Evaluation of Data Augmentation of MR Images for Deep Learning

Erik Andersson, Robin Berglund

LUND UNIVERSITY

Subject: Mathematical statistics (FMSM01)

Division: Centre for Mathematical Sciences,
Faculty of engineering, Lund University

Supervisor: Andreas Jakobsson

Co-supervisor: Stefan Adalbjörnsson

Examiner: Johan Swärd

Contents

Abstract

Data augmentation is a process to create new artificial data by altering the available data set. It has by many been considered a key factor for increasing robustness and performance of image classification tasks using deep learning methods, such as the convolutional neural network. This report aims to investigate the effect of data augmentation for the segmentation of magnetic resonance (MR) images using neural networks. Geometric data augmentation techniques such as rotation, scaling, translation and elastic deformation are evaluated for different data set sizes. Also, motion artifacts are simulated in MR images as an augmentation to increase segmentation performance on images containing real artifacts. A method for simulating these artificial motion artifacts is developed, where the k-space of the image is altered to simulate patient movement due to breathing.

Results showed that geometric data augmentation improved performance of the network for smaller data sets, but lost importance as the amount of available training data grew bigger. Segmentation performance of images containing motion artifacts significantly increased if the neural network had been trained on a training set augmented with simulated artifacts.

Acknowledgements

We would like to thank Spectronic Medical AB for giving us the opportunity to conduct our master thesis project with them. Special thanks to Stefan Adalbjörnsson for guidance and support during the whole project. We would also like to thank our supervisor at LTH, Andreas Jakobsson.

We would also like to express our gratitude to the Gentle Radiotherapy consortium for allowing access to the MRI data used in this work, constituting a subset of the data acquired during the MR-OPERA study (Persson et al.., Int J Radiat Oncol Biol Phys. Nov 2017). All MRI data was managed with the highest level of confidentiality and security throughout the entire project. All MR images marked "courtesy of Gentle Radiotherapy" reproduced in this report are approved by Gentle Radiotherapy.

Chapter 1

Background

1.1 Introduction

The last few years there has been a paradigm shift within the field of image analysis, where deep neural networks have shown themselves to be extremely effective in image classification and analysis. A neural network is defined by several layers of calculations where the connections between every layer consists of convolutions and activation functions that are trained using a large amount of training examples. One of many applications for this is automatic segmentation in medical images, for example in magnetic resonance imaging (MRI). Every MR image volume consists of many millions of data points which can be used for training the algorithm. The total amount of training data can easily become billions of data points from several image volumes that teach the network for example what is a hip bone and what is the surrounding tissue. Generally, the more qualitative training examples that a neural network has seen, the better it will perform.

However, the limited amount of training data is often a problem in the case of medical images, either because the images are protected for privacy reasons, or because the data simply does not exist. Even if a small number of image volumes contain a lot of data, it might not be well distributed, meaning that the data does not cover all the normal characteristics that are expected to be found. At the same time, it is widely claimed that the performance and robustness of a trained model can be directly linked to the amount of training data that is accessible, and its quality. How can a model be trained to find something which it has barely seen during its training? By using mathematical models that describe the natural variations that can be expected in unseen images, one can expand the training data with artificial training data, and thereby hope to increase the performance and robustness of the neural network.

1.2 Context and problem formulation

This master thesis was conducted at Spectronic Medical AB in Helsingborg. The company provides a software that produces synthetically created computed tomography (sCT) images by using an MR image. This data is then used for radiation therapy planning for prostate cancer treatment. By using the sCT, the correct radiation dosage can be given to the tumor, while minimizing the radiation dose to nearby sensitive healthy organs.

The software provided by the company uses a preprocessing step involving image segmentation, where bone segments are used to align images. Automatic segmentation of bone structures is therefore of great interested for the performance of the sCT algorithm.

Except for being useful in the context of the algorithm used to create sCTs, automatic segmentation of anatomical structures in medical images is interesting for several other reasons. Most medical image data today is processed manually by clinicians. This means that someone with expert knowledge has to look at the MR image in all three dimensions and manually segment different areas. This is obviously a time consuming and by extension expensive task, that is variable depending on the knowledge and experience of the expert conducting the segmentation. It would be of great value if an algorithm could automatically detect and segment risk organs such as the bladder, or pathological structures such as tumors.

In medical applications, an automatic process has to be very reliable. This means that the model created by the machine learning algorithm must perform very well and must be robust to all kinds of data. A neural network's performance and robustness depends entirely on the training data it is using to create its models. The training data needs to represent a large enough variation so that the network can generalize what it learns to new data. Unfortunately, the amount of available medical data is often insufficient.

To expand these insufficient data sets, one can create ”fake” data using data augmentation. By augmenting the existing data in ranges within the normal variation, the networks performance towards ”normal” data will, theoretically, increase. By augmenting the existing data with uncommon variations, the network could learn to perform well on this kind of data which would increase its robustness.

Augmentation of medical data to expand a training set has been done in several studies and has been said to greatly improve performance of machine learning algorithms, such as neural networks. However, much of the research is still very recent and is constantly evolving due to new state-of-the-art algorithms. It is therefore relevant to investigate the effect of data augmentation on neural networks for different data sets and tasks, such as segmentation in MR images.

1.3 Aim

The aim of this master thesis is to evaluate if data augmentation can improve the performance of convolutional neural networks. Another aim is to propose a data augmentation method based on the simulation of motion artifacts in MR images.

Anatomical planes

In medical terms the body is divided into different planes that is useful to know for the purpose of reading this thesis. The different planes can be seen in Figure 1.1 and are described below.

- *Transverse* plane: A horizontal plane that separates the body into a lower and an upper part.
- *Sagittal* plane: A vertical plane that splits the body in a left and a right side.
- *Coronal* plane: A vertical plane that divide the body in a front and a back.

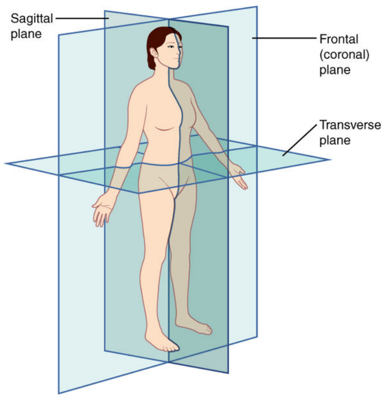


Figure 1.1: The different planes of the human body.[18]

1.4 Data

The primary data set that has been used consists of 50 magnetic resonance image volumes. The MR images has been acquired in hospitals using MR camera systems from two different producers and were taken specifically for the purpose of being used in the algorithm used by Spectronic Medical to produce synthetic CT images. The images are taken around the area of the prostate, from the diaphragm to the middle of the thighs. All data has been fully stripped of personal patient information.

The data has been collected in different hospitals, with different settings on the MR camera, and on two different imaging systems. For a physician, this non-uniform data might not make a big difference for their medical evaluation, however, the large variance in data can make big difference for an automated machine-based system. In Figure 1.2 two different images are presented that were acquired from two different MR camera systems. For example, notice the difference in intensity for the bladder in the middle of the body, and the difference in noise level outside the body.

This comparison also demonstrates the physiological variations between different templates. For example, the shape and size of the bladder varies greatly between templates, as does the shape of the body. The data that was provided for this master thesis was primarily provided in the form of *.nrrd* files, which is a format for representing n-dimensional raster data. In this report, these files will be referred to as *templates*.

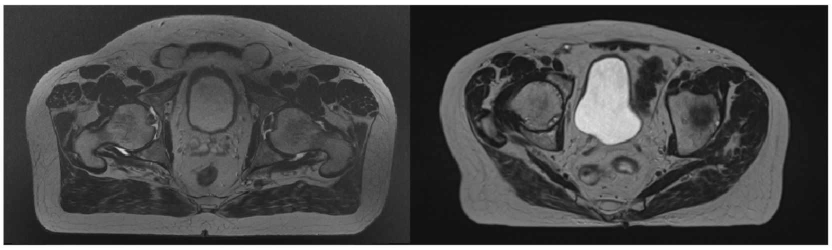


Figure 1.2: Comparison of MR images [1] in the transverse plane from two different camera systems.

The images has either been processed in its original 3D format, or have been converted into 2D arrays in the transverse plane. A 2D image taken from a 3D MR image is commonly referred to as a *slice*.

The data consists of both the MR images, as well as ground truth. The ground truth has been drawn out by experts with anatomical knowledge and consists of up to eight different segments: air, body, pelvis, spine, left femur, right femur, bladder and colon. These segments are also called *labels* or *classes*. For practical reasons, the 2D arrays were zero padded, technically adding a ninth label. Since this label exists for pure practical reasons, it will from now on be ignored when discussing labels. Different experiments used a different number of classes.

In Figure 1.3, an example of a transverse slice together with its labels is presented. The image in the middle shows a ground truth containing three labels, while the image to the right contains eight labels. The different labels are visually separated by different intensities of gray.

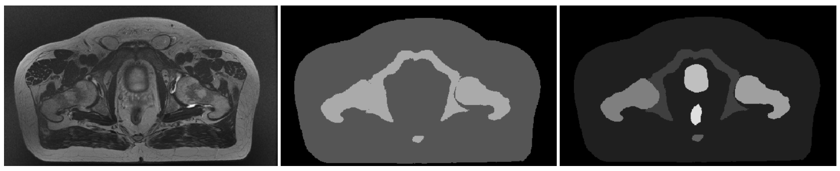


Figure 1.3: A MR image[1] to the left together with two different ground truths. The ground truth in the middle contain three labels and the one to the right contain eight labels.

During this project, primarily 50 different templates have been used for data augmentation, network training and network validation. The 50 templates are 3D volumes, and therefore they each become several slices when shown in 2D. In total, they contain 9424 unique slices, each with one image and one ground truth. One slice of a template has 454 x 466 pixels and there are about 200 slices per template, although the exact number varies. A number of other templates have also been used to validate the segmentation algorithms depending on the experiment. For example, when evaluating motion artifact augmentations, a test set containing templates with motion artifacts has been put together.

[1]Original MRI courtesy of Gentle Radiotherapy.

Chapter 2

Theory

2.1 Magnetic resonance imaging

Magnetic resonance imaging (MRI) differs a lot from most other imaging system and involves many concepts. The full details of how it works is beyond the scope of this thesis, and therefore a summary of the most important and relevant part is made here.

A key part of the functionality of the MRI is the strong magnetic field. This is formed by superconducting wires around the cylinder where the patient lies. When protons are placed inside this magnetic field they will align in one on two directions: either along the direction of the magnetic field or in the opposite direction. The chance for aligning with the magnetic field is slightly higher and a so called *net magnetization* is created by the slightly bigger proportion of protons aligned with the magnetic field. If not in a magnetic field, the protons are ordered randomly in all directions.

This is part of the main concept of MRI, that the protons in the nuclei of hydrogen atoms can act as tiny magnets. The proton also spins around its axis which also contribute to the physical properties that allows us to create images. Beside these magnetic properties and spin of the proton, it also precesses, meaning that the axis itself rotates. This can be likened with a spinning top toy that not only spins around its center, but where the center also starts to wobble in circles. The frequency of this precession increases with the strength of the magnetic field.

As a final action to create an image, radio frequency (RF) energy is needed. The RF must match the frequency of the precession of the protons, which in a normal magnetic field of 1.5 Tesla will be about 64 MHz. When an RF pulse is sent, this energy will be absorbed by the protons and the previously mentioned *net magnetization* will as a result move away from the direction of the magnetic field. How far away it will tilt depends on the strength and duration of the RF pulse. The *net magnetization* will begin to flip back to being parallel with the magnetic field, but it will take a different amount of time for different tissues. Utilizing this difference for imaging is called T1. The RF pulse also makes the precession of the protons become in phase so that they precess in the same way. When the pulse is gone, they start to become out of sync, and they are soon totally unsynchronized again. The speed of this to occur depends on several things, including the type of tissue. Using this property to acquire an image is called T2 [22].

By applying different gradient magnetic fields, it is possible to change the frequency and phase of each unique location in the image. Over the course of the MR scan, each location will have experienced a unique combination of field variations, which means that they are encoded with different phases and frequencies

Which direction that encodes for phase and which encodes for frequency can be arbitrarily chosen. In a transverse slice, the y-axis is often chosen as the phase encoding direction and the x-axis is chosen as the frequency encoding direction [17].

The signal received from these processes is collected in frequency space, known as *k-space*.

2.1.1 K-space

To understand the MR imaging process, it is fundamental to understand k-space, which is the data matrix directly obtained by the magnetic resonance scanner. By using the inverse Fourier transform, the k-space matrix can be reconstructed into the actual image depicting relevant physiological structures. In Figure 2.1 an example of a MR image and its k-space representation is shown. The k-space representation was created by Fourier transformation of the original MR image.

K-space represents the spatial frequency information of the image. This frequency describes how the features of the image changes depending on its position. Objects containing many edges will therefore have many high spatial frequencies in k-space, while large smooth objects will have an abundance of low spatial frequencies. High frequency components are mapped to the periphery of the k-space matrix, while low frequency components are mapped to the center of k-space. The center of k-space therefore contains the bulk of the information in the image [19].

