects of the illness. The adjuvant treatment aims at preventing illness recurrence, whilst the neo-adjuvant aims at reducing tumor size prior to other types of treatment. The nature of the cancer as well as the objective of the treatment determines the measures taken to care for the patient. Commonly, treatment will be comprised of a number of different actions, such as: *surgery*, *hormone therapy*, *chemotherapy* and *radiation therapy* (RT).

In 1895 Röntgen's discovery of X-rays drastically changed the landscape of medical care for both *malignant* and *benign* disorders [18, 51]. As a result, RT-treatment was enabled and has thence been used for cancer treatment in over a century [18]. Nowadays, approximately 50% of cancer patients attain RT as part of their treatment [12]. The RT consists of administration of ionizing radiation. The objective is to damage or kill malignant cells. This ultimately reduces the number of such cells and the pace of their future growth [57].

[1]cancers that are fatal if left untreated

Good RT-treatment plans are essential for the patient since the radiation may come with side effects directly related to the distributed dose. However, the time consuming and intricate task of designing RT plans presents difficulties, such as competing clinical priorities. For instance, one might wish to deliver a higher dose to the malignant cells than the healthy surrounding tissue can cope. A cancer tumor is usually referred to as *target* and surrounding organs which may be effected by the radiation as *organs at risk* (OARs). Conventionally, a specialist examines the patient image and depicts the biological structures of special interest in RT, usually referred to as the *RT structure set*, consisting of e.g. these OARs and target. This is however a task requiring not only time but also the resources of an expert medical professional. Furthermore, the RT planning process includes not only the *oncologist*, but also the *dosimetrists* and *medical physicist*, which illuminates some of the challenges that RT planning presents [38]. Additionally, both between and within institutions the planning procedure varies. This results in a variability in practice and quality of RT [41]. As a consequence, patients may be treated with sub-optimal RT plans [43]. As a result of these challenges and the growing burden, there is an increasing need for standardized and automated health care methods; resulting in development of statistical and machine learning (ML) applications that automatically infer new RT plans based on historical treatment plans or automatically creates a RT structure set from a patient image. The objective is that these automated schemes will save both time and resources, as well as enabling better and more equal treatment.

A cornerstone of this development is statistical methods and their progress. Let us contemplate the customary ML techniques applied in RT-treatment. Considering previously developed "[RT] planning pipelines [they] have traditionally incorporated historical treatment planning data with algorithms to estimate dose-volume objectives based on a limited number of features"[2] [37]. These dose-volume objectives (DVOs), characterize what dose of radiation that is to be accumulated in the patient's volumes of interest, and it is related to the *Dose Volume Histogram* (DVH), which conveys the dose level achieved in a fraction of a volume. In [1] they propose a statistical method that "improve[s] treatment plan quality using mathematical models that predict achievable organ-at-risk [...] dose volume histograms [...] based on individual patient anatomy" [3]. And in [58] they investigate an application for automatic treatment planning for *head-and-neck* cancer based on volume histograms. Moreover, in [37] they propose a method that predicts a spatial dose volume objective, i.e. a dose to be administered in each specific volume element, as supposed to a certain amount in a volume. This procedure

[2] [ ] denotes that the quote has been altered with this entity

[3] [...] denotes that part of the quote is omitted

predicts a RT dose, based on the *Computed Tomography* scan (CT-scan), of the novel patient. This is achieved by means of *regression forest* and *conditional random fields*. Now considering previously developed schemes for automatic segmentation, some works have applied a *Markov Random Field* (MRF) model [46] while more novel approaches applies e.g. the classical *Expectation Maximization* (EM) algorithm [48]. Moreover, in [11] they have used non-parametric classifiers for brain tissue. Hence, classical statistical methods are a common practice in these automated applications. Recently more modern applications has however been developed. As in [61] where they "propose to combine deep learning and marginal space learning (MSL) [...] for robust kidney detection and segmentation". Or as in [42] where they "have modified a convolutional deep network model, U-net [...], for predicting [RT] dose from patient image contours".

Let us now consider this statistical learning in more detail. In the 1960's statistical learning was first introduced, initially "it was a purely theoretical analysis of the problem of function estimation from a given collection of data" [56]. During the following decades the concept evolved from theory to algorithms, rendering the area applicable in both analysis and algorithmic estimation of functions [56]. Today the field of statistical learning is commonly multidisciplinary with applications in almost any field; such as finance, biology, health care, military, law enforcement, public policy and astrophysics [23]. The overall principle is to make use of data and knowledge in the field, to perform inference using some mathematical tool or reasoning, and ultimately gain insight and understanding of the data in question [23]. One may wish to predict outcomes, reduce dimensions or to classify, using *frequentist-* or *Bayesian methods*. Including *regression*, *tree based methods*, *neural nets*, *kernel methods*, *sampling*, *principal component methods* or expectation maximization and *approximate inference*. The last couple of years "the techniques developed from deep learning research have [...] been impacting [...] key aspects of machine learning and artificial intelligence"[13]. The concept of *deep learning* stems from the fact that the logic behind the class of methods is to render understanding by hierarchies of concepts. Where simpler concepts are connected successively to more abstract ones. Considering this structure in a graph, many layers will yield a deep structure, hence the name [19]. Applications built on deep learning have enabled large improvements in domains such as *speech recognition*, *visual object recognition*, *object detection* and *genomics* [32]. One key aspect in this development could be that deep learning "discovers intricate structure[s] in large data sets" [32]. Consequently, deep learning is assumed to gain ground and progress in new learning algorithms [32]. This presents promising new opportunities for development of cutting edge methods in health care and more specifically in applications regarding RT-planning and rendering of RT-structure sets.

1.1 Problem Statement

A common characteristic of the data used for ML in radiation therapy is that it is quite high dimensional. Feature extraction by dimensional reduction is commonplace regarding high dimensional data, and it is a common practice prior to inference. Considering this concept in relation to automation of methods in RT-treatment it is a fact that one vital aspect of automated RT-planning is the choice of data used to infer a new plan for a patient. In order to derive a plan one must be able to compare a novel patient with previous patients, and thence infer the plan based on the RT treatments for the historical patients with characteristics most favourable for prediction. This is achievable by comparison of the features of the patient images and the structure data; i.e. data for organ segmentation, consisting of an entity that demonstrates if a volume element in a patient image is within an organ or not. This image could come from *Magnetic Resonance Imaging* (MRI) or CT as in this thesis. The patient data is however high dimensional and quite extensive; the CT-scan is conventionally comprised of approximately 100 slices, where each slice has $(512, 512)$ pixels. Subsequently, the image data of a patient is of order 10^7 pixels. Thus, in order for comparison to be feasible one must extract prominent features somehow and thence reduce the dimension prior to inference. Some classical statistical methods for dimension reduction are e.g. various Gaussian filters such as the *Leung-Malik filter* [33] or *Principal Component Analysis* (PCA) [26]. However, the development of new methods has enabled usage of other techniques involving deep-learning for this purpose.

The characteristics chosen for this comparison of patient data will directly influence the quality of the inferred plans. And to derive a treatment plan of high quality one must extract the most salient features of the data and use these to choose the patients used in the inference. This presents two challenges: (i) how is one to derive the features of the patient data, and (ii) what metric should be used for the comparison of the features between patients?

Lastly, the amount of data available to train the ML models will have an effect on the predictive strength in the inference. If one is to automate generation of the RT structure set, by using deep learning applications, this will probably demand large amounts of training data. Hence, access to data for training is vital for both the treatment planning process and automatic generation of segmented data. And this matter is not trivial, since there are ethical and legal considerations regarding handling of patient data. Hence, the acquisition of training sets can present numerous challenges for the usage of ML models in health care.

1.2 Purpose

The purpose of this thesis is to implement and investigate probabilistic characteristics of deep generative models on medical patient data. Namely by considering the ability to extract prominent features of high dimensional data and to generate artificial patient data. This is done by applying probabilistic generalizations of autoencoders on CT-scans as well as organ segmentation data. This thesis may constitute a contribution towards enabling usage of these cutting edge methods in health care and more specifically in future applications for treatment of cancer.

1.3 Methodology

In this thesis the twofold purpose of both extracting salient features of high dimensional data and generating artificial data is made achievable by the deep learning application called a *Variational Autoencoder* (VAE). In some aspects this tool can be seen as a powerful stochastic generalization of PCA. Moreover, the VAE is a versatile and powerful technique where one can design and modulate the level of complexity in the architecture of the method. Hence the VAE is chosen for the investigation and work in this thesis.

1.4 Questions

The questions this thesis investigates are

1. How well can a VAE reconstruct an input consisting of medical data, such as a CT-scan?

2. How well can a VAE capture the salient features of a CT-scan and its adherent segmented organs?

3. How well can the representation of the patient data attained with a VAE be used for identifying similarity between patients?

4. How well can new artificial patient data be created by sampling from the decoder in the VAE network?

The outline of this thesis goes as follows: Chapter 2 is comprised of some basic aspects of *Statistical learning.* In Chapter 3 *Neural Networks* and adherent methods and concepts are discussed, as a theoretical prelude to the network structure of the VAE. This is followed by *Autoencoders* and related topics in Chapter 4. *Data* and *Implementation* are elaborated on in Chapter 5 and 6. Lastly, Chapter 7 and 8 are comprised of *Results* and *Discussion.*

Chapter 2

Statistical learning

The multidisciplinary field of statistical learning has as previously mentioned contributed immensely to a variety of fields. The notion of "[s]tatistical learning refers to a vast set of tools for understanding data" [23]. And this is highly central since a wave of data has swept into virtually all industries and it has become a key aspect along with *capital* and *labour* [36]. To exemplify this phenomenon, US healthcare alone could create more than \$300 billion in value yearly simply by adopting a creative and efficient usage of *big data* [36]. And since the amount of data is constantly increasing a need for automation of data analysis techniques come along with this shift [40].

A few central elements of this statistical learning will be discussed in this Chapter, for future reference. Namely, *supervised* and *unsupervised learning*, as well as considerations regarding generalization connected to *training* and *testing* statistical learning models.

2.1 Supervised vs unsupervised learning

In statistical learning the setting can be either supervised, unsupervised or *semi-supervised*. In the supervised case data $\mathcal{D} = \{\mathbf{x}_k, y_k\}_{k=1}^{K}$ is given, where there is a function $f : \mathbf{x} \to y$; entailing that both *predictors* $\mathbf{x}_k \in \mathbb{R}^N$ and a *response variable*, or *key*, y_k is prevalent. The response could be *categorical* or *nominal* s.t. $y_k \in \{1, \ldots, C\}$, or it could be a scalar s.t. $y_k \in \mathbb{R}$ [40]. Here *quantitative* or *qualitative* prediction is often of interest, e.g. to extrapolate in regression to predict the value of a real valued scalar response or to assign categorical labels y_k according to the predictors using e.g. *support vector machines*. For the unsupervised case on the other hand, the data available will be comprised of only $\mathcal{D} = \{\mathbf{x}_k\}_{k=1}^{K}$, entailing that no response variable y_k is provided. Consequently, the objective of the inference is ambiguous, unlike in the supervised setting. In other words, "[in] contrast to supervised learning, the objective of unsupervised learning is to discover patterns or features in the input data with no help" [16]. Consequently, prediction is not of interest, but the objective is rather to identify salient features and interesting qualities of the data. One such pattern of interest could be whether there are subsets of the data $\mathcal{D}$ that have similar characteristics and what features these groups demonstrate; this is usually called *clustering* of the input. One additional application may be dimensionality reduction by PCA [16]. The semi-supervised case has some predictors with response variables given. This entails that "[s]emi-supervised learning uses both labeled and unlabeled data to perform an otherwise supervised learning or unsupervised learning task" [62]. The setting of this thesis is unsupervised, since inference is to be performed on patient data from RT- therapy, with no response is given.

2.2 Training and testing in statistical learning

Some statistical methods can be powerful enough to identify and subsequently learn details unique for a specific data set. This phenomena is called *overfitting*, since the model will inherently encode the noise prevalent in the data. And as a consequence the model fits the training data "too well" and thence it will not perform as well on new unseen data, entailing that the method does not generalize. Conventionally this problem is counteracted with *regularization*, extensive and diverse data sets for training, *early stopping* during training or choice of less *flexible* models. A method of verifying whether a method has overfitted is to hold out some data during training, referred to as the *test set*. And then apply the model on this unseen data, if the training performance is superior to that of the test, one can expect that overfitting has occurred. If there is data enough one could also create a separate *validation set*, in this setting the test set can be used for hyperparameter modulation and model selection following training and the validation set can then quantify the chosen models ability to generalize.

Chapter 3

Neural Networks

3.1 Feedforward Neural Networks

Feedforward Neural Networks (FNNs) are an essential building block in many applications of ML. However, they were initially developed in the quest of modelling the neural system of the brain. These networks are comprised of a set of neurons entwined in a type of *Directed Acyclic Graph*, (DAG) [19]. This DAG is an ordered graph consisting of a number of vertices, whose edges are connected from the start of the graph to later in the sequence, i.e. the vertices have a *topological sorting* [55]. The *Feedforward* notion springs from the fact that the input $\mathbf{x}$ of the FNN is pushed forward thought the network, starting at the input nodes and thence rendering the output by the computations applied in the network [19]. Generally, in a supervised setting a FNN has the objective of estimating some mapping $f : \mathbf{x} \to y$, given training data $\mathcal{D} = \{\mathbf{x}, y\}$. This is achieved by deriving the parameters γ of an approximate mapping f_γ yielding the best estimate of f, e.g. in the *least squares sense.* In Figure 3.1 one can see in a DAG how such a mapping f_γ can be comprised in a network with three hidden layers.

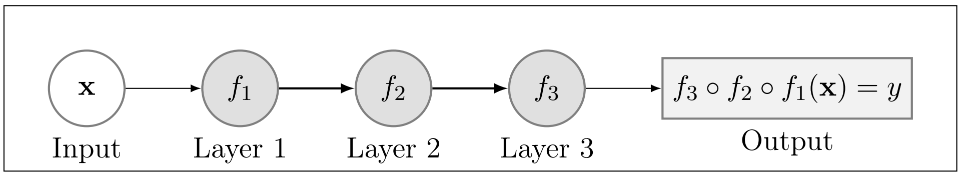


Figure 3.1: A visualization of a FNN with three hidden layers. The output is derived with the mapping $f_\gamma = f_3 \circ f_2 \circ f_1(\mathbf{x}) = y$, where $\circ$ denotes how the functions operate s.t. $h \circ g = h(g)$.

In a network setting the FNN shown in Figure 3.1 can have the architecture demonstrated in Figure 3.2. This network has tree hidden layers that are fully connected, i.e. all nodes in layer $\{l-1\}$ are connected to the nodes in $\{l\}$.

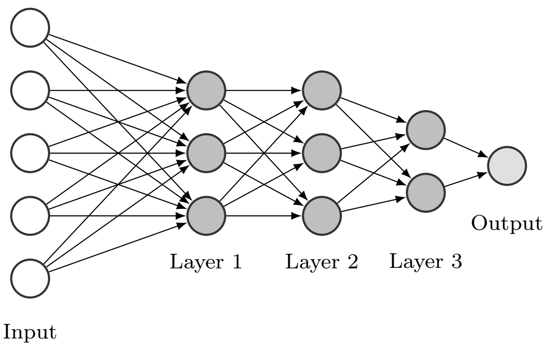


Figure 3.2: A fully connected FNN with three hidden layers.

Moreover, let the following entities be defined according to,

$$\begin{aligned} p_l &\ \text{number of vertices in layer } l,\\ b_j^l &\ \text{bias for layer } l,\text{ vertex } j,\\ \mathbf{w}_j^l = \{w_{j1}^l, \dots, w_{jp_{l-1}}^l\} &\ \text{weights for layer } l,\text{ vertex } j,\\ a^l = \{a_1^l, \dots, a_{p_l}^l\} &\ \text{outputs for layer } l. \end{aligned}$$

In our network the n-dimensional input $\mathbf{x}$ is transformed in the neuron by the weight vector $\mathbf{w}$ and bias b. The input layer is initialized by setting that the outputs equals the input data, i.e. $a_j^0 = x_j, \forall j \in \{1, 2, \dots, N\}$. The transformation in perceptron i is then performed according to

$$z_j^l = \mathbf{w}_j^l \cdot a^{l-1} + b_j^l, \quad \forall j \in \{1, 2, \dots, p_l\}.$$

An *activation function* $g_l(\cdot)$ is then applied and the neuron outputs the entity

$$a_j^l = g_l(z_j^l), \quad \forall j \in \{1, 2, \dots, p_l\}$$

to the subsequent vertices. In Figure 3.3 it is visualized how the input is weighted and subsequently summed with the node bias, thereafter the activation function is applied and the output is attained. This transformation occurs at all vertices in the hidden layers of the FNN. The output layer of the network will thereafter yield a^L, which in the supervised setting discussed earlier would correspond to the estimate $f_\gamma(\mathbf{x}) \approx y$. And where the parameters γ are given by the weights and biases $\{\mathbf{w}, b\}$ of the network.

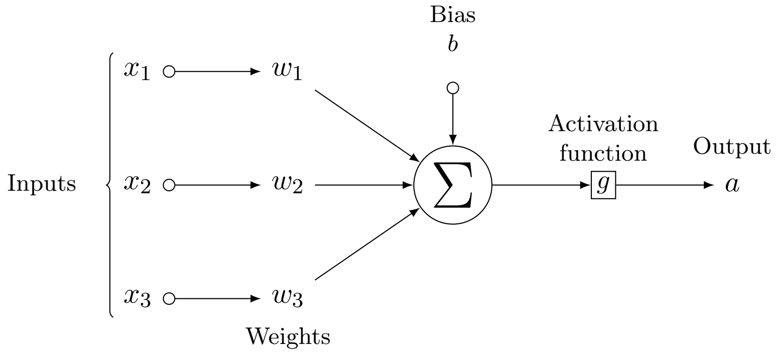


Figure 3.3: A visualization of the transformations in a neuron. The inputs are weighted, bias is added and the activation is subsequently applied.

3.1.1 Training of a Neural Network

Let us now consider the training or a FNN with $\{L-1\}$ hidden layers. As previously presented, the parameters $\{\mathbf{w}^l, b^l\}_{l=1}^L$, will in conjunction with the activation functions $\{g_l(\cdot)\}_{l=1}^L$ and the adherent architecture, characterize the output of the network. The problem at hand is to derive $\{\mathbf{w}^l, b^l\}_{l=1}^L$ such that the network performs its intended task well. This can be achieved by means of *Gradient descent*, (GD).

Loss function

In order to measure how well the FNN performs, one defines a *loss function* $\mathcal{L}(\mathbf{w}, b)$. Supposing we have a supervised setting, we could compare the output $a^L = f_\gamma(\mathbf{x})$ of the FNN for input $\mathbf{x}$ with corresponding response y. If the network renders an output close to the true value the objective is achieved. A loss function capturing this relation could be the *Mean Square Error* (MSE), defined according to

$$\mathcal{L}(\mathbf{w}, b) = \frac{1}{2K} \sum_{k=1}^{K} \left\| y_k - a^L(\mathbf{x}_k) \right\|^2. \tag{3.1}$$

Above it is assumed that there are K training pairs of data available, i.e. $\{\mathbf{x}_k, y_k\}_{k=1}^K$ and that $a^L(\mathbf{x}_k) = f_\gamma(\mathbf{x}_k)$ is the network's output for data $\mathbf{x}_k$.

Gradient descent

Now assume the network parameters are given by $\mathbf{v} \in \mathbb{R}^H$, then the loss function $\mathcal{L}(\mathbf{v})$ can be thought of as a billowy surface in the H-dimensional parameter space. The objective when training the network is to find the set of $\mathbf{v} \in \mathbb{R}^H$ that minimizes $\mathcal{L}(\mathbf{v})$. Hence, from a topological reasoning we wish to find the point in the parameter space yielding the lowest loss, positionally corresponding to the deepest valley of the ”functional landscape” given by $\mathcal{L}(\mathbf{v})$ [44]. One method of rendering the location of this point is gradient descent. The idea is that you start somewhere on the surface, you find the local shape around the current position and move in a direction that reduces the size of the loss function mostly. This is then repeated until either a *local* or *global minimum* is reached. In Figure 3.4 such a minimizing trajectory is visualized on level curves for $\mathbf{v} \in \mathbb{R}^2$.

