Using Deep Learning to Remove Computed Tomography Artifacts due to Hip Replacement

Adelina Zahirovic

Master's thesis
2020:E23

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Using Deep Learning to Remove Computed Tomography Artifacts due to Hip Replacement

Adelina Zahirovic

June 2019

Master's Thesis
Faculty of Engineering
Centre of Mathematical Sciences
Mathematics

Supervisor at Lund University: Professor Kalle Åström
Supervisor at EXINI Diagnostics AB: Konrad Gjertsson
Examiner: Mikael Nilsson

Abstract

Computed Tomography (CT) is the most common diagnostic method for cancer and prostate cancer is the most common cancer among men in Sweden. Some of the patients that get scanned have hip replacements that cause artifacts in CT scans that make the CT images unreadable for physicians. In this Master's Thesis an autoencoder model was implemented to reduce artifacts due to hip replacements in CT images. The model was trained, validated and tested on CT images provided by EXINI Diagnostics AB, Lund, Sweden.

The autoencoder was implemented using the deep learning framework Keras in Python. Autoencoders have been used to reduce different types of noise in other experiments and shown great results. The produced results show that the model is a good start for further work to completely reduce artifacts due to hip replacement.

Sammanfattning

Computed Tomography (CT) är den vanligaste metoden för undersökning av cancer och prostata cancer är den vanligaste cancerformen bland män i Sverige. En del patienter som genomför en CT undersökning har höftimplantat som orsakar artefakter och bidrar till att CT bilderna blir oläsliga för läkarna. I detta examensarbete implementerades det en autoencoder modell för att reducera artefakterna i CT bilder. Modellen tränades, validerades och testades på CT bilder framtagna av EXINI Diagnostics AB i Lund, Sverige.

Autoencodern var implementerad med hjälp av Deep Learning bibliotek i Keras i Python. I andra experiment har autoencoders visat sig vara en bra metod att reducera olika sorters brus i bilder. Även i detta fall visar den goda resultat samt möjligheter för vidare arbete för att reducera artefakterna helt.

Acknowledgements

I would like to thank EXINI Diagnostics AB and Progenics Pharmaceuticals Inc. for giving me this opportunity to do my Master's Thesis at EXINI in Lund and allowing me to use their resources when training, validating and testing my model constructed in this project. I would also thank my supervisor at Lund University Professor Kalle Åström and my supervisor at EXINI Konrad Gjertsson for all the guidance and advice they gave me during the project. I am very thankful that I was introduced to the field of machine learning by Professor Kalle Åström and Kerstin Johnsson, who also introduced EXINI.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

Every year about 10.000 men in Sweden get prostate cancer and about 2.500 men die of it each year. This makes prostate cancer the most common cancer among men in Sweden [1]. The disease usually affects men that are over 50 years old and is exceptionally rare among men under 40 years old.

The prostate is a small gland in the pelvic, located near the bladder. Prostate cancer means that the prostate gland is afflicted with a malignant tumor that is usually located on the outer part of the prostate gland. It can often take a long time before a patient get any symptoms of having been affected by prostate cancer. When a tumor grows larger, problems can arise and symptoms can occur such as difficulties during urination, weight loss and fatigue. However, some patients do not have any symptoms at all until the cancer spreads outside the prostate gland and on to the skeleton. [2]

Computed Tomography (CT scan) is a common test method for different types of cancer. The term tomography comes from the Greek words tomos and graphein which means a slice to write or record. A CT scan test is similar to a standard X-Ray. The main differences are that CT scans:

- Shows bones, organs and soft tissue in the images more clearly than X-Ray images.
- Can show a tumor and the shape, size, and location of it in the body.
- CT image slices are placed like layers on top of each other and makes a 3D-image in black and white. Most of the modern CT scan machines take continuous pictures in a spiral form rather than taking slice by slice. The described procedure results in lower acquisition time and higher quality images [3].

Due to the fact that CT images show bones, organs and soft tissue this makes it possible to look at the organs without any surgery. Physician can directly view CT images to detect cancer in the body, see the size of the tumors and locate them [4].

Artifacts are common in clinical CT images and may cause problems when physician evaluating the scans. There are several different types of artifacts, such as:

- Noise artifacts
- Motion artifacts
- Metal artifacts

Noise artifacts can cause different patterns in CT images. They are often caused by the statistical error of low photon counts and can result in random thin bright and dark streaks or random darker and brighter pixels similar to Gaussian noise. Motion on the other hand, often causes blurry images, double images and similar to noise artifacts, long streaks. If the patient is lying in one of the faster machines, motion artifacts are reduced automatically because the patient has less time to move during the acquisition. Metal artifacts are extremely common and are caused because of its high atomic number. Common metals with high atomic number used to make implants are iron and platinum. [5]

These metal artifacts are making some big problems for the physicians due to the fact that the CT images are worse when metal is scanned. Some of the patients who get prostate cancer also have some type of hip replacement where either one hip is replaced or both. The hip replacements cause artifacts in the scans and making it difficult to see the organs and soft tissue in the pelvic area. Most of the CT images are unreadable and are not giving any information at all in the pelvic region in the body.

1.1 Aim of the Master's Thesis

The primary goal of the Master's Thesis is to implement an algorithm that reduces or removes artifacts caused by hip replacements in a Computed Tomography (CT) image. The approach has been to implement an AutoEncoder which is a type of an Artificial Neural Network. All the CT images were provided by EXINI Diagnostics AB.

The work was divided into two steps:

- The first step was the implementation of the artificial neural network, and use it as a pure autoencoder used for reconstruction of CT images. Successful image reconstruction validates the efficacy of the designed model and shows its ability to capture the distribution of the data. In addition, reconstruction lacks requirements of annotated data, which in practice meant the work could commence before annotations were available.
- The final step was to train the model on training data that included artifacts in the CT images and see if the model could remove the artifacts in test images. Because of the small amount CT images, including artifacts, more images with artifacts were needed to be generated before training the model.

1.2 Literature Survey

A lot of different studies have tackled the problem of artifact reduction in CT images. Studies [6, 7] were reducing metal artifacts but with other methods than deep learning and autoencoders and they were not that useful in this Master's Thesis. There were some studies however, where deep learning algorithms were utilized, not to reduce and remove metal artifacts, but to reduce and remove noise in CT images and other kinds of images [8–13]. Their approaches to reduce noise were the first inspiration to how to build an autoencoder network that will learn how to, at least, reduce and take away some noise in CT images and other kinds of images. The only problem with these studies was that they were applying their models on color images and most of them only on 2D-images. It was difficult to find studies done on 3D-images were they use autoencoder or other deep learning methods to reduce noise. But there was one [14] that reduced noise in 3D CT images. Their 3D network architecture was the inspiration to this Master's Thesis and the model used in this work were all based on their proposed architecture.

Chapter 2

Machine Learning

In traditional computer vision algorithms, the developers write the rules that they want the program to follow. Each rule is based on a logical foundation and the system grows, additional rules have to be incorporated. The program can quickly become unsustainable to maintain and is difficult to write them as well as a data-driven model. This is where data-driven machine learning models come into the picture.

Often when somebody talks about Artificial Intelligence (AI) most of the times they are talking about machine learning. The word explains it well; it is about machines that learn from data. The machine learns how the input and output data are correlated to each other and writes its rules by itself. There are no developers needed to do this like in rule-based algorithms. The machine learns what it needs to learn in a similar way like humans. Humans learn from experience and situations that happens. When experiencing different situations we learn more and we grow more adept in understanding our environment and extrapolate from the current. We have a bigger chance to succeed with something we have been experiencing than when we get into a new unknown situation. The same principles govern the optimization procedures of data-driven machine learning models. The machine makes a model that predicts the new, unknown inputs as well as it can [15].

Machine learning is commonly divided into a number of sub-fields, where each field is characterized by the problem you want to solve depending on what kind of data you have access to. The division is as follows:

- Supervised Learning
- Unsupervised Learning
- Semi-Supervised Learning
- Reinforcement Learning

where a brief description is offered of each sub-field in subsequent sections.

2.1 Supervised Learning

In supervised learning, the algorithm learns how to associate some input with an associated output, where both input data and its annotation is provided to the model during the training phase. Usually, the annotated labels are difficult to collect automatically, so a human has to provide them, i.e. the supervisor. This is also the reason it is called supervised learning.

If the algorithm has an input variable X and an output variable y the algorithm will learn the mapping function

$$y = f(X)$$

We already know the correct outputs, so the algorithm makes predictions on the training data and is corrected by the supervisor. The main goal is to make this approximation of the mapping function so that when a new input X is given its output y can be predicted. [16]

Supervised learning can be divided into two different domains:

– Classification

– Regression

depending on what supervised machine learning problem has to be solved [17]. Both of these techniques are similar in nature. The only thing that separates them is the format of the output. In sections 2.1.1 and 2.1.2 classification and regression are explained and examples are given for each problem.

2.1.1 Classification

In classification, the output can be seen as a class label [17]. When working with classification problems it is very important to have examples of objects we want to classify (true examples) but it is also important to have examples of objects that do not belong to that class (false examples). A famous example of the classification problem is the MNIST dataset of handwritten digits. The goal with the classification problem is to construct a model that can classify which number the handwritten digit symbolizes. The MNIST dataset consists of 10 classes, the numbers $0-9$, and the images are black and white. The MNIST dataset consists of, 60.000 training images and 10.000 test images taken from American Census Bureau employees and American high school students [18].

As seen in Figure 2.1 far from all numbers are easy to identify, which makes this a perfect task for machine learning algorithms.

Another example of a classification problem could be to determine if a gene is healthy or has a disease like shown in Figure 2.2, where the points in the graph are used to train a logistic regression model.

Figure 2.1: Sample of the handwritten digits in the MNIST dataset [19].

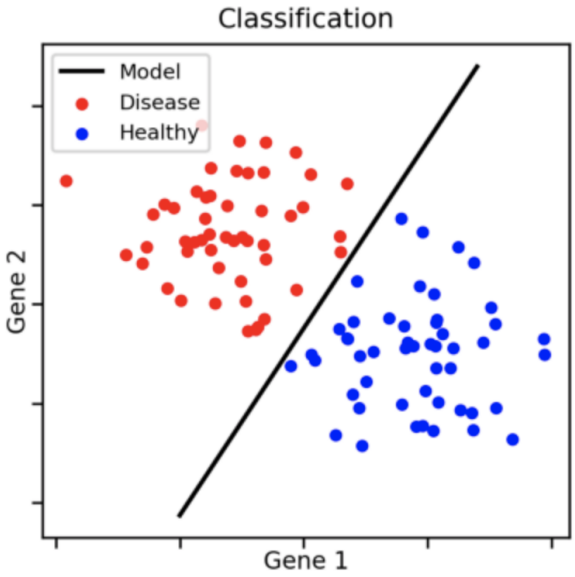


Figure 2.2: An example of a classification problem. The task is to classify if Gene 1 and Gene 2 are healthy or have a disease. The training data is used to learn and fit the model, decision boundary, that separates the two classes. [20]

2.1.2 Regression

In regression, the output consists of real values [17]. Unlike in the classification, in regression the output value is tried to be approximated using machine learning algorithms. The output value is generated by a function which could be anything that outputs continuous values, for example, determine the life expectancy of cancer patients, as shown in Figure 2.3.

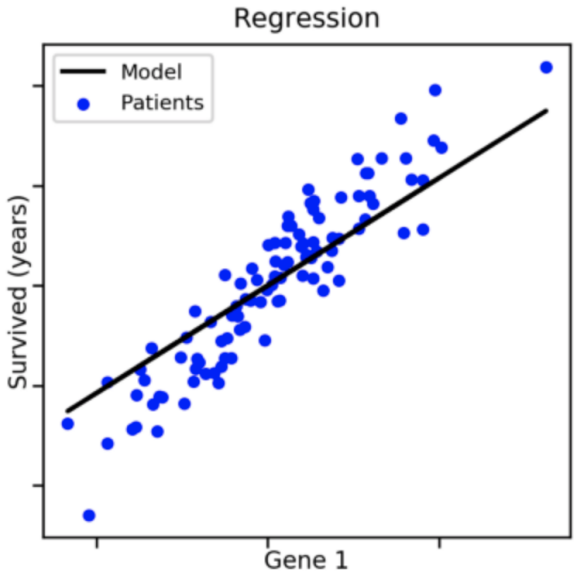


Figure 2.3: An example of a regression problem where the task is to predict how many years a cancer patient will survive [20].

2.2 Unsupervised Learning

Contrary to models trained in a supervised fashion, unsupervised training does not include annotations. This is the reason for the name, unsupervised learning - no supervisor can give feedback to the model and show what behavior is desired for each data point in X. The algorithm has no clear target of the learning to work against. This is the type of learning which has been used in this project and has resulted in this report.

Unsupervised learning can be divided into three different problems:

- Clustering
- Association
- Dimensional reduction

depending on which kind of problem is wanted to be solved [16]. In the following sections clustering, association and dimensional reduction are explained and some examples illustrated.

2.2.1 Clustering

In the clustering problem, similarities and differences are wanted to be found in the data (e.g. images) and assign them in clusters [21]. By clustering images you can, for example, find images with similar cars and identify different types of news [17].

One of the most popular examples of clustering in unsupervised learning is *K-means* clustering [16]. In K-means clustering you define a target number K, which represents how many clusters you want to divide the dataset into. The algorithm, shown in Equation 2.1 [22], will generate K numbers of centroids that represent the center of a cluster, like in Figure 2.4, with N different cases. The goal of the algorithm is to minimize the difference within the clusters and maximize the difference between the clusters [23]. The algorithm starts with randomly select centroids, c, to build the first group of datasets, which corresponds to the starting points for every cluster. After this, the algorithm performs repetitive calculations to optimize the positions of the centroids to minimize the differences within the clusters.