It should be noted that due to the relationship between k-space and the image, every point in k-space contains some information about every data point in the image domain. Removing a line in k-space, for example, will therefore affect the whole image and not just the corresponding line in the image. Theoretically, the whole image can be reconstructed from just half of the k-space matrix. This is possible due to the fact that k-space can be seen as a collection of cosine and sine waves of different frequencies, which are symmetrical and anti-symmetrical.

K-space trajectories

How to fill the k-space is called the k-space trajectory and can be done in several different ways. The most common strategy is the Cartesian trajectory which samples the k-space lines in a rectangular grid [32], [33]. This way allows for efficient reconstruction of the image using the algorithm fast Fourier Transform (FFT). Some other trajectories, with self-explaining names, are the radian trajectory, the spiral trajectory and the zig-zag trajectory. Different trajectories are related to different artifacts, image collecting time as well as other factors, and should be chosen depending on the purpose of the MR scan.

There are also other techniques for sampling the k-space faster, by for example collecting several lines at the same time. This can be done with a turbo spin echo sequence, also called fast spin echo. A number of echoes will be produced by one excitation pulse, each sampling a line of k-space. The number of echoes created by each pulse is called the *echo train length*, or *ETL* for short [22], [19].

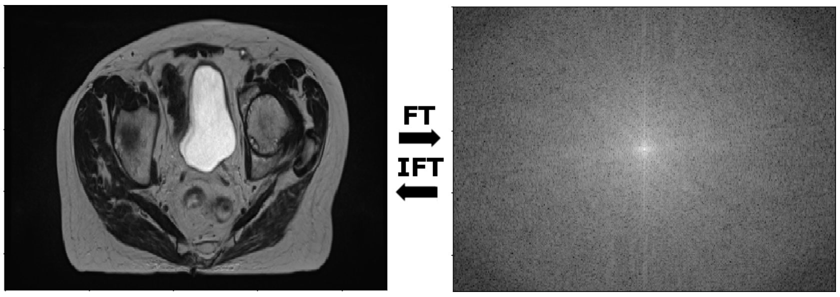


Figure 2.1: An MR image[1] and its k-space counterpart.

2.1.2 Motion artifacts

Artifacts are something that occurs in all images and due to the importance of medical images they are of great interest within this field. They can arise from hardware limitations or from the software collecting the image. MR images can contain a range of different artifacts that can appear as stripes, ghosts, spikes, shadows and distinct lines in the image. However, only one artifact will be further discussed in this thesis: the motion artifact.

Motion artifacts are one of the most common artifact present in MR images, and has been a challenge since MRI was introduced. The different kinds of movements that causes this kind of artifact can be categorized as rigid body motion, elastic motion and flow [32]. In this work, the focus will lie on rigid body motion, also called bulk motion. Examples of this kind of motion is movement of the diaphragm due to breathing, or movement of the head. Mathematically, the former can be modeled with a translation in one dimension, while the latter requires six degrees of freedom.

Acquisition of adjacent points in the frequency encoding direction is relatively short, while it is much longer for the phase encoding direction. This entails that the phase encoding direction is much more susceptible for motion, since most patient movement is faster than the data is encoded to the k-matrix. Object movement during acquisition of data in the phase encoding direction will lead to inconsistencies in k-space, since k-space lines will be sampled when the object has moved position. When the k-space is Fourier transformed, artifacts will appear in the resulting image.

Motion artifacts often appears as *ghosts* in the image, which are partial or full copies of the object being scanned along the phase encoding direction. If the motion affects the k-space in a way so that the lines are affected periodically, coherent ghosts will form in the image. If every fourth line is altered, four ghosts will appear in the image. If the lines are affected more randomly, the artifacts will look more like incoherent stripes.

In Figure 2.2, two examples of MR images containing motion artifacts are given. The images depict transverse slices in the area around the pelvis. The artifacts, which appears as thin stripes in the Y-direction is likely to have been caused by breathing.

[1]Original MRI courtesy of Gentle Radiotherapy.

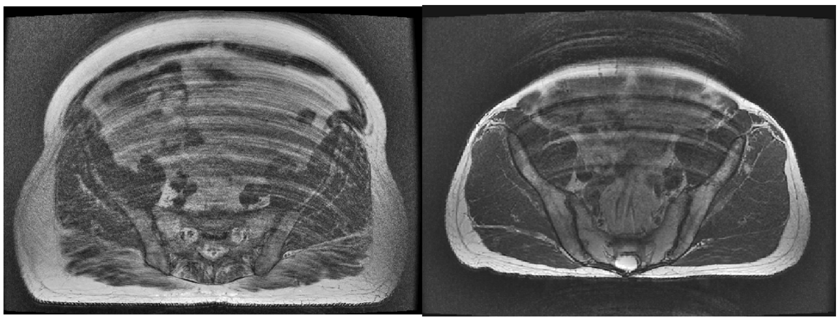


Figure 2.2: Two examples of breathing artifacts in MR images[1].

2.2 Machine learning and deep learning

A machine learning algorithm is an algorithm that is able to learn from the available data and use this ”knowledge” to become better at its task. One such task is *classification*, in which the computer program is asked to specify which of *k* categories some input belongs to [10]. A common classification problem is object recognition in images, where the task is to identify which of several classes an object belongs to. Another sort of classification is pixel-wise segmentation of images, where the algorithm is asked to assign a category to every pixel in an image.

When conducting machine learning experiments, it is common to partition your data set into three parts; training, validation and test set. The training set is the data set which the algorithm experiences and uses to build its model, while the test set is the actual data of interest, which the model is supposed to perform well on. The validation set is used to optimize the model during training and to tune the hyperparameters of the algorithm, something that will be discussed further in section 2.2.5.

There are two types of learning processes: *Unsupervised* and *Supervised*. In unsupervised learning, the algorithm is fed data and is expected to find useful patterns and properties of the data set. In supervised learning, each data point in the training set is associated with a *ground truth* or *label*. For example, in a image segmentation task, the ground truth is an image containing the different segments. By observing the connection between data point and ground truth, the algorithm can learn to predict labels for previously unseen data.

2.2.1 Neural networks

A kind of machine learning algorithm that has shown itself to be proficient in many tasks is the *feedforward neural network*. These algorithms consist of several different functions connected to each other, where each function is referred to as a *node*. The nodes are structured in different layers, as can be seen in Figure 2.3. The first layer is called the *input layer*, the last an *output layer* and the layers in between are referred to as *hidden layers*. An input x goes from the input layer, through different computations in the hidden layers which delivers an output through the output layer. During training, the neural network learns how to use the nodes to in way that minimizes a *cost function* The learnable parameters are called *weights* and *biases*, which are updated in each step of the algorithm. Fully connected layers,

[1]Original MRI courtesy of Gentle Radiotherapy.

like the one presented in Figure 2.3, have limitations when it comes to processing high-dimensional data like images due to something often referred to as *the curse of dimensionality* [21]. In an image with for example 1024 pixels, the input layer would have to treat 1024 individual parameters. This number becomes unmanageable as the image data scales up. A breakthrough in the processing of image data using neural networks have come with the *convolutional neural network*, described in more detail below.

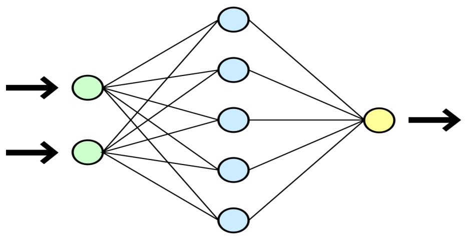


Figure 2.3: Example of the architecture of a fully connected neural network with only one hidden layer [31].

2.2.2 Convolutional neural networks

Convolutional neural networks, or *CNNs*, are a special kind of feedforward neural networks that is widely used with data sets consisting of images. This kind of network uses convolutional layers as their main component, which are essentially sets of convolutional filters or kernels. Each kernel only processes a subsample of the input data and creates so called *feature maps* which detects features in the image. The features detected could for example be edges and curves in the image. In Figure 2.4 the structure of a typical CNN network is presented. CNNs usually also contains *pooling* layers, which are layers that are used to reduce the number of parameters and spatial resolution in the network, and therefore to also control overfitting.

The performance of CNNs on classification of image data has been shown in a number of studies [14], [28], [26], and is currently considered state-of-the-art for image classification.

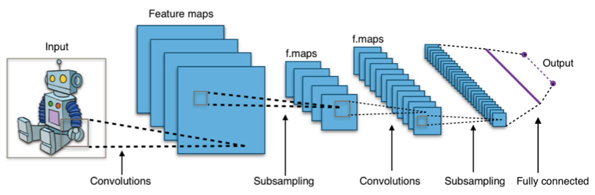


Figure 2.4: Example of the architecture of a typical convolutional network [29].

2.2.3 Gradient descent

Most machine learning algorithms are based on an optimization problem, for example to minimize a certain function $f(x)$. In machine learning context, this function is referred to as the *cost function* or *loss function*. A common cost function used in neural networks is the negative log-likelihood, which is referred to as the *cross-entropy* between the distributions of the model and the training data [10].

Gradient descent starts with a set of parameters and then iteratively moves towards values of the parameters that minimizes that cost function. The minimization is done by moving small *steps* in the opposite direction of the derivative of $f(x)$. How big of a step that is made is called the *learning rate*, and will be discussed further in section 2.2.5. Before a step can be made and the weights of the model can be updated, the cost and gradient of the entire training set must be evaluated.

Stochastic gradient descent

In most practical cases, it is too computationally demanding and ineffective to evaluate the entire training before updating the model. A solution to this problem is the *stochastic gradient descent*, which uniformly draws a *mini batch* from the training set for each step and determines the next step by evaluation of the batch. The mini batch can be as small as one data point.

2.2.4 Regularization

Machine learning algorithms need to be regularized so they can generalize to data other than what it has trained on. Goodfellow et.al [10] provides the following definition of regularization in the context of learning algorithms: *"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error."*. A common regularization approach is to add a norm penalty to the cost function, called L_2 parameter regularization. In neural networks this regularization is implemented by decaying the weights of the network.

Another regularization technique commonly used for neural networks is called *dropout*. This method aims to stop overfitting by randomly deactivating, or dropping out, units in the network [27].

2.2.5 Hyperparameters

Parameters that are set prior to training and that is not learned by the machine learning algorithm are called hyperparameters. *hyperparameters* [10]. Hyperparameters need to be tuned depending on the network and the data it is training on. This process is called *hyperparameter optimization* and should be done as a first step when using a machine learning algorithm for a task. This process involves setting the parameters to different values, training the network with them and evaluating the created model on the validation set. The correct values are searched for either manually, through a grid search or through random search [6].

One of the most important hyperparameter is the *learning rate* [5]. This parameter tells the optimizer how far to move the weights in the direction of the gradient. Smaller steps lowers the risk of missing the global minima but also increases the time it takes to converge. A common practice is to take large steps in the beginning of the training and then smaller as the learning process evolves. This is called *learning rate decay* and is often a hyperparameter. Another hyperparameter related to the learning rate is *momentum*, which is used to prevent the algorithm getting stuck in a local minima. A higher momentum means a larger fraction of the current weights will be used to update the new weights.

Parameters that are used for regularization of the network, such as weight decay and dropout, are also often used as hyperparameters.

2.2.6 Inference

Inference is the process of using the trained model on new data to perform the task it was trained for. For example, in an image object classification task the model would be fed new data and predict which class each object belongs to.

2.2.7 Overfitting and underfitting

The idea of machine learning is that the algorithm should learn from a training set and then perform well on a test set of unseed data. Performing well on the training set but much worse on the test set is referred to as *overfitting*. The algorithm have then tuned itself to fit the training set to a degree that it has lost its ability to generalize. *Underfitting* is a term used for when the learning algorithm is not performing well enough on the training set. Usually overfitting is the bigger challenge of the two and is combated with regularization and data augmentation.

2.2.8 Imbalanced data

In classification tasks there is often different amount of data points belonging to each class. This can lead to a model that is optimized in a way so it will work best on the dominating class while ignoring small classes. This is called *imbalanced data*, and is a problem that can be countered by weighting each class by a factor. This will make all the classes important to the decision making of the network and ensure that smaller classes will not be ignored.

2.2.9 Performance metrics

Several different performance metrics can be used to calculate performance of a neural network and varies greatly depending on the specified task of the network. The *Dice coefficient* gives a measure of the overlap between two segments, which makes is a popular metric to compare a networks segmentation with the ground truth. It is calculated by using formula 2.1.

$$Dice = \frac{2|X \cap Y|}{|X| + |Y|} \tag{2.1}$$

X and Y denotes the prediction and the ground truth respectively. In the case of segmentation, one can see it as dividing the overlapping area between the prediction and ground truth with the sum of their total area. This value is then multiplied with 2 to give a measure between 0 and 1.

The segmentations can also be evaluated visually to estimate the performance of the model. While this method is subjective, it gives more insight into the segmentations made and eventual problems with it than a simple numerical value.