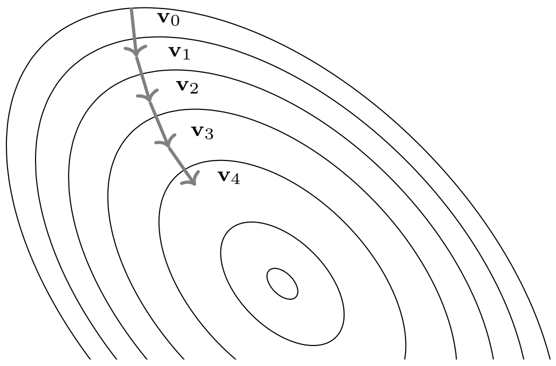


Figure 3.4: A visualization of level curves of the loss function $\mathcal{L}(\mathbf{v})$ for $\mathbf{v} \in \mathbb{R}^2$. The initial parameters are given by $\mathbf{v}_0$, the parameters are then iteratively updated as the parameter position moves downhill in the level curves, thence minimizing $\mathcal{L}(\mathbf{v})$.

Formally, gradient descent entails computation of derivatives of $\mathcal{L}(\mathbf{v})$ w.r.t. $\mathbf{v}$ and motion in the direction of the largest negative change.

Let $\Delta\mathbf{v}$ denote the change in the parameters such that

$$\Delta\mathbf{v} = (\Delta v_1, \Delta v_2, \ldots, \Delta v_H)^{\mathrm{T}}.$$

Furthermore, let the gradient of the loss be defined according to

$$\nabla\mathcal{L}(\mathbf{v}) = \left(\frac{\partial\mathcal{L}(\mathbf{v})}{\partial v_1}, \frac{\partial\mathcal{L}(\mathbf{v})}{\partial v_2}, \ldots, \frac{\partial\mathcal{L}(\mathbf{v})}{\partial v_H}\right)^{\mathrm{T}}, \tag{3.2}$$

this entity will convey how a change in a parameter relates to a change in the loss. One can approximate the change of the loss according to

$$\Delta\mathcal{L}(\mathbf{v}) \;\approx\; \nabla\mathcal{L}(\mathbf{v})\Delta\mathbf{v} \;=\; \sum_{h=1}^{H} \frac{\partial\mathcal{L}(\mathbf{v})}{\partial v_h}\Delta v_h,$$

i.e. as a decomposition of the change in loss with respect to each parameter and the change for each respective parameter. One method of identifying a direction in which the loss is reduced is to set

$$\Delta \mathbf{v} \;=\; -\eta \nabla \mathcal{L} \quad \rightarrow \quad \Delta \mathcal{L} \;=\; -\eta \nabla \mathcal{L} \nabla \mathcal{L} \;=\; -\eta \left\| \nabla \mathcal{L} \right\|^2 ,$$

where $\left\| \nabla \mathcal{L} \right\|^2 \geq 0$, and η is refereed to as the *learning rate* [44]. An alteration of the parameters according to $\Delta \mathbf{v}$ would entail a reduction in the objective function $\forall \left\| \nabla \mathcal{L} \right\|^2 > 0$. Thus, the parameters are updated iteratively according to

$$\mathbf{v}_{i+1} \;=\; \mathbf{v}_i - \eta \nabla \mathcal{L}(\mathbf{v}_i)$$

until a minimum is reached [44]. One can discern that the hyperparameter η will modulate the step size taken by the optimization algorithm and as such it will influence the pace at which the parameters are learned, it has thence earned its epithet. Consequently, for weights and biases of the network one updates the parameters according to

$$\mathbf{w}_{i+1} \;=\; \mathbf{w}_i - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} \quad \text{and} \quad b_{i+1} \;=\; b_i - \eta \frac{\partial \mathcal{L}}{\partial b_i}.$$

Stochastic gradient descent

Let us now reconsider the loss function (3.1), upon inspection it is evident that $\mathcal{L}(\mathbf{w}, b)$ can be decomposed according to

$$\mathcal{L}(\mathbf{w}, b) \;=\; \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_{\mathbf{x}_k}(\mathbf{w}, b) \;\; \text{where} \;\; \mathcal{L}_{\mathbf{x}_k}(\mathbf{w}, b) \;\triangleq\; \frac{1}{2} \left\| y_k - a^L(\mathbf{x}_k) \right\|^2 . \tag{3.3}$$

Consequently, the derivation of the gradient (3.2) will involve computation of

$$\nabla \mathcal{L}_{\mathbf{x}_k}(\mathbf{w}, b) \;\; \forall \;\; \mathbf{x}_k \in \{\mathbf{x}_1, \ldots, \mathbf{x}_K\},$$

and subsequently the gradient is computed by averaging over the gradient of each training input. If K is large this involves computations of a considerable number of gradients for one iteration in the GD algorithm. This can be computationally unfeasible, hence to remedy this issue *Stochastic Gradient Descent* (SGD) may instead be applied. This version of GD will not utilize the entire training data set for the computation of the gradients, it will rather choose a *mini batch* of S samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_S\}$ that are randomly drawn from the training set and approximate the gradient

$$\nabla \mathcal{L}(\mathbf{w}, b) \;\approx\; \frac{1}{S} \sum_{s=1}^{S} \nabla \mathcal{L}_{\mathbf{x}_s}(\mathbf{w}, b).$$

Thence, the network parameters can be updated according to,

$$\begin{aligned} \mathbf{w}_{i+1} \;&=\; \mathbf{w}_i - \frac{\eta}{S} \sum_s \frac{\partial \mathcal{L}_{\mathbf{x}_s}}{\partial \mathbf{w}_i}, \\ b_{i+1} \;&=\; b_i - \frac{\eta}{S} \sum_s \frac{\partial \mathcal{L}_{\mathbf{x}_s}}{\partial b_i}. \end{aligned}$$

A new mini batch is subsequently chosen, and the procedure is repeated until all samples in the training data set have been used once; then one *epoch* of training is said to have been executed. Generally, in the SGD algorithm training is comprised of a large number of epochs. If one however has data of an extensive dimension, such that a number of samples S would entail too large of a set for the computations, *incremental learning* is an option. Here one simply sets $S = 1$ and perform SGD on one training sample at a time [44].

Backpropagation

One cornerstone of the presented GD algorithms is the computation of the gradients of the loss w.r.t. to parameters, one method of deriving these entities is *backpropagation* (BP). The loss function $\mathcal{L}$ must fulfill the following two conditions in order to make backpropagation feasible [44]

1. The loss must be a function of the output a^L of the network.

2. One must be able to decompose the loss as a sum over the loss $\mathcal{L}_{\mathbf{x}_k}$, of individual training data.

If one studies the loss (3.3) it is evident that the function is comprised of contributions given by each training input, and furthermore the loss is a function of $a^L(\mathbf{x}_k)$. Hence, both requirements for back propagation are fulfilled in this supervised setting and for the chosen loss function.

Error in layer l

Let us begin with contemplating the error in each layer of the network, i.e. the extension of the loss in each part of FNN. Suppose the error in layer l and neuron j is given by

$$\delta_j^l = \frac{\partial \mathcal{L}}{\partial z_j^l}, \tag{3.4}$$

i.e. a measure on how the loss changes with respect to the weighted input to the vertex. If the layer in question is the output s.t. $l = L$, one can re-express (3.4) with the chain rule according to

$$\delta_j^L = \frac{\partial \mathcal{L}}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial \mathcal{L}}{\partial a_j^L}\frac{\partial g(z_j^L)}{\partial z_j^L} = \frac{\partial \mathcal{L}}{\partial a_j^L} g'(z_j^L).$$

Which can be written in vector form

$$\delta^L = \nabla_a \mathcal{L} \odot g'(z^L) \quad \text{for} \quad \nabla_a \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial a_1^L}, \ldots, \frac{\partial \mathcal{L}}{\partial a_{p_L}^L}\right).$$

Where $\odot$ denotes the *Hadamard product*. By propagating δ^L back through the network the error in an arbitrary layer l can be derived according to

$$\delta^l = ((\mathbf{w}^{l+1})^{\mathrm{T}} \delta^{l+1}) \odot g'(z^l). \tag{3.5}$$

Entailing that the weights that interconnect layer $l \to (l+1)$ will reflect how the error evolves through the network, in conjunction with the change in the activation for the weighted input of the layer [44]. The error quantities hereby defined will, as a matter a fact, characterize the gradient of the loss with respect to the parameters of the network, according to

$$\frac{\partial \mathcal{L}}{\partial b_j^l} = \delta_j^l \qquad \text{and} \qquad \frac{\partial \mathcal{L}}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l.$$

Consequently, the gradient with respect to any parameter in the network can be computed with (3.5), and GD can subsequently be applied. In Algorithm 1, BP can be seen, comprised of a *forward pass* where the weighted inputs and outputs to each vertex is derived, followed by the *backward pass* where the gradients are computed.

```
Result: Backpropagation for computation of gradients
For each update of parameters $\{\mathbf{w}, b\}$ in GD algorithm;
Set activation for input layer $\to a^0$
for $l = 1 : L$ do
    $z^l = \mathbf{w}^l a^{l-1} + b^l$
end
Compute $\delta^L = \nabla_a \mathcal{L} \odot g'(z^L)$;
for $l = (L-1) : 1$ do
    $\delta^l = (\mathbf{w}^{l+1})^{\mathrm{T}} \delta^{l+1} \odot g'(z^l)$
end
Compute $\frac{\partial \mathcal{L}}{\partial b_j^l} = \delta_j^l$ and $\frac{\partial \mathcal{L}}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$.
```

Algorithm 1: Backpropagation in a FNN.

3.1.2 Activation functions

Let us now consider activation functions $g_l(\cdot)$. These operators will modulate the extent to which each weighted input in a vertex is propagated further through the network, i.e. they will decide whether each neuron fires or not, in addition to the strength of the signal.

Heaviside step function

One activation function is the *Heaviside step function*

$$H(z) = \begin{cases} 1 \text{ if } z > 0 \\ 0 \text{ else .} \end{cases}$$

The neuron will consequently be activated and fire only for positive values of the weighted input. However, the application of the FNN might require more nuances than this binary activation. Additionally, GD which is conventionally used to find the optimal network parameters γ of a FNN, generally requires a differentiable activation. The gradient of the Heaviside step function gives rise to problems with learning since $\partial H(z)/\partial z = 0 \ \forall \ z \neq 0$.

Sigmoid

One alternative is the *sigmoid* $\sigma(\cdot)$, characterized by

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

This activation is differentiable $\forall z$ and will render outputs in $[0, 1]$. One might think of the function as a saturating force on the input of the vertex. Nonetheless, the sigmoid, which may be seen in Figure 3.5, can cause slow learning of the network parameters. Due to the fact that *saturated neurons* with weighted input z far from the origin, will reduce the gradients of the sigmoid w.r.t. z i.e. $\partial\sigma(z)/\partial z \approx 0 \ \forall \ |z| \geq 10$. This will cause problems when applying GD for learning.

Hyperbolic tangent

Yet another activation is the *hyperbolic tangent* $\tanh(\cdot)$, given by

$$\tanh(z) = \frac{e^{z} + e^{-z}}{e^{z} - e^{-z}}.$$

This activation will instead render outputs $\in [-1, 1]$. This zero-centered characteristic is beneficial since the gradients may then have variable signs for inputs where all data is either positive or negative; which enables faster learning with GD. However the issue with saturated neurons "killing" gradients is still occurring, in Figure 3.5 one can see how inputs $|z| \geq 2$ renders a near non-existing gradient of the activation w.r.t. the input z, since the function becomes almost constant.

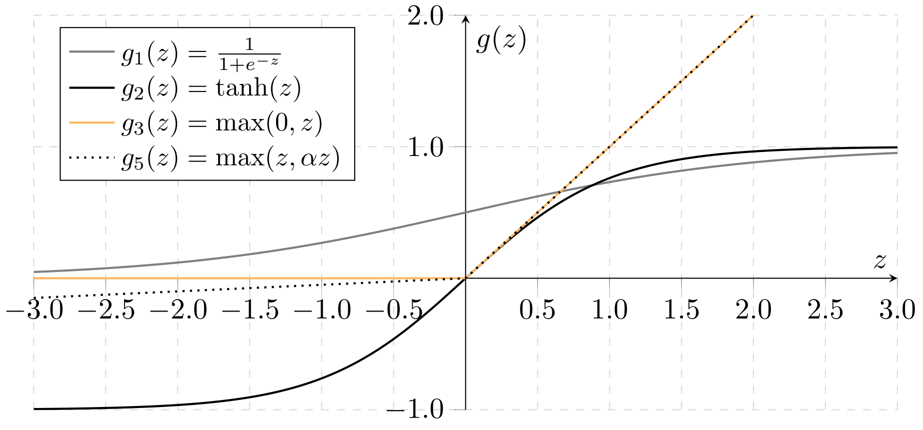


Figure 3.5: Visualization of activation functions commonly used in feed forward networks.

Rectified Linear Unit

Conversely, an activation that does not saturate in $\mathbb{R}^+$ is the computationally efficient *Rectified Linear Unit* (ReLU) given by

$$\text{ReLU}(z) = \max\big(0, z\big).$$

This function will enable fast convergence of network parameters, since the gradients are not killed by the activation in large regions of the plausible input space. Nonetheless, in the region $\mathbb{R}^-$ the problem stills stands, which can be clearly seen in Figure 3.5, since the activation is constant for all values. This will practically implicate that data $\in \mathbb{R}^-$ will belong to "dead" ReLU nodes, that will not contribute to updates of the weights during training. This activation is differentiable $\forall z \neq 0$.

Leaky Rectified Linear Unit

One function that can remedy the problem occuring in the ReLU is the *leaky* ReLU activation

$$\text{leaky ReLU}(z) = \max(z, \alpha z).$$

Where α usually is a small positive number, commonly 0.01. As one can see in Figure 3.5 this will entail some leakage for $z \in \mathbb{R}^-$ which counteracts nodes being inactive $\in \mathbb{R}^-$, as with the ReLU.

Furthermore, it is noteworthy that the activation function $g_l(\cdot)$ is generally chosen to be non linear. Since if the activations were linear, the output would be comprised of linear combinations of the input; meaning that several hidden layers in the FNN could be replaced with a single layer.

3.2 Convolutional Neural Networks

A *Convolutional Neural Network* (CNN) is a FNN specialized in handling data with a grid structured topology. They enable architectural encoding of these properties which presents computational advantages [19]. More specifically, inputs such as images or *time series* are well suitable for these networks. There are several important factors that are characterizing for a CNN, namely *sparse interactions*, *parameter sharing* and *equivariant representations* [19]. In a traditional FNN every input node will interact with the final outputs, whilst a CNN will interact sparsely which entails that certain subsets of the input data will render some of the outputs. Moreover, in the ordinary FNN the weights of the network are all applied once in the propagation; whilst in a CNN, the network applies the weights of the kernel at all inputs in a layer. The FNN will hence have to learn a substantial set of unique parameters for each position of the input, when a CNN only has to render one set. Furthermore, the CNN is equivariant in its representation, which entails that a displacement of the input will render the equivalent displacement in the output. However, there are transformations such as rotations or scaling that convolution is not equivariant to, hence these actions must be handled in another manner [19].

Convolutional layer

A convolutional layer will replace the matrix multiplication from the ordinary FNN with a convolution. Mathematically, discrete convolution of an *input* $x(i)$ and a *kernel* $w(a)$ is performed according to

$$s(i) = (x * w)(i) = \sum_{a=-\infty}^{\infty} x(a)w(i-a),$$

resulting in a *feature map* $s(i)$. One feature map will reflect where in the input a certain feature is prevalent. Hence the kernel for one specific map will become a detector for that one characteristic. For a two dimensional input, such as an image $I \in \mathbb{R}^{H\times W\times C}$, where C is the number of color channels, H the height and W the width; the discrete convolution is performed with a kernel $K \in \mathbb{R}^{k_1\times k_2\times C}$ and bias $b \in \mathbb{R}^1$ according to

$$s(i,j) = (I * K)(i,j) = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} K_{m,n} I_{i-m,j-n} + b. \qquad (3.6)$$

For ease of notation in (3.6) it is supposed that I is in gray scale i.e. that the channel $C = 1$. In Figure 3.6 such a convolution is visualized in a small scale example. For an image $I \in \mathbb{R}^{H\times W}$ and weight kernel $K \in \mathbb{R}^{k_1\times k_2}$ the resulting feature map fulfills $s \in \mathbb{R}^{(H-k_1+1)\times(W-k_2+1)}$.

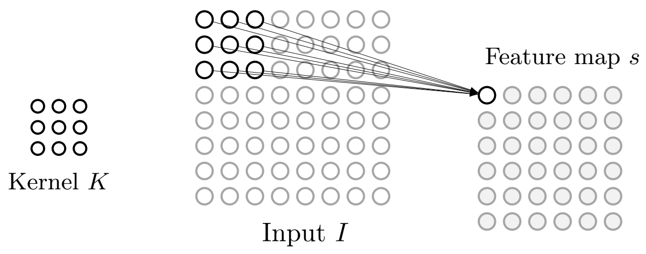


Figure 3.6: A visualization of a convolution between the kernel and input. Each element in the feature map is rendered as the kernel sweeps over the input, analogous to how the first element of the map is found, indicated by the arrows. In this example the *stride* S_t equals one.

When performing the convolution numerically, *cross correlation* with a flipped kernel is conventionally applied, entailing that the kernel is rotated 180 degrees and the *Hadamard product* of a subset of I and the flipped kernel is computed. Then the entries in the matrix are summed to derive an entry in the feature map, yielding the subsequent form of (3.6), where K is assumed to have been rotated

$$s(i,j) = (I * K)(i,j) = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} K_{m,n} I_{i+m,j+n} + b. \tag{3.7}$$

Moreover, let the following entities be defined according to,

b^l	bias for layer l,
$w^l_{m,n}$	filters connecting layer l and $l-1$,
z^l_{ij}	weighted input to layer l,
a^l_{ij}	outputs for layer l.

Where the weighted input and output are derived according to,

$$z^l_{ij} = \sum_m \sum_n w^l_{m,n} a^{l-1}_{i+m,j+n} + b^l, \tag{3.8}$$

$$a^l_{ij} = g(z^l_{ij}) \quad \text{and} \quad a^0_{ij} = I. \tag{3.9}$$

In Figure 3.6 it can be seen how the kernel sweeps over the first subset of I which renders the first entry in the feature map. This is successively repeated as the kernel moves along the input, with a motion characterized by the *strides*. In Figure 3.6 the stride S_t is one, which entails that the kernel moves one index to derive the next element in the feature map, according to the visualization in Figure 3.7.

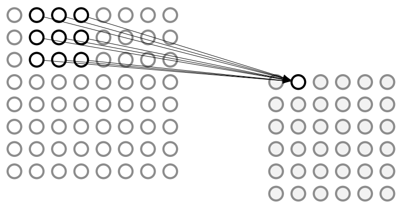


Figure 3.7: A visualization of how the next element in the feature map is generated, relative to Figure 3.6.

Furthermore, "strides constitute a form of *subsampling* [...] [they modulate] how much of the output [that] is retained" [15]. Namely, one may increase the strides and attain an analogous feature map with a unit stride where only some of the outputs are kept. The nodes in the input where the kernel sweeps and renders an element in the feature map, is called the *local receptive field* for that element [44]. Subsequent convolutional layers will be comprised of a combination of the receptive fields of the earlier layers, since it is rendered from the previous nodes. Thence, the deeper layers of a CNN will learn more abstract features of the input. Generally, one wants the latter feature maps to intake as much as possible from the previous receptive fields in its vertices, since this will entail that much of the information is preserved between the convolutional layers, much like in a fully connected FNN. In Figure 3.6 it is apparent that the interactions of a convolutional layer is indeed sparse, as only some of the nodes interconnect with each vertex in the next layer. However, this sparsity does not entail that deeper layers do not connect to the input, as there are indirect connections between most nodes with the input I.

When applying convolution to images one generally wishes to identify the most salient features, i.e. several of the localized features predominant in the input data. Since one feature map will reflect but one image characteristic, this entails that numerous maps ought to be rendered in order for several features to be captured. As a result, convolutional layers are commonly comprised of a set of different kernels, rendering a set of varying feature maps.