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N} \|x_i^{(j)} - c_j\|^2 \tag{2.1}$$

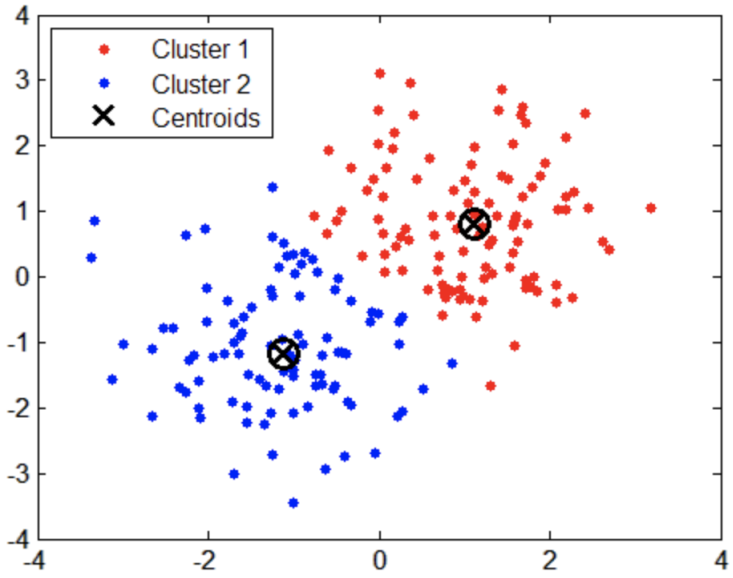


Figure 2.4: Illustration of two clusters, $K = 2$, identified from the source dataset [24].

2.2.2 Association

The association rule problem is applied to problems where we want to find some interesting relationships between the variables in a dataset [21]. An example of an association derived from data examination is that people who buy a new house often tend to also purchase new furniture.

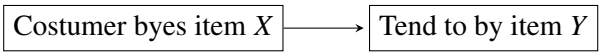


Figure 2.5: Example of an association rule.

2.2.3 Dimensional reduction

Dimensional reduction is a technique used to reduce number of features in datasets without any negative effects on machine learning problems [25]. Dimensional reduction can both be linear or non-linear depending on which method is used. The most common methods used in dimensional reduction are:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

By using dimensional reduction the computation time reduces, it helps the compression of input data and reducing the storage space required [26].

A popular method used to reduce dimentional is an *AutoEncoder*. An autoencoder is a type of artificial neural network that aims to copy its input to their output [27]. The autoencoder is more explained later on in section 3.4.

2.3 Semi-Supervised Learning

Semi-Supervised learning is a combination of supervised learning and unsupervised learning that are described in section 2.1 and 2.2 respectively. In semi-supervised learning, we have as usual input data x but we also have some labeled output data Y. As mentioned in supervised learning all output data are labeled and in unsupervised learning, no output is labeled, and in semi-supervised learning, some of the output data are labeled and some do not have any labels.

Most of the real-world problems fall into this area. It takes time and money to get labeled data and most of the time you do not get a lot of this type of data. While unlabeled data, are easy to collect and cheap [16].

2.4 Reinforcement Learning

Reinforcement learning is a goal-oriented algorithm of how to achieve a complex goal. It is a bit different in how it works compared to supervised and unsupervised learning. In reinforcement learning the algorithm learns how to achieve a specific goal by discovering an environment by itself.

Reinforcement learning algorithms make decisions sequentially, where decisions are dependent on each other. To simplify the model, it can be explained that the output depends on the state of the current input, and the next input depends on the output of the previous input. Depending on which decision is made, the model gets rewards. This means that the model acts in a certain way in the environment depending on which type of rewarding it got from the previous action. The goal of the model is to get a lot of positive rewards and as little negative rewards as possible. This is how the model learns the environment all by itself. It is exploring the different decisions and what type of reward it gives [28]. This is a common model used when a computer plays e.g. chess [29].

Chapter 3

Deep Learning

Deep learning is the main key technology of many items today, like driverless cars, recognition of items like images, sound and voice control of phones, TVs, hands-free speakers and so on. Thanks to deep learning today we can achieve results that we could not do before.

Deep learning is a special form of machine learning technique that learns the computer things that are natural for humans. Deep learning can be used to for example, identify objects in an image, translate language and describing images with text. It was in the 1980s deep learning was theorized [30], but it was not until recently it became useful. The main reasons why it was not useful before are that deep learning requires a large amount of labeled data and substantial computing power. Now-days we have access to a much larger amount of data and much better computers.

Deep learning methods use neural network architectures for modeling and learns features directly from the data without the need for manual feature extraction. Before machine learning engineers had to go through all data to find features by hand, which was fed into to complete the task, e.g. some form of classification. This was and still is a time consuming and expensive process which often requires highly specific knowledge. But this is not the only problem, it is very difficult to select which features actually contribute to the model performance. The methods in deep learning, strives to perform a so-called end-to-end learning which means that they automatically learn how to perform a given task based on available data [31]. Deep learning has been introduced with the goal to move machine learning closer to one of its original goals - Artificial Intelligence (AI) [32].

The word *deep* is a technical term and refers to the number of layers in a network. A deep network, usually has more than one hidden layer, because it is the hidden layers that allow the deep neural network to learn a representation of the data. Neural networks with many hidden layers allow the features to pass through more mathematical operations than in networks with few layers. This is why these networks with a lot of hidden layers are more intensive to train and need a lot of computing power. [33]

3.1 Artificial Neural Network

Artificial neural networks (ANNs) were introduced in 1943 [34] and represents a type of machine learning model. Initially, the primary objective was to model the biological neural system in our brain. Due to its complexity, the research split into two parts. One part is actually focusing on trying to understand the biological process of what happens in our brain, and the second part focuses on applications of the ANNs.

It was not until 1974 that artificial neural networks with more than a single hidden layer were trained successfully and not until 1986 the method got famous [31] when the backpropagation algorithm was introduced. The backpropagation algorithm is covered in further detail in section 3.2.

Initially, a single artificial neuron, called a perceptron, was used to model relationships in the data until the scientists at the time realized that the human brain has billions of neurons to do the processing when solving a problem. Instead of the single perceptron, multiple artificial neurons were combined to create multilayer perceptrons, i.e. artificial neural networks, in an attempt to more closely replicate the structure and design of the human brain. Our biological computer is made up of about 86 billion neurons [35] and all these neurons are connected to each other through about 10^{15} synaptic connections [36] in complex structures. We have different kinds of neurons in our brain and all of them have their different properties, but the basic structure is the same in all of the neurons. In Figure 3.1 the basic biological neuron is represented by a mathematical model, i.e. an artificial neuron.

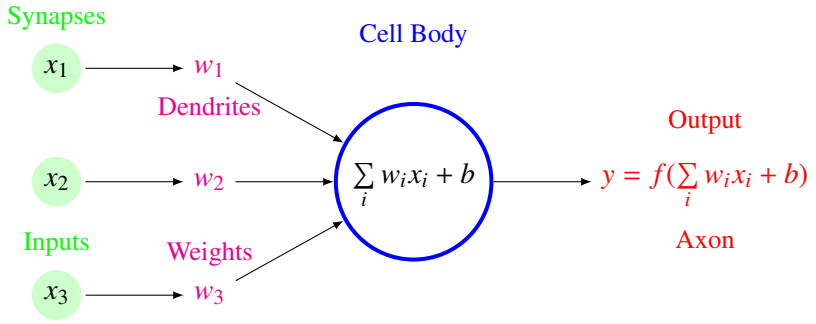


Figure 3.1: A mathematical representation of a biological neuron - An artificial neuron.

The biological neuron has three main parts, a cell body that has a nucleus, small branches called dendrites and a bigger branch called an axon. All signals from other neurons are passed into the cell body through receptors on the dendrites called synapses. The synapses in the mathematical model in Figure 3.1 are represented as the input data x_1, x_2 and x_3. Before the signals come to the cell body, they are either getting stronger or weaker in the dendrites, which means that the input data is being weighted before coming to the artificial neuron. In machine learning, the weighting process enhances specific signals and ignores others [37]. If the signal

in the biological neuron is strong enough, it will be transformed through a complex chemical process and be sent through the axon to the next neuron. But if it is weak and not the right pattern in the signal nothing will happen. Because of this, the bias term b is added to the sum in the artificial neuron before it is processed using an activation function, as close as possible to the biological process.

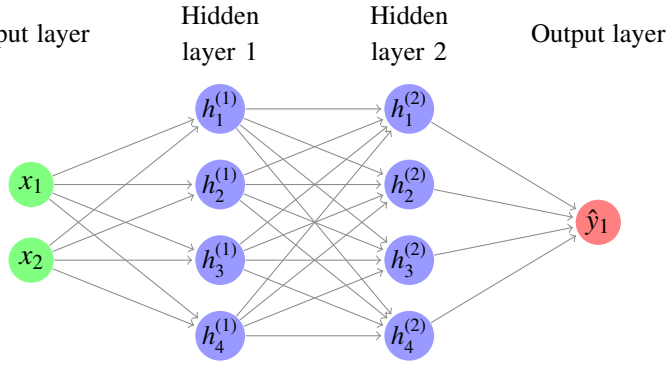


Figure 3.2: An concept graph of a fully connected ANN. The variable in each node represents the output of the node.

As mentioned the structure in the human brain is very complex and it is impossible to model such complex structures. But with careful observations of the biological axons and their communications, the researchers have found more organized versions of the biological neural networks to create artificial neural networks that can represent this. A simple concept graph of a fully connected artificial neural network is visualized in Figure 3.2.

3.2 Backpropagation

As previously mentioned, the backpropagation algorithm was introduced in 1970s, but it was not until 1986 its importance was fully discovered when the paper by David Rumelhart, Geoffrey Hinton and Ronald Williams got famous. In the paper, they talked about how backpropagation works much faster than earlier approaches to learning and this made it possible to solve problems, which had been insoluble, by using neural nets. [31]

It is thanks to the backpropagation algorithm the artificial neural network is the powerful machine learning model it is today. The backpropagation algorithm calculates the gradients by using the chain rule and propagates the gradients of the loss function back through the network without any overlapping calculations needed. The chain rule is a formula that tells us how to find the derivative of the composition of two or more functions. A function is composite if it can be written as a function of a function, $F(x) = f(g(x))$. The chain rule shows then that the composition function $F(x)$ can be calculated as:

$$F'(x) = f'(g(x)) \cdot g'(x)$$

The chain rule can also be expressed as a product of derivatives if we assume that $g(x) = y$ and $F(x) = z$, then it can be expressed as

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

To make the backpropagation algorithm easier to understand, it will be demonstrated by an example. The example will be constructed as in [38] shown in Figure 3.3.

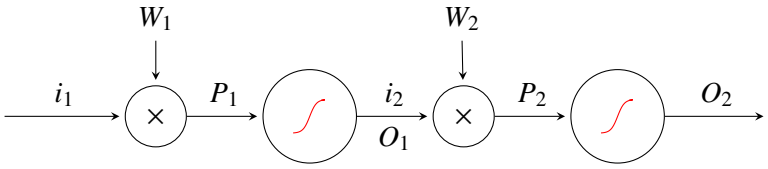


Figure 3.3: An artificial neural network is defined as two or more perceptrons that work together to approximate a function, where the simplest only consists of two perceptrons [39]. This figure shows an example of the simplest kind. The input i_1 is multiplied with the weight W_1 which output is P_1. The output P_1 is then fed through a sigmodial activation function. This makes now the first neuron in the network. The output from the activation function is then the input to the next artifical neuron where the same steps happens. The last output of the multilayer perceptron O_2 is compared to the label d in error function E.

Before starting the calculation of the gradients the error function E and the activation function S has to be defined. To make it simple, we define the error function E as the mean squared error:

$$E = \frac{(d - z)^2}{2}$$

where d is the true label and z is the output of the network. And the activation function S will be defined as the sigmoid function

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

where x is the input value to the neuron.

Now when we have defined the error function E and the activation function S we can start to find the gradient of the error with respect to the weights W_1 and W_2. The gradients are found by using the chain rule backward through the network. This means that we want to start with finding:

$$\frac{\partial E}{\partial W_2}$$

and then

$$\frac{\partial E}{\partial W_1}$$

But first we have to find

$$\frac{\partial E}{\partial O_2}$$

since O_2 is the only variable that has a direct dependence on our error function E. To find it we start to express it by the chain rule:

$$\frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial O_2} \times \frac{\partial O_2}{\partial W_2} = \frac{\partial E}{\partial O_2} \times \frac{\partial O_2}{\partial P_2} \times \frac{\partial P_2}{\partial W_2} \tag{3.1}$$

where

$$\frac{\partial E}{\partial O_2} = \frac{\partial}{\partial z}(\frac{(d-z)^2}{2}) = z - d, \quad \frac{\partial P_2}{\partial W_2} = \frac{\partial (i_2 W_2)}{\partial W_2} = i_2$$

and the last part $\frac{\partial O_2}{\partial P_2}$ have a dependence on x in the activation function S. The derivation of the activation function is calculated:

$$\begin{aligned} \frac{\partial S}{\partial x} = \frac{\partial}{\partial x}(\frac{1}{1+e^{-x}}) = \frac{\partial}{\partial x}(1+e^{-x})^{-1} = (-(1+e^{-x})^{-2})(-e^{-x}) = \\ (\frac{1}{1+e^{-x}})(\frac{e^{-x}}{1+e^{-x}}) = S(x)(1-S(x)) = z(1-z) \end{aligned} \tag{3.2}$$

where z is the output from the activation function. Equation 3.2 gives us the last part we need to find the gradient of the error for the weight of W_2.

$$\frac{\partial E}{\partial W_2} = (z-d)z(1-z)i_2 \qquad \Leftrightarrow \qquad \frac{\partial E}{\partial W_2} = (O_2 - d)O_2(1-O_2)i_2 \tag{3.3}$$