2.2.10 Cross validation

Machine learning algorithms generally builds better models if trained on more data. However, the model has to be validated on data that is has not trained on. Partitioning the data set into fixed sets for training and testing might not be a good idea if the data set is small. This would give a high uncertainty regarding the result since it would only have been tested on a small test set. Another solution would be to train multiple models with the same settings but with different training and test sets. Then more or even all of the data can be used for training and an average error can be calculated. One common way is to use the k-fold cross validation. This divides the data set into *k* subsets and using one such subset for testing, and the rest for training. This is then repeated for all subsets. Other types of cross validation also exist like repeated random sub-sampling, where the training and test set are randomly drawn from the data set, and then repeated for as many times necessary. The drawback of cross validation is the additional cost of resources [10].

2.3 Data augmentation

One of the key aspects in convolutional neural networks is data. Having access to large data sets gives the network a better chance of becoming better at whatever task it is supposed to do [15]. Everything the network learns will come from the available training data, and it is therefore important that it contains all major aspects needed to make a decision. Having too little data results in overfitting [15]. For example: if you want to create a network that can detect a cat in an image, it is important to know that a cat can be seen in almost every environment. This means that the cat can be seen in snow, on grass, on the road, on a wooden floor and so on. If all the training data would come from pictures made in a studio environment with a white background, this would prevent the network from learning all the necessary parameters that exist in the real-world application. Therefore, it is in general better to have bigger data sets, which will contain more versions of the same type of object [24], [10]. Apart from the different backgrounds of the cat, it may contain different angles, lightning, or colors.

Image data is especially suitable for augmentation since many of the variations are easily simulated. Several works have been presented were performance of a machine learning algorithm has increased after the image data was augmented [25], [3], [20]. There are some augmentations that are widely used in many types of image applications, and some that are more specialized for a particular problem. Some of the more general augmentations are rotation, flipping, cropping and adding noise.

Rotation is very simple and is suitable for most applications. If you rotate an image of a cat, the image would still contain a cat. This can generate many new images from one original. Flipping or mirroring is in many ways similar to rotation. It preserves the information and label but is limited in how many new images it can create.

If much of the training data is in some way standardized so that the object of interest is always in the same position, it might be a good idea to augment using translation to move the object to different places in the image. Otherwise the network might favor one position over another, which might not be desirable.

Scaling is another way to augment images. It can be useful when the desired object can appear in different sizes, but still with the same characteristics.

The rotation, translation and scaling all belong to a group of transformations called *affine transformations*. One way of distinguishing an affine transformation is that it preserves parallel lines in an image. To mathematically perform these affine transformations, one can multiple a vector of the old coordinates with a 3x3 matrix, A, as seen in equation 2.2. Depending on how the matrix A looks, the transformation can become for example translation or rotation. Other affine transformations can be seen in Figure 2.5.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.2}$$

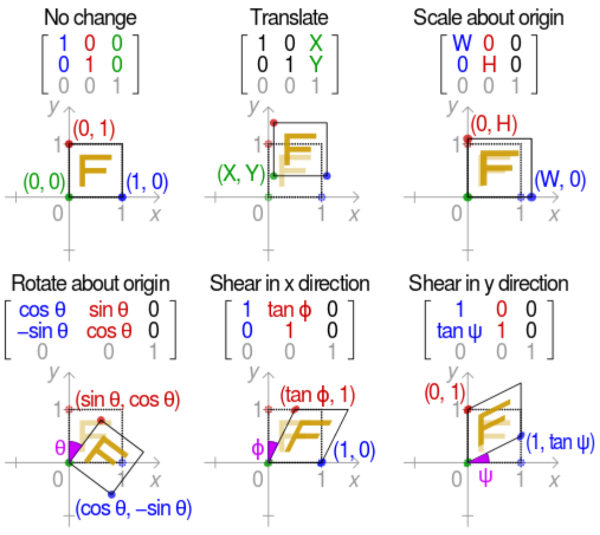


Figure 2.5: Summary of some affine transformations [30].

Another type of data augmentation is elastic deformation. This is more specialized and might not be a good idea for all applications. This augmentation involves deforming the image in ways that may destroy lines, meaning that it is not an affine transformation. It can be seen as dividing an image into a grid of squares and then transform each square in some way. One way is to randomly move the corners of the small squares while stretching the image inside, as seen in Figure 2.6. The local deformation of each small square can be made in different ways. If the deformation of each small square is made by an affine transformation, such as a combination of shearing and rotation seen in Figure 2.5, the transformation as a whole can be called *piecewise affine* [7].

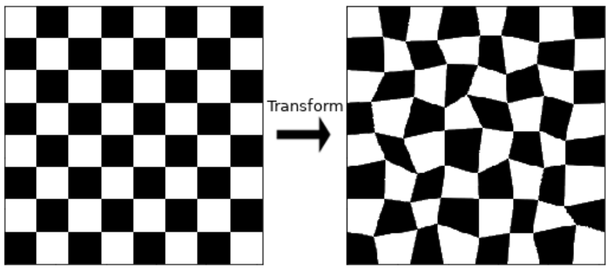


Figure 2.6: Example of elastic deformation on a grid.

It is important that the data augmentation chosen is fitting for the specific type of task. For example, rotating might not be a good idea if the network should be able to differentiate between a 6 and a 9, or if it should know left from right on street signs [15]. Similar things can be said about many types of augmentation, for example when the object to be recognized have many straight lines it may be unwise to use elastic deformation.

2.3.1 Online vs Offline augmentation

Augmentations can be made either *online*, meaning that it is made during training and that the augmented data does not exist afterwards, or *offline* meaning that a new complete data set is created before the training starts. They both have different advantages. For example, offline training requires more storage available beforehand, but it also makes the training process faster. Offline augmentation has a fixed number of possible augmented images, whereas the online training can generate a larger number of unique images if training set is iterated over multiple times.

Chapter 3

Related work

It is worth mentioning that while many of the ideas behind neural networks and data augmentation are not new, much of the research is very recent. Many articles have been written the past year and the field is evolving quickly. This section describes some of the work that is related to the subject of this thesis.

3.1 Networks

Two different neural networks available open-source were used in this thesis. The networks are described briefly below, but the interested reader should see the full articles corresponding to each network for more detail.

3.1.1 U-Net and tf_unet

U-Net is a neural network architecture created by Olaf Ronneberger, Philipp Fischer, and Thomas Brox, designed for the purpose of segmenting biomedical images [23]. The network has been shown to perform very well in segmentations tasks of electron microscopic stacks using a limited amount of training samples.

The network consists of a contractive path together with a symmetric expansive path which gives the network a U-shape as can be seen in Figure 3.1. The contractive part applies convolutions and max pooling operations at each step, which downsamples the input and feature maps that are created. At each step in the expansive path the feature maps is upsampled and then convolved. To map each feature vector to the desired number of classes a final convolutional layer is applied.

A generic version of U-Net, called *tf_unet*, has been implemented in Tensorflow by Joel Akeret, and the entirety of the implementation has been published on the code-sharing platform *github.com*. Joel Akeret et.al used this implementation to classify clean signals and RFI signatures acquired from a radio telescope [2]. However, the network is designed to be generic and can be used for any segmentation

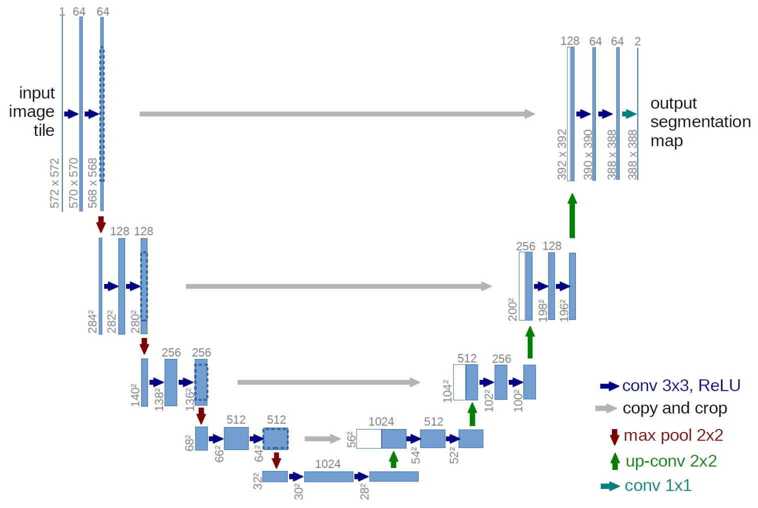


Figure 3.1: Architecture of U-Net [23].

task.

The network takes 2D matrices as input. In this project, this implementation has been altered to fit the data and resources worked with. For example, the network was adjusted to take whole images as input instead of working with smaller patches of the image. The cost function used with this network implementation was cross-entropy.

3.1.2 NiftyNet and HighRes3DNet

NiftyNet is an open source convolutional neural networks platform for medical image analysis and image-guided therapy [9]. The platform is built for use with Tensorflow. Several different network structures has been implemented in NiftyNet that can be used to segment medical images.

One such network is *HighRes3DNet*, which is a NiftyNet implementation of a neural network solution proposed for volumetric image segmentation in [16]. Li, W. et al.. used this network to segment neuroanatomical structures from MR images of the brain and achieved results that could be compared to state-of-the-art segmentation networks.

The network consists of 20 convolutional layers that uses dilated convolutions which create feature maps of different scales. Residual connections between each pair of convolutional layers enables fusion of features from different scales. See Figure 3.2 for a full illustration of the network's architecture.

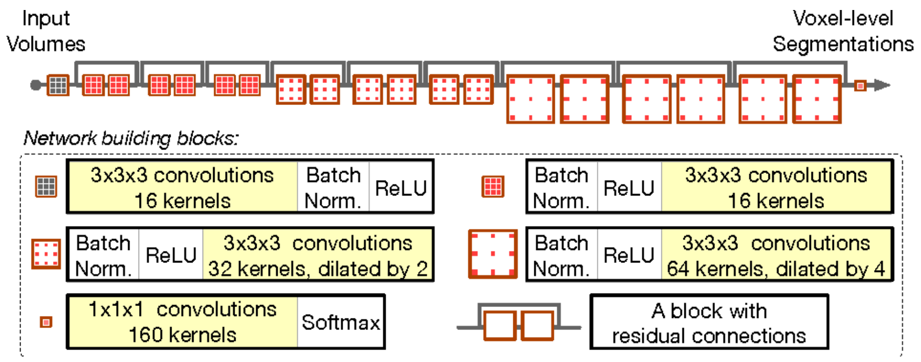

Input Volumes
Voxel-level Segmentations
Network building blocks:
3x3x3 convolutions 16 kernels
Batch Norm.
ReLU
Batch Norm.
ReLU
3x3x3 convolutions 16 kernels
Batch Norm.
ReLU
3x3x3 convolutions 32 kernels, dilated by 2
Batch Norm.
ReLU
3x3x3 convolutions 64 kernels, dilated by 4
1x1x1 convolutions 160 kernels
Softmax
A block with residual connections

Figure 3.2: Architecture of HighRes3DNet [16].

3.1.3 Data augmentation

In their article [25] , J. Shijie et al. describes the impact of different data augmentations on an image classification task. Their experiments are made using a number of different sized data sets with small images, of varying sizes down to 32 x 32 pixels, with a number of different augmentation techniques. These include flipping, cropping, color shifts and adding noise. Their result is that almost all augmentations had a positive effect on the classification accuracy.

Biomedical data augmentation

Many articles that involve medical images and neural networks shortly discuss the special challenges that arise in this area compared to many other non-medical fields. The main problem is lack of data, which result in very small training sets and contribute to overfitting.

In *GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification* by Frid-Adar et al. [8] it is discussed that most researchers try to solve the problem of small data sets using data augmentation. This is usually made by using geometrical transformations such as rotation, scaling, translation and flipping. Aside from discussing this, the authors suggest a method called *Generative Adversial Networks*. This can be described as one network that generates fake images, and one that evaluates them to see if they are good or bad. The article shows that this approach could improve the performance when using a small training set.

Due to the lack of sufficient data, the work by Ronneberger et al. [23] also use excessive data augmentation as a way to increase the network's invariance and robustness properties. Rotation, shift and deformation augmentations are used on the training samples for this purpose. The authors argue that elastic deformation augmentations are particularly important in biomedical segmentation tasks, since the most common variations in tissue is deformations.

Another article by Asperti et al. [3] use similar data augmentations as the previous articles. Rotation, translation, shearing and scaling are used in their experiments. Like in many other articles about medical uses for neural networks, they conclude that the data augmentation helped to increase the performance when using a small data set.

Havaei et al. [11] tried data augmentation on brain tumor segmentation on MR images with a convolutional neural network. In contrast to many others they state that the data augmentation did not improve the overall accuracy of their model.

3.2 Modeling of motion artifacts

Modeling of artifacts in MR images is relevant for several reasons. Knowing how artifacts occur makes it easier to prevent them from being created and can also aid in the removal of them in the post processing of the acquired image.