3.2.1 Backpropagation in Convolutional Neural Networks

To derive optimal filters and biases for a CNN, GD can be used like in a fully connected FNN; where BP can compute the gradients in the GD algorithm. In order to derive the gradients one must contemplate the error in each layer of the network. Let the error in neuron $\{i, j\}$ and layer l correspond to

$$\delta_{ij}^{l} = \frac{\partial \mathcal{L}}{\partial z_{ij}^{l}}. \tag{3.10}$$

Entailing that the change of the loss $\mathcal{L}$ w.r.t. weights in the kernel is given by

$$\frac{\partial \mathcal{L}}{\partial w^l_{\tilde{m},\tilde{n}}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \frac{\partial \mathcal{L}}{\partial z^l_{ij}} \frac{\partial z^l_{ij}}{\partial w^l_{\tilde{m},\tilde{n}}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta^l_{ij} \frac{\partial z^l_{ij}}{\partial w^l_{\tilde{m},\tilde{n}}}. \tag{3.11}$$

Considering (3.8), the partial derivative of the weighted input and the filter can be expressed according to

$$\begin{aligned} \frac{\partial z^l_{ij}}{\partial w^l_{\tilde{m},\tilde{n}}} &= \frac{\partial}{\partial w^l_{\tilde{m},\tilde{n}}} \left(\sum_m \sum_n w^l_{m,n} a^{l-1}_{i+m,j+n} + b^l \right) \\ &= \frac{\partial}{\partial w^l_{\tilde{m},\tilde{n}}} \left(w^l_{\tilde{m},\tilde{n}} a^{l-1}_{i+\tilde{m},j+\tilde{n}} + b^l \right) = a^{l-1}_{i+\tilde{m},j+\tilde{n}}. \end{aligned}$$

Inserting this in (3.11) yields

$$\frac{\partial \mathcal{L}}{\partial w^l_{\tilde{m},\tilde{n}}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta^l_{ij} a^{l-1}_{i+\tilde{m},j+\tilde{n}}. \tag{3.12}$$

Let us now consider δ^l_{ij} further. In order to compute this derivative, given by (3.10), one must know where in the subsequent layer $\{l+1\}$ the entity z^l_{ij} has an effect. For a filter $K \in \mathbb{R}^{k_1 \times k_2}$ the pixel $\{i,j\}$ in layer l will influence parts of the feature map given by $\{(i-k_1+1,i),(j-k_2+1,j)\}$. Thence δ^l_{ij} can be decomposed in the subsequent manner

$$\delta^l_{ij} = \frac{\partial \mathcal{L}}{\partial z^l_{ij}} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \frac{\partial \mathcal{L}}{\partial z^{l+1}_{i-m,j-n}} \frac{\partial z^{l+1}_{i-m,j-n}}{\partial z^l_{i,j}} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta^{l+1}_{i-m,j-n} \frac{\partial z^{l+1}_{i-m,j-n}}{\partial z^l_{i,j}}.$$

Where the partial derivative of the weighted inputs can be expressed using (3.8)

$$\begin{aligned} \frac{\partial z^{l+1}_{i-m,j-n}}{\partial z^l_{i,j}} &= \frac{\partial}{\partial z^l_{i,j}} \left(\sum_{\tilde{m}} \sum_{\tilde{n}} w^{l+1}_{\tilde{m},\tilde{n}} a^l_{i-m+\tilde{m},j-n+\tilde{n}} + b^{l+1} \right) \\ &= \frac{\partial}{\partial z^l_{i,j}} \left(\sum_{\tilde{m}} \sum_{\tilde{n}} w^{l+1}_{\tilde{m},\tilde{n}} g(z^l_{i-m+\tilde{m},j-n+\tilde{n}}) + b^{l+1} \right) \\ &= \frac{\partial}{\partial z^l_{i,j}} \left(w^{l+1}_{m,n} g(z^l_{i,j}) \right) = w^{l+1}_{m,n} g'(z^l_{i,j}) \end{aligned}$$

Consequently,

$$\delta^l_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta^{l+1}_{i-m,j-n} \frac{\partial z^{l+1}_{i-m,j-n}}{\partial z^l_{i,j}} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta^{l+1}_{i-m,j-n} w^{l+1}_{m,n} g'(z^l_{i,j}),$$

which can be inserted in (3.12) in order for the gradient to be derived. [1] The gradient with respect to the bias can be derived according to

$$\frac{\partial \mathcal{L}}{\partial b^l_{ij}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta^{(l+1)}_{i,j}.$$

Generalization to more channels and dimensions

The so far discussed convolution operations in neural networks can be generalized to inputs $I \in \mathbb{R}^{H\times W\times D\times C}$, where D is the depth of the image and $C > 1$; the same methods and principles still prevail. When several channels occur, each channel is given a separate $k_1 \times k_2 \times k_3$ dimensional filter in $K \in \mathbb{R}^{k_1\times k_2\times k_3\times C}$, answering to the dimension of I.

3.2.2 Padding in a Convolutional layer

The dimension attained after convolution might not coincide with the sought one, to modulate the dimension of the output feature map one can apply *zero padding*. This entails that the input I is altered by *padding* with zeros around its borders, yielding a new input $\tilde{I}$. Subsequently, the kernel will sweep over $\tilde{I}$, comprised of the trivial inputs as well as the regular I. Hence, the spatial dimension will be altered according to the amount of padding. Let P correspond to the depth of the padded rows around the input. In Figure 3.8 a convolution with a zero-padded input can be seen.

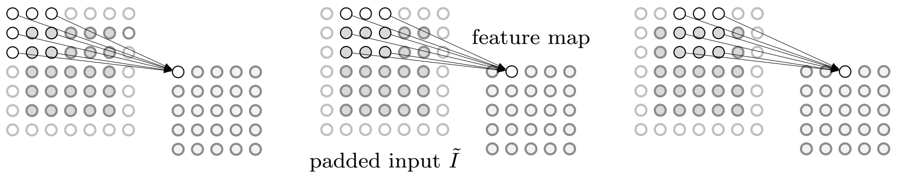


Figure 3.8: Visualization of convolution with zero *half padding* [15] and unit stride, with a kernel $K \in \mathbb{R}^{3\times 3}$ and where $I \in \mathbb{R}^{5\times 5}$ is seen as the gray nodes in the center of $\tilde{I}$. The arrows indicates how the kernel sweeps over the input, thence rendering each element in the resulting feature map. The padded zeros are seen in the weakly colored edges of $\tilde{I}$, here $P = 1$.

The padding seen in Figure 3.8 maintains the spatial dimension of I. In many applications this is desirable, since several convolutions then can be applied on an input of unaltered dimension, and consequently the receptive field can be expanded between convolutional layers whilst preserving the size of the input. This practically implies that deeper network structures can be built. That the depth matches the complexity of its task is actually crucial

[1]A large portion of the derivations were found in the tutorial on CNN and backporpagation by Jefkine Kafunah, *Backpropagation In Convolutional Neural Networks.*

in learning, since "an architecture with insufficient depth can require many more computational elements, potentially exponentially more [...] than architectures whose depth is matched to the task" [2].

If one contrastingly wishes to extend the dimension of I one can apply *full-padding* [15], which entails that the output feature map will be dimensionally greater than the input as an effect of the padding. If one assumes that $H = W$, i.e. that the image height is analogous to the width, and that $k = k_1 = k_2$, i.e. that the kernel is symmetric, the dimension of the output feature map $s \in \mathbb{R}^{F \times F}$ corresponds to

$$F = \frac{W - k + 2P + S_t}{S_t}.$$

3.2.3 Max pooling layer

In order to further reduce the size of the input in a CNN, in addition to the spatially decreasing effect of the convolutional layers, *pooling* may be applied. This type of layer "provide[s] invariance to small translations of the input" [15]. Which may be regarded as a consequence of applying an infinitely strong prior on the function constituted by the layer, to achieve translational invariance [19]. Moreover, the number of parameters is lessened and over fitting is counteracted. This layer, in contrast to a convolution layer, does not require any learning. Instead it is obtained by applying a predefined function, and thence locally aggregate the data [15]. One usual pooling function is *max-pooling*. In this type of pooling the input is commonly divided into disjoint subsets, then the max of each subset is propagated further to the next layer.

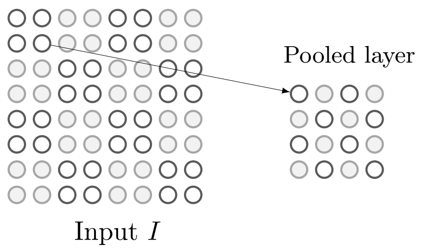


Figure 3.9: Visualization of a max-pooling layer, each pixel in I is divided into subsets, from which the maximum entity is propagated to the next layer. The varying colors in the inputs indicates the division into subsets, and the arrow demonstrates the propagation for the first patch.

In Figure 3.9 one can see max-pooling, where the input is divided into disjoint patches indicated by the colors. One entity in each such patch is propagated to the next layer, namely the maximum entity in each subset. This entails that the most salient features in each local part of the image is extracted and preserved between layers. There are other types of non-linearities than the max-operation that may be applied in pooling layers, such as *average pooling*, where an average over each subset is propagated instead of the max. Nonetheless, max-pooling is found to be superior to average pooling in a variety of settings [6].

3.2.4 Dropout layer

When training neural networks the expressive strength of the method may entail that intricate relationships are learned from the training data. However, these may not be present in other data, and as a result the network will not generalize properly and perform well with unseen data. This impediment in machine learning is as previously mentioned known as overfitting. In order to counteract this problem the network can be regularized in a range of manners, one method that is presented in [21] is the *dropout neural network*. In this type of network one samples a *thinned* version of the original network for each training input and then train on each such specific architecture. Hence, for a network with n units or vertices, there are 2^n unique thinned versions of the network, over which the weights are shared [53]. This dropout network is formed by setting

$$\begin{aligned} r_j^l &\sim \text{Bernoulli}(p), \quad 0 \leq p \leq 1 \text{ and } r_j^l \in \{0,1\}, \\ \tilde{a}^l &= r^l \odot a^l, \\ \mathbf{z}^{l+1} &= \mathbf{w}_j^{l+1} \tilde{a}^l + b_j^{l+1}. \end{aligned}$$

Consequently, parts of the network is rendered inactive by $r_j^l = 0$, subsequently the thinned architecture is formed. In Figure 3.10 a thinned version of the network in Figure 3.2 can be seen.

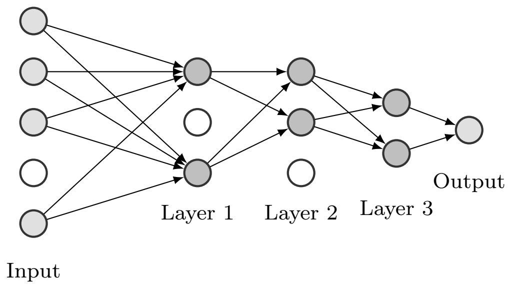


Figure 3.10: A dropout neural network, i.e. a thinned version of a fully connected FNN with three hidden layers. The white vertices are dropped out from the network.

Even though there are exponentially many thinned versions of the network, the number of utilized thinned networks in training will correspond to $m \cdot e$, where e is the number of epochs and m corresponds to dimension of the training set [59]. In a CNN the parameter sharing and sparse interactions make the network less inclined to over fit [59]. Nonetheless it is shown in [53] that dropout convolutional neural networks can outperform ordinary CNNs, hence such networks can also benefit from further regularization.

In contrast to a fully dropout neural network, one can incorporate dropout in some chosen layers of a network. These dropout layers will regularize the architecture in an analogous manner. Here one commonly places a separate layer which propagates an input with probability p and a trivial output "0" with probability $(1-p)$. Moreover the expected value of the input is usually preserved, by re-scaling the propagated inputs by the inverse propagation probability.

3.2.5 General structure of a CNN

A convolutional neural network is generally comprised of a variety of components, *parametric* as well as *non parametric*. In addition to the purely convolutional layers the previously discussed: (i) activation layers, (ii) pooling layers as well as (iii) dropout layers are commonplace. Conventionally, the convolution layer is followed by an activation layer such as a ReLU. Thence pooling is usually applied [19], s.t. the most prominent features of the map is extracted. If the network has dropout regularization this layer usually follows the pooling. In Figure 3.11 an example of this architecture can be seen.

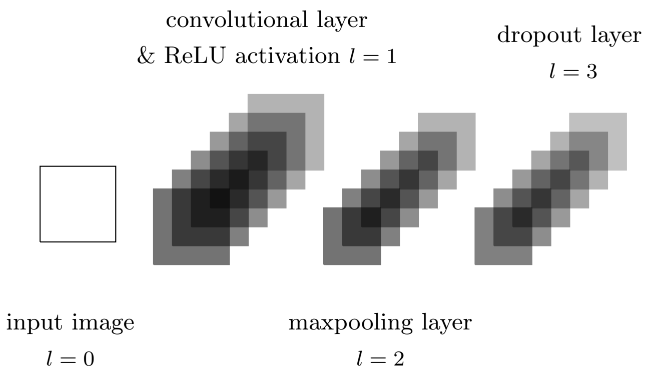


Figure 3.11: A visualization of the common architecture of layers in a CNN. Here the number of feature maps in the convolution corresponds to 6.

Furthermore, the network might also contain fully connected layers. Nonetheless, these entail such an extensive amount of parameters, such that in order to make it computationally feasible, these layers are usually applied on an input of quite reduced dimension, entailing that they are applied after e.g. the convolution operation or max pooling have spatially reduced I.

3.2.6 Transpose convolution

According to [52], the deconvolutional network, and its inherent *transpose convolutional layer*, was first proposed in [60]. Since then the layer has been applied in a variety of applications, such as *semantic segmentation* [34], and *generative modeling* [50]. The transpose convolution solves the problem of going in the opposite direction of an ordinary convolution, i.e. the output of the layer is the input to the traditional layer. This is accomplished "while maintaining a connectivity pattern that is compatible with said convolution" [15]. This operation reconstructs the shape of the input to a convolutional layer, consequently it does not act as the inverse to the convolutional operand [15]; it could however retain the input. In Figure 3.12 a convolution can be seen, and in Figure 3.13 a visualization of a transpose convolution to said convolution can be seen.

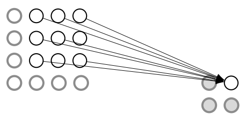


Figure 3.12: A convolution between an input $I \in \mathbb{R}^{4\times4}$, and a kernel $K \in \mathbb{R}^{3\times3}$ yielding with a one-stride convolution, the gray 2×2 feature map. The generation of the second element in the feature map is seen in the visualization.

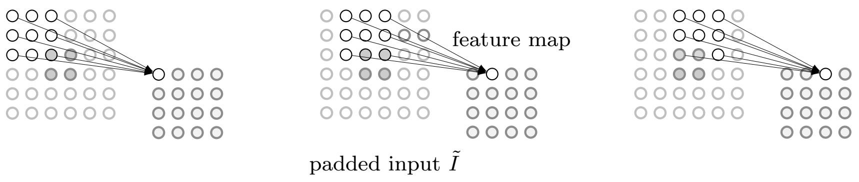


Figure 3.13: Visualization of transpose convolution. The original convolution is seen in Figure 3.12, the gray 2×2 feature map generated is visible in the center of the padded input $\tilde{I}$ here. The transpose convolution is then applied on a $P = 2$ zero padded input $\tilde{I}$, with a kernel $K \in \mathbb{R}^{3\times3}$ and unit stride. This produces a feature map of the same dimension as I.

To *up-sample*, i.e. enlarge a feature map, one may use various interpolation methods or transpose convolution. Since the transpose convolution has learnable parameters one attains an up-sampling modulated to the task by using this method.

Chapter 4

Autoencoders

4.1 Introduction to Autoencoders

An *Autoencoder* (AE) is essentially a type of neural network that consists of two parts: an *encoder* and a *decoder*. The objective of the AE is to encode a set of data $\mathbf{x}$ in a representation referred to as *code* $\mathbf{z}$, and then be able to use this representation to reconstruct the data, thence attaining $\tilde{\mathbf{x}}$. Hence, the autoencoder will map $\mathbf{x} \mapsto \tilde{\mathbf{x}}$ through the encoder $f(\mathbf{x}) \to \mathbf{z}$ and the decoder $g(\mathbf{z}) \to \tilde{\mathbf{x}}$. In Figure 4.1 a simple visualization of the architecture of an AE can be seen.

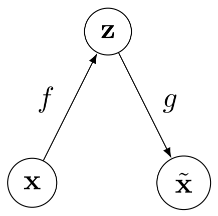


Figure 4.1: A visualization of an AE. The encoder receives data $\mathbf{x}$ and creates the representation $\mathbf{z}$ through $f(\mathbf{x})$. The decoder then creates a reconstruction $\tilde{\mathbf{x}}$ through $g(\mathbf{z})$.

Dimensionality reduction as well as *feature learning* are two applications where the AE is conventionally used [19]. The essential AE task, of copying the input to the output might appear odd, however one is usually not that interested in the decoder output, $\tilde{\mathbf{x}}$. The entity of focus is rather the code $\mathbf{z}$ produced by the encoder; the objective being that the AE will capture salient features of the data in the representation $\mathbf{z}$ [19]. This may occur as a bi-product of being able to perform the copying task $\mathbf{x} \mapsto \tilde{\mathbf{x}}$ well. The AE framework is unsupervised since no keys y_k are given. Moreover, the process of learning for the AE can essentially be described as minimizing the loss function,

$$\mathcal{L}(\mathbf{x}, g \circ f(\mathbf{x})) = \mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}).$$

Since the training objective is that the AE learns to copy the input $\mathbf{x}$ to the output $\tilde{\mathbf{x}}$, this loss function could e.g. be the MSE between these entities. Considering that the AE is principally a type of FNN, they can be trained with the techniques used to train standard FNNs, such as GD and BP [19]. Moreover, there are connections between the AE and classical representation learning techniques, namely *Principal Component Analysis*. If the AE network is trained with the MSE criterion and is comprised of one linear hidden layer with p units, the encoder will project the input onto the span of the p first principal components of the data [7]. Non-linearity in the hidden layer will however yield quite different characteristics, including an ability to convey other aspects of the input distribution [20]. We can thence achieve “a more powerful nonlinear generalization of PCA” [19]. Furthermore, more abstract and deep representations can be rendered by stacking AEs, i.e. interconnecting several autoencoders in a network structure [3].

The dimension of the entities included in the AE will render quite different characteristics of the network. If the code $\mathbf{z}$ is of smaller dimension than the data $\mathbf{x}$, the AE is referred to as *undercomplete*. An undercomplete AE can be successful in capturing key features of the data in its code, since the network is forced to reconstruct the data by using a representation in a lower dimension. A too large capacity of the AE can limit the usefulness of the features learned, in the sense that the AE can perform the task $\mathbf{x} \mapsto \tilde{\mathbf{x}}$ well, but is unable to capture characterizing features from the data [19]. Furthermore, if the dimension of the code is larger then that of the data we have an *overcomplete* AE. In the overcomplete case we can also encounter the problem of a code unable to reflect important features of the data, even though the AE can excel in the copying task. These implications, connected to the dimension, illuminates that one have to be considerate when designing the architecture of any AE; in order to attain a network that functions well according to its intended task. Consequently, the capacity and dimensions must be modulated according to the characteristics of the data at hand.

There is a certain type of AE that, in addition to being able to perform the copying task well, can handle the tradeoffs regarding capacity and dimension, called the *Regularized Autoencoder*. This AE is attained by altering the loss function $\mathcal{L}$ such that the AE gains characteristics such as *robustness to noise* and *sparsity of representation* [19]. To render a *Sparse Autoencoder* the loss function has an additional penalty term $\Omega(z)$ which encourages sparsity of the code, hence the loss function for this type of regularized AE is given by

$$\mathcal{L}(\mathbf{x}, g(\mathbf{z})) + \Omega(\mathbf{z}).$$

There are additional types of AEs, such as the *denoising Autoencoder*, the *contractive Autoencoder* and the *Variational Autoencoder* (VAE) [19].

4.2 Variational Autoencoders

4.2.1 Deep learning interpretation

A VAE is essentially a stochastic generalization of the AE. Here the encoder and decoder are instead comprised of stochastic mappings. The analogue to the encoding function $f(\mathbf{x})$ is an encoding distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$, and the analogue to the decoding function $g(\mathbf{z})$ is a decoding distribution $p_\theta(\mathbf{x} \mid \mathbf{z})$, where ϕ denotes the *recognition model parameters* and θ denotes the *generative model parameters* [28]. A visualization of the structure of a VAE can be seen in Figure 4.2.

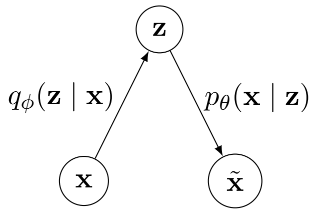

$\mathbf{z}$
$q_\phi(\mathbf{z} \mid \mathbf{x})$
$p_\theta(\mathbf{x} \mid \mathbf{z})$
$\mathbf{x}$
$\tilde{\mathbf{x}}$

Figure 4.2: A visualization of a variational autoencoder. The encoder receives data $\mathbf{x}$ and creates the representation $\mathbf{z}$ through $q_\phi(\mathbf{z} \mid \mathbf{x})$. The decoder then creates a reconstruction $\tilde{\mathbf{x}}$ through $p_\theta(\mathbf{x} \mid \mathbf{z})$.

Imposing a prior on the code $\mathbf{z}$ will regularize the VAE, a typical choice of such distribution is a multivariate isotropic Gaussian $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ [30]. Note that this particular prior has no parameters θ.