Now we can find the gradient of the error respect to the weight W_1:

$$\begin{aligned} \frac{\partial E}{\partial W_1} &= \frac{\partial E}{\partial O_2} \times \frac{\partial O_2}{\partial W_1} \\ &= \frac{\partial E}{\partial O_2} \times \frac{\partial O_2}{\partial P_2} \times \frac{\partial P_2}{\partial W_1} \\ &= \frac{\partial E}{\partial O_2} \times \frac{\partial O_2}{\partial P_2} \times \frac{\partial P_2}{\partial O_1} \times \frac{\partial O_1}{\partial W_1} \\ &= \frac{\partial E}{\partial O_2} \times \frac{\partial O_2}{\partial P_2} \times \frac{\partial P_2}{\partial O_1} \times \frac{\partial O_1}{\partial P_1} \times \frac{\partial O_1}{\partial W_1} \end{aligned} \tag{3.4}$$

where we already know:

$$\frac{\partial E}{\partial O_2} = O_2 - d, \qquad \frac{\partial O_2}{\partial P_2} = O_2(1-O_2)$$

The rest of the partial derivatives are:

$$\frac{\partial P_2}{\partial O_1} = W_2, \qquad \frac{\partial O_1}{\partial P_1} = O_1(1-O_1), \qquad \frac{\partial O_1}{\partial W_1} = i_1$$

This gives the gradient of the error respect to the weight W_1:

$$\frac{\partial E}{\partial W_1} = (O_2 - d)O_2(1-O_2)W_2 O_1(1-O_1)i_1 \tag{3.5}$$

If we simplify Equation 3.5 and say that the part we got from Equation 3.3 is $\delta_2 = (O_2 - d)O_2(1 - O_2)$ and $\delta_1 = W_2O_1(1 - O_1)\delta_2$ gives us simple expressions for our gradients as functions of the input to the neuron:

$$\frac{\partial E}{\partial W_2} = \delta_2 \times i_2$$
$$\frac{\partial E}{\partial W_1} = \delta_1 \times i_1$$

With these gradient functions the weights are updated according to:

$$W_1^+ = W_1 - \eta\frac{\partial E}{\partial W_1}$$
$$W_2^+ = W_2 - \eta\frac{\partial E}{\partial W_2}$$

If more neurons add on to the left the computation just repeats by computing the delta for the new layer and the input to that layer. The final formulas will always only consist of known values and the values that have been calculated in the previous steps of the backpropagation algorithm. [38]

3.3 Convolutional Neural Network

A convolutional neural network (ConvNet or CNN) is a deep learning algorithm which can take an image as an input, define the importance in the image and be able to differentiate the image from other images [40]. CNN takes biological inspiration from the visual cortex. The visual cortex has small regions of cells that are sensitive to specific regions of the field. It was explained by D.H Hubel and T.N Wiesel in 1962 with an experiment where they showed that some neuronal cells in the brain responded only the vertical edges, some others on horizontal edges and so on. It was only specific neuronal cells that responded on a specific orientation of an edge. D.H Hubel and T.N Wiesel also found out that all of these neurons were organized in a specific way and that they together produced visual perception. This is also the basis of convolutional neural networks, and all the components in the model have a specific task to complete in the machine. [41]

When we teach children to recognize what is what and how something looks like we need to show them a lot of pictures and that is the same thing we have to do to teach a CNN algorithm to recognize something. The difference in how we see pictures and how a computer sees it, is that the computer only sees numbers. If you have a 2D-image the computer will see the image like a 2D-array consisting of numbers, shown in Figure 3.4. Each number corresponds to a pixel. The fact that a machine sees images different from us humans does not mean that the machine is not able to be trained to recognize things. We only have to think that the image looks slightly different for computers.

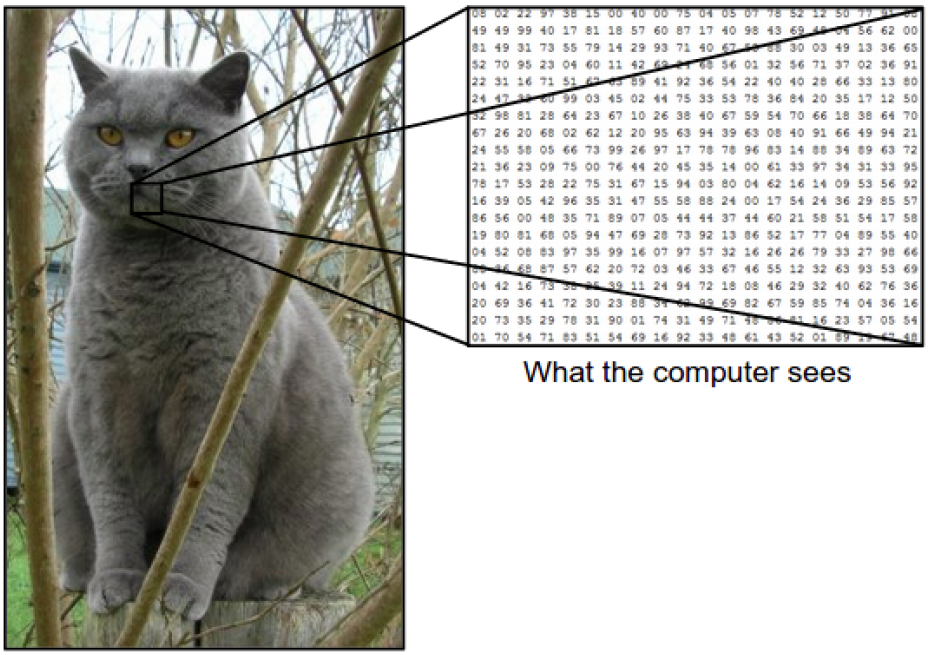


Figure 3.4: A computer sees an image like numbers where each number corresponds to a pixel in the image [42].

Convolutional neural networks have a bit different architecture compared to a regular feed forward neural network. A feed forward artificial neural network looks like Figure 3.2 and transforms an input by putting it through hidden layers. Each layer consists of a set of neurons where each layer is connected to all neurons in the previous layer. In the end, there is an output layer that represents a prediction. CNN is a bit different. The main differences are:

1. The layers in the CNN are organized in dimensions. How many dimensions the CNN consists of depends on the components that make up a layer, which maintain the spatial dimensions of the image. For example, in Figure 3.5 the neurons are organized in three dimensions - height, width, and depth.

2. The neurons in one layer do not connect to all of the neurons in the next layer, but only a small part of it.

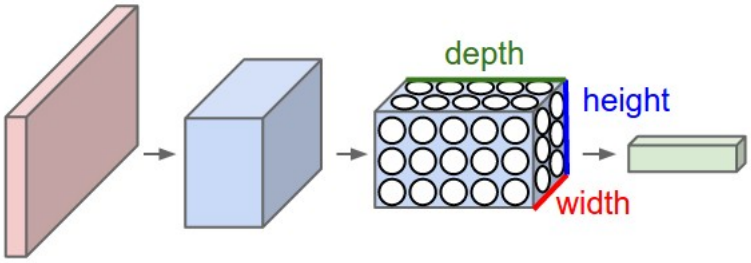


Figure 3.5: An example of a CNN organized in three dimensions — height, width and depth — as seen in one of the layers [43].

In convolutional neural network there are three main components:

– Convolutional Layer

– Pooling Layer

– Activation function

where a brief description is offered of each main component in subsequent sections.

3.3.1 Convolutional Layer

The convolutional layer is the main building block in the convolutional neural network. It is the convolutional layer that carries the main portion of the computational load. The layer performs a dot product between two matrices which are:

1. The set of learnable parameters, known as a kernel

2. The restricted portion of the receptive field.

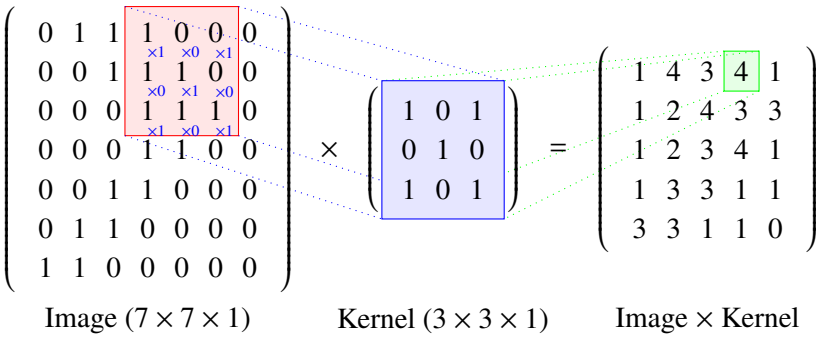


Figure 3.6: Convoluting a $7 \times 7 \times 1$ image with a $3 \times 3 \times 1$ kernel and getting a $5 \times 5 \times 1$ convolved feature.

As seen in Figure 3.6 the kernel matrix is smaller than the image. During the forward pass, the kernel slides to the right with a certain sliding size called stride till

it passes the whole width. The kernel matrix is producing the image representation of that receptive region. Each spatial position of the image gives a response at the kernel and produces the activation map [44]. When the width is passed the kernel matrix moves down and starts the same procedure again until the whole image is traversed. The movement of the kernel matrix across an image is visualized in Figure 3.7. The whole process is called convolution.

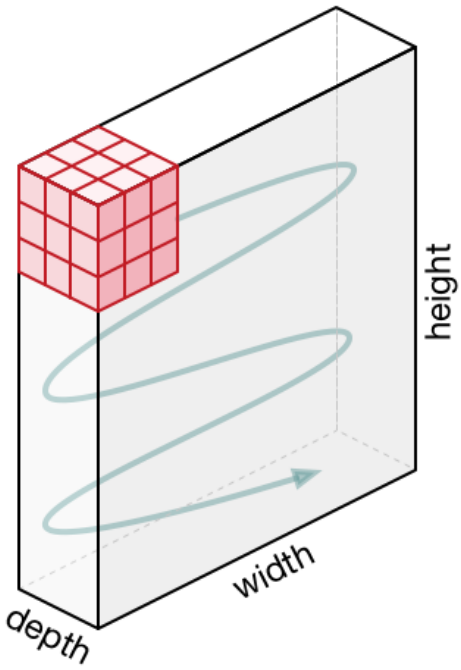


Figure 3.7: The movement of the kernel matrix across an image [40].

The output volume of the image can be calculated by Equation 3.6, where W is the height or width depending on which one you want to calculate W_{out} for, F is the spatial size of the kernel with stride S and P is the amount of padding.

$$W_{out} = \frac{W - F + 2P}{S} + 1 \tag{3.6}$$

If the input image has size $W \times W \times D$ then the output volume will be of size $W_{out} \times W_{out} \times D_{out}$ [44], where D_{out} is the number of filters.

3.3.2 Pooling Layer

Usually, when constructing a convolutional neural network pooling layers are inserted between the convolutional layers. Pooling layers reduces the spatial size of the data which decreases the computational cost. The two most common pooling types are:

1. Max Pooling: Return the maximum value of the region covered by the kernel.

2. Average Pooling: Return the average value of the region covered by the kernel.

No matter which type of pooling is used, they have the same goal to achieve. Just like in convolutional layers, the kernel is smaller than the image. The kernel matrix will move from the left top, across the whole width, moving down a step and start over till the whole image is pooled. Depending on which type of pooling layer is used, different values are given. In Figure 3.8 we see an image that is pooled with Max Pooling and Average Pooling with stride 2. This means that the image is reduced by a factor 4 for a 2D-image.

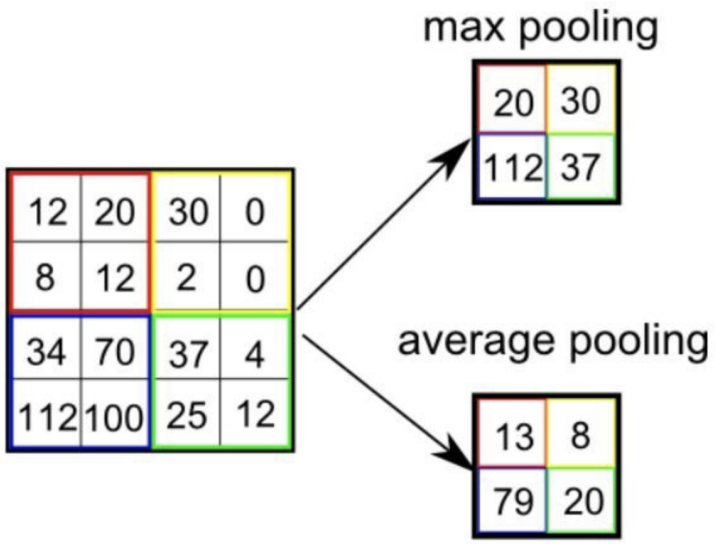


Figure 3.8: Two of the most common pooling types are Max Pooling and Average Pooling. The input data is pooled with a kernel matrix (2×2) consisting of only ones with stride 2. Max Pooling is returning the maximum values of the region covered by the kernel, and Average Pooling is returning the average values. [40]

Usually, the Max Pooling layer is used as a layer between the convolutional layers. The Equation 3.7 is giving the output volume you get after using max pooling on an image, where W are width or height depending on which one is calculated, F is the spatial size of the kernel and S is the stride. [44]

$$W_{out} = \frac{W - F}{S} + 1 \tag{3.7}$$

3.3.3 Activation function

In a convolutional neural network, the activation function is responsible for helping to decide if the neuron will fire or not [46]. There are a lot of different types of activation functions and some of them are shown in Figure 3.9.

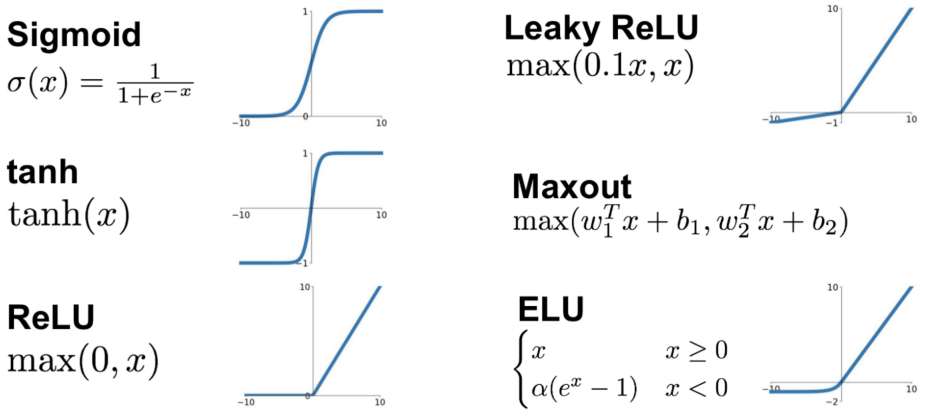


Figure 3.9: Different types of activation functions [46].