Since motion artifacts occur due to inconsistencies in the acquired k-space, it is also possible to simulate motion artifacts by introducing inconsistencies in an image's k-space representation. A simple example of this is done by Moratal et al. [17], where phase encoding lines along the y-axis is canceled out. A self-produced example of this type of artifact simulation can be seen in Figure 3.3. Here, every fourth line along the y-axis has been set to zero. Because of this periodicity, four ghosts are present in the simulated image.

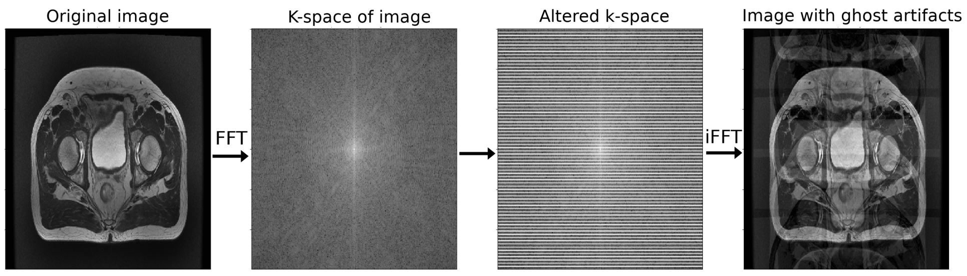


Figure 3.3: Simulated ghosting artifact in MR image [1].

Motion of an object can mathematically be modeled as a translation. This produces the possibility to simulate motion in an MR image simply by translating the imaged object a certain distance. If the acquisition time and trajectory of lines in k-space is known, it would be possible to exactly simulate the motion artifacts produced by a certain motion.

An often more desirable goal is to correct images with artifacts rather than introducing them [13]. In their paper *Matrix Description of General Motion Correction Applied to Multishot Images* [4], Batchelor et al. proposes a method to correct images with motion artifacts by assuming a motion model and make alterations to k-space according to that model. The authors simulate motion by random affine transformations. Motions in MR images are also simulated with affine rotations and translations on a digital phantom by M. Zaitsev et al. in their article where the origin of motion artifacts is discussed [32].

[1]Original MRI courtesy of Gentle Radiotherapy.

Chapter 4

Method

4.1 Software and hardware

4.1.1 Python

All code was written in the *python* programming language, version 2.7. Some open source libraries for python that were used extensively during the project is described below.

TensorFlow

TensorFlow is an open source framework implemented in Python for machine learning algorithms [1]. It allows execution of algorithms such as neural networks and an interface for visualization of the training process.

imgaug

imgaug is a python library created by Alexander Jung for augmenting data and is designed especially for machine learning experiments [12]. The library provides a range of different augmentation techniques on both 2D and 3D matrices.

4.1.2 Hardware

The main hardware that has been used for implementation of code and for training the *tf_unet* network has been two laptops equipped with one NVIDIA GeForce GTX 1050 graphic card each. The *High3DResNet* was trained on a computer equipped with a NVIDIA GeForce GTX 1080 Ti graphic card.

4.2 General methodology

The bulk of the experiments conducted was done using the Tensorflow implementation of U-net, *tf_unet*, but *HighRes3DNet* implemented in *Niftynet* was also used in experiments regarding motion artifacts.

Execution of training

Since the training process took much of the computers resources, the training was made during nights and weekends. Scripts were therefore written to easily allow multiple trainings with different settings to execute from one script. The same was done for scripts that used the trained models to predict on new images and calculate various performance metrics. Most of the code written for this project has been for scripts aiming to create an efficient pipeline to conduct the experiments and extract the results.

Continuously during the project data logs were created to document the training and inference. These logs were helpful when looking at the data after training to see what parameters and settings were used.

Training length

The first models trained was done so with an excessive amount of iterations to ensure that the models were fully trained. The learning curves, see figures 4.1 and 4.2, of these models were then studied to decide after what point in time the model did not improve. Mainly the accuracy and the loss curves were examined to make this decision.

For the *U-Net* the number of iterations was set to **40 000**, which took approximately **8 hours** to run. For the *HighRes3DNet* network the number of iterations was set to **10 000**, which took approximately **3 hours** to run.

Downsampled data

Because of the lengthy training time of 8 hours for the *U-Net*, it was decided that the data set used should be downsampled to allow for more experiments. The 2D images used were scaled from 454 x 466 pixels to 254 x 266 pixels. The exact number was chosen because it fitted the size requirements for U-net architecture and were approximately half of the original size in each direction. This reduced training time to approximately 3 hours.

4.3 Hyperparameter optimization

The specific hyperparameters for the U-Net implementation were studied and the following parameters were tuned:

- learning rate
- momentum
- learning decay rate
- regularizer
- dropout

To save time, a fully random search for parameters was not conducted, but rather a more structured grid search. The interval of values to search was limited by educated guesses and a reasonable amount of values in this interval was used to create different models. Because of the importance of the learning rate [5], this parameter was the first to be investigated and set. Thereafter several models were trained with different values and combination of the hyperparameters and then compared against each other.

One way to compare the effect of different hyperparameters was to use *Tensorboard*, which is a visualization tool built into TensorFlow. Figure 4.1 and Figure 4.2 show information from two different training runs. The top three plots in the figures display the dice coefficient for the different labels, and the left bottom plot displays the overall accuracy. Note the top right plot, which shows that the dice coefficient for bone, is much lower in Figure 4.1 than in Figure 4.2. One can also see that the loss function, in the down right corner of both figures, has stagnated at a non-zero value for the training run in 4.2 already after 15 000 iterations. This indicates that the network stopped learning at that point of time. The loss function in Figure 4.1 is continuously decreasing towards zero, indicating a better training run.

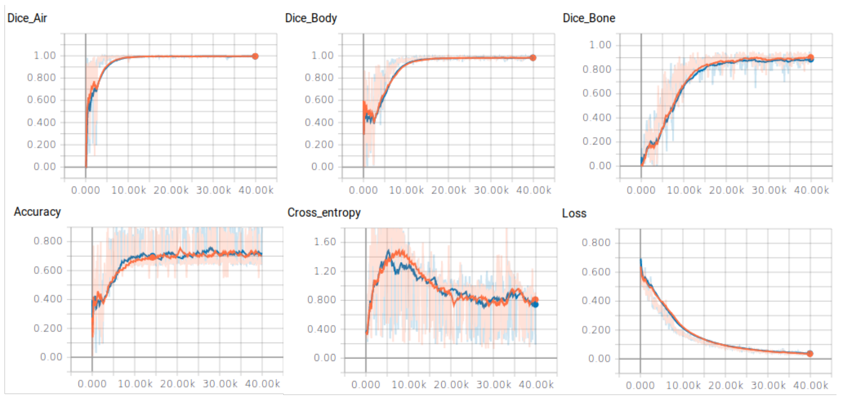


Figure 4.1: Training information for a training run that resulted in a bad model. The blue line represents training data and the orange validation data.

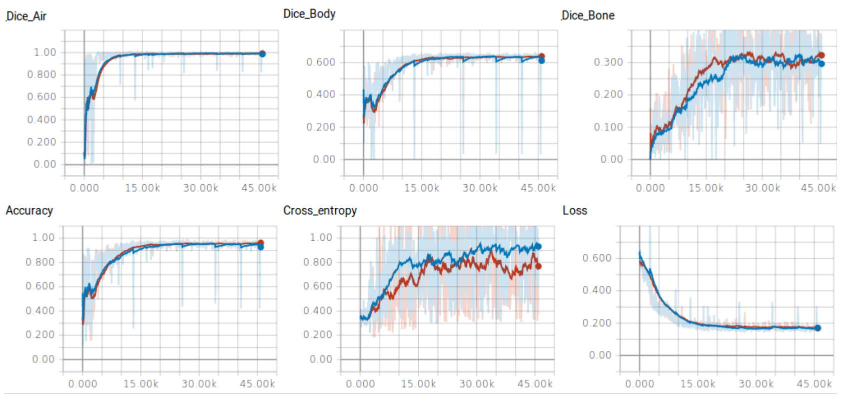


Figure 4.2: Training information for a training run that resulted in a good model. The blue line represents training data and the red validation data.

The hyperparameters that were set for the first experiment can be seen in Table 4.1. After the first experiment, hyperparameter optimization was continued and the final hyperparameters, that were used for all other experiments, were set to the ones in Table 4.2.

Hyperparameter	Value
learning rate	0.0050
momentum	0.9500
learning decay rate	0.9930
regularizer	0.0001
dropout	0.9000

Table 4.1: Early hyperparameters.

Hyperparameter	Value
learning rate	0.0050
momentum	0.9900
learning decay rate	0.9930
regularizer	0.0001
dropout	1.0000

Table 4.2: Final hyperparameters.

4.4 Data augmentation

4.4.1 Geometric data augmentation

For the geometric augmentations of the MR images, both existing python libraries have been used and new algorithms have been developed. The library *imgaug* [12] created by Alexander Jung, which consists of augmentation methods especially designed for machine learning experiments, has been particularly useful.

By investigating earlier work, it was decided that the standard augmentations to be tested in this project would be rotation, scaling, translation and elastic deformation. These were made online during training and implemented as a function that was called during the pre-processing of data during training. The function takes an image and ground truth labels as input and outputs augmented versions of them. A stochastic parameter determines whether or not to perform the augmentation.

The augmentations were mostly made by using functions from the *imgaug* library, with some additional code added to make sure that the data structures of the input images and the provided functions matched.

Geometric augmentations were used separately to train several models whose performance was evaluated. However, the geometric transformations were later split into two groups. One group, named *affine augmentations*, consisted of rotation, scaling and translation. The affine augmentations were used in such a way that one, two, or all three of them were applied to the input data. A simple description of this augmentation can be seen in Algorithm 1. The other augmentation was elastic deformation, which was investigated separately since the article that presents U-Net [23] claims that it was a particularly important augmentation.

Algorithm 1 Affine augmentation

```
Input: Image to be augmented $I$, ground truth to be augmented $GT$ and chance for augmentation $p$
 1: Generate random number $r$ in range (0,1)
 2: if $r < p$ then
 3:     Generate random number $rand$ in range (1,7) and apply one of the following transformations to $I$ and $GT$.
 4:     if $rand == 1$ then
 5:         Translation
 6:     if $rand == 2$ then
 7:         Rotation
 8:     if $rand == 3$ then
 9:         Scaling
10:     if $rand == 4$ then
11:         Translation and rotation
12:     if $rand == 5$ then
13:         Translation and scaling
14:     if $rand == 6$ then
15:         Rotation and scaling
16:     if $rand == 7$ then
17:         Translation, rotation, and scaling
18: return Augmented $I$ and $GT$
```

To decide on reasonable parameters, such as rotation angle and scaling percentage, images were augmented and visually evaluated. Two conditions were set: all anatomical structures had to be preserved inside the borders of the image and the anatomical structures should not be deformed unrealistically. The ranges of the parameters are summarized below.

- The image was translated randomly in both x-axis and y-axis in the ranges limited by the image borders.
- The rotation angle was randomly set to a value in the continuous range of (-15,15) degrees.
- Scaling was randomly set to a value from 0.9 to the maximum scaling that could be conducted without breaking the border condition, but never exceeded 1.2.
- The elastic deformation was made randomly to a degree that the anatomical structures still remained realistically intact.

Example of all the geometric augmentations can be seen in Figure 4.3.

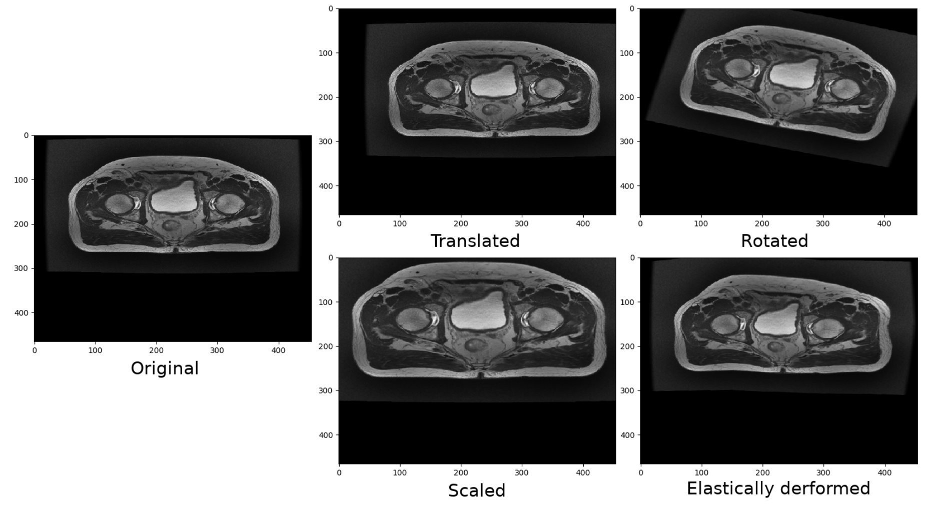


Figure 4.3: MR image[1] and augmented version of it. Original image to the left, augmented versions to the right.

[1]Original MRI courtesy of Gentle Radiotherapy.

4.4.2 Simulation of motion artifacts

A literature review was conducted to investigate how breathing could affect the acquisition of MR image to the degree that motion artifacts appeared. A number of different methods to simulate breathing were tested to gain an understanding of the consequences of altering k-space.