In a VAE the loss function $\mathcal{L}_{\text{VAE}}$ used to train the neural network is comprised of the reconstruction error, given by the expected log-likelihood term, as well as a regularization term imposed by the chosen prior [30]. Specifically we have

$$\begin{aligned}
\mathcal{L}_{\text{VAE}} &= -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\right] \\
&= -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log q_\phi(\mathbf{z} \mid \mathbf{x}) - \log p_\theta(\mathbf{z})\right] \\
&= -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] + D_{\text{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p_\theta(\mathbf{z})) \\
&= \mathcal{L}_{\text{logl}} + \mathcal{L}_{\text{prior}}.
\end{aligned}$$

Where D_{KL} denotes the *Kullback-Leibler divergence* [29], hence we have that $\mathcal{L}_{\text{prior}}$ enforces the prior $p_\theta(\mathbf{z})$ by penalizing divergence from this distribution. Effectively, this entails that the encoding distribution is encouraged to follow the distributional encoding of the prior. Furthermore, the reconstruction term $\mathcal{L}_{\text{logl}}$ is actually the *cross entropy* between $q_\phi(\mathbf{z} \mid \mathbf{x})$ and $p_\theta(\mathbf{x} \mid \mathbf{z})$.

4.2.2 Probability model interpretation

One can also interpret the VAE as a certain type of probability model, on which joint inference is applied, by "perform[ing] maximum likelihood (ML) or maximum a posteriori (MAP) inference on the (global) parameters, and variational inference (VI) on the latent variables" [28]. This inference procedure can be applicable in spite of intractable inherent distributions. The probabilistic framework is characterized by two stochastic entities

$$\begin{aligned} X &: \Omega_{\mathbf{x}} \to \mathbb{R}, \\ Z &: \Omega_{\mathbf{z}} \to \mathbb{R}. \end{aligned}$$

Where $\mathbf{x} \in \Omega_{\mathbf{x}}$ is an N-dimensional observable random variable and $\mathbf{z} \in \Omega_{\mathbf{z}}$ its adherent J-dimensional latent correspondence. The joint distribution of these variables corresponds to

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}),$$

where the prior is usually an isotropic centered Gaussian

$$p_\theta(\mathbf{z}) = p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

This framework provides a *generative process*,

$$\begin{aligned} &\text{Draw } \mathbf{z}_k \sim p_\theta(\mathbf{z}), \\ &\text{Draw } \mathbf{x}_k \sim p_\theta(\mathbf{x} \mid \mathbf{z}_k). \end{aligned}$$

Hence draws from the prior characterizes the likelihood from which data can be generated. We can represent the process by the *graphical model* seen in Figure 4.3. This graph conveys the dependence structure in the probabilistic model.

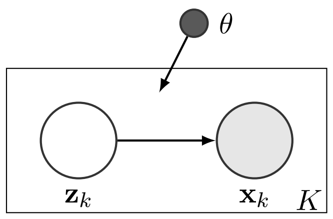


Figure 4.3: Directed graph visualizing the Autoencoder probabilistic model, where θ are the generative model parameters, $\mathbf{x}_k$ is the observed data and $\mathbf{z}_k$ the latent variable.

Consequently, a generative process is the foundation of the setting, where the true parameter values as well as the latent variables are unknown [28]. Hence, ML or MAP inference regarding θ could be of interest. Furthermore, *approximate posterior inference*: where the connection between the latent

$\mathbf{z}$ given data $\mathbf{x}$, for a specific value of θ, may also be relevant concerning e.g. *data representation tasks* [28]; in settings where the true posterior is intractable. According to Bayes' theorem the posterior $p_\theta(\mathbf{z} \mid \mathbf{x})$ is given by

$$p_\theta(\mathbf{z} \mid \mathbf{x}) = \frac{p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})}.$$

Where the *evidence* $p_\theta(\mathbf{x})$ can be attained by marginalization of the joint distribution

$$p_\theta(\mathbf{x}) = \int_{\Omega_\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) \mathrm{d}\mathbf{z} = \int_{\Omega_\mathbf{z}} p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}) \mathrm{d}\mathbf{z}.$$

However, evaluation of this integral is computationally expensive, and may require exponential time [10]. Moreover, intractability for the posterior or integrals inherent in inference procedures, could render the EM algorithm as well as the *mean-field Variational Bayes* (VB) algorithm inapplicable in this setting [28]. This illuminates the need for a more general inference procedure, as the VAE which essentially is " a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case" [28]. Firstly, the posterior ought to be approximated in order for this inference to be computationally feasible in some cases. Let $q_\phi(\mathbf{z} \mid \mathbf{x})$ be such an estimate. Here the parameters ϕ do not have to be derived by a closed form expectation, and the density does not have to factorize, as supposed to the setting in mean-filed VB [28]. A metric conveying the quality of this approximation is the Kullback-Leibler divergence

$$D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p_\theta(\mathbf{z} \mid \mathbf{x})) \;=\; \mathbb{E}_{q_\phi}\left[\log q_\phi(\mathbf{z} \mid \mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})\right] + \log p_\theta(\mathbf{x}),$$

which effectively reflects how congruent the estimate is with the true posterior, i.e. it is " a measure of the inefficiency of assuming that the distribution is q when the true distribution is p" [49]. Consequently, we wish to find an optimal $q_\phi(\mathbf{z} \mid \mathbf{x})$ such that the divergence is minimized i.e.

$$\hat{q}_\phi(\mathbf{z} \mid \mathbf{x}) \;=\; \arg\min_{\phi} \; D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p_\theta(\mathbf{z} \mid \mathbf{x})).$$

Minimizing D_{KL} exactly is however unfeasible in most cases, since the intractable evidence is included. The tractable measure *Evidence Lower Bound* (ELBO) is however analogous to D_{KL} up to a specific constant; hence, to minimize the Kullback-Leibler Divergence one can study this evidence instead, which is the procedure in the VAE inference. Let us look at the marginal likelihood $p_\theta(\mathbf{x})$, we have that

$$\begin{aligned} \log p_\theta(\mathbf{x}) \;&=\; \log \int_{\Omega_\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \;=\; \log \int_{\Omega_\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) \frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{q_\phi(\mathbf{z} \mid \mathbf{x})} d\mathbf{z} \\ &=\; \log \mathbb{E}_{q_\phi}\left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\right] \;\geq\; \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \;\log q_\phi(\mathbf{z} \mid \mathbf{x})\right]. \end{aligned}$$

Where *Jensen's inequality* [24] has been been used in the last step. The RHS of the inequality corresponds to the ELBO, which upon inspection is related to the Kullback-Leibler divergence D_{KL} according to

$$D_{\text{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p_\theta(\mathbf{z} \mid \mathbf{x})) \;=\; -\text{ELBO}(\phi) + \;\log p_\theta(\mathbf{x}).$$

Consequently, the constant in the relation between these entities is not dependent on ϕ, thence we can disregard it and simply set

$$\hat{q}_\phi(\mathbf{z} \mid \mathbf{x}) = \arg\max_\phi \;\text{ELBO}(\phi) = \arg\max_\phi \;\mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \;\log q_\phi(\mathbf{z} \mid \mathbf{x})\right].$$

Thus, the recognition model parameters, or *variational parameters* ϕ are simultaneously optimized with the recognition model parameters θ in the quest of maximizing this lower bound [28]. Where an encoder can be trained to *parameterize* the posterior, i.e. take data as input and yield optimal parameter estimates ϕ. Thence, a decoder can parameterize the likelihood and yield data parameters θ. Moreover, in the VAE framework each data point has its own unique latent correspondence. As a result, one can decompose ELBO(ϕ) such that it is comprised of a sum over the data set, i.e.

$$\text{ELBO}(\phi) = \sum_{k=1}^{K} \text{ELBO}_k(\phi) = \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}_k, \mathbf{z}) - \;\log q_\phi(\mathbf{z} \mid \mathbf{x}_k)\right].$$

Furthermore, note that the ELBO can be written according to

$$\begin{aligned}
\text{ELBO}(\phi) &= \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \;\log q_\phi(\mathbf{z} \mid \mathbf{x})\right] \\
&= \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}) - \;\log q_\phi(\mathbf{z} \mid \mathbf{x})\right] \\
&= \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] - \mathbb{E}_{q_\phi}\left[\log q_\phi(\mathbf{z} \mid \mathbf{x}) - \;\log p_\theta(\mathbf{z})\right] \\
&= \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] - D_{\text{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p_\theta(\mathbf{z})) \\
&= -\mathcal{L}_{\text{VAE}}
\end{aligned}$$

Hence, by studying the VAE from a probabilistic perspective we have found that the objective function for inference of an optimal posterior or maximization of the ELBO in a latent variable model, is analogous to the negative of the loss function, $\mathcal{L}_{\text{VAE}}$, used when training the VAE network.

4.2.3 VAE in a strictly Gaussian case

Now, suppose the stochastic entities all follow a Gaussian distribution, according to

$$\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{0}, \mathbf{I}), && (4.1) \\
p_\theta(\mathbf{x} \mid \mathbf{z}) &= \mathcal{N}(\boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\sigma}^2(\mathbf{z})\mathbf{I}), && (4.2) \\
q_\phi(\mathbf{z} \mid \mathbf{x}) &= \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}^2(\mathbf{x})\mathbf{I}). && (4.3)
\end{aligned}$$

In Figure 4.4 the network and its adherent parameters, for this Gaussian case, can be seen in a schematic graph.

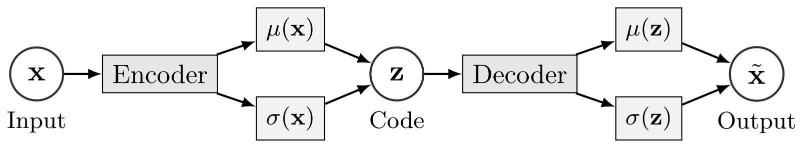


Figure 4.4: Visualization of the architecture of a Gaussian VAE.

The loss function for a VAE with these distributions corresponds to

$$\begin{aligned}\mathcal{L}_{\text{VAE}} &= -\,\mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] + D_{\text{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z})) \\ &= -\,\mathbb{E}_{q_\phi}[\log \mathcal{N}(\boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\sigma}(\mathbf{z})^2\mathbf{I})] + D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}(\mathbf{x})^2\mathbf{I}) \mid\mid \mathcal{N}(\mathbf{0}, \mathbf{I})).\end{aligned}$$

Suppose that $\mathbf{z} \in \mathbb{R}^J$ and let k denote the data point at which evaluation occurs, in [28] it is shown that the Kullback-Leibler divergence for these Gaussians reduces to

$$\boxed{\begin{aligned}D_{\text{KL}}\big(q_\phi\big(\mathbf{z} \mid \mathbf{x}^{(k)}\big) \mid\mid p(\mathbf{z})\big) &= D_{\text{KL}}(\mathcal{N}\big(\boldsymbol{\mu}(\mathbf{x}^{(k)}), \boldsymbol{\sigma}^2\big(\mathbf{x}^{(k)}\big)\mathbf{I}\big) \mid\mid \mathcal{N}(\mathbf{0}, \mathbf{I})) \\ &= -\frac{1}{2}\sum_{j=1}^{J}\Big(1 + 2\log\big(\sigma_j^{(k)}\big) - \big(\mu_j^{(k)}\big)^2 - \big(\sigma_j^{(k)}\big)^2\Big).\end{aligned}}$$

Reparameterization trick

Moreover, one can sample from the posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ with a reparameterization trick, by a " differentiable transformation [...] of an (auxiliary) noise variable ϵ" [28], i.e.

$$\begin{aligned}\mathbf{z}^{(k)} &\sim q_\phi\big(\mathbf{z} \mid \mathbf{x}^{(k)}\big) \quad \text{according to} \\ \mathbf{z}^{(k,u)} &= \boldsymbol{\mu}\big(\mathbf{x}^{(k)}\big) + \boldsymbol{\sigma}\big(\mathbf{x}^{(k)}\big) \odot \epsilon^{(u)}, \quad \text{where} \quad \epsilon^{(u)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).\end{aligned}$$

and $u \in \{1, U\}$ conveys the number of samples drawn [28]. *Monte Carlo integration* can subsequently be used to estimate the expected value in $\mathcal{L}_{\text{VAE}}$ since by the LNN we have that

$$\frac{1}{U}\sum_{u=1}^{U} f\big(z^{(u)}\big) \xrightarrow{\text{a.s.}} \mathbb{E}[f(Z)] \quad \text{as} \quad U \to \infty.$$

Consequently, the cross entropy of the posterior and the likelihood can be estimated according to,

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta\big(\mathbf{x}^{(k)} \mid \mathbf{z}^{(k)}\big)\big] \approx \frac{1}{U}\sum_{u=1}^{U}\log p_\theta\big(\mathbf{x}^{(k)} \mid \mathbf{z}^{(k,u)}\big),$$

for large enough U. If we now consider that the likelihood is Gaussian as defined in (4.2), the above expectation can be decomposed according to,

$$\boxed{\begin{aligned}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log\mathcal{N}(\boldsymbol{\mu}(\mathbf{z}),\boldsymbol{\sigma}^2(\mathbf{z})\mathbf{I})\big] \;\approx\; &\frac{1}{U}\sum_{u=1}^{U}\log\mathcal{N}(\boldsymbol{\mu}(\mathbf{z}^{(u)}),\boldsymbol{\sigma}^2(\mathbf{z}^{(u)})\mathbf{I}) \\ \propto \frac{1}{U}\sum_{u=1}^{U}-\frac{1}{2}\Big(\log\prod_{n=1}^{N}\big(\sigma_n(\mathbf{z}^{(u)})\big)^2 + &\frac{1}{\sigma(\mathbf{z}^{(u)})^2}\big(\mathbf{x}-\boldsymbol{\mu}(\mathbf{z}^{(u)})\big)^{\mathrm{T}}\big(\mathbf{x}-\boldsymbol{\mu}(\mathbf{z}^{(u)})\big)\Big).\end{aligned}}$$

Where, temporarily the notation $\frac{1}{\sigma(\mathbf{z}^{(u)})^2} = \big(\sigma(\mathbf{z}^{(u)})^2\mathbf{I}\big)^{-1}$ is applied.

Fixed variance Variational Autoencoders

To simplify inference and reduce the number of parameters to infer in the framework, *fixed variance* VAEs are commonplace. Here it is assumed that $\boldsymbol{\sigma}(\mathbf{z})^2 = \boldsymbol{\sigma}^2$ and $\sigma_n^2 = \sigma^2$, hence the variance is assumed equal $\forall\ n \in \{1, N\}$ and predefined as supposed to inferred by the generative network [14]. In addition to reduction of sought parameters this renders a more simplistic form of reconstruction loss, since

$$\mathbb{E}_{q_\phi}\left[\log\mathcal{N}(\boldsymbol{\mu}(\mathbf{z}),\sigma^2\mathbf{I})\right] \propto -\frac{1}{U}\sum_{u=1}^{U}\frac{1}{2}\left(\frac{1}{\sigma^2}(\mathbf{x}-\boldsymbol{\mu}(\mathbf{z}))^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}(\mathbf{z}))\right)$$

when the variance is known. As such it will not influence the maximization of the expected value, more than as a simple scaling factor. In Figure 4.5 a fixed variance VAE can be seen in a schematic graph.

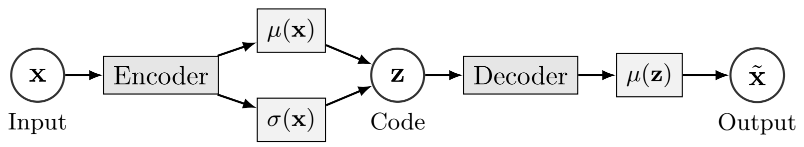


Figure 4.5: Visualization of the architecture of a VAE with fixed variance.

4.3 Reconstruction with decoder network

Samples $\mathbf{z}$ from the latent variables can be mapped to the observation space $\Omega_{\mathbf{x}}$ by the decoder, entailing that new data is artificially generated. Supposing the likelihood is given by (4.2) new data can be rendered by

$$\tilde{\mathbf{x}} = \boldsymbol{\mu}(\mathbf{z}) + \boldsymbol{\xi}(\mathbf{z}),$$

where $\boldsymbol{\xi}(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}(\mathbf{z})^2\mathbf{I})$ or $\boldsymbol{\xi}(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Nonetheless, in the reconstruction from the decoder, $\boldsymbol{\xi}(\mathbf{z})$ and its adherent variance is rarely used [14]. This is because the addition of noise in the outer layer of the decoder simply adds white noise to the reconstruction [14]. Thence, artificially generated data is often comprised of a deterministic mapping of the latent $\mathbf{z}$ according to

$$\tilde{\mathbf{x}} = \boldsymbol{\mu}(\mathbf{z}).$$

In Figure 4.5 a schematic graph of this reconstruction in the VAE framework can be seen.

4.4 Learned manifold

To investigate how the latent space $\Omega_{\mathbf{z}}$ relates to the image space $\Omega_{\mathbf{x}}$ one can study the learned data manifold. This "manifold is a connected region [consisting of] a set of points associated with a neighborhood around each point" [19]. Moreover, there is a resemblance to a *Euclidean space* in a local neighborhood of each point on the manifold. In machine learning one conventionally refers to a manifold as a "connected set of points that can be approximated well by considering only a small number of degrees of freedom, or dimensions, embedded in a higher-dimensional space" [19]. In the VAE application, the low-dimensional latent space $\Omega_{\mathbf{z}}$ will be related to the high dimensional $\Omega_{\mathbf{x}}$ image space, which can be characterized in terms of a manifold. By moving in the latent space and finding the equivalence in the image space one can get a grip of what latent representation the VAE has rendered.

One method of finding the subspace of $\Omega_{\mathbf{z}}$ that gives rise to non trivial decoded representations is to use the *Cumulative Distribution Function* (CDF), $F_{\mathbf{z}}$ of the prior

$$F_{\mathbf{z}}(\mathbf{y}) = \mathbb{P}(\mathbf{Z} \leq \mathbf{y}) = \int_{\mathbf{z} \in \Omega_{\mathbf{z}} | \mathbf{z} \leq \mathbf{y}} P(\mathbf{z}) \mathrm{d}\mathbf{z}.$$

This function fulfills $0 \leq F_{\mathbf{z}} \leq 1$, and reflects the distribution of probability in $\Omega_{\mathbf{z}}$. In Figure 4.6 the CDF of a standard one dimensional Gaussian can be seen.

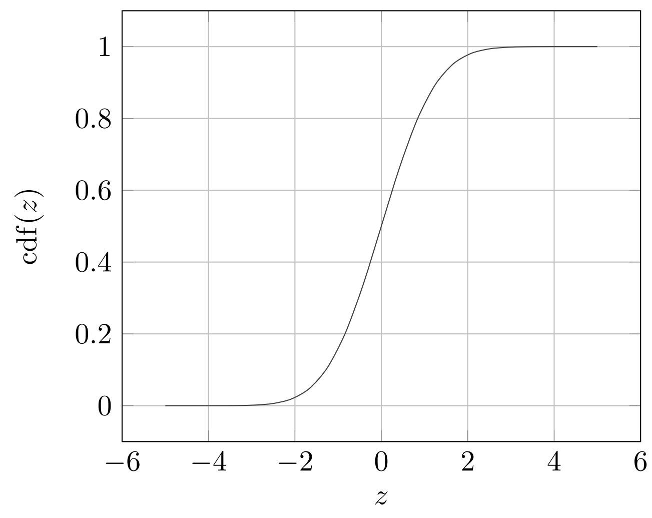


Figure 4.6: CDF of $\mathcal{N}(0,1)$.

In the Figure it is evident that the majority of the probability mass is situated in $|z| \leq 2$.

Hence, to find the subspace in the $\Omega_{\mathbf{z}}$ where probability mass of the prior is situated one can use the inverse of the CDF: $F_{\mathbf{z}}^{-1}$. More specifically, if we consider a latent space $\Omega_{\mathbf{z}} \subset \mathbb{R}^2$ we can form an equidistant grid on the unit square $\{(c_1, c_2) : 0 \leq c_1 \leq 1,\ 0 \leq c_2 \leq 1\}$; map each coordinate $\{(i,j)\}$ on the grid by the inverse CDF to its equivalent $\mathbf{z}$-position. Then a grid of latent variables has been formed, over a subset of $\Omega_{\mathbf{z}}$ which holds probability mass. Subsequently, one may decode each $\left\{z_1^{(i,j)}, z_2^{(i,j)}\right\}$ on the latent grid, and study how the latent space relates to the image space.