The Rectified Linear Unit (ReLU) is the most common one in deep learning models, and also used in this project. The function returns 0 if it receives a negative value and for any positive value x it returns the same value. So its function can be written as:

$$f(x) = max(0, x)$$

and its graph looks like Figure 3.10. This means that when the ReLU function gets an output that is negative the neuron does not get activated. This makes it more computational effective as few neurons are activated each time. It does not saturate the positive region. In practice ReLU converges about 6 times faster than sigmoid activation function [46]. Its fast convergence and that its linear means that the slope does not have a plateau when x gets larger. Thanks to this the ReLU activation function does not have a vanishing gradient problem like the activation functions sigmoid or tanh have.

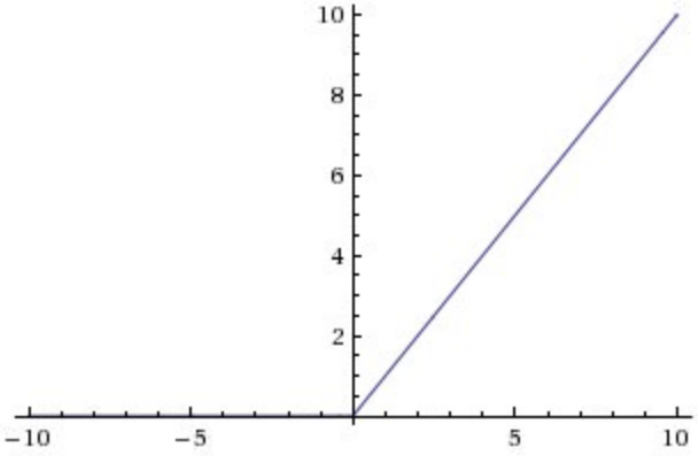


Figure 3.10: ReLU activation function. The function returns 0 if it receives a negative value and for any positive value x it returns the same value back [47].

3.4 AutoEncoder

An autoencoder is an unsupervised deep learning model that does not use any labeled data [50]. An autoencoder is trained to attempt to copy its input to its output. An autoeconder consists of two parts, as shown in Figure 3.11:

1. An encoder function $h = f(x)$
2. A decoder function that produces the reconstruction $r = g(h)$

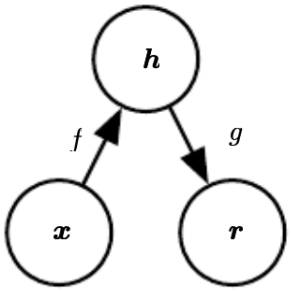


Figure 3.11: A general structure of an autoencoder. Mapping an input x to an output r through an internal representation h. The autoencoder has two components: The encoder f that maps x to h and the decoder g that maps h to r. [51]

If an autoencoder succeeds to learn to set $g(f(x)) = x$ everywhere, then the autoencoder is not really useful. Autoencoders are designed to be unable to learn how to copy something perfectly. They are restricted in ways that only allow them

to copy approximately and only copy inputs that are similar to the training data. The model is forced to prioritize what should be copied into the input and this is the reason why it often learns useful properties of the data. [51]

Autoencoders are designed similarly to neural networks with a bottleneck, like in Figure 3.12, in the network, which gives a compressed representation of the original image. The network takes an unlabeled dataset as an input and a reconstruction $\hat{x}$ of the original input x is made. The network can be trained by minimizing the reconstruction error

$$\mathcal{L} = \mid x - \hat{x} \mid \tag{3.8}$$

which measures the differences between the original input and the reconstruction. The bottleneck is really important in the network. Without the bottleneck, the network could learn to memorize the input values and directly passing them forward in the network. The important information that is needed to force learning of the input data is in the bottleneck [52].

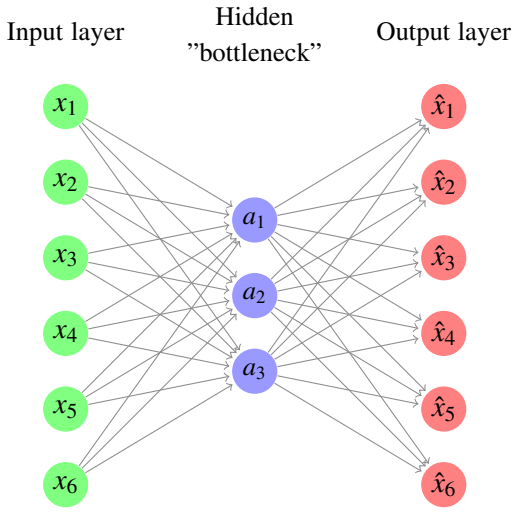


Figure 3.12: Illustriation of how an autoencoder network looks like including a bottleneck.

The idea of autoencoders has been a part of the neural networks for decades [51]. Traditionally autoencoders were used to reduce the dimensionality. It was not until recently autoencoders that were started to use for other things like denoising images.

3.4.1 Denoising AutoEncoder

As shown in the previous section an ordinary autoencoder minimizes some function

$$\mathcal{L} = (x, g(f(x))) \tag{3.9}$$

This function encourages $g \cdot f$ learn to be like an identity function if there is capacity to do so.

A denoising autoencoder minimizes instead:

$$\mathcal{L} = (x, g(f(\tilde{x}))) \tag{3.10}$$

where $\tilde{x}$ is a copy of input x that has been damaged by some form of noise. Denoising autoencoders have to take away the noise rather than just copy the inputs [51]. The whole approach of the denoising autoencoder is to train a model that can take a corrupted input, but still maintain the uncurrupted data as its output, like shown in Figure 3.13. The model is not able to memorize the training data because of the bottleneck in the network.

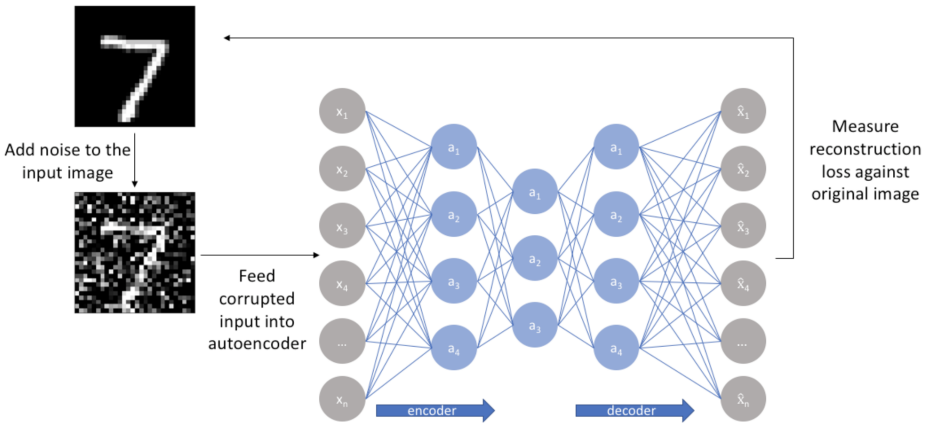


Figure 3.13: Illustration of a denoising autoencoder. Noise is added to the input image before the image is fed into the autoencoder. The output representation is then compared to the original input image and losses are calculated. [52]

Chapter 4

Software

In this chapter a short description of the used programming language Python and the used deep learning framework Keras will be presented.

4.1 Python

Python is an open source high-level programming language which was developed and created by Guido van Rossum in 1989 [53]. Python is a powerful, flexible and ”easy-to-learn” language. It is used in many different fields by many companies because of the large amount of packages that are freely available for everyone. Thanks to all the libraries and special commands in Python, Python is a very powerful tool [54]. In this Master's Thesis the Python v.3.7.1 was used.

4.2 Keras

Keras is one of many deep learning frameworks that are available today. Keras is easy to learn and use, so it allows the user to be more productive and test different ideas. Keras has the ability to work on top of TensorFlow with GPU support [55]. Thanks to the functional API in Keras it gives the opportunity to create more complex models and gives a lot of flexibility [56]. In this Master's Thesis the Keras v.2.2.4 was used.

Chapter 5

Data Material

All of the data used in this Master's Thesis have been provided by EXINI Diagnostics AB, Lund, Sweden. The dataset contains of 126 Computed Tomography (CT) images, where 19 of the images contains artifacts due to hip replacement. The CT images were split into two sets according to:

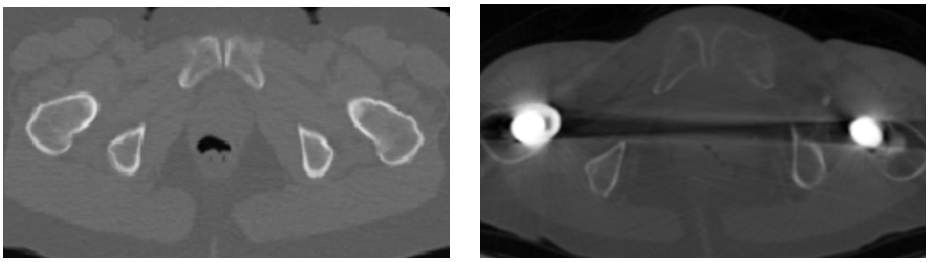
- CT images without artifacts: 107 CT images
- CT images with artifacts: 19 CT images

Further on the dataset with artifacts were split into two sets according to:

- CT images with artifacts used to make more training data including artifacts: 13 CT images
- CT images with artifacts used as final test set: 6 CT images

The dataset without artifacts containing 107 CT images and the dataset including artifacts containing 13 CT images were used during the training to construct the model and evaluate it. The small dataset including artifacts containing 6 CT images was only used at the end to evaluate the final model.

All of the CT images used had the size $(91, 146, 251)$, were black and white and were cropped to only include the pelvis area. Due to that there was only 19 of the CT images that included artifacts caused by hip replacement, and 6 of them were excluded during the training procedure so they could be used as a final test set, so we had to make more data that included artifacts so the training could be done right. The dataset with artifacts containing 13 CT images was used to make more data including artifacts.



(a) (b)

Figure 5.1: Two examples how a slice in the images can look like when it has not any hip replacements (a) and how it can look when it has two hip replacements (b).

A CT image attenuation values are expressed as Hounsfield Units (HU) that is a linear density scale. The name Hounsfield Units is given after the inventor of CT scanning Sir Godfrey Newbold Hounsfield. [57] In Hounsfield scale water corresponds to 0 HU and all other CT values can be calculated by equation 5.1.

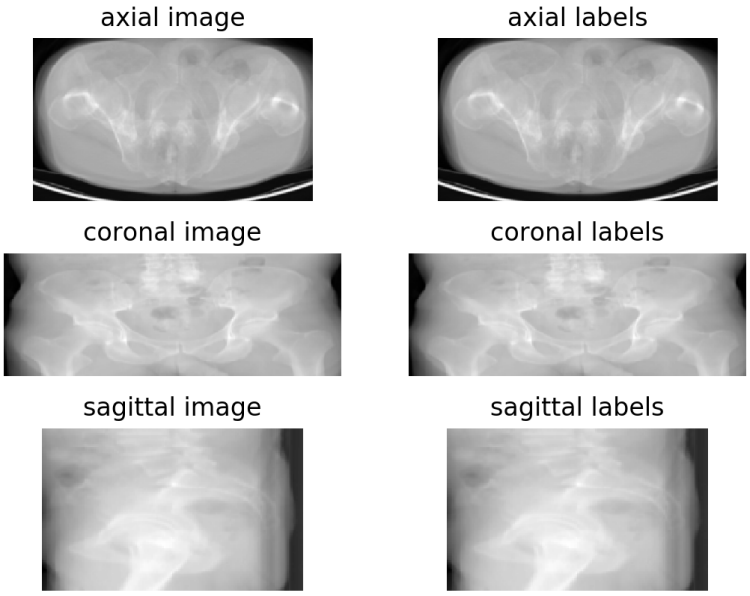
$$HU = 1000 \times \frac{\mu_{Tissue} - \mu_{H_2O}}{\mu_{H_2O}} \tag{5.1}$$

In Table 5.1 some examples of different HU are given for different things in a CT image.

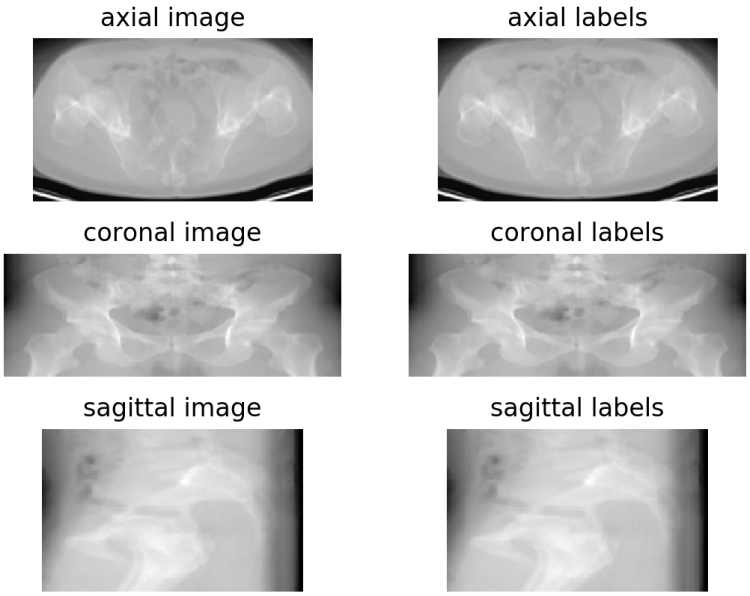
Hounsfield Units, HU	Tissue
-1000	Air
-700 to -600	Lung
-120 to -90	Fat
0	Water
-700 to 900	Soft tissue
30	Kidney
10 to 50	Muscle
> 1000	Bone, Calcium, Metal

Table 5.1: Some approximate Hounsfield Units for CT images and what they corresponds to.