Finally, a process was developed that was based on the Fourier transformation of an image to k-space, alteration av k-space lines in the phase encoding direction, and transformation of the k-space back to an augmented image. In this case the phase encoding direction was set along the y-axis in a transverse slice.

This process was developed into a simulation program with three different modes of complexity. These modes were named *Simple*, *Advanced*, and *Very advanced*. The least advanced mode simply replaces certain lines in the image's k-space with zeros [17], either periodically or randomly. The mode *Advanced* simulates the breathing movement using scaling and translation of the input image in the Y-direction, and then replaces lines in the original image's k-space with the simulated image's k-space. This is supposed to imitate the vertical movement of the stomach during image acquisition. To make sure the position of the patient's back stays in the same place in the simulated image as in the original, the image is translated in the negative y-direction after scaling.

The *Advanced* mode treats all slices in the image volume the same and will therefore create motion artifacts at the legs as well as as the stomach. The *Very advanced* mode includes a more complex scaling. It is not done uniformly in the Y-direction of the transversal plane as in the *Advanced* mode, but it varies depending on the X and Z-direction of the transversal plane. The variability of the scaling depending on the Z-direction is made so that the simulated breathing artifacts does not affect the legs and makes a smooth transition into the body located above the bladder. The scaling also depends on the position in the X-direction so that the center of the body is scaled more than the sides. This correspond to the center of the stomach moving further out than the sides during a breath. The scaling in the X-direction is modeled by a full period of a sinus wave, whereas the scaling in the Z-direction is modeled by connecting two scaling levels by half a sinus wave. This creates a 3D scaling map that is visualized in Figure 4.4 below.

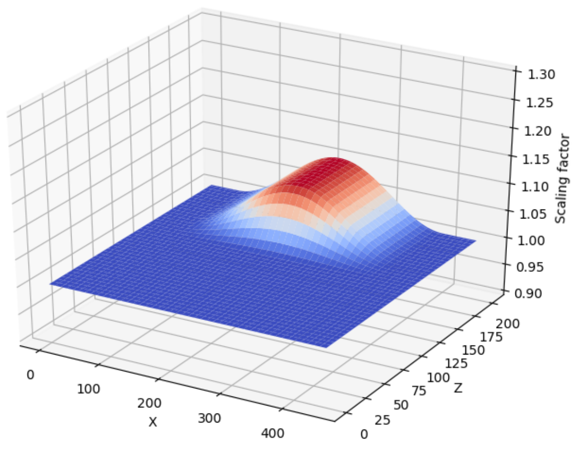


Figure 4.4: Example of a scaling map for simulation of breathing artifacts in the *Very advanced* mode.

The maximum scaling of the image and the width of the sinus wave used for scaling in the X-direction was set as input parameters to the program. A number of scaling maps are generated and used on each template, corresponding to the variable *number of copies*. Another parameter related to this scaling map is a parameter to adjust where in the X-direction the scaling will start.

Altering lines in the middle of k-space often resulted in very drastic changes of the image, such as a great loss in detail and high blurriness. The simulation program therefore includes the option to avoid altering the center of k-space with a certain number of lines.

Since most modern MR cameras collects several k-space at a time, the k-space lines are replaced in chunks, which is another changeable parameter in the program. They can be replaced either periodically, as to say every n:th chunk, or randomly with a certain probability.

Below two of the different algorithms are presented in a simplified form, *Simple mode* and the *Very advanced mode*. For full descriptions of the algorithms see appendix A.

Algorithm 2 Simple mode - short

```
Input: Image volume = $V_{in}$ and parameters deciding size of chunks, number of lines at middle
    to avoid and how to alter lines.
Output: Image volume augmented with motion artifacts, $V_{out}$
  1: Fourier transform each slice in $V_{in}$ to get $K$.
  2: In $K$, set certain chunks of lines in the phase encoding direction to zero, either periodi-
     cally or randomly depending on input. Avoid a certain area around the middle of k-space
     depending on input.
  3: Let $V_{out}$ be the inverse Fourier transform of $K$
```

Algorithm 3 Very advanced mode - short

```
Input: Image volume = $V_{in}$ and parameters deciding size of chunks, number of lines at middle
    to avoid, how to alter lines and scaling limitations.
Output: Image volume augmented with motion artifacts, $V_{out}$
  1: Fourier transform each slice in $V_{in}$ to get $K$.
  2: Create a number of copies depending on input.
  3: Scale each copy in all directions to simulate breathing, according to parameters deciding
     X, Y and Z limitations.
  4: Translate the copies so that the back of the body is at the same position as the original.
  5: Fourier transform each copy.
  6: In $K$, replace chunks of lines in the phase encoding direction with the same lines in the
     copies, either periodically or randomly depending on input. Avoid altering certain area
     around the middle of k-space depending input.
  7: Let $V_{out}$ be the inverse Fourier transform of $K$
```

Augmentation of data set

A graphical user interface (GUI) was developed to create and evaluate motion artifacts. For a detailed description of this GUI, see appendix B.

Since the algorithms for artifact simulation contained many different parameters, a large number of different models for creating the artifacts were possible. Educated guesses relying on previous articles about k-space and motion artifacts was aided in deciding values for these parameters. Using the developed GUI, simulations with different parameters could be created swiftly and different values for the parameters could be evaluated. The chosen combination was the one that gave the most similar artifacts compared to the images with actual breathing artifacts. The data sets used for the experiments was augmented offline.

Examples of simulated artifacts can be seen in Figure 4.5. For comparison, two examples of how the actual breathing artifacts can look like are seen in Figure 2.2.

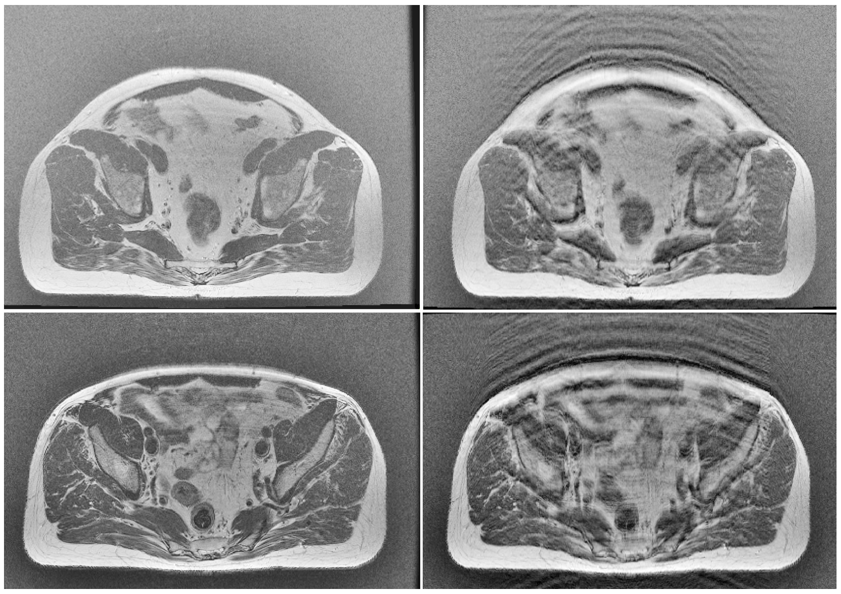


Figure 4.5: Two examples of motion artifacts simulations on two different MR slices [1]. The slices to the left are unaltered, and the augmented versions of them are shown to the right.

[1]Original MRI courtesy of Gentle Radiotherapy.

4.5 Experiments

4.5.1 Geometric augmentations

The first experiment involved using the previously mentioned geometric augmentations on our full data set. The network used for this experiment was U-Net. All the data set was randomly divided into nine groups of five templates, and one group of four templates (the smaller group was due to the fact that one of the templates was at a late time discovered to be unusable). The group with four templates plus eight of the nine groups of five were used as training data, while the ninth group of five was used as validation. This resulted in a training set that always had 44 templates, and a validation set that had five templates. This was repeated for all of the nine groups of five templates to make a total of nine cross validations. These nine cross validations were made once without augmentation, referred to as the *baseline*, once with affine augmentations, and once with elastic deformation. Affine augmentation and elastic deformation were applied online 50% of the times the training data was iterated over. This means a total of 27 models of the network were made.

The results from the first experiment raised a number of questions, such as why many articles seemed so confident that their augmentations improved their results. To answer these, a longer experiment was conducted. This time, the test was done using downsampled versions of the templates. This was due to time constraints and the fact that the time consumption scales with the template size. This time however, a new list of hyperparameters were used, see Table 4.2. A number of short preliminary trainings were conducted to verify that there was no abnormal behavior when using the downsampled templates. After that, a large number of test were conducted using cross validation on several different sizes of data sets. The data set sizes were 5, 10, 20 and 45 templates and the number of cross validations per data set size was five. The cross validations here were made by using repeated random sub-sampling, where the desired number of templates for training and validation are chosen randomly. Each cross validation contained one baseline, one training with affine transformations and one with elastic deformation, giving 60 trained models in total. This tested how the difference in performance changed depending on the size of the data set.

All models were also used to predict on training data to get a measure of overfitting. The resulting dice coefficients from the predictions were put in spreadsheets and mean values and standard deviations were calculated.

4.5.2 Motion artifacts

This experiment investigated the effect of simulated motion artifacts for training of models in neural networks. Experiments were carried out on two networks, *U-Net* and *HighRes3DNet*. Three classes were used in the training of U-Net: *body*, *air*, *bone*. Eight classes were used in the training of HighRes3DNet: *body*, *air*, *bone* bladder, *colon*, *left femur*, *right femur*, *pelvis*.

First, a test set for the evaluation of the models were determined. This was done simple by visually investigating the available templates and looking for images with obvious motion artifacts. Five templates were found that suited the criteria. None of these were a part of the data set used for training and validation.

For both networks, five pairs of models were trained; five baseline models and five augmented models. A data set of 50 templates were used for training and validation. The baseline models were trained on a data set that was unaltered while the augmented models trained on the same data set that had been augmented with motion artifacts.

Two augmented data sets were created from the original data set of 50 templates, one for each network. The augmentations were done offline, prior to training. The augmentations were created with the developed algorithm seen in Algorithm 3 with two different sets of parameters. The settings used for the U-Net experiments are presented in Table 4.3, and the settings used for the HighRes3DNet experiments are presented in Table 4.4.

Using the fully augmented data set and the original data set, a new data set was put together for each of the five pairs of training runs. The data set was randomly partitioned into a training set of 45 templates and a validation set of 5 templates. For an augmented training, 80% of the templates in the training set was drawn from the augmented data set. In a baseline training, all were drawn from the original data set. For both the augmented and the baseline runs the validation set was drawn from the original data set.

This procedure led to a cross validation process which yielded 10 different models. All models were used to predict the labels of the test set containing actual motion artifacts. These segmentations were evaluated visually due to lack of ground truth data. All models were also evaluated on their respective validation set to determine performance on data not containing motion artifacts. This evaluation was done both visually and by computing the Dice coefficient for the different classes.

Parameter	**Value**
Chunk size	16
Random or periodical	0.80
Avoid middle with	30
Max scale	1.15
Number of copies	3
Width adjustment	0

Table 4.3: Settings used to simulate breathing artifacts for training of U-Net.

Parameter	**Value**
Chunk size	2
Random or periodical	20
Avoid middle with	30
Max scale	1.10
Number of copies	2
Width adjustment	0

Table 4.4: Settings used to simulate breathing artifacts for training of HighRes3DNet.

Chapter 5

Result

5.1 Geometric augmentations

In this section the results for the two experiments investigating geometric augmentations using U-Net will be presented.

5.1.1 First experiment

Below in Table 5.1 the computed Dice coefficients for the four different classes from nine cross validated models are shown. Tables 5.2 and 5.3 show the difference from the baseline when augmenting with elastic deformation and affine transformations respectively.

In 5.4 the mean values and their standard deviations for the results in tables 5.2 and 5.3 are presented. Table 5.4 show a slight improvement on all predicted classes for the models using data augmentation.

The p values for a two tailed t test on the bone segmentation results are presented in Table 5.5. This shows that the slight improvements in Table 5.4 cannot be said to be a significant difference between the Dice coefficients of the baseline and augmented models.

Model	Dice air	Dice body	Dice bone	Total
1	0.9944	0.9651	0.8245	0.9740
2	0.9937	0.9548	0.8005	0.9658
3	0.9962	0.9701	0.8362	0.9741
4	0.9961	0.9690	0.8269	0.9723
5	0.9934	0.9685	0.8129	0.9696
6	0.9962	0.9722	0.8131	0.9755
7	0.9972	0.9726	0.8512	0.9794
8	0.9965	0.9727	0.8529	0.9766
9	0.9953	0.9702	0.8314	0.9743
Mean	0.9954	0.9684	0.8277	0.9735

Table 5.1: Dice coefficients of the nine cross validated baselines.