4.5 Evaluation of encoder

One method of evaluating the performance of the encoder (4.3) is to see how well this density estimates the prior distribution. This is since

$$p(\mathbf{z}) \;=\; \int_{\Omega_{\mathbf{x}}} p(\mathbf{x},\mathbf{z})\mathrm{d}\mathbf{x} \;=\; \int_{\Omega_{\mathbf{x}}} p_\theta(\mathbf{z} \mid \mathbf{x})p(\mathbf{x})\mathrm{d}\mathbf{x} \;=\; \mathbb{E}_{p(\mathbf{x})}[p_\theta(\mathbf{z} \mid \mathbf{x})],$$

which can subsequently be estimated via Monte Carlo integration together with the encoder distribution according to

$$\mathbb{E}_{p(\mathbf{x})}[p_\theta(\mathbf{z} \mid \mathbf{x})] \;\approx\; \frac{1}{K}\sum_{k=1}^{K} p_\theta(\mathbf{z} \mid \mathbf{x}_k) \;\approx\; \frac{1}{K}\sum_{k=1}^{K} q_\phi(\mathbf{z} \mid \mathbf{x}_k) = \tilde{p}(\mathbf{z}).$$

Hence, if the chosen estimate $q_\phi(\mathbf{z} \mid \mathbf{x})$ of the posterior distribution $p_\theta(\mathbf{z} \mid \mathbf{x})$ is sufficient, and $\{\mathbf{x}_k\}_{k=1}^{K}$ is diverse and extensive enough; one expects the

MC estimate $\tilde{p}(\mathbf{z})$ of the prior to be approximately distributed according to $p(\mathbf{z})$, entailing that

$$\tilde{p}(\mathbf{z}) \xrightarrow{\text{d.}} p(\mathbf{z}).$$

4.5.1 Gaussian encoder

For an encoder distribution

$$q_\phi(\mathbf{z} \mid \mathbf{x}_k) = \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{x}_k), \boldsymbol{\sigma}^2(\mathbf{x}_k)\mathbf{I}\big),$$

the estimation of the prior consequently fulfills

$$\tilde{p}(\mathbf{z}) \;=\; \frac{1}{K}\sum_{k=1}^{K} q_\phi(\mathbf{z} \mid \mathbf{x}_k) \;=\; \sum_{k=1}^{K} \pi_k \, \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{x}_k), \boldsymbol{\sigma}^2(\mathbf{x}_k)\mathbf{I}\big),$$
$$\text{where} \quad \pi_k = \frac{1}{K} \quad \text{hence} \quad \sum_{k=1}^{K} \pi_k = 1.$$

As a result $\tilde{p}(\mathbf{z})$ is a mixture of Gaussians where the *mixing coefficients* π_k are uniformly distributed over each Gaussian yielding the distribution.

4.5.2 Data augmentation by approximate prior

Let us now suppose that the data $\mathcal{D} = \{\mathbf{x}_k\}_{k=1}^{K}$ is not sufficient for the approximation of the prior. One way of counteracting this is to use *data augmentation*, which essentially entails that $\mathcal{D}$ is extended with additional data in some manner. In this instance one may use the estimation of the prior to this end. Namely, one can sample $\{\tilde{\mathbf{z}}_a\}_{a=1}^{A}$ from $\tilde{p}(\mathbf{z})$ and push those samples through the decoder, rendering $\{\tilde{\mathbf{x}}_a\}_{a=1}^{A}$. Subsequently, $\mathcal{D}$ can be extended with $\{\tilde{\mathbf{x}}_a\}_{a=1}^{A}$, and the network can approximate the prior by the encoder once more. This can then be repeated in an iterative fashion and the behaviour of the approximate prior can be studied. In Algorithm 2 it is demonstrated how one samples from a mixture of Gaussians, i.e. the approximate prior for a Gaussian encoder.

```
**Result:** Sampling from a mixture of Gaussians
Sample $A$ indices $I_a$ from categorical distribution yielding
$\{1, \dots, K\}$ according to $\{\pi_k\}_{k=1}^{K}$
for $a = 1 : A$ do
    draw $\tilde{\mathbf{z}}_a \sim \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{x}_{I_a}), \boldsymbol{\sigma}^2(\mathbf{x}_{I_a})\mathbf{I}\big)$
end
```

Algorithm 2: Sampling for data augmentation.

4.6 Feature extraction

The latent representation $\mathbf{z}$ of the data $\mathbf{x}$ will, as previously elaborated on, hopefully capture prominent features. Subsequently, one may compare the potentially high dimensional $\mathbf{x}$ in the latent space $\Omega_{\mathbf{z}}$ using various probabilistic distance metrics. One potential measure is the previously mentioned Kullback-Leibler divergence. Suppose one wants to compare the similarity of $\mathbf{x}_1$ and $\mathbf{x}_2$, this can be quantified by

$$\begin{aligned} D_{\mathrm{KL}}(p_{\theta_1}(\mathbf{z} \mid \mathbf{x}_1) \mid\mid p_{\theta_2}(\mathbf{z} \mid \mathbf{x}_2)) &= \mathbb{E}_{p_{\theta_1}}[\log p_{\theta_1}(\mathbf{z} \mid \mathbf{x}_1) - \log p_{\theta_2}(\mathbf{z} \mid \mathbf{x}_2)] \\ &= \int_{\Omega_{\mathbf{z}}} [\log p_{\theta_1}(\mathbf{z} \mid \mathbf{x}_1) - \log p_{\theta_2}(\mathbf{z} \mid \mathbf{x}_2)] p_{\theta_1}(\mathbf{z}_1 \mid \mathbf{x}_1) \mathrm{d}\mathbf{z}. \end{aligned}$$

Let us now consider the approximate posterior, where

$$q_i = q_{\phi_i}(\mathbf{z} \mid \mathbf{x}_i) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}_i), \Sigma(\mathbf{x}_i)) \text{ for } i \in \{1, 2\}$$

and where we hereby denote $\boldsymbol{\mu}(\mathbf{x}_i) = \mu_i$ and $\Sigma(\mathbf{x}_i) = \Sigma_i$. Then the following holds

$$\begin{aligned} \log q_i &= \log \left[\frac{1}{(2\pi)^{\frac{J}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp\Big(- \frac{1}{2} (z - \mu_i)^{\mathrm{T}} \Sigma_i^{-1} (z - \mu_i) \Big) \right] \\ &= -\log(2\pi)^{\frac{J}{2}} |\Sigma_i|^{\frac{1}{2}} - \frac{1}{2} (z - \mu_i)^{\mathrm{T}} \Sigma_i^{-1} (z - \mu_i). \end{aligned}$$

As a consequence, the first factor, $\mathbb{G}_{\log} = \log q_1 - \log q_2$, in the Kullback-Leibler integral can be reduced to

$$\mathbb{G}_{\log} = \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} + (z - \mu_2)^{\mathrm{T}} \Sigma_2^{-1} (z - \mu_2) - (z - \mu_1)^{\mathrm{T}} \Sigma_1^{-1} (z - \mu_1) \right].$$

If we now consider the separate terms in $\mathbb{G}_{\log}$

$$\begin{aligned} \int_{\Omega_{\mathbf{z}}} \log \frac{|\Sigma_2|}{|\Sigma_1|} q_1 \mathrm{d}\mathbf{z} &= \log \frac{|\Sigma_2|}{|\Sigma_1|} \int_{\Omega_{\mathbf{z}}} q_1 \mathrm{d}\mathbf{z} = \log \frac{|\Sigma_2|}{|\Sigma_1|} \\ \int_{\Omega_{\mathbf{z}}} (z - \mu_i)^{\mathrm{T}} \Sigma_i^{-1} (z - \mu_i) q_1 &= \mathbb{E}_{q_1} \left[(z - \mu_i)^{\mathrm{T}} \Sigma_i^{-1} (z - \mu_i) \right]. \end{aligned}$$

Using proprieties from [47] one can express the expected value according to

$$\mathbb{E}_{q_1} \Big[(z - \mu_i)^{\mathrm{T}} \Sigma_i^{-1} (z - \mu_i) \Big] = (\mu_1 - \mu_i)^{\mathrm{T}} \Sigma_i^{-1} (\mu_1 - \mu_i) + \mathrm{Tr}(\Sigma_i^{-1} \Sigma_1),$$

hence

$$D_{\mathrm{KL}}(q_1 \mid\mid q_2) = \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - J + (\mu_1 - \mu_2)^{\mathrm{T}} \Sigma_2^{-1} (\mu_1 - \mu_2) + \mathrm{Tr}(\Sigma_2^{-1} \Sigma_1) \right]$$

Since $\mathrm{Tr}(\Sigma_1^{-1} \Sigma_1) = \mathrm{Tr}(I_J) = J$, where I_J denotes a J-dimensional identity matrix. The above expression can subsequently be applied as a similarity metric for data $\mathbf{x}$ in the lower dimensional $\mathbf{z}$, by parameterizing the approximate posterior using the encoder network.

Chapter 5

Data

The data constituting the basis for the experiments in this thesis is given by *The Cancer Imaging Archive* [5]. It is yielded by a "randomized phase III Trial of Radiation Therapy and Chemotherapy for stage III and IV Head and Neck carcinomas" [5]. Entailing that the patients have a head-neck cancer sprung from *epithelial cells*, i.e. cells lining outer or inner surfaces in the head-neck area [35]. Both CT-scans and RT structure data will be of interest hence these entities and adherent subjects will be discussed in this Chapter.

5.1 CT-scans

The X-ray is an electromagnetic wave [9], and when it passes through biological tissue different amounts of radiation will be absorbed according to the characteristics of the impacted radiated matter. As a consequence, varying amounts of the rays are propagated through the tissue, all according to this absorption. And by measuring the amount of radiation that is emitted after piercing the tissue, i.e. the attenuation, one attains a *radiograph*. This image will be a projection of the internal appearance of the radiated entity, where organs and other structures are visible. The CT-scan is rendered by using a set of X-ray measurements and it produces a cross-sectional image of the scanned entity. This image is constructed by making use of X-rays taken over a range of different angles, over the cross section in question. One may render a number of such cross sectional scans along the same axis and subsequently produce an image volume which conveys a 3-dimensional view of the internal structure. This is of great use in a range of medical practises and disciplines. Moreover, this rendered volume can be divided in 3-dimensional units refereed to as *voxels*. The intensity of each voxel in the image volume can then be measured in *Hounsfield units*. This metric is a relative scale with densities yielded by the CT-scan, in the sense that it is given by a linear transformation of the attenuation coefficients attained when scanning. Voxel-wise this unit is defined as the prevalent attenuation value from the radiation, subtracted and then divided with the attenuation of distilled water, and lastly multiplied by the value 10^3 [25]. The span of the unit differs depending on the body part. In the thorax the range is: $(-800, 700)$, which "is wider than in any other part of the body" [45]. Different windows of Hounsfield units will display matters with an attenuation answering to that range, as a consequence the units chosen will display specific anatomical features. Furthermore, let CT data be denoted $\mathcal{D}_{ct}$. In Figure 5.1 chosen slices of a *Head-Neck* CT-scan can be seen.

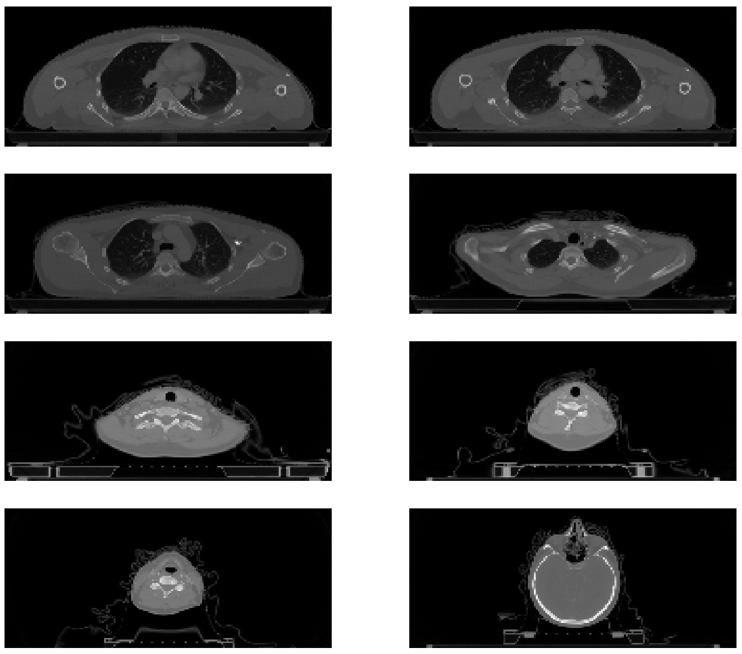


Figure 5.1: Slices of a CT scan for one patient from [5].

5.2 Structure data

The purpose of taking scans of a patient can be e.g. therapeutic or diagnostic. These applications commonly require that organs and internal structures are depicted in the patient image, such that their spatial location in the anatomy is evident. This spatial structural information is called *structural data*, and in radiation therapy several entities are of special interest for the treatment, these constitute the *RT Structure Set.* Where more specifically, the "RT Structure Set [...] defines a set of areas of significance in radiation therapy, such as body contours, tumor volumes [...], OARs, and other ROI" [31]. Let this structure data be denoted $\mathcal{D}_{rt}$.

5.3 Preprocessing

The CT data is given in DICOM format, which is shorthand of *Digital Imaging and Communication in Medicine* [39]. This format and system of communication has become a standard in medicine [39] for handling, transmitting and printing patient images and data [1]. This image data is firstly transformed to Hounsfield units, subsequently the data is caped into the unit window of interest, in this case the entities ranging over biological tissue.

The RT data is given in the DICOM files along with the CT scan; where the ROIs are depicted either manually by a specialist or with support of some image processing tool. Prior to inference of the segmented data, each ROI is given its own image volume, where an entity $\neq 0$ in a pixel indicates that the area belongs to one of the segmented regions. The structures are treated such that each pixel can belong exclusively to a ROI or to the complement or those subsets. In Figure 5.2 the CT-slice of a patient's torso, as well as the contour of segmentation of the skin, can be seen.

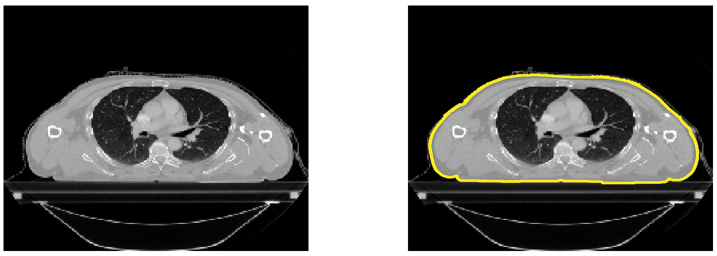


Figure 5.2: Left: Slice of CT- scan of a torso. Right: Contour of segmentation of the skin can be seen as a colored line.

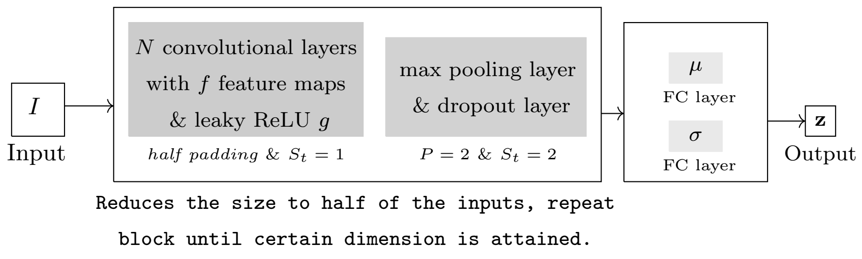
[1] *About DICOM* in the DICOM Library

Chapter 6

Implementation

6.1 Architecture of network

Let us now characterize the network structure designed for the VAE framework. The purpose of the network is outlined in Chapter 4 and the data $\mathcal{D}$, is described in Chapter 5.1 and 5.2. The network will be a FFN comprised mainly of convolutional layers, since the inputs are images of an extensive size. The entire network consists of two connected structures: (i) the encoder network and (ii) the decoder network. As previously elaborated on, the encoder will take an image I as input and yield a latent representation $\mathbf{z}$. Subsequently, $\mathbf{z}$ will be fed as an input to the decoder and the output will be a reconstruction of I. Since the latent $\mathbf{z}$ is of lower dimension than I the encoder network will successively reduce the spatial dimension of the input and thence render its output. On the other hand the decoder network will have to enlarge its input to attain the reconstruction of I. Entailing that the encoder will be built by convolutional layers as well as pooling layers, whilst the decoder network will have transpose convolutional layers. In Figure 6.1 a schematic graph of the encoder network can be seen.



I
Input
N convolutional layers with f feature maps & leaky ReLU g
half padding & $S_t = 1$
max pooling layer & dropout layer
$P = 2$ & $S_t = 2$
μ
FC layer
σ
FC layer
$\mathbf{z}$
Output
Reduces the size to half of the inputs, repeat block until certain dimension is attained.

Figure 6.1: Encoder network consisting of two blocks: (i) convolutional block, which reduces the spatial dimension of I. This block is repeated until a sought dimension of its output is attained. (ii) Fully connected block, here the output of the first block is fed through one FC layer, then it is separated into two FC-layers to yield the parameters $\{\mu, \sigma\}$.

In Figure 6.1 one can see that the encoder firstly convolves the input N times, followed by max pooling and dropout. In this scheme the padding in the convolution preserves the dimension and the max pooling reduces it to half its input size; all according to the strides S_t, the pool size P and kernel size K. To attain a dimension that is computationally sensible for fully connected layers, this first block is repeated until the spatial dimension is significantly reduced, e.g. suppose $I \in \mathbb{R}^{512\times512}$ then 6 stacked such convolutional blocks yield an output $\in \mathbb{R}^{8\times8}$; which would yield a fully connected layer with $8 \cdot 8 = 64$ input nodes. In addition to being computationally feasible, this deep structure with repeated convolutions entails that more abstract and meaningful features of the images can be captured.

The decoder network consists mainly of transpose convolutional layers. With a certain stride S_t, kernel K and pooling, these layers will yield an enlarged version of its input. To preserve as much as possible, from the latent $\mathbf{z}$, the decoder network starts with two fully connected layers, consequently each pixel of the input to the first convolutional layer will inherently reflect the characteristics of $\mathbf{z}$. In Figure 6.2 a schematic graph of the decoder network can be seen.

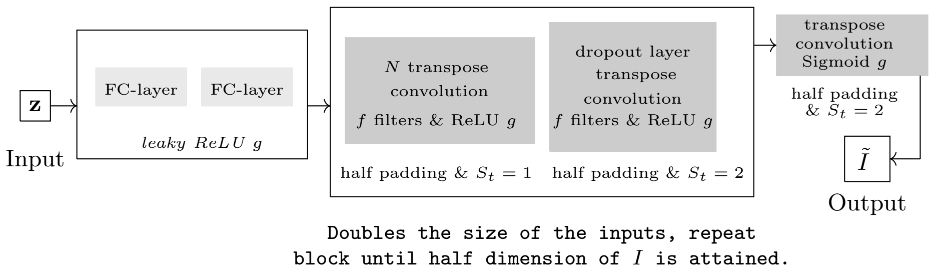


Figure 6.2: Decoder network consisting of two blocks: (i) fully connected block, here the output of the encoder is fed through two FC layers, then it is propagated. (ii) The convolutional block, which enlarges the spatial dimension of I. This block is repeated until a sought dimension of its output is attained. Lastly, a transpose convolutional layer will yield the reconstruction $\tilde{I}$.

In Figure 6.2 it is evident that the decoder initiates with fully connected layers, followed by transpose convolutional layers and dropout layers. Much like the encoder network, this network has a convolutional block that is repeated until a dimensional constraint is fulfilled. The number of filters as well as the ReLU activation is equal for all layers in this block. When half the dimension of the input I is attained the output is fed to the last convolutional layer, this entity has one filter and a stride that doubles its input, $\tilde{I}$ is subsequently generated. The hereby described architecture is applicable on $\mathcal{D} \in \mathbb{R}^2$ and $\mathcal{D} \in \mathbb{R}^3$, i.e. on individual slices of CT as well as on a stack of slices.