In Figure 5.2 two randomly plotted CT images and labels from the dataset without artifacts are visualized. The CT images including artifacts, were at the beginning in different sizes, before they were cropped to include only pelvic area. Two randomly plotted CT images and labels from the dataset including artifacts are shown in Figure 5.3.

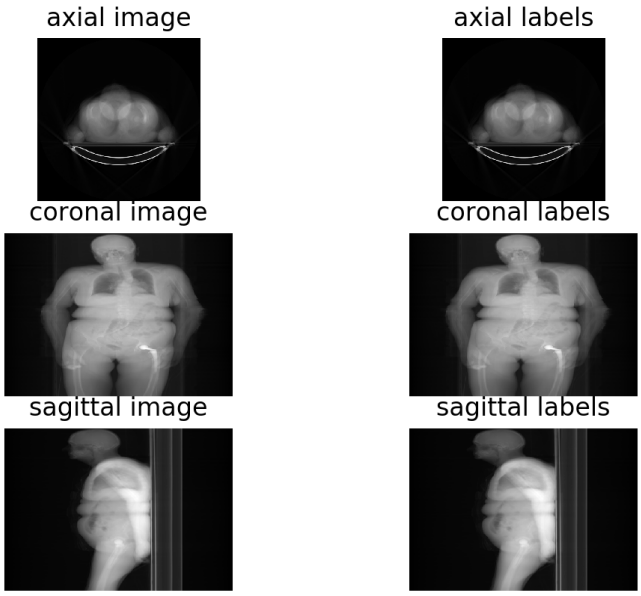


(a)

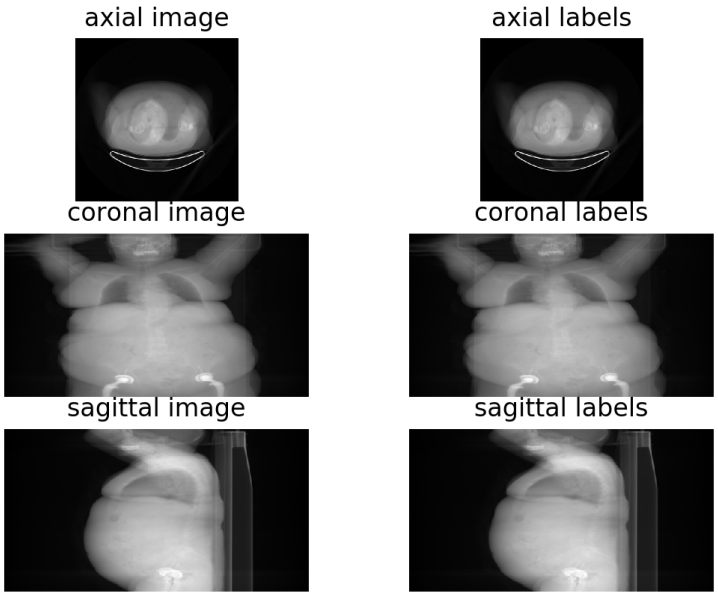


(b)

Figure 5.2: Subfigures (a) – (b) are two randomly plotted CT images and labels including only pelvic area from the dataset without artifacts. Each CT image and label is plotted in axial, coronal and sagittal plane.



(a)



(b)

Figure 5.3: Subfigures (a) – (b) are two randomly plotted CT images and labels including artifacts. Each CT image and label is plotted in axial, coronal and sagittal plane.

Chapter 6

Model

The first data set that EXINI Diagnostics AB provided had the size $(91, 146, 251)$ and did not include any artifacts. This was the starting dataset for the model setup. The inspiration for the model setup came from [14]. All of the images were cropped, so that only pelvic area were included. In Figure 5.2 some of the first dataset images and corresponding labels were randomly plotted. Each CT image and label is plotted in three different planes: axial, coronal and sagittal plane.

The model was constructed for input images with size $(91, 146, 251)$, so when the second dataset was given that included different kinds of hip replacements, they needed to be cropped into the right size. The second dataset including hip replacements and artifacts were in different size - $(274, 512, 512)$, $(574, 512, 512)$ and $(324, 512, 512)$. All of the images included the whole upper body and some also included a part of the legs. There were also some pictures that also included the whole legs, as shown in Figure 5.3 where some of the CT images with artifacts and corresponding label were randomly plotted. All the images in the second dataset needed to be cropped into the same size as the first dataset, $(91, 146, 251)$, including only pelvic area.

Due to the small amount of images with hip replacements and artifacts, only 19 CT images in total and 6 of them were the test set, more images needed be generated whom includes hip replacements and artifacts. Hip replacements from the 13 CT images, including artifacts, were taken to generate more CT images including hip replacements and artifacts. To achieve this the dataset including 13 CT images with artifacts were all threshed with a threshold of 1400, where all of the pixels with an intensity higher than 1400 were kept with their actually intensity and at the same place in the CT image and the rest pixels were set to zero. Each threshold CT image, including hip replacements and artifacts were randomly applied on the dataset without any artifacts in the training set. When applying the hip replacements and artifacts, each pixel with an intensity higher than 0 in the threshed CT image were inserted at their current location but in the CT image without any artifacts. In Figure 6.1 two random CT images and its labels from the training set are plotted in different planes to illustrate how a training image could look like without

and with a hip replacement randomly applied.

The model represented in Figure 6.2 was normalized trained on the training set with randomly applied hip replacements and artifacts. As mentioned the inspiration of the model came from [14] where they reduce noise in 3D images. The model is constructed as an autoencoder with three branches. Each branch first uses a convolutional layer which consists of 25 filter kernels of size $5 \times 5 \times 5$. These 25 filters corresponds to the elements that the model seeks to learn. The obtained 25 feature maps are threshold by using the ReLU function.

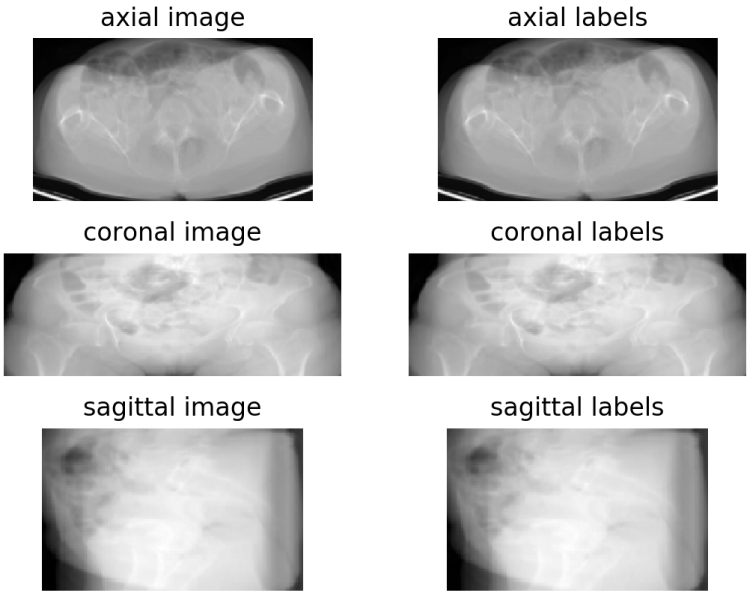
Each branch in the model operates on the same input image, but in different sizes. The MaxPooling layer is reducing the size of the input CT image in two of the branches before the branch use convolutional layer. In each branch a bottleneck is constructed and at the end the decoded CT images are added. The added CT image is given as an output of the model.

The loss of the model is optimized with the Adam optimizer from Keras [58] and mean squared error (MSE) function. The used loss is optimized with a modified Adam optimizer according to:

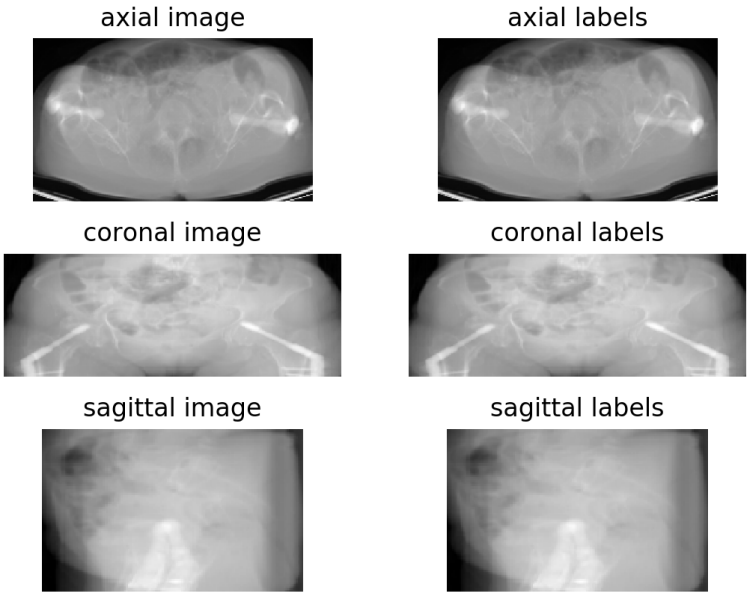
- **lr** = 1×10^{-6} (learning rate)
- **beta**$_1$ = 0.995
- **beta**$_2$ = 0.999
- **epsilon** = 1×10^{-8}

and a written mean squared error function (Equation 6.1).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{6.1}$$



(a) CT image without applied hip replacement.



(b) CT image with applied hip replacement.

Figure 6.1: One randomly generated CT images from the training set without and with randomly applied hip replacements from the validation set.

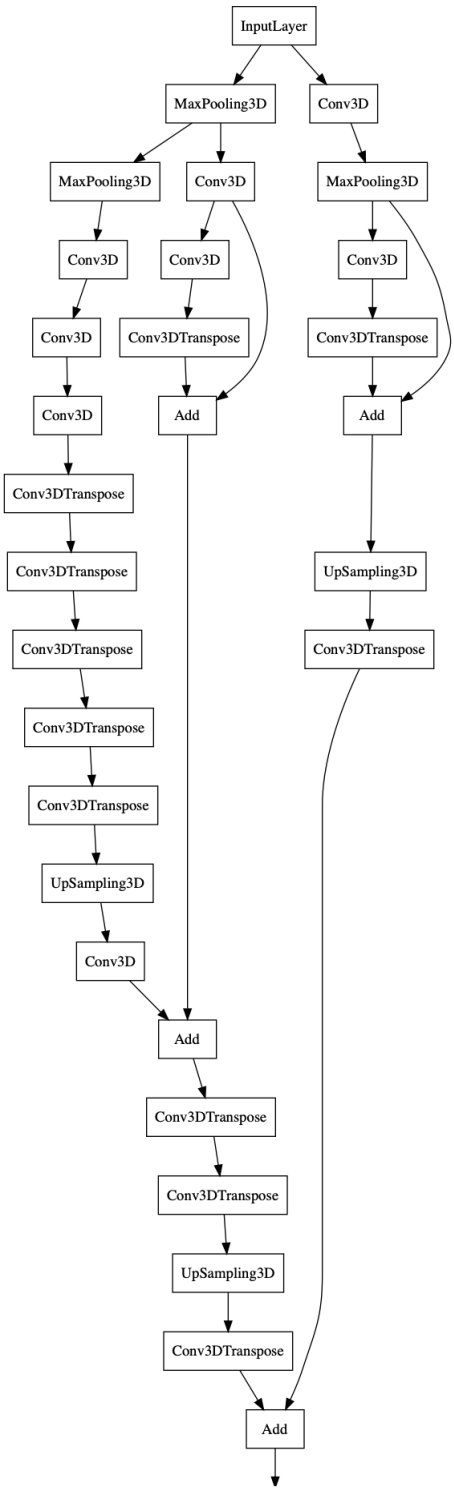


Figure 6.2: Graphical illustration of the autoencoder model which has been constructed and used in this Master's Thesis.

Chapter 7

Result

In this chapter, the result of the two steps in the work will be represented. The first step is a proof of concept, and the final one is to actually reduce artifacts due to hip replacements in CT images.

7.1 Proof of concept

The proof of concept is a validation part to see that the model is able to encode relevant information from which it may reconstruct the original input. See if the model can reconstruct the CT images. In Figure 7.1 a randomly selected image, including the original image and the reconstructed image from the model can be viewed. Some more figures of this result can be viewed in Appendix A. The images to the left in each image is the original CT image plotted in axial, coronal and sagittal plane and to the right the reconstructed CT image plotted in the same planes. In Figure 7.2 the loss is shown. The dataset was divided according to:

- Training set: 80% of the dataset
- Validation set: 20% of the dataset

The loss in Figure 7.2 is optimized with the modified Adam optimizer described in the previous chapter 6, and with the same mean squared error function shown in equation 6.1. The model is run for 200 epochs.

In Figure 7.2 the loss function is the loss value is getting smaller and smaller for each epoch. This means the model learns by the training and the predictions is not so different to the actual image. The small value of loss value is the reason why the reconstructed CT images are very similar to the original CT images.

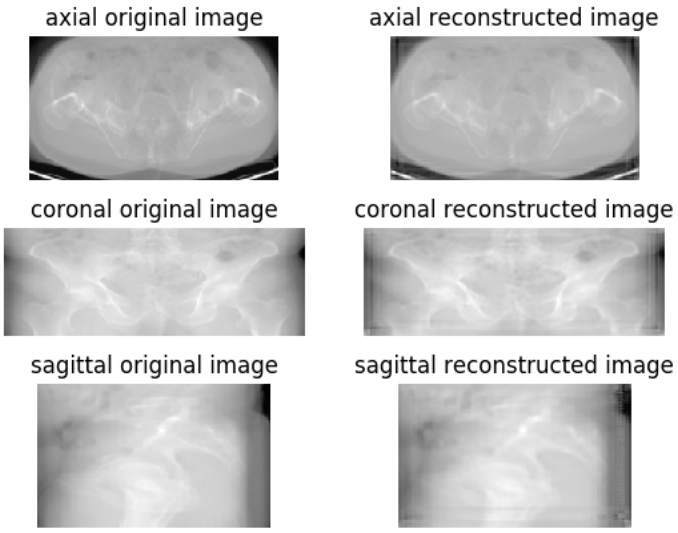


Figure 7.1: A randomly plotted CT image and the reconstructed image by the model. To the left the original image is plotted and to the right the reconstructed image. The original image and the reconstructed image is plotted in axial, coronal and sagittal plane.