Model	Dice air	Dice body	Dice bone	Total
1	0.0001	0.0057	0.0290	0.0040
2	0.0009	0.0021	0.0071	0.0014
3	-0.0004	-0.0020	-0.0085	-0.0018
4	-0.0008	0.0027	0.0175	0.0023
5	0.0009	-0.0009	-0.0093	-0.0008
6	0.0008	0.0030	0.0126	0.0025
7	0.0000	-0.0026	-0.0133	-0.0020
8	-0.0006	-0.0054	-0.0237	-0.0045
9	0.0010	0.0039	0.0181	0.0034

Table 5.2: The difference of Dice coefficients between elastic deformation and baseline models.

Model	Dice air	Dice body	Dice bone	Total
1	0.0013	0.0089	0.0385	0.0067
2	0.0015	0.0087	0.0280	0.0058
3	0.0002	0.0050	0.0261	0.0043
4	0.0002	0.0031	0.0151	0.0028
5	0.0014	-0.0024	-0.0201	-0.0023
6	0.0002	0.0062	0.0385	0.0053
7	-0.0005	-0.0034	-0.0113	-0.0023
8	-0.0006	-0.0040	-0.0171	-0.0033
9	0.0007	-0.0021	-0.0145	-0.0018

Table 5.3: The difference of Dice coefficients between affine transformations and baseline models.

Augmentation	Air	Body	Bone	Total
Elastic deformation				
Mean	0.0002	0.0007	0.0033	0.0005
Std(m)	0.0002	0.0011	0.0055	0.0009
Affine transformations				
Mean	0.0005	0.0022	0.0092	0.0017
Std(m)	0.0002	0.0017	0.0078	0.0013

Table 5.4: The mean differences in Dice coefficients between the baseline and the models trained with elastic deformation and affine transformations, as well as the standard deviation.

Augmentation result	**P value**
Elastic deformation	0.6956
Affine transformations	0.3433

Table 5.5: P values from a two tailed t test for Dice coefficients for bone in tables 5.1-5.3. Significance level = 0.05.

5.1.2 Second experiment

Here the results from the second experiment regarding geometric augmentations are presented. Table 5.6 shows the mean Dice coefficients for the baselines for the different training set sizes. It can be seen that the performance increases with the amount of training data. The tables 5.7 - 5.10 show the mean difference between the baseline and the augmented models and standard deviation of the mean for the models trained on smaller data sets containing 5, 10, 20 and 45 templates. It can be seen that augmentations on both 5 and 10 templates improve the performance of the model to different degrees, whereas it on 20 and 45 templates in general decreased the performance.

Figures 5.1 and 5.2 show graphically the results of bone from tables 5.6-5.10. They show the difference between the performance of the baselines and the performance with augmentation and how it changes for different sizes of the data set. Table 5.11 show the results from a statistical t test on the bone class. It shows that in the results from tables 5.7 - 5.10, the improvement is significant on the smallest data set of 5 templates and that there are in general no significant results for the larger data sets of 10 or more templates. Figures 5.3 - 5.5 show the results when predicting on the bone class on the validation data versus the training data, in order to measure overfitting.

Number of templates	Air	Body	Bone	Total
5	0.9840	0.9626	0.8278	0.9664
10	0.9915	0.9751	0.8801	0.9780
20	0.9935	0.9811	0.9040	0.9829
45	0.9947	0.9829	0.9104	0.9847

Table 5.6: Mean Dice coefficients for the cross validations of the baseline models of 5, 10, 20 and 45 templates.

Augmentation	Air	Body	Bone	Total
Elastic deformation				
Mean	0.0023	0.0088	0.0533	0.0077
Std(m)	0.0024	0.0020	0.0065	0.0020
Affine transformations				
Mean	0.0030	0.0098	0.0549	0.0086
Std(m)	0.0023	0.0021	0.0073	0.0021

Table 5.7: The mean differences in Dice coefficients between the baseline and the models trained with elastic deformation and affine transformations, as well as the standard deviation when 5 templates are used for training data.

Augmentation	Air	Body	Bone	Total
Elastic deformation				
Mean	0.0017	0.0038	0.0170	0.0035
Std(m)	0.0006	0.0011	0.0054	0.0010
Affine transformations				
Mean	0.0028	0.0036	0.0085	0.0034
Std(m)	0.0007	0.0013	0.0056	0.0012

Table 5.8: The mean differences in Dice coefficients between the baseline and the models trained with elastic deformation and affine transformations, as well as the standard deviation when 10 templates are used for training data.

Augmentation	Air	Body	Bone	Total
Elastic deformation				
Mean	-0.0011	-0.0028	-0.0098	-0.0024
Std(m)	0.0010	0.0008	0.0012	0.0008
Affine transformations				
Mean	0.0007	-0.0007	-0.0056	-0.0004
Std(m)	0.0010	0.0018	0.0074	0.0016

Table 5.9: The mean differences in Dice coefficients between the baseline and the models trained with elastic deformation and affine transformations, as well as the standard deviation when 20 templates are used for training data.

Augmentation	Air	Body	Bone	Total
Elastic deformation				
Mean	0.0006	-0.0012	-0.0110	-0.0011
Std(m)	0.0005	0.0017	0.0078	0.0015
Affine transformations				
Mean	0.0010	-0.0038	-0.0261	-0.0029
Std(m)	0.0007	0.0015	0.0087	0.0013

Table 5.10: The mean differences in Dice coefficients between the baseline and the models trained with elastic deformation and affine transformations, as well as the standard deviation when 45 templates are used for training data.

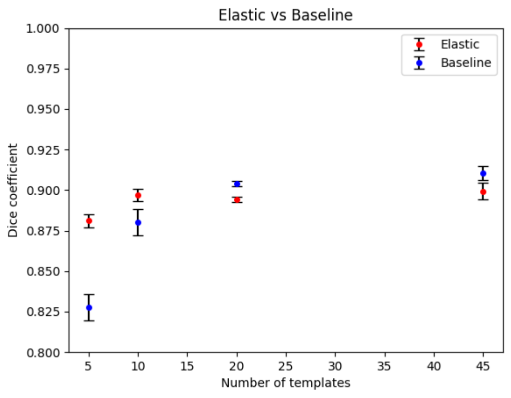


Figure 5.1: Mean Dice coefficients and standard deviations for the bone class for 5, 10, 20 and 45 templates with elastic deformation and baseline models.

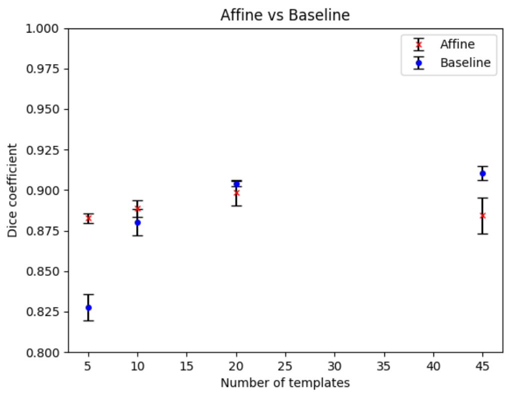


Figure 5.2: Mean Dice coefficients and standard deviations for the bone class for 5, 10, 20 and 45 templates with affine transformations and baseline models.

Augmentation	5	10	20	45
Elastic deformation	**0.0008**	0.1215	**0.0059**	0.1740
Affine transformations	**0.0005**	0.4497	0.5535	0.0875

Table 5.11: The p values from a two tailed t test for Dice coefficients for bone for 5, 10, 20 and 45 templates. Significance level = 0.05. Significant values are boldfaced.

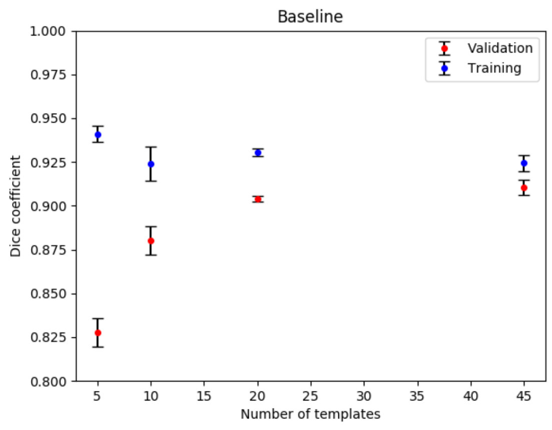


Figure 5.3: Mean Dice coefficient for bone with standard deviations for 5, 10, 20 and 45 baseline templates when predicting with validation data vs training data.

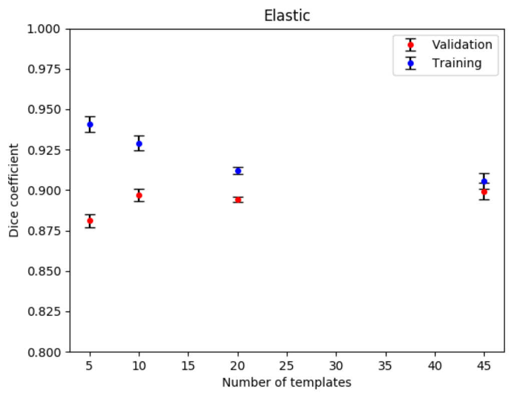


Figure 5.4: Mean Dice coefficient for bone with standard deviations for 5, 10, 20 and 45 templates with elastic transformation, when predicting with validation data vs training data.

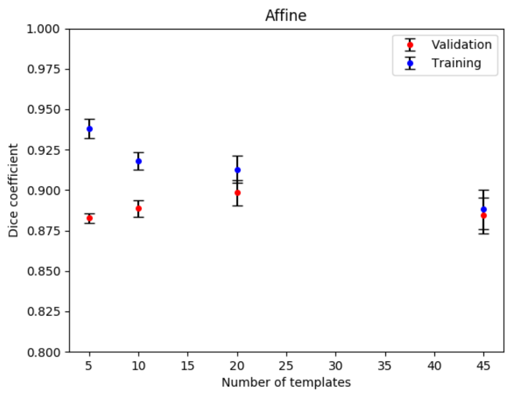


Figure 5.5: Mean Dice coefficient for bone with standard deviations for 5, 10, 20 and 45 templates with affine transformations, when predicting with validation data vs training data.

5.2 Motion artifact augmentations

In this section the result from training models with the simulated motion artifacts are presented. First the result from using the U-Net network is presented and then the results from using HighRes3DNet are presented.

5.2.1 U-Net

It was found that the models trained with augmentation either performed better than the baseline model or did not improve results. Below in Figure 5.6 - 5.8, predicted segmentations on three different problematic templates where the improvement is the most prevalent are presented. The segmentations have been made from three different pairs of models. The baseline segmentation is shown to the left whereas the segmentation made from the model that was augmented is shown to the right. Slices from all three different anatomical views are shown.

Generally, both segmentation of the body as well as segmentation of bone structures improved with the model trained on augmented data. Figure 5.6 shows a clear example of better segmentation of the body where motion artifacts are present. Cases were improvement of bone segmentation is obvious can be seen in figures 5.7 and 5.8.

The ten models were also evaluated on their validation data to evaluate performance on training data without severe motion artifacts. These results are presented as Dice coefficients for the different classes below in Table 5.12 and 5.13. P values resulting from a t test conducted to test for significant differences in the results are presented in 5.14. A decrease performance in body segmentation was detected for the augmented models.

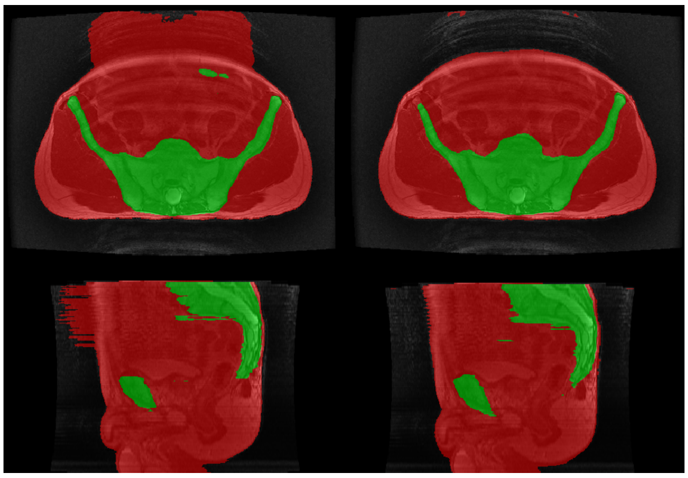


Figure 5.6: Comparison of predictions from two models. To the left, predictions from models trained on baseline data. To the right, predictions from models trained on data with simulated breathing artifacts. Top images are from transverse view, bottom are from sagittal view. Red marks for body class and green for bone class.