6.1.1 RT-Structure data in the VAE framework

When the data also contains RT structure data, s.t. $\mathcal{D} = \mathcal{D}_{\text{ct}} \cup \mathcal{D}_{\text{rt}}$, the network structure is altered, firstly by adding channels C for $\mathcal{D}_{\text{rt}}$. Entailing that one channel will contain the CT-data, and the remaining channels structure data. Each ROI is given one channel and an additional channel comprised of the complement to the ROIs is also included, implying that each pixel can be either in one of the ROIs or in their complement. In reality this might not be the case, since e.g. target could intertwine with organs, and there might be volume elements where several ROIs are adjacent. Secondly, the framework is altered by adding an entropy component to the loss function. In Chapter 4.2 the loss function $\mathcal{L}_{\text{VAE}}$ was characterized according to

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{logl}} + \mathcal{L}_{\text{prior}},$$

where $\mathcal{L}_{\text{logl}}$ reflects the level of similarity attained between I and $\tilde{I}$ for $\mathcal{D} = \mathcal{D}_{\text{ct}}$, and $\mathcal{L}_{\text{prior}}$ quantifies how well the posterior follows the distribution of the prior. When this framework is extended to include structure data, the level of similarity for CT-data and RT-structure data will be treated differently. Like before, the discrepancy of the CT-data will be measured as the difference between two continuous gray scale values, whilst the deviance for RT- structure data will be given by how well the network assigns each pixel to the prevalent classes. Hence, the output of the channels with structure data will be a ROI-wise probability distribution over the possible classes. In the table below an example of this is given, where $\sum_{r=1}^{4} p_r = 1$ and the pixel is from the ROI: *lung.*

Considered ROI	*Target*	*Lung*	*External*	*Complement*
Input pixel (i, j)	$q_1 = 0$	$q_2 = 1$	$q_3 = 0$	$q_4 = 0$
Output pixel (i, j)	p_1	p_2	p_3	p_4

Ultimately, the network assigns high probability to p_2 for pixel (i, j). To make the RT-structure output from the network a valid probability distribution a softmax function σ is applied; defined as,

$$\sigma : \mathbb{R}^R \to \left\{ p \in \mathbb{R}^R \mid p_r > 0, \ \sum_{r=1}^{R} p_r = 1 \right\}$$
$$\text{where} \quad \sigma(z_j) = \frac{e^{z_j}}{\sum_{r=1}^{R} e^{z_r}} \quad \text{for} \quad j \in (1, \ldots, R).$$

Subsequently, $\{p_r\}_{r=1}^{R}$ is attained. The similarity between input membership between classes q and output p can be measured with cross entropy $\mathcal{H}$, where

$$\mathcal{H}(p, q) = -\sum_{r=1}^{R} p_r \log q_r.$$

This entity will constitute the loss contribution from the RT-structure data to the VAE framework. Entailing that

$$\mathcal{L}_{\text{cre}}(\mathcal{D}_{\text{rt}}) = \mathcal{H}(p,q) = -\sum_{r=1}^{R} p_r \log q_r$$

and that the entire loss is given by

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{logl}}(\mathcal{D}_{\text{ct}}) + \mathcal{L}_{\text{prior}}(\mathcal{D}) + \mathcal{L}_{\text{cre}}(\mathcal{D}_{\text{rt}}).$$

Where $\mathcal{L}_{\text{logl}}(\mathcal{D}_{\text{ct}})$, as before reflects the similarity between input and output of the CT-data, $\mathcal{L}_{\text{prior}}(\mathcal{D})$ the divergence of the posterior from a standard Gaussian and $\mathcal{L}_{\text{cre}}(\mathcal{D}_{\text{rt}})$ the discussed entropy for the RT-structure data.

Furthermore, the decoder network will also be altered somewhat to cater for ROI channels. This is done by dividing the network at some point in the convolutional block, and thence having a number of separate network layers for the CT-data and RT-structure data respectively.

6.1.2 Special considerations for 3-dimensional inputs

The 3-dimensional input $\mathcal{D} \in \mathbb{R}^3$ is commonly of spatial extension $\{H \times W \times D\} = \{512 \times 512 \times 100\}$. Hence, for ease in the implementation the dimension can be altered by interpolation, such that $H = W = D$. This commonly entails that the CT-stacks are interpolated in their respective depth direction. Subsequently, the network acts on a symmetrical entity which simplifies the handing of the data in the successive reduction or magnification of the dimension. This interpolation constitutes a form of *upsampling*, i.e. a manner to increase the dimension of an entity.

6.2 Stochastic considerations

The probabilistic setting characterized in Section 4.2.3 will be applied in the architecture of the VAE. Where all inherent distributions are spherical Gaussians and where the VAE has fixed variance, entailing that the decoder will reconstruct the network input in a deterministic fashion, according to reasoning in Section 4.3.

6.3 Deep learning library and applied algorithms

The numerical implementation was constructed with *TensorFlow* 1.5.0, which is an open source library developed by *Google Brain* "for high performance numerical computation" [1]. Where the *Adam algorithm* is applied to optimize the network parameters, this algorithm is similar to SGD, here the learning rates are however adaptive. "The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients" [27]. Consequently, it is appropriate for large scale problems [27].

[1]About TensorFlow

Chapter 7

Results

There are several results and metrics of interest for evaluation of the VAE network's performance. Firstly, one evident measure is visual inspection of the input to the encoder versus reconstruction from the decoder network, optimally these entities are alike. Secondly, one can look at the convergence of training- and test error during training. Moreover, the samples from the decoder can also be inspected, to see whether they have the anatomical appearance of a CT-scan. One can also investigate how the latent space $\Omega_{\mathbf{z}}$ relates to the image space $\Omega_{\mathbf{x}}$ in a range of manners. All these evaluation methods can be applied for various latent dimensional spaces, varying configuration of the network as well as different hyper parameter setups; additionally, the specific data set used for inference is also a matter of choice. Hence, results rendered from several aspects, data sets as well as setups will be presented, to convey the effects on the result of all these factors.

7.1 CT-scans

7.1.1 2 dimensional case

Let us study the performance of the VAE for latent $\mathbf{z} \in \mathbb{R}^{12}$. The network architecture characterized in Section 6 is applied, with a dropout probability of $p = 0.7$ for regularization and 24 filters in each convolutional layer. The data is comprised of CT-slices from 40 patients, where two nearly situated images in proximity of the heart have been chosen for each patient and the (128, 128) pixels in the center are selected. The objective when choosing this set and pixel size is to verify whether the framework can extract details from these images and properly reconstruct them, since this is more evident in small scale images. Larger structures will be examined later on. In Figure 7.1 the average of the loss function can be seen, as a function of the number of epochs run, for this initial experiment.

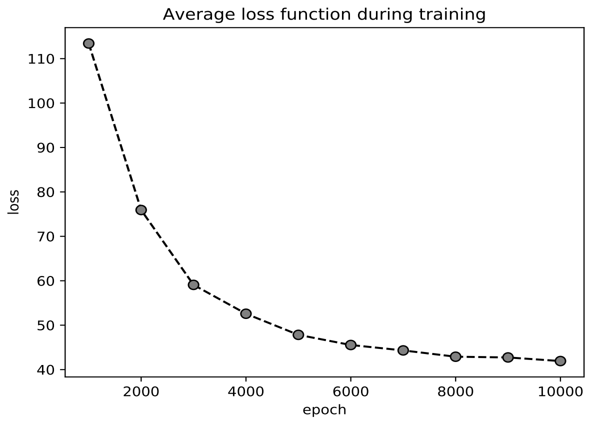


Figure 7.1: Investigation of loss elaborated on in Section 4.2.1, derived on training images.

In Figure 7.1 it is evident that the error monotonically decreases as the epochs increases, which constitutes a sought converging behaviour. In Figure 7.2 one can see a VAE training input and the network reconstruction, at each of the points where the average loss is derived during training.

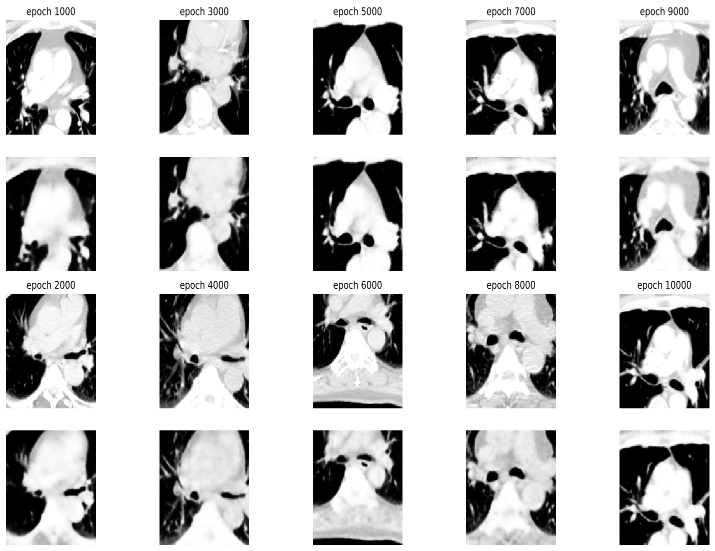


Figure 7.2: Training input and output of VAE network at certain epochs. The upper figure for each epoch constitutes the network input and the lower figure is the reconstruction from the network.

In Figure 7.2 it is evident that the VAE gradually increases its performance as the epochs are run, as the input and reconstruction becomes more and more alike. This is congruent with the decrease of loss apparent in Figure 7.1. To further study the behaviour of the VAE, a set of training images and their respective reconstruction at the last epoch, is presented in Figure 7.3.

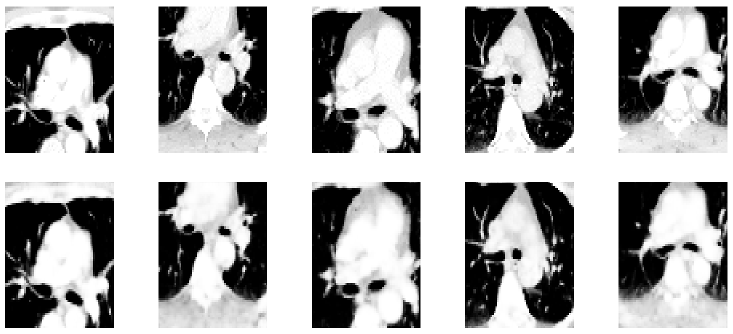


Figure 7.3: Training input to encoder in upper row, and their respective outputs from the decoder in lower row. Taken at last epoch of training.

In Figure 7.3 the network's ability to capture and reconstruct details can be seen. The quality of these training reconstructions indicates that the VAE has the ability to reconstruct medical images. Moreover, to investigate the network's ability to generalize to unseen data, following training on few images, test images are also studied. These images are held out during training, thence the performance on this data will indicate how well the network fares with new CT-scans. In Figure 7.4 two test images and their decoded reconstructions can be seen.

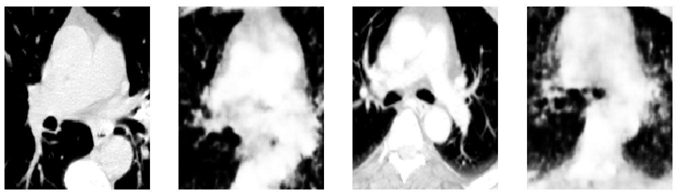


Figure 7.4: Test data, Left: patient 1, input to the left and output to the right. Right: patient 2, input to the left and output to the right.

In Figure 7.4 It is apparent that the performance of the network is not as good as on training data, since the output images are not as similar to the input as with training data. This is however an expected result since some level of overfitting can render a better performance on training, and on this modest data set this will likely occur; even though measures to counteract over fitting is taken. In 7.25 the trade off between training and test error is conveyed.

If one considers the appearance of the anatomical CT-data, the images are evidently quite varied, both in overall structure and in the inherent details. This makes the learning and subsequent test reconstruction intricate. Hence, to get a better sense of how well the network fares test-wise on data with an overall structure that is known to the network, one additional test experiment was carried out. This experiment consists of training on 40 CT-slices and subsequently testing on 40 slices situated quite close to those trained on, examples of this data can be seen in Figure 7.27. This experimental construction will entail that the network has seen at least one image with an overall structure prevailing in the test images. In Figure 7.5 the result of the experiment can be seen, the network had 24 filters, a latent $\mathbf{z} \in \mathbb{R}^2$ and no dropout.

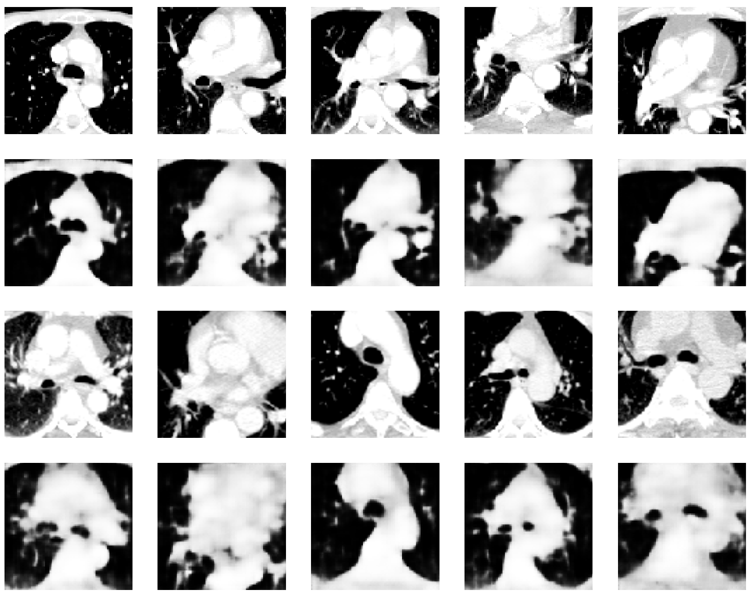


Figure 7.5: After 1000 epochs of training, the input to the network is situated above the output for both rows of examples shown.

In 7.5 it can be seen that the network generalizes better, when early stopping at 1000 epochs during training is applied and when the network has seen something similar to the images tested. This illuminates some considerations regarding training on VAEs and their ability to generalize. To compare the test- with the training performance after 1000 epochs, a training input and output after 1000 epochs of training can be seen in Figure 7.6.

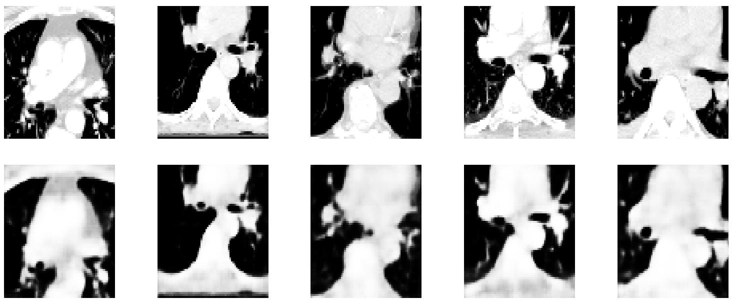


Figure 7.6: After 1000 epochs of training, training input in upper row and output in lower. The data and experimental setup is analogous to the setting described in Section 7.1.1.

If one compares Figure 7.5 with 7.6 it is evident that the train versus test performance appear to be similar, which indicates that special care must be taken regarding representation in the training set, of the anatomy apparent in the test set.

Let us now return to the experiment characterized in Section 7.1.1. To verify how well the network can generate artificial data, samples were drawn and these can be inspected in Figure 7.7.

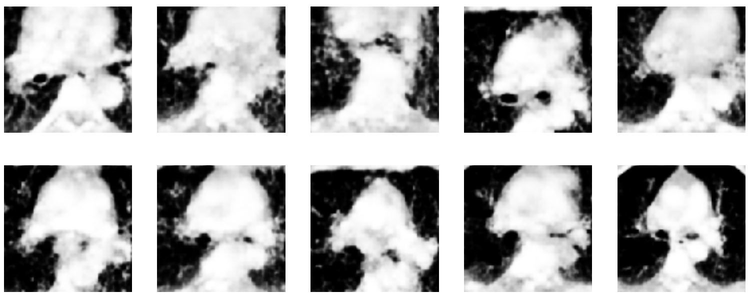


Figure 7.7: Samples drawn from decoder.

In Figure 7.7 it is apparent that some samples are distorted yet they have anatomical structures, since *gross anatomy* is clearly evident [1]. This leads us to conclude that the VAE-framework may be applied for generation of artificial patient data, for applications such as augmentation in ML.

The relation between the latent space $\Omega_{\mathbf{z}}$ and the image space $\Omega_{\mathbf{x}}$ can be investigated by drawing two samples $\{\mathbf{z}_1, \mathbf{z}_2\}$ from the prior, and interpolate between these points. Subsequently, the motion between these latent points can be visualized in the image space, by decoding each such latent position. In Figure 7.8 such a motion in the image space can be seen, where the underlying connection between the images is the described interpolation in the latent space.

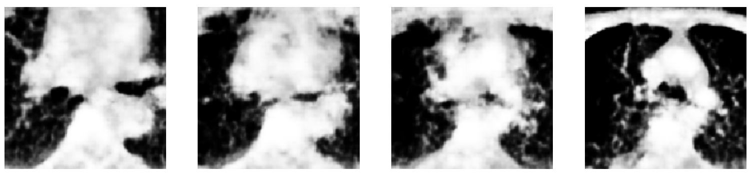


Figure 7.8: Decoded latent points, interpolated in $\Omega_{\mathbf{z}}$ between the latent positions yielding the outermost images.

One can also study how the patient images are related to the latent space, by encoding two patient images $\{\mathbf{x}_1, \mathbf{x}_2\}$, thence attaining their latent representations $\{\mathbf{z}_1, \mathbf{z}_2\}$. Then one can interpolate between these encoded representations in the latent space, and decode these coordinates to see how the latent grid and its image representation is altered between patients. In Figure 7.9 the inputs $\{\mathbf{x}_1, \mathbf{x}_2\}$ can be seen and in Figure 7.10 the decoded interpolation is shown, where the outermost images correspond to $\{\mathbf{z}_1, \mathbf{z}_2\}$.

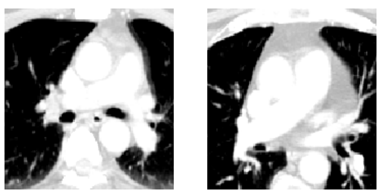


Figure 7.9: Two patient images, $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively.

[1]comments regarding how realistic the samples are comes from consultation with a *radiologist*

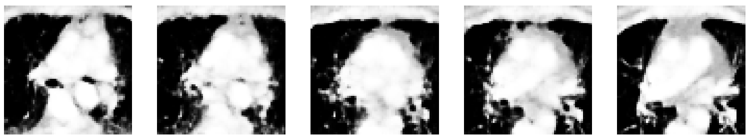


Figure 7.10: Decoded latent points, interpolated in $\Omega_{\mathbf{z}}$ between the latent positions yielded by the outermost images.

In both Figure 7.8 and 7.10 a successive alteration in the decoded images can be seen, this implies that the latent space has related specific anatomical behaviours of $\mathbf{x}$ to certain subsets of $\Omega_{\mathbf{z}}$. This constitutes sought behaviour since this enables inference in the latent space, regarding $\mathbf{x} \in \Omega_{\mathbf{x}}$. To more fully comprehend how the latent space relates to the patient images one can visualize the learned manifold, which is elaborated on in Section 4.4. For a $\mathbf{z} \in \mathbb{R}^2$ and an architectural setup analogous to that of the experiment with $\mathbf{z} \in \mathbb{R}^{12}$, but with 2000 epochs, yielded the manifold seen in Figure 7.11.

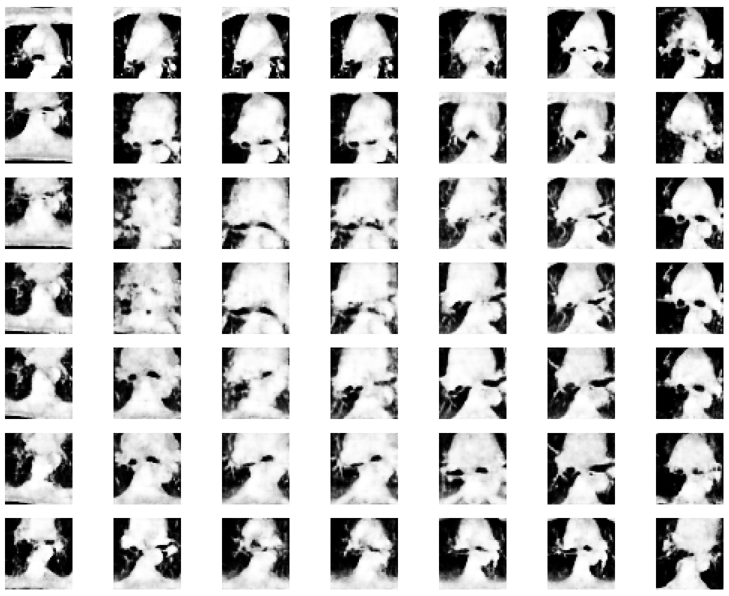


Figure 7.11: Learned manifold, 2000 epochs, latent space $\Omega_{\mathbf{z}} \subset \mathbb{R}^2$, 0.7 dropout and 24 filters.

In the manifold above one can see how some anatomical features are related to parts of the latent space. And yet again one can note how the representation is gradually altered as one moves in $\Omega_{\mathbf{z}}$. The proximity of the various anatomical structures seen in this manifold demonstrates how the VAE apprehend these biological entities and how close they relate to one another.