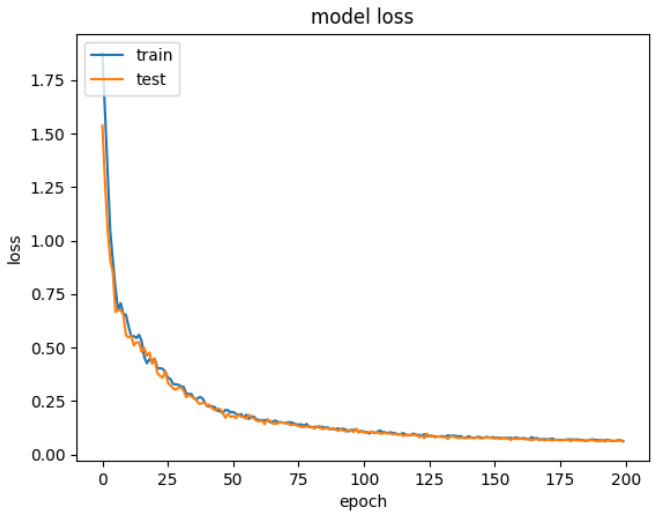


Figure 7.2: The loss function during the training of the proof of concept.

7.2 Reducing artifact

In Figure 7.3 and Figure 7.4 two test images from the test set is visualized, and the rest of the images from the test set can be viewed in Appendix B.1. None of the test images were shown to the produced network prior the test phase. The images include the original test images to the left and the reconstructed images with the model to the right in different planes. In Figure 7.5 the loss of the model is shown.

The loss in Figure 7.5 is optimized with the same modified Adam optimizer as for the proof of concept in the previous section, and with the same mean squared error function shown in equation 6.1. The model is run for 400 epochs.

In Figure 7.5 the loss function values are much higher than the values of the loss function for our proof of concept, Figure 7.2. The values are much higher in this case due to the fact that the reconstructed images are different from the original test images. The reconstructed images has lost most of the hip replacements and artifacts.

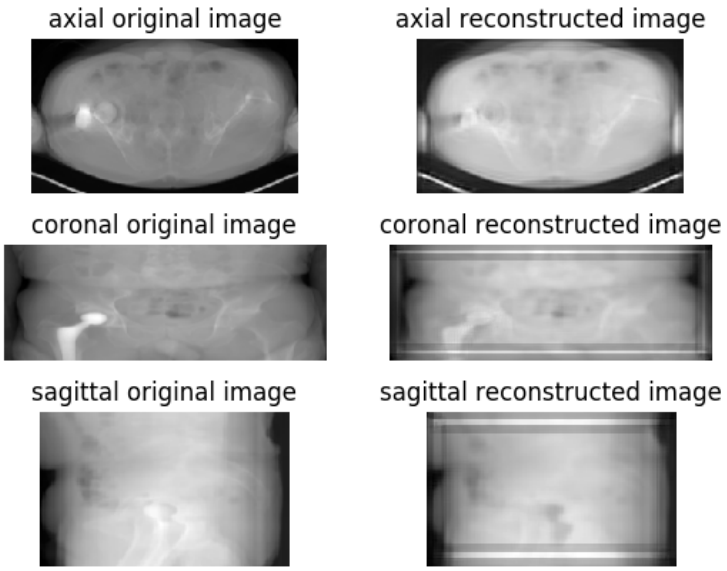


Figure 7.3: A CT image from the test set and its reconstructed image with the model. To the left is the original image plotted and to the right the reconstructed image. The original CT image and the reconstructed CT image is plotted in axial, coronal and sagittal plane.

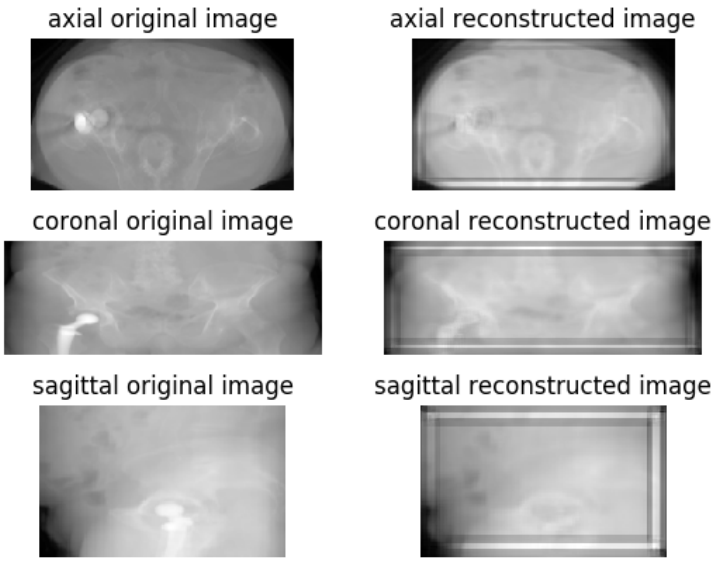


Figure 7.4: A CT image from the test set and its reconstructed image with the model. To the left is the original image plotted and to the right the reconstructed image. The original CT image and the reconstructed CT image is plotted in axial, coronal and sagittal plane.

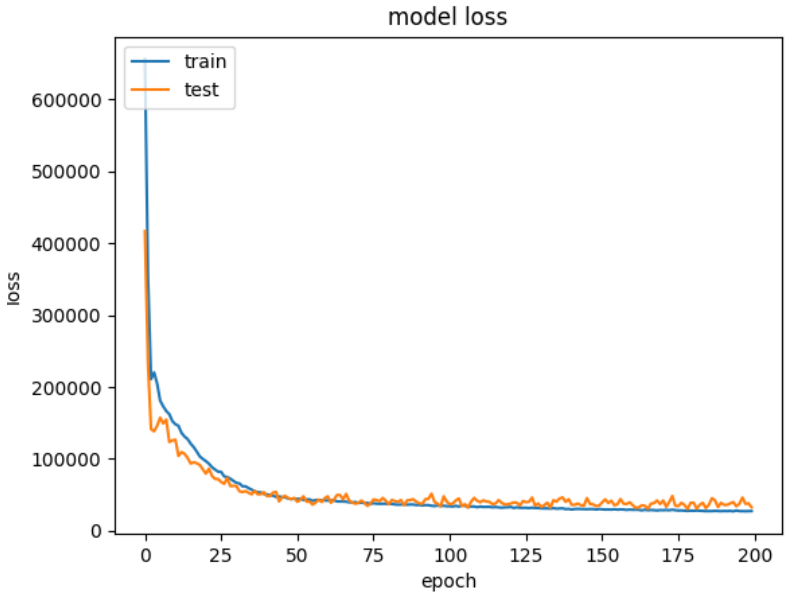
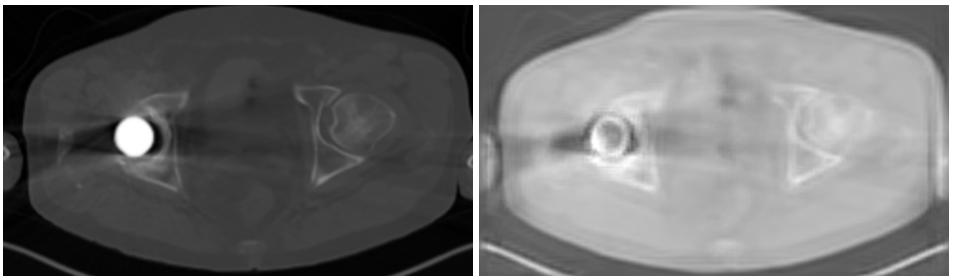


Figure 7.5: The loss function during the training when trying to reduce the artifacts.

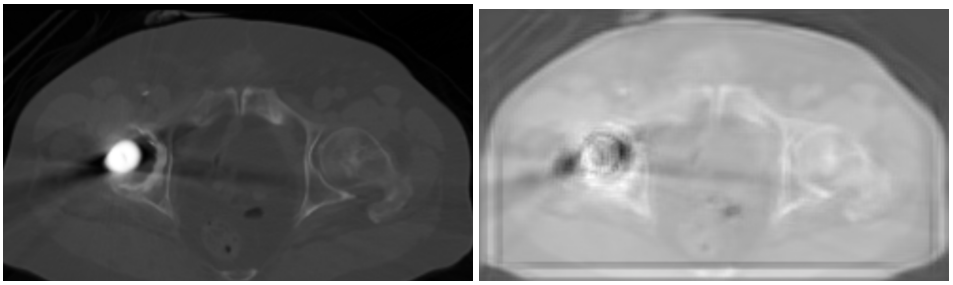
In Figure 7.6 and Figure 7.7 slices from two CT images are shown from the original test image and the corresponding slice in the reconstructed CT image. In these images the results are shown more clearly than in the previous images plotted in different planes. The hip replacements are nearly completely gone and the artifacts are reduced. More images of the slices can be found in Appendix B.2.



(a) A slice in the original image.

(b) The corresponding slice in the reconstructed image.

Figure 7.6: A slice of the original image and the corresponding slice in the reconstructed image shown in Figure 7.3.



(a) A slice in the original image.

(b) The corresponding slice in the reconstructed image.

Figure 7.7: A slice of the original image and the corresponding slice in the reconstructed image shown in Figure 7.4.

Chapter 8

Discussion

We have successfully implemented an autoencoder that reduces artifacts due to hip replacements in a CT image.

The first part of the work was just a proof of concept part to show that the constructed model actually works and is able to reconstruct an input image. By showing that the model is able to reconstruct CT images, we could say that the constructed model works as a pure autoencoder for the reconstruction of images. The first dataset did not include any artifacts, so this part was done before the second dataset was available. Looking at the result of the Proof of concept part in section 7.1 the result is good. The reconstructed image looks nearly exactly the same as the original image. There are some differences at the edges, but looking at the whole CT image in the different planes the images look good and similar to the originals. These differences at the edges in the plane-images could be avoided by changing the parameters, e.g. learning rate or beta_1, used in the Adam optimizer and changing little in the constructed model.

Due to the fact that our proof of concept was good, we could draw the conclusion that our model was good enough and could be used in the final part trying to reduce artifacts due to hip replacements. This part was the biggest part of this Master's Thesis. Looking at the result of the reducing artifacts part in section 7.2 the result is quite good. The reconstructed images have some differences close to the edges, but the main goal of trying to reduce the artifacts is beginning to reach. The artifacts are starting to disappear from the CT images, if the reconstructed and the original image is compared. This is even better viewed in the slice images. Some artifacts have nearly completely disappeared, and some are not. The model is also trying to remove the hip replacements. This result is quite good compared to how difficult this task is and how many different approaches, there are to handling this type of problem, but also how little amount of data was used to complete this task. To improve this result, there is a need for a larger amount of data to train the model on and a larger amount of data including artifacts. The lack of data is a common problem when working with medical data, e.g. CT images and PET/SPECT images. It is difficult to get this type of data, probably because of the rights to

anonymity. This is maybe something companies want to work with and develop a model that can reduce the features in a face and make all faces smooth or deformed so it is impossible to recognize the person in e.g. a CT image.

In our model, the result was quite good but can get better. There are several improvements that can be done, and one of them is already mentioned - a bigger dataset. Next improvement is a hard one to improve, but it is a problem in this case - remake an artifact. It is easy to extract a hip replacement from a CT image to put it into another to make a larger dataset with hip replacements, but the artifacts due to hip replacements are hard to extract from an image. The difficulties to extract artifacts in CT images are because its HU values are similar to the bone value and some are even similar to the soft tissue values. If the artifacts could be simulated in some way this would probably improve the model a lot. The best solution of this problem would be to take a CT image before putting the implant into a person and take another CT image after, so there is a ground truth how the person actually looks. But this is impossible.

Another thing that could be improved is to reduce the differences there are now towards the edges in the reconstructed images. The differences towards the edges are certainly caused due to that the branches in the model operate on images of different sizes. In the reconstruction faze of these images in different sizes, some information is lost and the differences appear in the reconstructed images. By chancing the reshaped sizes of the original image in the model the reconstructed image could give a better result.

Overall the implemented autoencoder model in this Master's Thesis is quite good and a good starting point for further work for reducing artifacts and remove hip replacements in CT images.

Bibliography

[1] Redaktionen. 2015. Prostatacancer [Prostate Cancer]. *Doctorn*. 02-23. https://www.doktorn.com/artikel/prostatacancer (Accessed on 05/14/2019)

[2] Bratt Ola Professor i urologi vid Sahlgrenska akademin, Göteborgs universitet. 2018. Prostatacancer [Prostate Cancer].*Cancerfonden*. 6 April. https://www.cancerfonden.se/om-cancer/prostatacancer (Accessed on 05/14/2019)

[3] National Cancer Institute at the National Institutes of Health. 2013. Computed Tomography (CT) Scans and Cancer. *National Cancer Institute*. July 16. https://www.cancer.gov/about-cancer/diagnosis-staging/ct-scans-fact-sheet (Accessed 06/12/2019)

[4] The American Cancer Society medical and editorial content team. 2015. CT Scan for Cancer. *American Cancer Society*. 30 November. https://www.cancer.org/treatment/understanding-your-diagnosis/tests/ct-scan-for-cancer.html (Accessed 06/12/2019)

[5] F Edward Boas and Dominik Fleischmann. CT artifacts: Causes and reduction techniques. *Stanford University School of Medicine, Department of Radiology*. 300 Pasteur Drive. Stanford USA. CA 94305.

[6] S. Shellikeri, E. Girard, R. Setser, J. Bao, A.M. Cahill. Metal artefact reduction algorithm for correction of bone biopsy needle artefact in paediatric C-arm CT images: a qualitative and quantitative assessment. *Clinical Radiology*. 71 (2016) 925e931.