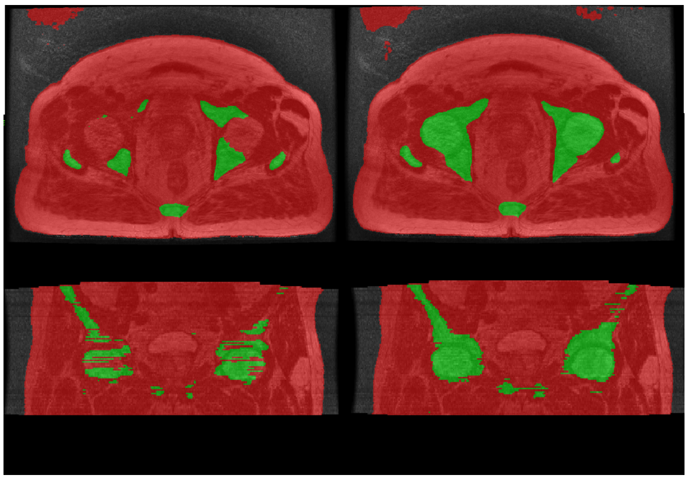


Figure 5.7: Comparison of predictions from two models. To the left, predictions from models trained on baseline data. To the right, predictions from models trained on data with simulated breathing artifacts. Top images are from transverse view, bottom are from sagittal view. Red marks for body class and green for bone class.

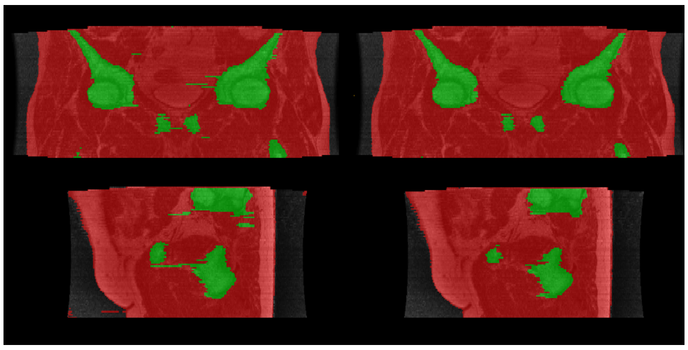


Figure 5.8: Comparison of predictions from two models. To the left, predictions from models trained on baseline data. To the right, predictions from models trained on data with simulated breathing artifacts. Top images are from transverse view, bottom are from coronal view. Red marks for body class and green for bone class.

Model	Air	Body	Bone	Total
Baseline				
1	0.9955	0.9862	0.9159	0.9863
2	0.9963	0.9868	0.9163	0.9870
3	0.9950	0.9851	0.9086	0.9851
4	0.9950	0.9840	0.9014	0.9841
5	0.9961	0.9855	0.9068	0.9856
Augmented				
1	0.9963	0.9847	0.9125	0.9870
2	0.9960	0.9828	0.9052	0.9852
3	0.9954	0.9843	0.8998	0.9844
4	0.9962	0.9787	0.9057	0.9841
5	0.9962	0.9798	0.8783	0.9819

Table 5.12: Dice coefficient for all classes on baseline and augmented models on non-problematic templates.

Model	Air	Body	Bone	Total
Baseline				
Mean	0.9956	0.9855	0.9098	0.9856
Std(m)	0.0002	0.0004	0.0025	0.0004
Augmented				
Mean	0.9960	0.9821	0.9003	0.9845
Std(m)	0.0001	0.0011	0.0052	0.0007

Table 5.13: Mean Dice coefficient and standard deviation of the mean for all classes on baseline and augmented models on non-problematic templates.

Class	Air	Body	Bone	Total
P value	0,2785	**0,0282**	0,1824	0,2872

Table 5.14: P values from a two tailed t test comparing difference in Dice coefficients for baseline models and augmented models. Significance level = 0.05. Significant values are boldfaced.

5.2.2 HighRes3DNet

Ten models were trained using the network HighRes3DNet. Five were used as baseline and trained on normal data while the other five was trained on data that had been augmented prior to training.

The baseline models and the augmented models were evaluated on the same problematic test set as in the 2D U-Net case. However, significant visual differences in performance was only found for one particular template. Comparison of segmentations on this template with three different models are presented below. The segmentations made with the baseline model are shown to the left while the segmentation made with the augmented model is shown to the right.

As can be seen in the images below in figures 5.9 - 5.11, the augmented model greatly outperforms the baseline model in these slices. Improvements are made especially in the left femur, marked by the color turquoise, and in the pelvis bone, marked by the color green. The images are taken from three different models from the cross validation.

The ten models were also evaluated on their validation set to evaluate performance on training data without severe motion artifacts. These results are presented as Dice coefficients for the different classes below in Table 5.15. P values resulting from a two tailed t test conducted to test for significant differences in the results are presented in 5.17. No significant change in performance was detected.

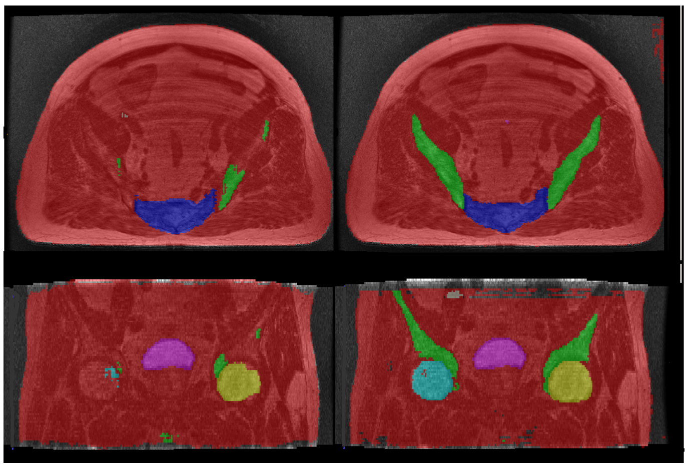


Figure 5.9: Comparison of predictions from two models. Left - baseline model. Right - augmented model. The two upper rows of images are from the transversal view. The bottom images are from a coronal view. Bladder marked in purple, spine in blue, pelvis in green and femur bones in yellow and turquoise.

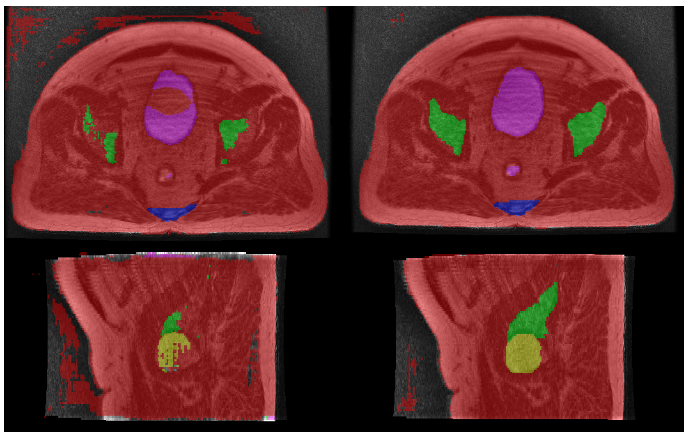


Figure 5.10: Comparison of predictions from two models. Left - baseline model. Right - augmented model. The two upper rows of images are from the transversal view. The bottom images are from a sagittal view. Bladder marked in purple, spine in blue, pelvis in green and femur bones in yellow and turquoise.

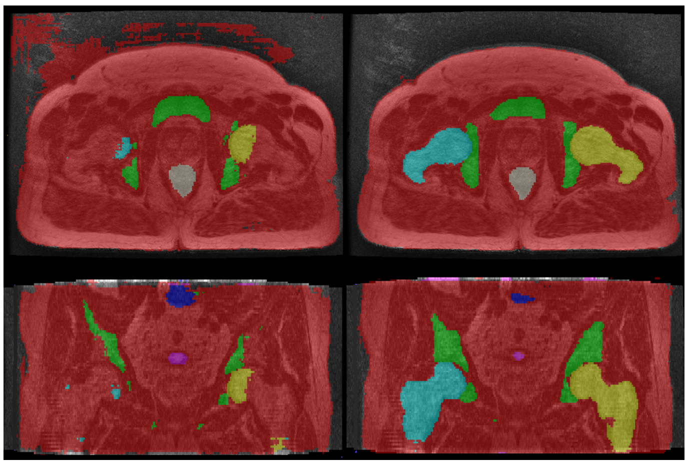


Figure 5.11: Comparison of predictions from two models. Left - baseline model. Right - augmented model. The two upper rows of images are from the transversal view. The bottom images are from a coronal view. Bladder marked in purple, spine in blue, pelvis in green, colon in gray and femur bones in yellow and turquoise.

Model	Air	Body	Pelvis	Spine	Left femur	Right femur	Bladder	Colon	Total
Baseline									
1	0.9812	0.9602	0.8874	0.9260	0.9292	0.9363	0.8689	0.8554	0.9706
2	0.9828	0.9673	0.9042	0.9024	0.9393	0.9312	0.9469	0.8144	0.9735
3	0.9801	0.9701	0.8838	0.9200	0.9308	0.9341	0.9143	0.8786	0.9717
4	0.9859	0.9649	0.9010	0.9303	0.9381	0.9388	0.9371	0.8798	0.9750
5	0.9817	0.9681	0.8926	0.9137	0.9326	0.9230	0.8946	0.8460	0.9727
Augmented									
1	0.9739	0.9476	0.8906	0.9187	0.8997	0.9321	0.8445	0.8098	0.9613
2	0.9770	0.9599	0.8990	0.8960	0.9381	0.9344	0.9421	0.8487	0.9673
3	0.9802	0.9691	0.8637	0.9206	0.9274	0.9177	0.8996	0.8148	0.9706
4	0.9931	0.9734	0.8804	0.9193	0.9280	0.9275	0.8759	0.8450	0.9810
5	0.9867	0.9695	0.8762	0.8945	0.9215	0.9162	0.9151	0.8459	0.9757

Table 5.15: Dice coefficient for baseline model from HighRes3DNet on non-problematic templates.

Model	Air	Body	Pelvis	Spine	Left femur	Right femur	Bladder	Colon	Total
Baseline									
Mean	0.9823	0.9661	0.8938	0.9185	0.9340	0.9327	0.9124	0.8548	0.9727
Std(m)	0.0009	0.0015	0.0035	0.0044	0.0018	0.0024	0.0127	0.0108	0.0007
Augmented									
Mean	0.9822	0.9639	0.8820	0.9098	0.9229	0.9256	0.8954	0.8328	0.9712
Std(m)	0.0031	0.0041	0.0054	0.0053	0.0057	0.0033	0.0149	0.0076	0.0030

Table 5.16: Mean Dice coefficient and standard deviation of the mean for the baseline and augmented model from HighRes3DNet on non-problematic templates.

Class	Air	Body	Pelvis	Spine	Left femur	Right femur	Bladder	Colon	Total
P value	0,9656	0,6651	0,1392	0,2942	0,1374	0,1610	0,4615	0,1731	0,6732

Table 5.17: P values from a two tailed t test comparing difference in Dice coefficients for baseline models and augmented models. Significance level = 0.05.

Chapter 6

Discussion

6.1 Discussion of methodology

6.1.1 Geometric data augmentation

The choice of which geometric transformations to use was done based on previous practices for augmentation of neural networks [8], [23], [3]. However, the specific parameters of the augmentations, like the rotation angle and the scaling and translation percentage seems to have been chosen arbitrarily in many works and was also done so here. The choice was made based on what seemed reasonable considering the physiological appearance of the image. However, it is very possible that other geometric transformations would have achieved different results. The amount of training data that should be augmented with each transformation and combination of transformation can also be discussed. Perhaps one could conduct a random search of the parameters for the geometric transformations and how often they should be applied. However, because of the great number of parameters and combination of parameters, this would require training of an unfeasible number of models. This was outside the scope of this project and probably many others. As seems to be the case in many areas within machine learning, one has to rely on educated guesses.

As mentioned, geometric augmentations were divided into two different groups: affine transformations and elastic deformation. The affine augmentations were used several different articles and is very commonly used in deep learning, which is why it was a representative choice. It could have been divided into its different components, but that would also have led to many more trainings of models. There simply was no time for that, and since all the affine augmentations are closely related, it was a simple choice to put them all together. It was also thought to be a better augmentation than any individual one, since it presents more different possible variations. The elastic deformation is not as common in general deep learning, but sometimes appear in medical applications. Especially since Ronnerberg et al. [23] claimed that it was a very important augmentation for their use of U-Net, it early became a good candidate. As is explained in the U-Net article, elastic deformations are common in the body, especially in soft tissue, and is therefore a logical choice of augmentation.

6.1.2 Simulation of motion artifacts

The simulation of motion artifacts came with a number of assumptions, estimations and guesses. For example, it was not known exactly how k-space was sampled for the acquisition of the MR images used. With this said, the basis of these simulations, the alteration of k-space in the phase encoding direction, is well grounded from theory and has been reproduced several times before.

It was found that altering chunks of k-space rather than single lines at a time yielded better results. This probably corresponds to multi-shot sampling where the echo train length is simulated with our chunk size parameter. It also seems like creating more than one scaled copy of the image and joining their k-space with the original image's k-space yielded better results than with one copy. This could model for gradual movement of the patient, something that could be closer to reality. Other parameters that were used were set completely to mimic the visual appearance of artifacts and are harder to connect with physical and physiological properties. It was noticed that altering the center of k-space resulted in unrealistically severe artifacts, and an option to avoid altering the center was therefore added. It is possible that this corresponds to a acquisition strategy where the center of k-space is more resistant to inconsistencies.