To investigate how well the VAE can extract and reconstruct larger structures as well as the inherent details, one can study how the network fares for images of dimension (256, 256) pixels. The data in the previous experiment is chosen, but here with the enlarged (256, 256) pixel dimension. The network setup is analogous to that in Section 7.1.1, but the dropout probability is now set to $p = 0.95$. In Figure 7.12 examples of the test input and output of the network can be seen. In the Figure it is apparent that the overall structure of the anatomy is captured.

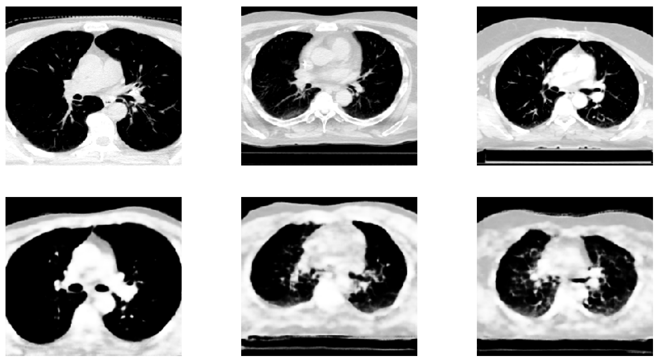


Figure 7.12: Test data, input in upper row and output from network in lower.

Now let us study the behaviour of the latent space for these larger structures. In Figure 7.13 interpolated points in latent space and their decoded representations can be seen.

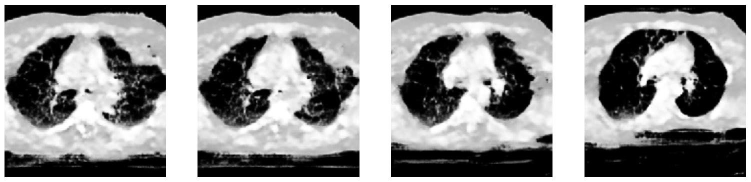


Figure 7.13: Decoded latent points, interpolated in $\Omega_{\mathbf{z}}$ between the latent positions yielding the outermost images.

Like before one can see a gradual alteration of the decoded representation in Figure 7.13. Now consider the ability to create new artificial data of this larger dimension. In Figure 7.14 draws from the decoder can be seen.

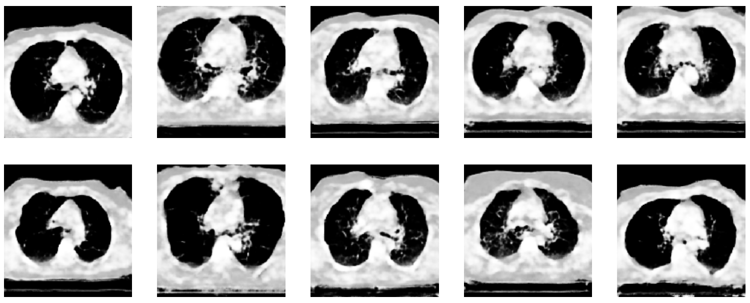


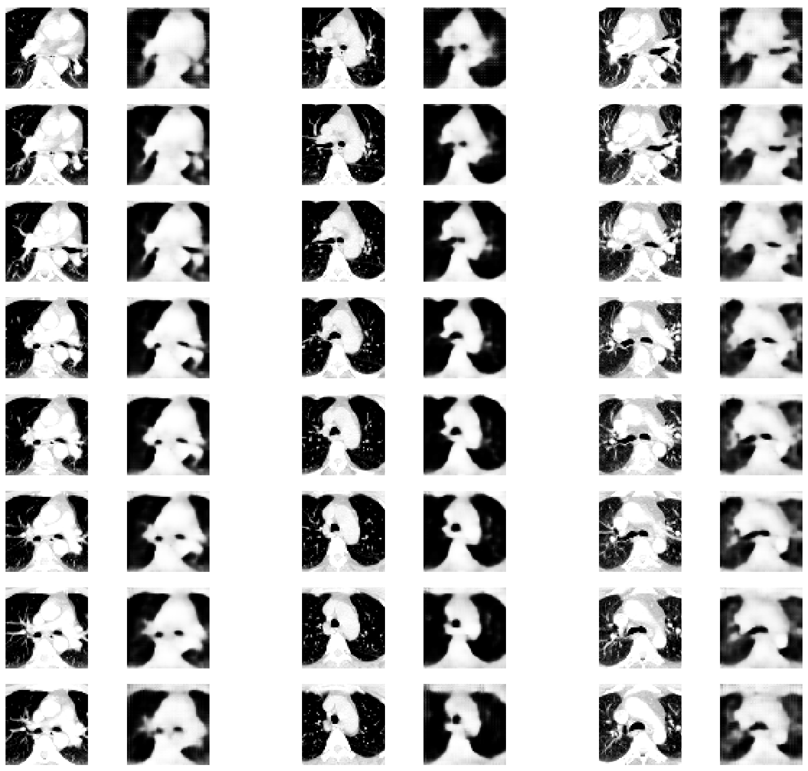
Figure 7.14: Samples drawn from decoder.

In Figure 7.14 it can be seen that the anatomical structural behaviour is quite realistic with body like gross anatomy rendered by the network.

Comparing the networks performance for the smaller $I \in \mathbb{R}^{(128,128)}$ images to those of $I \in \mathbb{R}^{(256,256)}$ pixels, one can note that the test performance appear to have been somewhat heightened, which can be seen when comparing Figure 7.4 to 7.12. The overall structure seems to have been better captured in the larger dimension. One reason for this might be that the anatomical structure in the larger images is more coherent between patients than in the smaller case, since the rib-cage and heart is apparent in all samples $\mathbf{x} \in \mathbb{R}^{(256\times256)}$, whilst various diverse parts of the heart region is prevalent in the smaller $\mathbf{x} \in \mathbb{R}^{(128\times128)}$ images. Much like the better test performance attained in Figure 7.5 relative to 7.4, where the network had seen similar structures prior to testing.

7.1.2 3 dimensional case

Let us now consider entire CT-images, i.e. stacks of individual CT-slices which renders a 3-dimensional VAE setting. For this experiments 10 subsequent slices in proximity of the heart and with extension $(128, 128)$ pixels, are extracted from each of 40 patient images. Prior to inference these stacks are interpolated in their depth directions, such that $10 \times (128, 128) \rightarrow 128 \times (128, 128)$. Thereafter, the VAE network is trained on the data and the results can be seen in Figure 7.15.



(a) Patient 1. (b) Patient 2. (c) Patient 3.

Figure 7.15: Training input and output after 5000 epochs of training. Every 16:th slice of the 128 available is shown. The left column for each patient constitutes the input and right column the reconstructions for the image on the same row.

In Figure 7.15 it can be seen that the overall structure is captured in the network output. Moreover the nuances and changes from input slice 1 to slice 128 is also apparent in the 8 reconstructions for all patients. Indicating that a VAE may be applicable on 3-dimensional medical image data. The performance of a VAE in a 3-dimensional framework is comparable to that of the 2-dimensional. The network reconstructions appear to have successfully captured the anatomical structures, much like in the lower dimensional case. The inherent details in the images is however not preserved in the same detail, which is somewhat expected since fewer epochs of training was run and the larger structure and increased dimension presents a tougher learning problem.

7.2 CT scans and RT structure data

The framework is extended to include RT-structure data as well. The architectural implications of this is elaborated on in Chapter 6.1.1. Moreover, the basis for the subsequent experiment is data from 40 patients, where both CT images $\mathcal{D}_{ct}$ and RT-structure data $\mathcal{D}_{rt}$ from ROI: *skin* and *spinal cord*, $(256, 256)$ pixels are included. The network is comprised of 24 filters, a 12 dimensional latent space, no dropout and a network separation in the decoder at size 32×32 in the convolutional block. In Figure 7.16 the results during training for one ROI and the CT can be seen.

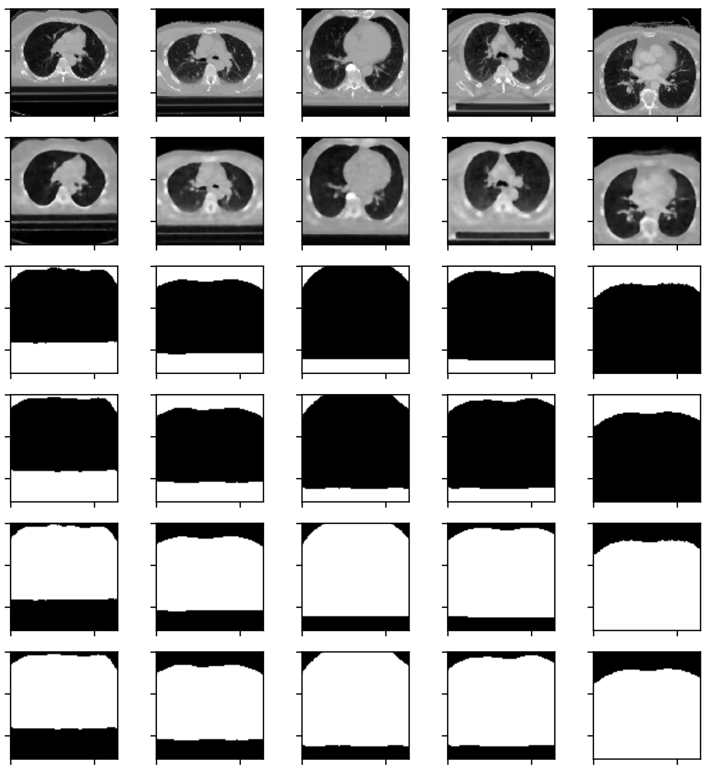
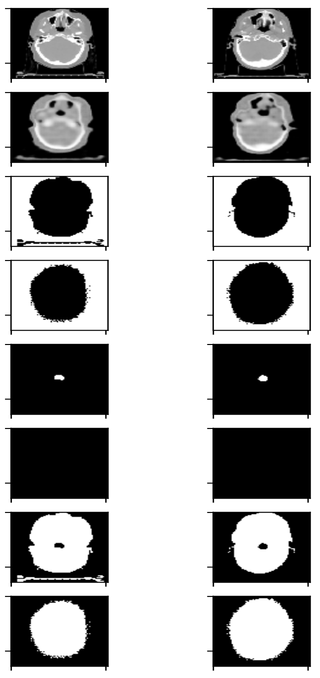


Figure 7.16: Training input and output of network after 4000 epochs. The top image is the input CT-image and the one below the output. On the third row the complement to the RT-structure data can be seen, and on the subsequent row its network output. On the last two rows the input contour for skin and output can be seen, respectively.

In Figure 7.16 it is apparent that both input of CT and RT-structure data has a close likeness to the network output. The segmentation of the spinal cord was omitted since the results were that no ROI was identified by the network. The reason for this is probably that this ROI has such a modest amount of pixels in relation to skin or complement, which may render the cross entropy loss negligible for this ROI compared to the others. One way to remedy this can be to use *weighted cross entropy*, here the contributions to the loss from different ROIs can be modulated. To investigate this further, one more CT- and RT structure experiment was carried out. The network setup is analogous to that of the previous experiment. Here CT-slice from the head was extracted, along with the ROIs *skin* and *brain stem*. In Figure 7.17a the result with ordinary cross entropy can be seen and in Figure 7.17b the result by applying weighted cross entropy is shown.


(a) Ordinary cross entropy loss.

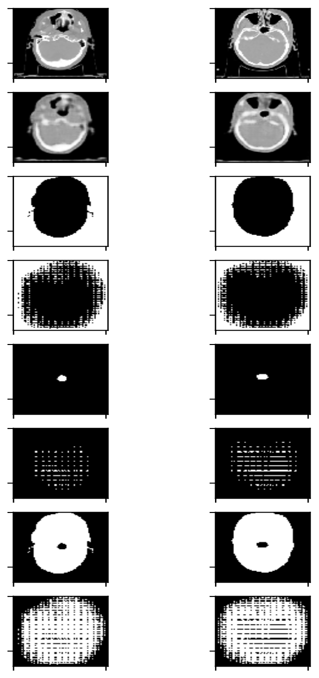

(b) Weighted cross entropy loss.

Figure 7.17: Training input and output of network. The top images constitutes the input CT-image and the ones below the output. On the third row the complement to the RT-structure data is seen, and on the following row its network output. The subsequent two rows shows brain stem input and output. And on the last two rows the input contour for skin and output can be seen.

In Figure 7.17 one can see that the small ROI brain stem is not identified by the network when the cross entropy in not weighted. When the weights are applied however, the network identifies that a ROI is somehow present in the picture. The training was nonetheless less robust when the entropy was weighted and hence diverged more often. Further experiments could be carried out to investigate this matter, e.g. by looking at ROIs with slightly more equal extension. The reason this is not performed here is that the data set did not appear to have a moderately sized common ROI for all patients.

Moreover, these preliminary results demonstrates that the VAE may very well be applicable on not only CT-data but also segmented data. This shows promise towards VAE applications concerning RT structure data as well as general segmentation.

7.3 Evaluation of encoder

The relationship between the prior distribution and the chosen approximate posterior can be investigated by evaluation of the encoder, according to the reasoning in Section 4.5. In Figure 7.18 $\tilde{p}(\mathbf{z})$ for a Gaussian encoder and $\mathbf{z} \in \mathbb{R}^2$ can be seen. It is evident that the probability mass of $\tilde{p}(\mathbf{z})$ is centered around the origin, and it has the appearance of a mixture of Gaussians. The variance of the density exceeds that of the true prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ somewhat.

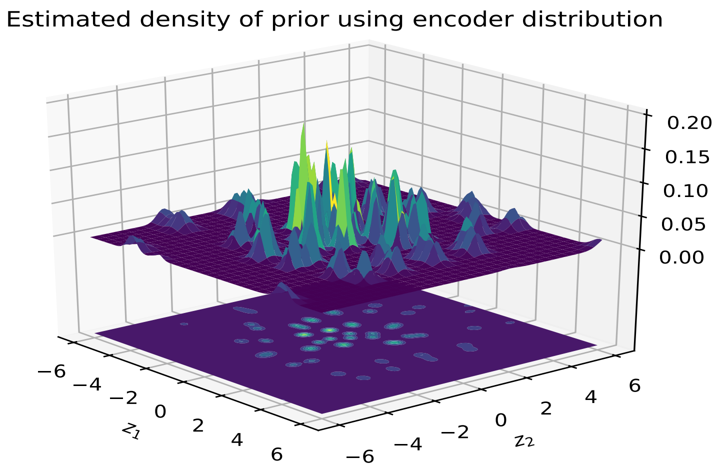


Figure 7.18: Estimated density of prior, derived after 5000 epochs for the 40 patients characterized in Section 7.1.1.

Let us now study the characteristics of $\tilde{p}(\mathbf{z})$ when the data is augmented, according to the reasoning in Section 4.5.2; this can be seen in Figure 7.19. This augmentation was derived with $\{\tilde{\mathbf{x}}_a\}_{a=1}^{10}$ every hundredth epoch, the previous augmented data was exchanged with new draws every round. It is evident that the data augmentation sharpens the probability-peaks and reinforces the disconnection between the separate clusters of probability mass.

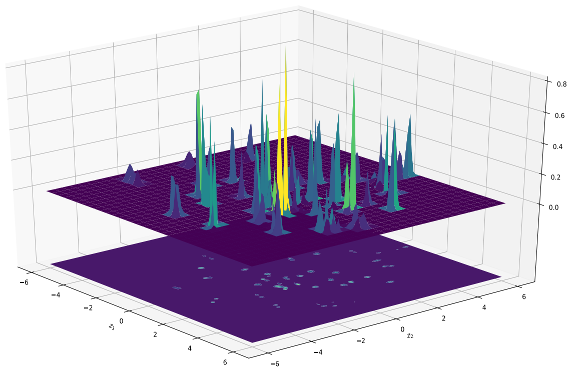


Figure 7.19: Estimated density of prior using encoder distribution. Derived after 500 epochs for the 40 patients characterized in 7.1.1, with augmentation executed every hundredth epoch.

The behaviour of the estimated prior can be a result of the few training examples available, each of the patient images appear to be related to isolated parts in the latent space, thence rendering one separate probability peak each, apparent in Figure 7.18 and 7.19. The reason behind this could be that each image has a quite specific characteristic appearance, that is isolated in its own subset of $\Omega_{\mathbf{z}}$. If the set was more extensive these subsets will probably have more overlap, thence rendering a better estimate of the prior.

7.3.1 Data augmentation by approximate prior

Let us now study the characteristics of the encoder and decoder under augmentation. The latent space is set to $\mathbf{z} \in \mathbb{R}^2$ and 24 filters are prevalent in the networks. After 10000 epochs of training the network demonstrates the test behaviour seen in Figure 7.20

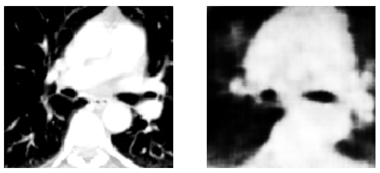


Figure 7.20: Test slice run after last epoch of training, input to the left and output to the right.

In Figure 7.20 it is apparent that the details are not preserved but that the overall structures are captured. Now consider the samples drawn from a decoder trained with this augmented data, this can be seen in Figure 7.21.

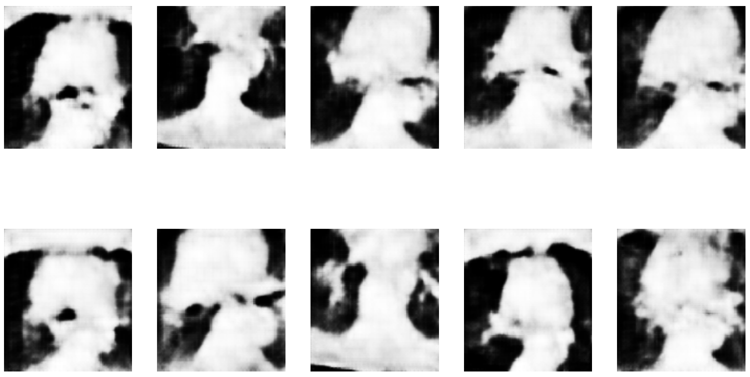


Figure 7.21: Samples drawn from decoder trained with augmentation.

The samples seen in Figure 7.21 hold the structural appearance of anatomical data, the inherent details of such data is however not apparent.

Let us now study how the representation changes as one interpolates in the latent space $\Omega_{\mathbf{z}}$, and decode the points; this can be seen in Figure 7.22. It is evident that the decoded representation gradually alters, and that the structure of the samples demonstrates somewhat anatomical features.

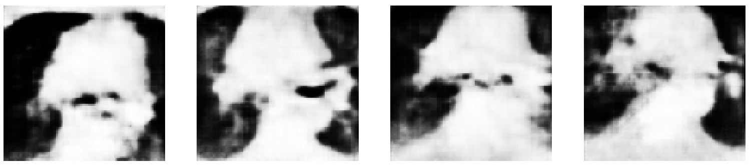


Figure 7.22: Decoded latent points, interpolated in $\Omega_{\mathbf{z}}$ between the latent positions yielding the outermost images.

To further examine how the augmentation effects the VAE performance one can investigate the learned manifold resulting from this training. In Figure 7.23 this manifold can be seen; evidently there is a successive change as one moves in $\Omega_{\mathbf{z}}$, and in parts of the space the decoded latent positions yields anatomically looking structures. There is however parts where the representation is quite distorted and has a non-anatomical appearance.

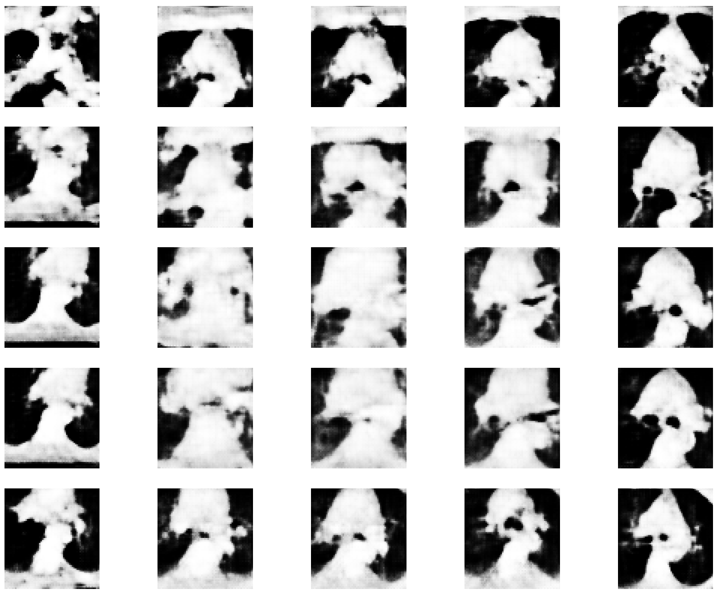


Figure 7.23: Learned manifold over the latent space $\Omega_{\mathbf{z}} \subset \mathbb{R}^2$ yielded with augmentation.