[7] Lifeng Yu, PhD, Hua Li, PhD, Jan Mueller, MS, James M. Kofler, PhD, Xin Liu, PhD, Andrew N. Primak, PhD, Joel G. Fletcher, MD, Luis S. Guimaraes, MD, Thanila Macedo, MD, and Cynthia H. McCollough, PhD. 2009. Metal Artifact Reduction From Reformatted Projections for Hip Prostheses in Multislice Helical Computed Tomography. *Health of National Institutes*. November. 44(11): 691–696. doi:10.1097/RLI.0b013e3181b0a2f9.

[8] Mizuho Nishio, Chihiro Nagashima, Saori Hirabayashi, Akinori Ohnishi, Kaori Sasaki, Tomoyuki Sagawa, Masayuki Hamada, Tatsuo Yamashita. Convolutional auto-encoder for image denoising of ultra-low-dose CT. *Heliyon 3*. (2017) e00393. doi: 10.1016/j.heliyon.2017. e00393

[9] Heyi Li and Klaus Mueller. Low-Dose CT Streak Artifacts Removal using Deep Residual Neural Network. *Xi'an*. (June 2017). Stony Brook, NY 11794–2424.

[10] Hu Chen, Yi Zhang, Mannudeep K. Kalra, Feng Lin, Yang Chen, Peixo Liao, Jiliu Zhou and Ge Wang. Low-Dose CT with a Residual Encoder-Decoder Convolutional Neural Network (RED-CNN). *IEEE*

[11] Dufan Wu, Kyungsang Kim, Georges El Fakhri and Quanzheng Li. A Cascaded Convolutional Nerual Network for X-ray Low-dose CT Image Denoising. (28 Aug 2017). arXiv:1705.04267v2 [cs.CV]

[12] HU CHEN, YI ZHANG, WEIHUA ZHANG, PEIXI LIAO, KE LI, JILIU ZHOU AND GE WANG. Low-dose CT via convolutional neural network. *BIOMEDICAL OPTICS EXPRESS 679*. Vol. 8, No. 2. 1 Feb 2017.

[13] Lei Xiang, Yu Qiao, Dong Nie, Le An, Weili Lin, Qian Wang, Dinggang Shen. Deep auto-context convolutional neural networks for standard-dose PET image estimation from low-dose PET/MRI. *Neurocomputing 267*. (2017) 406–416.

[14] Katrin Mentl, Boris Mailhe, Florin C. Ghesu, Frank Schebesch, Tino Haderlein, Andreas Maier, Mariappan S. Nadar. NOISE REDUCTION IN LOW-DOSE CT USING A 3D MULTISCALE SPARSE DENOISING AUTOENCODER. 2017 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING. SEPT. 25–28. 2017. TOKYO JAPAN.

[15] Machine Learning Tutorial for Beginners. *Guru99*. `https://www.guru99.com/machine-learning-tutorial.html` (Accessed on 06/05/2019)

[16] Browniee Jason. 2016. Supervised and Unsupervised Machine Learning Algorithms. *Machine Learning Mastery*. March 16. `https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/` (Accessed on 05/31/2019)

[17] Mrukwa Grzegorz. 2018. Types of Machine Learning Algorithms: Supervised and Unsupervised Learning. *Netguru*. Oct 16. `https://www.netguru.com/blog/types-of-machine-learning-/algorithms-supervised-and-unsupervised-learning` (Accessed on 05/31/2019)

[18] Orhan Gazi Yalcin. 2018. Image Classification in 10 Minutes with MNIST Dataset. *Towards Data Science*. Aug 18. https://towardsdatascience.com/image-classification-in-10-/minutes-with-mnist-dataset-54c35b77a38d (Accessed on 05/31/2019)

[19] Katariya Yash. 2017. Applying Convolutional Neural Network on the MNIST dataset. *GitHub*. April 15. https://yashk2810.github.io/Applying-Convolutional-Neural-/Network-on-the-MNIST-dataset/ (Accessed on 05/31/2019)

[20] Hariharan Ashwin. 2018. How to Use Machine Learning to Predict the Quality of Wines. *freeCodeCamp*. 7 February. https://www.freecodecamp.org/news/using-machine-learning-to-predict-the-quality-of-wines-9e2e13d7480d/ (Accessed on 05/31/2019)

[21] Amnah Khatun. 2018. Let's know Supervised and Unsupervised in an easy way. *Chatbots Magazine*. Jul 10. https://chatbotsmagazine.com/lets-know-supervised-and-unsupervised/-in-an-easy-way-9168363e06ab?gi=38b16dbee382 (Accessed on 05/31/2019)

[22] DataFlair Team. 2019. Data Science K-means Clustering – In-depth Tutorial with Example. *DataFlair*. May 3. https://data-flair.training/blogs/k-means-clustering-tutorial/ (Accessed on 04/24/2020)

[23] Czar Yobero. 2018. K-Means Clustering Tutorial. *RPubs*. January 2. https://rpubs.com/cyobero/k-means (Accessed on 05/31/2019)

[24] Kaushik Raghupathi. 2018. 10 Interesting Use Cases for the K-Means Algorithm. *DZone*. March 27. https://dzone.com/articles/10-interesting-use-cases-for-the-k-means/-algorithm (Accessed on 05/31/2019)

[25] Rejat S. 2019. Understanding Dimensionality Reduction for Machine Learning. *Towards Data Science*. Nov 30. https://towardsdatascience.com/understanding-dimensionality-reduction-/for-machine-learning-ad9a3811bd89 (Accessed on 04/24/2020)

[26] Rinu Gour. 2019. Dimensionality Reduction in Machine Learning. *Medium*. Apr 8. https://medium.com/@rinu.gour123/dimensionality-reduction-in-machine-/learning-dad03dd46a9e (Accessed on 04/24/2020)

[27] Judy T Raj. 2019. A beginner's guide to dimensionality reduction in Machine Learning. *Towards Data Science*. Mar 11. https://towardsdatascience.com/dimensionality-reduction-for-/machine-learning-80a46c2ebb7e (Accessed on 04/24/2020)

[28] A.I. Wiki. A Beginner's Guide to Deep Reinforcement Learning. *Skymind*. https://skymind.ai/wiki/deep-reinforcement-learning (Accessed on 06/03/2019)

[29] Prateek Bajaj. Reinforcement learning. *GeeksforGeeks*. https://www.geeksforgeeks.org/what-is-reinforcement-learning/ (Accessed on 06/16/2019)

[30] Deep Learning. What Is Deep Learning?. *MathWorks*. https://www.mathworks.com/discovery/deep-learning.html (Accessed on 05/13/2019)

[31] Michael A. Nielsen. Neural Network and Deep Learning. *Determination Press*. 2015.

[32] Gulcehre Caglar. 2015. Welcome to Deep Learning. *Deep Learning*. December 1. http://deeplearning.net (Accessed on 05/13/2019)

[33] A.I Wiki. Artificial Intelligence (AI) vs. Machine Learning vs. Deep Learning. *skymind*. https://skymind.ai/wiki/ai-vs-machine-learning-vs-deep-learning (Accessed on 05/21/2019)

[34] McCulloch WS. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics 5*.

[35] Azevedo FA. 2009. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *The Journal of comparative neurology 5*. DOI: 10.1002/cne.21974.

[36] EX-job proposal. 2003. Matematiskt-Neurobiologiskt examensarbete [Mathematical-Neurobiological degree project]. *Chalmers Tekniska Högskola*. 02 March.

[37] Dang Johnny. 2016. Classification in Bone Scintigraphy Inages Using Convolutional Neural Network. *Lund University*.

[38] 6.034 Artificial Intelligence. *MIT - Massachusetts Institute of Technology* http://web.mit.edu/6.034/wwwbob/recit-nets.pdf (Accessed on 05/20/2019)

[39] Gjertsson Konrad. 2017. Segmentation in Skeletal Scintigraphy Images using Convolutional Neural Networks. *Lund University*.

[40] Saha Sumit. 2018. A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. *Towards Data Science*. Dec 15. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-/neural-networks-the-eli5-way-3bd2b1164a53 (Accessed on 06/06/2019)

[41] Adit Deshpande. 2016. A Beginner's Guide To Understanding Convolutional Neural Networks. *GitHub*. July 20. https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/ (Accessed on 06/06/2019)

[42] Machine Learning. 2018. An intuitive guide to Convolutional Neural Networks. *freeCodeCamp*. 24 April. https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/ (Accessed on 06/06/2019)

[43] cs231n. Convolutional Neural Networks (CNNs / ConvNets). *GitHub*. http://cs231n.github.io/convolutional-networks/ (Accessed 06/06/2019)

[44] Mayank Mishra. 2019. Convolutional Neural Networks, Explained. *Learn Data Science*. March 7. https://www.datascience.com/blog/convolutional-neural-network (Accessed 06/06/2019)

[45] SuperDataScience Team. 2018. Convolutional Neural Networks (CNN): Step 4 - Full Connection. *SuperDataScience*. Aug 18. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection (Accessed 06/10/2019)

[46] Udeme Udofia. 2018. Basic Overview of Convolutional Neural Network (CNN). *Medium*. Feb 13. https://medium.com/@udemeudofia01/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17 (Accessed 06/10/2019)

[47] Dan B. 2018. Rectified Linear Units (ReLU) in Deep Learning. *kaggle*. https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning (Accessed 06/10/2019)

[48] Danqing Liu. 2017. A Practical Guide to ReLU. *TyniMind*. 30 nov. https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7 (Accessed on 06/11/2019)

[49] Eddins Steve. 2018. Defining Your Own Network Layer. *MathWorks*. 27 Sep. https://blogs.mathworks.com/deep-learning/2018/01/05/defining-your-own-network-layer/ (Accessed on 06/11/2019)

[50] Renu Khandelwal. 2018. Deep Learning Autoencoders. *Medium*. Dec 2. https://medium.com/datadriveninvestor/deep-learning-autoencoders-db265359943e (Accessed on 06/12/2019)

[51] Ian Goodfellow and Yoshua Bengio and Aaron Courville. 2016. Deep Learning. *MIT Press*. Goodfellow-et-al-2016.

[52] Jordan Jeremy. 2018. Introduction to autoencoders. 19 March. https://www.jeremyjordan.me/autoencoders/ (Accessed on 06/12/2019)

[53] Sharwood Simon. 2018. Python creator Guido van Rossum sys.exit()s as language overlord. *The Register*. 13 July. https://www.theregister.co.uk/2018/07/13/python_creator_guido_van_rossum_quits/ (Accessed on 05/13/2019)

[54] Computer History Museeum. 2018. Guido van Rossum. https://www.computerhistory.org/fellowawards/hall/guido-van-rossum/ (Accessed on 05/13/2019)

[55] Keras Documentation. https://keras.io/why-use-keras/ (Accessed on 05/13/2019)

[56] Browniee Jason. 2017. How to Use the Keras Functional API for Deep Learning. *Machine Learning Mastery*. October 27. https://machinelearningmastery.com/keras-functional-api-deep-learning/ (Accessed on 05/13/2019)

[57] M. H. Lev and R. G. Gonzalez. CT Angiography and CT Perfusion Imaging. *Department o f Radiology, Massachusetts General Hospital and Harvard Medical School*. Boston, Massachusetts 02114.

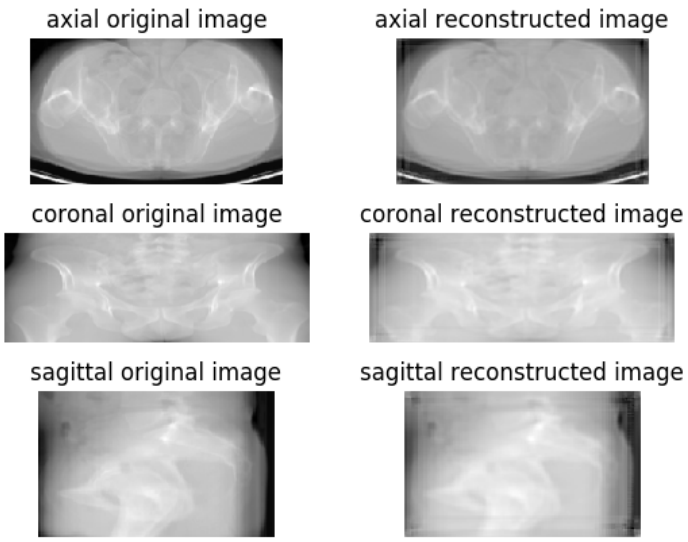
[58] Keras Documentation. Usage of optimizers. https://keras.io/optimizers/ (Accessed on 06/16/2019)

[59] Keras Documentation. Usage of loss functions. https://keras.io/losses/ (Accessed on 06/16/2019)

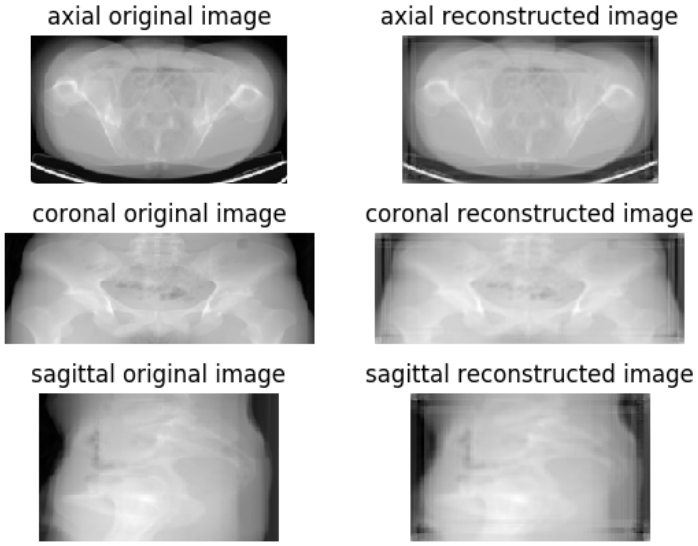
Appendix A

Proof of concept

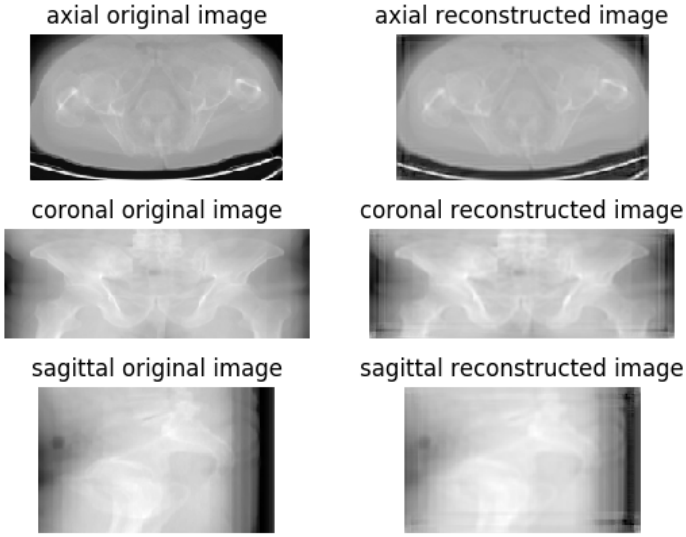
Some more randomly plotted results of the proof of concept.



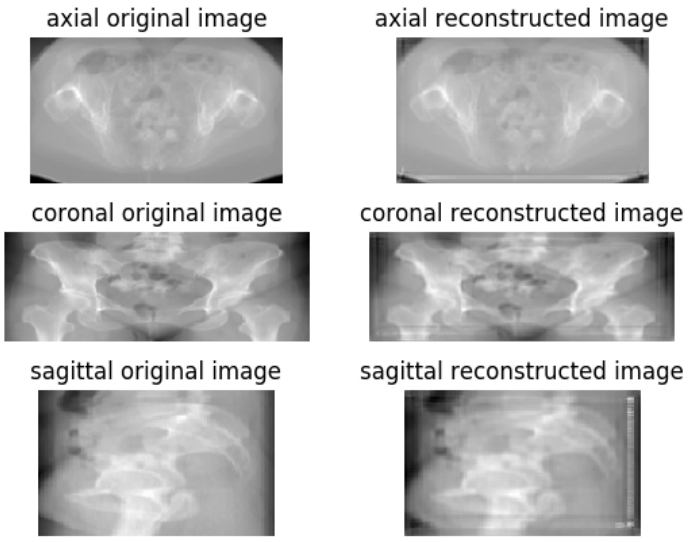
(a)


axial original image
axial reconstructed image
coronal original image
coronal reconstructed image
sagittal original image
sagittal reconstructed image

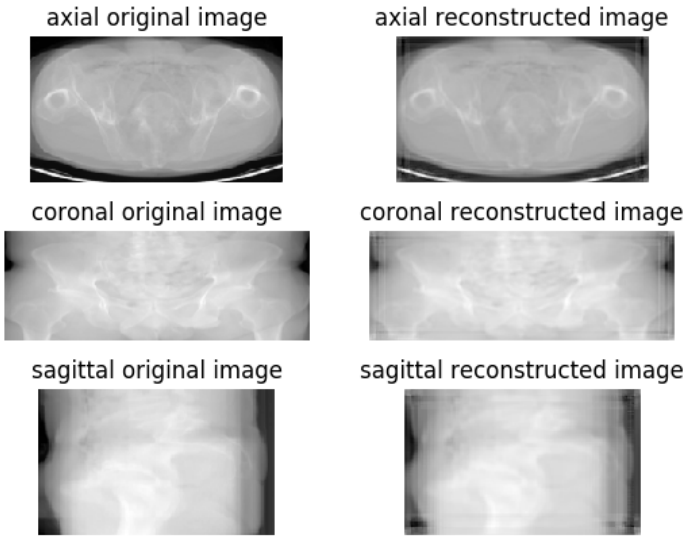
(b)


axial original image
axial reconstructed image
coronal original image
coronal reconstructed image
sagittal original image
sagittal reconstructed image

(c)



(d)



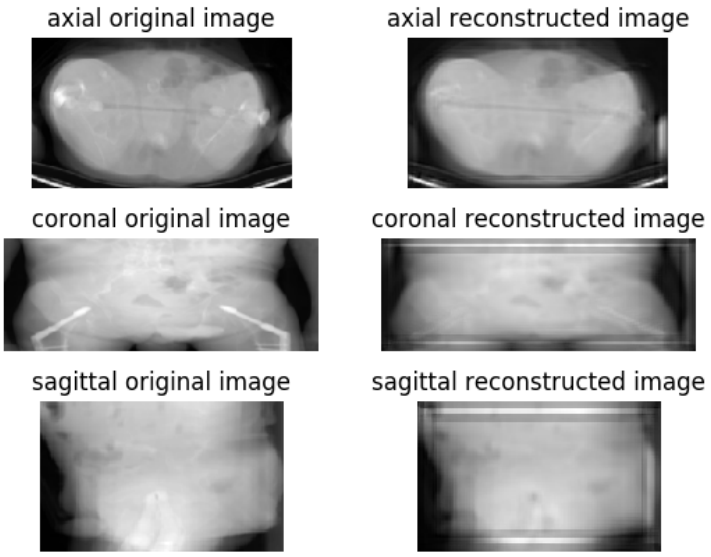
(e)

Figure A.1: Subfigures (a) – (e) are six randomly plotted CT images and their reconstructed image with the model. To the left in each image, is the original image plotted and to the right the reconstructed image. Each image and reconstructed image is plotted in axial, coronal and sagittal plane.

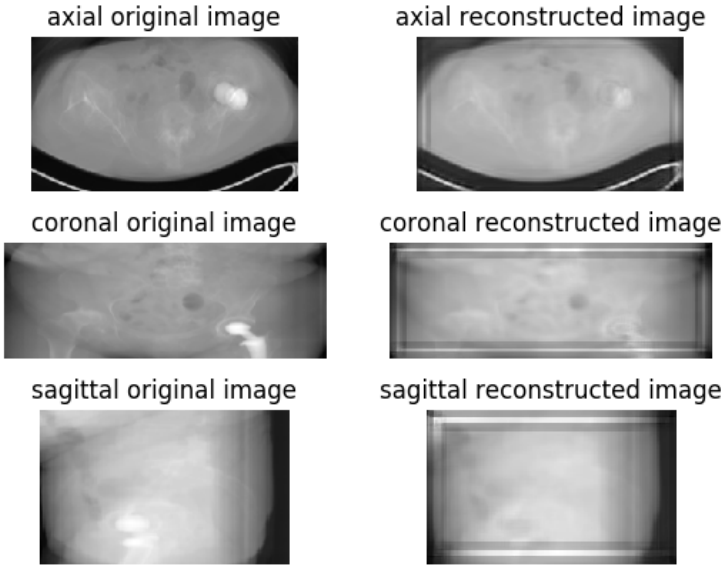
Appendix B

Reducing artifact

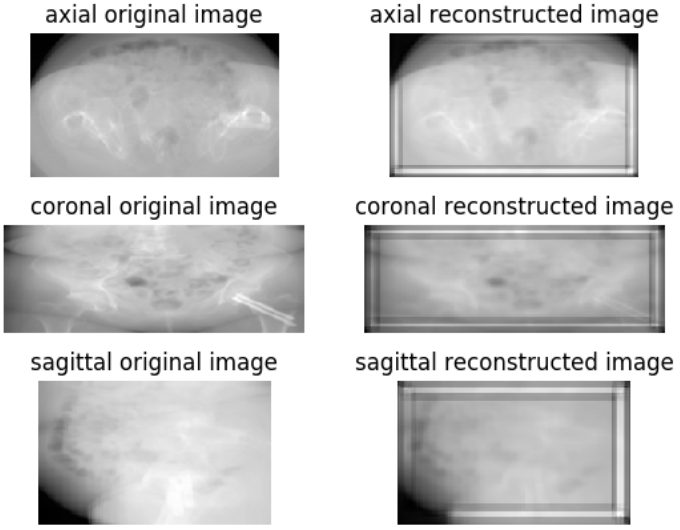
B.1 Test set



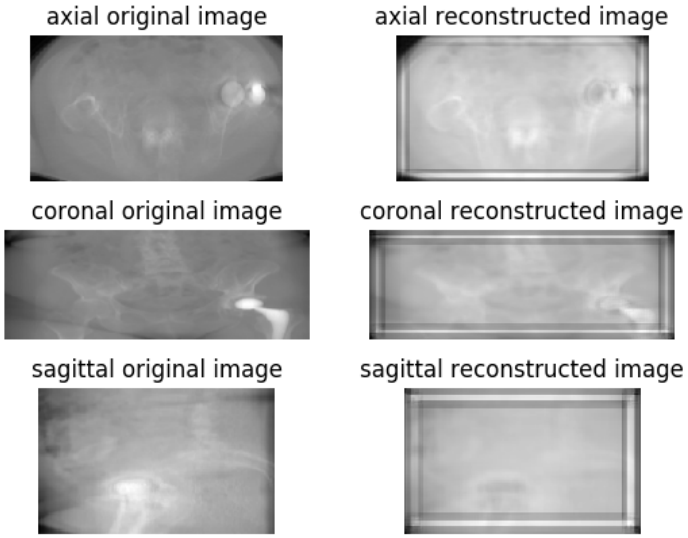
(a)


axial original image
axial reconstructed image
coronal original image
coronal reconstructed image
sagittal original image
sagittal reconstructed image

(b)


axial original image
axial reconstructed image
coronal original image
coronal reconstructed image
sagittal original image
sagittal reconstructed image

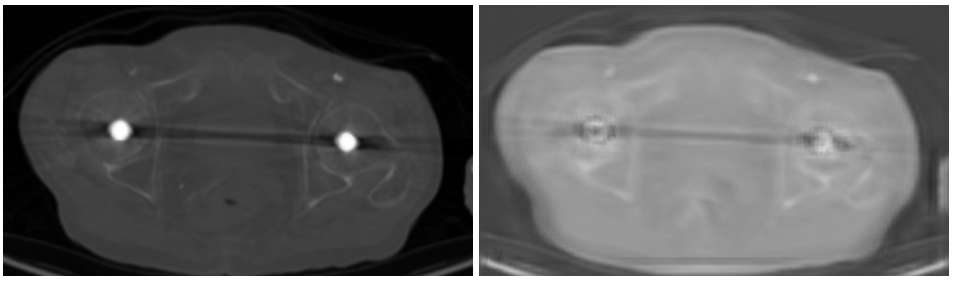
(c)



(d)

Figure B.0: Subfigures (a) – (d) are all of the CT images in the test set and their reconstructed image with the model. To the left in each image, is the original image plotted and to the right the reconstructed image. Each image and reconstructed image is plotted in axial, coronal and sagittal plane.

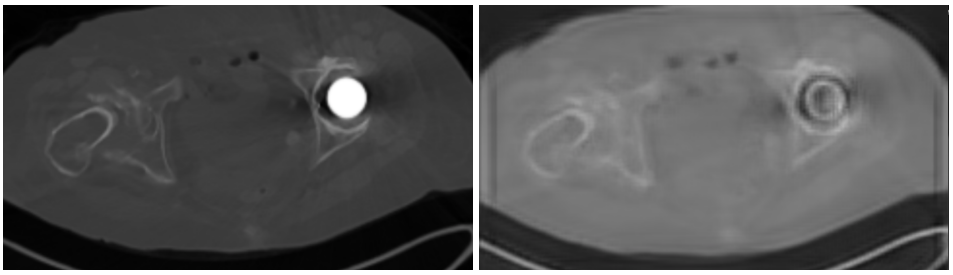
B.2 Slices of the test set



(a) A slice in the original image.

(b) The corresponding slice in the reconstructed image.

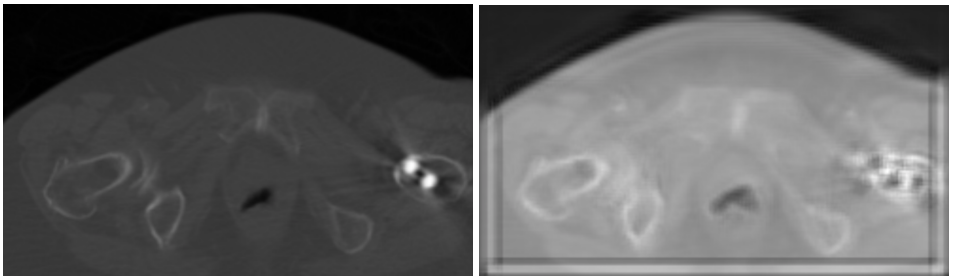
Figure B.1: A slice of the original image and the corresponding slice in the reconstructed image shown in Figure B.0a.



(a) A slice in the original image.

(b) The corresponding slice in the reconstructed image.

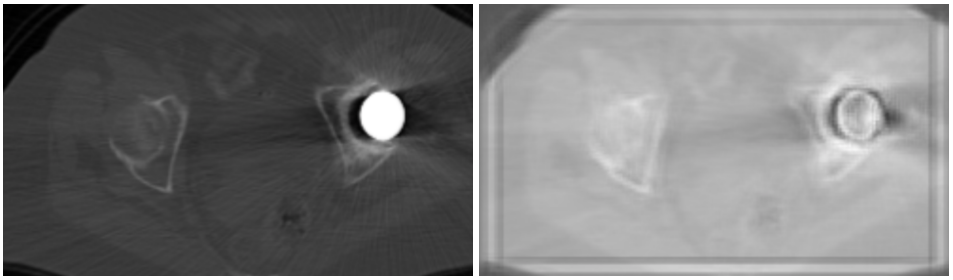
Figure B.2: A slice of the original image and the corresponding slice in the reconstructed image shown in Figure B.0b.



(a) A slice in the original image.

(b) The corresponding slice in the reconstructed image.

Figure B.3: A slice of the original image and the corresponding slice in the reconstructed image shown in Figure B.0c.



(a) A slice in the original image.

(b) The corresponding slice in the reconstructed image.

Figure B.4: A slice of the original image and the corresponding slice in the reconstructed image shown in Figure B.0d.

Master's Theses in Mathematical Sciences 2020:E23
ISSN 1404-6342

LUTFMA-3406-2020

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/