The augmentation by the approximate prior did not appear to heighten the performance of the VAE. The less detailed samples and reconstructions may be an effect of the augmented samples missing these entities. Hence, encoding of those features is not encouraged by those training examples. Upon inspection of the manifold in Figure 7.23 one can see that parts of the latent space has encoded structures that are non anatomical. A reason for this may be the small amount of data available, rendering parts of the latent space none-related to anatomical structures. Since the framework seems to have mapped anatomical structures to a small local neighborhood around each $\mathbf{z}$ point in the latent space, leaving parts of $\Omega_{\mathbf{z}}$ near unrelated to the input. Subsequently, these subsets will yield a curious decoded appearance.

7.4 Investigation of loss

Now let us consider how different entities will influence the loss during training. One evident quantity of interest is the dimension of the latent space $\mathbf{z} \in \mathbb{R}^J$. In Figure 7.24 this can be seen for 5000 epochs and varying J.

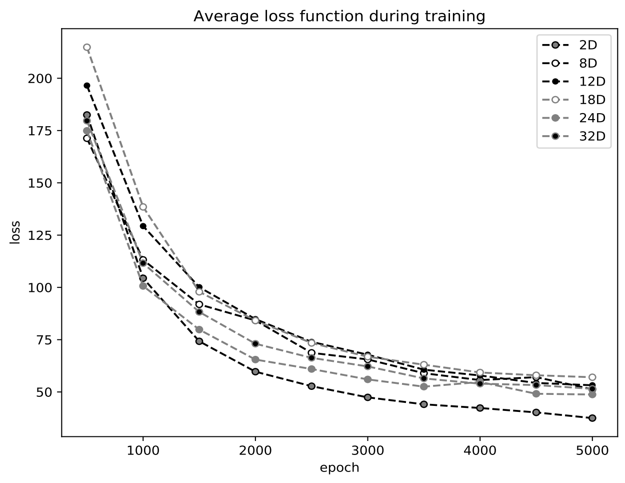


Figure 7.24: Loss during training, for varying dimension of latent space, no dropout and 24 filters, (128, 128) pixels.

In Figure 7.24 it is apparent that the dimension of the latent space does not have a dramatic effect on the convergence of the loss. However the low dimensional $\mathbf{z} \in \mathbb{R}^2$ does seem to be somewhat superior to the other dimensions in this specific regard. Moreover, one additional property of interest is the train- versus test error, since this will indicate how well the network can generalize on small data sets. In Figure 7.25 this loss comparison can be seen.

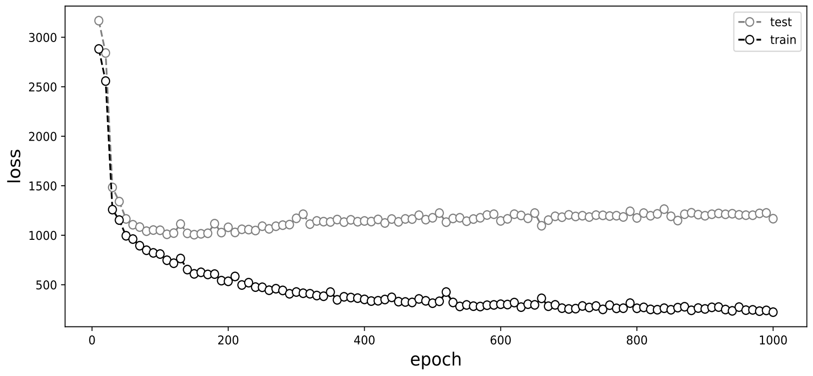


Figure 7.25: Average loss during training, on both training set and test set. The experiment is run on images from 40 patients, where 10 CT-slices are extracted from each patient. This yields 400 images, 200 for test and 200 for training. A dropout of 0.9, 32 filters, and (128, 128) pixels are applied.

The train- versus test loss in Figure 7.25 demonstrates that the test loss starts to increase quite early during training, which may indicate that over fitting is commenced, even though it is counteracted with dropout. The comparatively small data set to train on may be the reason for this. Furthermore, let us now consider how the entropy of the prior influences the loss during training. In Figure 7.26 the loss for various prior variance, and hence also entropy, can be seen.

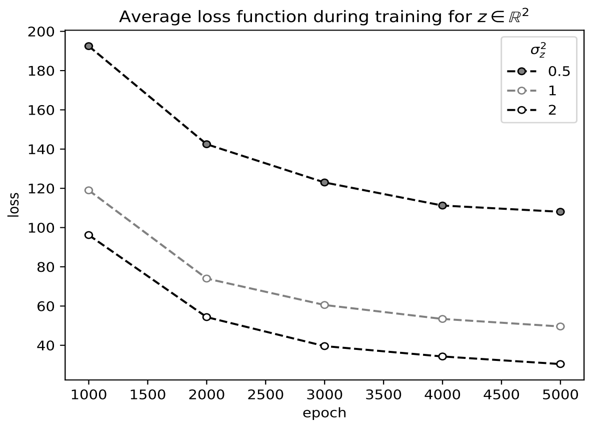
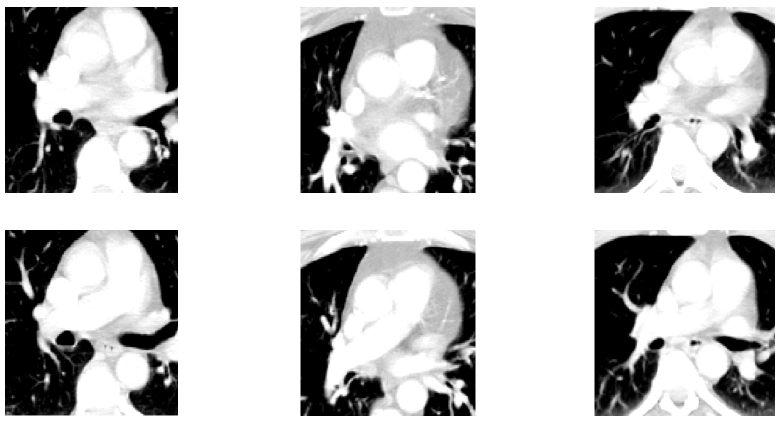


Figure 7.26: Loss during training, for varying prior variance, no dropout and 32 filters, (128, 128) pixels for the 40 patients described in 7.1.1.

In Figure 7.26 one can see that the variance and resulting entropy of the prior effects the loss notably. For the variances $\sigma_{\mathbf{z}}^2$ tested, a higher variance leads to a lower loss.

7.5 Investigation of feature extraction

Let us consider the feature extracting ability of the VAE framework. The basis of the subsequent experiment is 20 CT-slices chosen from each of the CT-scans of 10 patients, where the first and third slices where extracted from each patient. In Figure 7.27 examples of this data can be seen.



(a) Patient 1. (b) Patient 2. (c) Patient 3.

Figure 7.27: 2 slices of a CT-scan for 3 patients. The upper slice is the first one in the image stack and the lower the third slice.

The network is trained on the first image from all 10 patients, thence the resulting parameters $\{\boldsymbol{\mu}(\mathbf{x}_{1i})), \boldsymbol{\sigma}(\mathbf{x}_{1i})\}_{i=1}^{10}$ are rendered. Next, the 10 third images in the slices are pushed through the encoder and the resulting parameters $\{\boldsymbol{\mu}(\mathbf{x}_{3i})), \boldsymbol{\sigma}(\mathbf{x}_{3i})\}_{i=1}^{10}$ are obtained. Ultimately, the distribution characterized by $\{\boldsymbol{\mu}(\mathbf{x}_{1i})), \boldsymbol{\sigma}(\mathbf{x}_{1i})\}$ is similar to that of $\{\boldsymbol{\mu}(\mathbf{x}_{3i})), \boldsymbol{\sigma}(\mathbf{x}_{3i})\}$. Since this entails that the encoder maps similar images to similar distributions in the latent space. One may measure how congruent the distributions are by using Kullback-Leibler divergence, according to the reasoning in Section 4.6. This toy experiment demonstrated that the images from the same patient had an average divergence of $\bar{D}_{KL} \approx 1749$ and that the images from different patients rendered an average divergence of $\bar{D}_{KL} \approx 117854$. Furthermore nine out of ten patients had the lowest divergence between its own respective images. A slightly larger experiment with 20 patients and hence 20 images from each respective slice, demonstrated a divergence of $\bar{D}_{KL} \approx 10074$ for pictures from the same patient, versus $\bar{D}_{KL} \approx 225168$ for pictures from different patients. This further supports the feature extracting ability of the VAE.

Chapter 8

Discussion

8.1 Reflection upon results

Let us now discuss and reflect upon the attained results and the exhibited behaviour of the VAE network. If one begins with beholding the ability of the VAE to reconstruct an input consisting of medical images, it is evident that the method is proficient and quite precise on training data; thus demonstrating that the method has the potential to capture the versatile and intricate details of medical images. Moreover, by considering the feature extracting ability of the VAE one can establish that the quality of the training reconstructions indicates that the network indeed finds salient features of the data. Since there is a distinct similarity between the reconstruction and the input, the network must be successful in encoding the features that are prevalent in the reconstruction. Furthermore, let us reflect upon the ability of using the code for feature comparison between patients. The result of the performed toy experiment demonstrates that the approximate posterior and resulting code indeed seem to reflect likeness between patient data; since the closely situated CT-slices from the same patient had a generally lower divergence than other images. Additionally, inspection of the artificial patient data generated by the decoder, conveys that the samples indeed exhibit anatomical characteristics. This shows promise toward the possibility of generating new artificial patient data using the VAE framework.

The various test performances have conveyed the strength and flexibility of the method, since few epochs of training yield a strong connection between the latent space and specific inputs of data; yielding a superior training versus test performance. Hence, to attain a VAE that generalizes well, large amounts of diverse data, additional regularization methods or augmentation could be beneficial. And especially, one must make sure that the network has seen examples of the anatomical structure in question, in order to perform better during testing.

8.2 Future work

There is a plenitude of aspects of the VAE that could be considered and investigated in the future. Firstly one could consider other types of covariance structures on both likelihood, prior and posterior, as supposed to the spherical assumed throughout this work. Considering computational limitations where inference of a full covariance matrix may be unfeasible, one could e.g. have a covariance between pixels in proximity to each other in both encoder and decoder, such that local structures and patters in the image are potentially enhanced and better preserved. Moreover one could test other types of more flexible distributions, in both encoder decoder and prior. Regarding the posterior this would enable that the approximate inference becomes more precise. Unlike flexibility in other senses in machine learning, the flexibility of the approximate posterior can not yield overfitting but rather improve precision [10]. Moreover, application of a more dynamic prior can have a larger importance than previously emphasized, it " has been shown that a simple prior over-regularizes the latent space leading to poor reconstructions" [4] in work by Hoffman and Johnson [22]. Additionally, one may consider other inference approaches in the VAE framework, than optimizing on approximate distributions as in approximate inference. One could e.g. consider applying *Markov Chain Monte Carlo* (MCMC) methods, i.e. methods that deduce an exact limit distribution by constructing a *Markov Chain* with an *ergodic distribution* $\tilde{p}_x$ that coincides with the true sought distribution p_x. I.e. one constructs a stochastic process where dependence is only inherent between x_n and x_{n-1} and where $\tilde{p}_x(n) \to p_x$ as $n \to \infty$. And this inference procedure could potentially replace e.g. the decoder network.

8.3 Conclusion

Reflection on the presented results leads me to conclude that the VAE may very well be a new addition to the statistical tools applied in automation of health care applications, since it demonstrates great promise in the various aspects considered in this thesis. Future work can possibly strengthen these hopes further.

Bibliography

[1] Lindsey M Appenzoller et al. "Predicting dose-volume histograms for organs-at-risk in IMRT planning". In: *Medical physics* 39.12 (2012), pp. 7446–7461.

[2] Yoshua Bengio et al. "Learning deep architectures for AI". In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[3] Yoshua Bengio, Guillaume Alain, and Salah Rifai. "Implicit density estimation by local moment matching to sample from auto-encoders". In: *arXiv preprint arXiv:1207.0057* (2012).

[4] Erik Bodin et al. "Nonparametric Inference for Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1712.06536* (2017).

[5] Walter R Bosch et al. *Data From Head-Neck Cetuximab.* `http://http://doi.org/10.7937/K9/TCIA.2015.7AKGJUPZ.com`. The Cancer Imaging Archive, 2015.

[6] Y-Lan Boureau et al. "Learning mid-level features for recognition". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE. 2010, pp. 2559–2566.

[7] Hervé Bourlard and Yves Kamp. "Auto-association by multilayer perceptrons and singular value decomposition". In: *Biological cybernetics* 59.4-5 (1988), pp. 291–294.

[8] Freddie Bray et al. "Global cancer transitions according to the Human Development Index (2008–2030): a population-based study". In: *The lancet oncology* 13.8 (2012), pp. 790–801.

[9] Quinn B Carroll. *Radiography in the digital age.* Charles C Thomas, 2011.

[10] M Bishop Christopher. *Pattern Recognition and Machine Learning.* Springer-Verlag New York, 2016.

[11] Chris A Cocosco, Alex P Zijdenbos, and Alan C Evans. "A fully automatic and robust brain MRI tissue classification method". In: *Medical image analysis* 7.4 (2003), pp. 513–527.

[12] Geoff Delaney et al. "The role of radiotherapy in cancer treatment". In: *Cancer* 104.6 (2005), pp. 1129–1137.

[13] Li Deng, Dong Yu, et al. “Deep learning: methods and applications”. In: *Foundations and Trends® in Signal Processing* 7.3–4 (2014), pp. 197–387.

[14] Garoe Dorta et al. “Structured Uncertainty Prediction Networks”. In: *arXiv preprint arXiv:1802.07079* (2018).

[15] Vincent Dumoulin and Francesco Visin. “A guide to convolution arithmetic for deep learning”. In: *arXiv preprint arXiv:1603.07285* (2016).

[16] Andries P. Engelbrecht. *Computational intelligence : an introduction.* John Wiely Sons, Ltd, 2007.

[17] Jacques Ferlay et al. “Cancer incidence and mortality worldwide: sources, methods and major patterns in GLOBOCAN 2012”. In: *International journal of cancer* 136.5 (2015).

[18] Serena Gianfaldoni et al. “An Overview on Radiotherapy: From Its History to Its Current Applications in Dermatology”. In: *Open access Macedonian journal of medical sciences* 5.4 (2017), p. 521.

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

[20] Nathalie Japkowicz Stephen Jos ée Hanson and Mark A Gluck. “Nonlinear autoassociation is not equivalent to pca”. In: ().

[21] Geoffrey E Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).

[22] Matthew D Hoffman and Matthew J Johnson. “Elbo surgery: yet another way to carve up the variational evidence lower bound”. In: *Workshop in Advances in Approximate Bayesian Inference, NIPS.* 2016.

[23] Gareth James et al. *An introduction to statistical learning.* Vol. 112. Springer, 2013.

[24] Johan Ludwig William Valdemar Jensen. “Sur les fonctions convexes et les inégalités entre les valeurs moyennes”. In: *Acta mathematica* 30.1 (1906), pp. 175–193.

[25] Shervin Kamalian, Michael H Lev, and Rajiv Gupta. “Computed tomography imaging and angiography–principles”. In: *Handbook of clinical neurology.* Vol. 135. Elsevier, 2016, pp. 3–20.

[26] Ada Rajneet Kaur. “Feature extraction and principal component analysis for lung cancer detection in CT scan images”. In: *International Journal of Advanced Research in Computer Science and Software Engineering* 3.3 (2013).

[27] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[28] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[29] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[30] Anders Boesen Lindbo Larsen et al. "Autoencoding beyond pixels using a learned similarity metric". In: *arXiv preprint arXiv:1512.09300* (2015).

[31] Maria YY Law and Brent Liu. "DICOM-RT and its utilization in radiation therapy". In: *Radiographics* 29.3 (2009), pp. 655–667.

[32] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[33] Thomas Leung and Jitendra Malik. "Representing and recognizing the visual appearance of materials using three-dimensional textons". In: *International journal of computer vision* 43.1 (2001), pp. 29–44.

[34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[35] David Lowe. *General Pathology Vivas*. Greenwich Medical Media, 2001.

[36] James Manyika et al. "Big data: The next frontier for innovation, competition, and productivity". In: (2011).

[37] Chris McIntosh and Thomas G Purdie. "Voxel-based dose prediction with multi-patient atlas selection for automated radiotherapy treatment planning". In: *Physics in Medicine & Biology* 62.2 (2016), p. 415.

[38] Chris McIntosh et al. "Fully automated treatment planning for head and neck radiotherapy using a voxel-based dose prediction and dose mimicking method". In: *Physics in Medicine & Biology* 62.15 (2017), p. 5926.

[39] Peter Mildenberger, Marco Eichelberg, and Eric Martin. "Introduction to the DICOM standard". In: *European radiology* 12.4 (2002), pp. 920–927.

[40] Kevin P. Murphy. *Machine Learning, A probabilistic Perspective*. The MIT press, 2012.

[41] Benjamin E Nelms et al. "Variation in external beam treatment plan quality: an inter-institutional study of planners and planning systems". In: *Practical radiation oncology* 2.4 (2012), pp. 296–305.

[42] Dan Nguyen et al. "Dose Prediction with U-net: A Feasibility Study for Predicting Dose Distributions from Contours using Deep Learning on Prostate IMRT Patients". In: *arXiv preprint arXiv:1709.09233* (2017).

[43] Hanne Melgaard Nielsen et al. "Audit of the radiotherapy in the DBCG 82 b&c trials - A validation study of the 1538 patients randomised to postmastectomy radiotherapy". In: *Radiotherapy and oncology* 76.3 (2005), pp. 285–292.

[44] Michael A. Nielsen. *Neural Networks and Deep Learning*. `http://neuralnetworksanddeeplearning.com/`. Determination Press, 2015.

[45] Simon PG Padley and David M Hansell. "Imaging techniques". In: *Clinical Respiratory Medicine (Fourth Edition)*. Elsevier, 2012, pp. 63–121.

[46] Nikhil R Pal and Sankar K Pal. "A review on image segmentation techniques". In: *Pattern recognition* 26.9 (1993), pp. 1277–1294.

[47] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. "The matrix cookbook". In: *Technical University of Denmark* 7.15 (2008), p. 510.

[48] Kilian M Pohl et al. "Anatomical guided segmentation with non-stationary tissue class distributions in an expectation-maximization framework". In: *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on.* IEEE. 2004, pp. 81–84.

[49] Pantelimon G Popescu et al. "Bounds for Kullback-Leibler divergence". In: *Electronic Journal of Differential Equations* 2016 (2016).

[50] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[51] WK Röntgen. "Über eine neue Art von Strahlen: vorläufige Mitteilung". In: *Sitzungsber. Phys. Med. Gesell.* (1895).

[52] Wenzhe Shi et al. "Is the deconvolution layer the same as a convolutional layer?" In: *arXiv preprint arXiv:1609.07009* (2016).

[53] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[54] BWKP Stewart, Christopher P Wild, et al. "World cancer report 2014". In: *Health* (2017).

[55] K Thulasiraman and MNS Swamy. "5.7 acyclic directed graphs". In: *Graphs: Theory and Algorithms* 118 (1992).

[56] Vladimir Naumovich Vapnik. "An overview of statistical learning theory". In: *IEEE transactions on neural networks* 10.5 (1999), pp. 988–999.

[57] M Veness and S Richards. "Radiotherapy". In: *Dermatology* 2 (2012), pp. 2291–2301.

[58] Binbin Wu et al. "Using overlap volume histogram and IMRT plan data to guide and automate VMAT planning: a head-and-neck case study". In: *Medical physics* 40.2 (2013).

[59] Haibing Wu and Xiaodong Gu. "Towards dropout training for convolutional neural networks". In: *Neural Networks* 71 (2015), pp. 1–10.

[60] Matthew D Zeiler et al. "Deconvolutional networks". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE. 2010, pp. 2528–2535.

[61] Yefeng Zheng et al. "Deep Learning Based Automatic Segmentation of Pathological Kidney in CT: Local Versus Global Image Context". In: *Deep Learning and Convolutional Neural Networks for Medical Image Computing.* Springer, 2017, pp. 241–255.

[62] Xueyuan Zhou and Mikhail Belkin. "Semi-supervised learning". In: *Academic Press Library in Signal Processing.* Vol. 1. Elsevier, 2014, pp. 1239–1269.

TRITA -SCI-GRU 2018:232