Compared to other methods that have simulated motion using one dimensional transformation, our method considers all dimensions. It also uses scaling as the main transformation instead of translation, although translation is used to compensate for the movement of the back. This simulated movement of the back was the reason scaling was chosen in the first place. Experiments with translation of the whole body gave large artifacts originating from the change in position of the lower border of the body. Using scaling in two directions also made it possible to simulate movement concentrated to middle of the body, rather than using the same transformation over all elements in the x-axis. Finally, by using the whole volume and taking advantage of anatomical data, such as the position of the bladder, artifacts were simulated in places were breathing is anatomically relevant.

Scaling does of course have its disadvantages, since it creates unrealistic deformations of the anatomical structures. Unlike translation, scaling actually deforms the image and stretches it out. While this might somewhat simulate the stretching of the soft tissue during a breath, parts like the pelvis bones and spine will also deform and change size, which is obviously not realistic.

With this said, the aim of the simulations was not to make a realistic physiological model of the breathing, but to simulate the visual artifact in the MR images. To this end, the simulations were very much successful. Considering how to make the model itself physiologically realistic was however a way to reach the goal of simulating the artifacts.

For the purpose of this thesis, neither the physiological correctness nor the visual appearance is what actually matters. The network only cares about the information in the image, not how it was created or its appearance. What is important is the neural networks' response to the artifacts. The aim should be that it uses the simulations, together with the untampered ground truths, to realize that the artifacts are only noise and not important data. It is therefore possible that simulations of a simpler kind than the one presented could yield similar results.

It should also be noted that there was many different parameters that could be altered to change the appearance of the breathing artifacts. Only a few combinations were tested due to time constraints, but the result may have improved if more combinations of parameter values had been tested.

6.2 Discussion of results

6.2.1 Geometric augmentations

First experiment

The results of the first experiment made on the full data set are shown in tables 5.1-5.5. They show that there are no significant improvements when using any of the augmentations. The mean Dice coefficient of bone is slightly higher with the affine augmentations, but it is in no way statistically significant. There is not much more to say about the actual result, other than it raised questions about data set size that later led to the second experiment. As mentioned, the set of hyperparameters changed after the first experiment, yielding in better performance for later models. This means that the results of this experiment cannot be directly compared to the other experiments, but instead it acted like a stepping stone into later experiments.

The results are however great for showing why cross validation and multiple models of the network is necessary for a good result. The first models showed an improvement with the affine augmentations, while the later ones showed the opposite. A shorter experiment might have concluded that the augmentations had an effect, which would contradict the final results. Some might argue that some models do show an increased performance when using augmentations, but the idea of the cross validation is to show whether or not it can be said to be a general rule and not just a coincidence.

Second experiment

A number of articles state that data augmentation is an important part of their work with images in neural networks, including the paper by Ronneberger et al. [23]. Many of these articles have a lot smaller training sets than in this project, which might imply that the improvement is greater when the amount of data is lower. Many articles state clearly that augmentation improve their models, but also add that it works well on ”small data sets”[3], [8], without discussing it further. This is why different sizes of training sets were tested as seen in tables 5.6 - 5.10 and figures 5.1-5.2. This might give a better insight to whether there is an improvement at all, and if it might depend on the amount of training data. The graphs 5.1 and 5.2 show that the improvement was the greatest for the smallest data set. These graphs only show the performance on bone as an example, because they had the lowest performance on the baseline and appeared to be affected the most by the augmentations. All the results can however be seen from the tables 5.6 - 5.10 if one wants to do the same comparison for the other classes.

These results might indicate that these types of data augmentations work best when the training set is small. This can also be seen in 5.11, which show that the improvement on bone from the augmentations is significant for the smallest data set, but not in general for the bigger ones. One explanation could be that the augmented, but still similar, images reduce the overfitting of the models. It is shown in figures 5.3-5.5 that smaller data sets suffer more from overfitting, just like the literature says. It can also be seen that the overfitting is reduced for the small data sets when augmentations were made, which may strengthen this hypothesis. One obvious question here is ”What is a small data set?”. That is not an easy question to answer and will depend on what type of data is available, what network and parameters that are chosen, etc.

It can be seen in 5.1 and 5.2 that the performance decreases somewhat when augmenting the larger data sets, although 5.11 show that it is not in general significant. It could be a sign that the augmentations are not good enough to add any information to the network. Instead it might be that it dilutes the overall quality of the training set. Where this breaking point between improving and not improving occurs will probably depend on what network structure that is used, and what type the data is.

These results can be seen as both agreeing and disagreeing with some of the related work mentioned earlier in this paper. On one hand, most papers agreed that data augmentation improved their result. This is not true for the larger data sets tested here. Although, the related articles also state that their augmentation worked well on their *small* data set. This present the possibility that their data set were small enough to be compared to our smallest set of training data. It all depend on what is a small data set. If the question is reversed, then maybe one can say that a large data set is a set that is big enough to require no augmentation to achieve its best result?

6.2.2 Motion artifact augmentations

For segmentation of images containing motion artifacts, the networks' performance increased significantly if trained on data containing simulated motion artifacts. This shows that the simulations served its purpose – to make the networks robust against actual artifacts. It is also very satisfactory that the augmented models performed as well as baseline models on images without motion artifacts. The decrease in Dice coefficient for body segmentation using U-Net can likely be contributed to excessive augmentation. Perhaps the number of augmented images can be decreased so that the network is able to train on a sufficient amount of "normal" data but is still able to become robust to artifacts.

It should be noted that the most problematic template contained an excessive amount of noise compared to other templates. It would therefore be interesting to see if similar results could be achieved using an augmentation that simulates noise in the training data.

Much of the improvement was seen in the contour of the body, which is reasonable since motion artifacts create distortions outside of the body where there should only be air as seen in Figure 5.6. A model that has not trained on these kind of images will not learn that these lines are to be treated as noise and will instead treat the lines as part of the contour of the body.

More interesting for practical uses is the improvement of bone segmentation, noticeable in experiments using both networks. As seen in figures 5.9 - 5.11, the segmentation of the femoral head was of very low quality for baseline models, and was in one case completely ignored. The augmented models succeeded much better at this segmentation. This anatomical structure is especially vulnerable to radiation and needs to be protected during radiation therapy. It is therefore of great importance that it is segmented correctly. If a system for segmenting these structures were to be fully automated, it would have to be incredibly robust to a large range of variations appearing in the images. Artifacts is something that will always exist as long as an image capturing system is used. It is therefore pleasing to see these variations can be easily simulated and used to make an algorithm robust towards real artifacts.

Although all eight classes were used with the HighRes3DNet experiments, the greatest improvement was still found in bone segmentation. By inspecting images with motion artifacts visually, is can be seen that the artifacts distort the borders of the bone structures to a degree that they blend in to the surrounding tissue. Other classes such as the bladder generally have stronger borders and a different gray scale than the surrounding soft tissue, which could explain why it was easier for the network to predict their segment even with the distortions created by artifacts. However, there are likely several other segments in MR images not used in these experiments whose segmentation would be as problematic for neural networks as bone structures. Hopefully the work in this thesis can inspire others to investigate augmentation with simulated motion artifacts in their own deep learning applications using MR data.

6.3 General discussion

6.3.1 Generalization to other networks and data sets

Although the results have been cross validated it is important to know that one cannot necessarily generalize this to every network type and every kind of data. Only two different networks have been tested in this project: U-Net and HighRes3DNet. Even from only those two networks it became clear that they had different advantages and different problems. Only MR images from two different producers were used, and all images were collected for the same specific purpose. It is not known how the networks would perform on other MR images created with different settings and purpose.

The exact point where the geometric transformations start to improve the model is with all likelihood specific for this data set and network. Using other images, like for example MR images of the brain, or using another network architecture, will likely change this. It will probably also depend on the difficulty of the specific task. The main conclusion to draw from this is that there probably is a similar breaking point for other networks and data sets but finding where it is will have to be made in every individual case. This means that one cannot blindly use data augmentation and think it will improve the results.

6.3.2 Performance metrics

Another topic that can be discussed at length is the choice of performance metric. During this project two main ways of evaluating the segmentation results has been used: numerically, using the Dice coefficient, and visually, by investigating the MR images and its segmentations with a 3D visualization tool. The first method was used mainly for the experiments aiming to increase performance of the network with geometric augmentations, while the latter method was used mostly evaluating the experiments aiming to increase the network's robustness against motion artifacts.

While the Dice coefficient is a very common metric used to evaluate biomedical segmentation tasks, one must be careful when using it as a strictly numerical performance metric. It is always important to look at the results visually as well. The error from a given Dice coefficient might be the result of a segment that is overall slightly to big, or a segment that is perfect, but also has small miss classified outliers without physiological meaning. Visually large differences in the images used in this project could yield a very small change in the Dice coefficient. If the images had not been inspected visually, these results would be misleading. Also, a model trained with a certain augmentation might improve the segmentation task in some areas while making it worse in others, making the net improvement zero. This would not be discovered only using the Dice coefficient.

If this project were to be continued, a good next step could be to develop a customized performance metric that would fit the problem at hand for this specific data. This metric should be able to detect subtle but relevant changes in segmentation performance, which would make automation of the experiment process easier and therefore greatly increase the rate at which they could be done.

6.4 Future work

There is still much more to be investigated regarding data augmentation in neural networks, especially in the case of medical data. First of all, it would be interesting to investigate whether the results from this report could be replicated using other networks or using a different data set of MR images.

Several variations of the augmentations used in this thesis that has not been investigated. The different parameters for the geometric augmentations and motion artifact augmentations could also be investigated further.

Several other augmentations were only investigated briefly in the project but not more extensively due to time constraints. Noise augmentations are for example something that seems to have potential, as do gray scale augmentation to give the network an invariability towards different gray scale values. Only one kind of MR artifact was investigated in this thesis. However, there is a great range of artifacts that could be simulated in a similar manner to augment data. Building a model that is robust against several types of MR artifacts would of course be very beneficial for many applications.

Chapter 7

Conclusion

The results showed that commonly used affine augmentations of the training data increased the models' performance on test data when the amount of training data was below a certain level. However, above a certain amount of training data, in this specific case around 10 templates, the affine augmentations of the data did not increase the performance of the model. The same result was found when using elastic deformations as augmentation method.

Using information of how the raw k-space data is acquired and why motion artifacts appears, motion artifacts in MR images could be simulated by Fourier transforming the image, altering the k-space matrix and transforming it back to image space. The most realistic simulations were reached when breathing was simulated with scaled and translated copies of the original image, to simulate breathing during image acquirement.

By simulating images with motion artifacts and then using these images as training data for two different neural networks, the performance of the models produced by the network increased for segmenting problematic images containing actual motion artifacts. The performance on non-problematic images did not notably decrease.

Appendix A

Algorithms

Algorithm 4 Simple mode - complete

Input: Image volume = V with dimensions (X,Y,Z) and parameters $chunk\_size$, per_or_prob and $avoid\_middle$
Output: Image volume augmented with motion artifacts

```
 1: Create matrix K with dimensions (X,Y,Z)
 2: for each slice $i$ in dimension $Z$ in V do
 3:     $k$ = Fourier transform of $i$
 4:     Shift zero-frequency components of $k$ to center
 5:     Put $k$ into K
 6: $mid$ = round($Y/2$)
 7: $middle\_area$ = ($mid$-$avoid\_middle$,$mid$+$avoid\_middle$),)
 8: if $per\_or\_prob$ > 1 then
 9:     for $y$ = 0 to $nmbr\_chunks$ do
10:         if $y$%per_or_prob= 0 and $y$ is not within $middle\_area$ then
11:             Set chunk $y$ in $K$ to zero in all slices $z$
12:         else
13:             Do nothing
14: else
15:     for $y$ = 0 to $nmbr\_chunks$ do
16:         if $y$ is not within $middle\_area$ then
17:             Generate $r$, a random number between 0 and 1
18:             if $r$ < $per\_or\_prob$ then
19:                 Set chunk $y$ in $K$ to zero in all slices $z$
20:             else
21:                 Do nothing
22: for each slice $z$ in dimension $Z$ in $K$ do
23:     Shift zero-frequency components back to original spectrum
24:     Inverse Fourier transform of $z$
25: return $K$
```

Algorithm 5 Very advanced mode - complete

```
Input: Image volume = $V$ with dimensions (X,Y,Z) and parameters $chunk\_size$,
    $per\_or\_prob$, $avoid\_middle$, $max\_scale$, $nmbr\_cop$ and $width\_adjustment$
Output: Image volume augmented with motion artifacts
 1: Create matrix K with dimensions (X,Y,Z)
 2: for each slice $i$ in dimension $Z$ in V do
 3:     $k$ = Fourier transform of $i$
 4:     Shift zero-frequency components of $k$ to center
 5:     Put $k$ into K
 6: Create $nmbr\_cop$ number of copies of $V$
 7: for $i$ = 1 to $nmbr\_cop$ do
 8:     Create 3D scaling matrix according to $max_scale$ in the Y-direction,
    $width_adjustment$ in the X-direction and position of the bladder in the Z-direction, if
    data is available. Else use middle of volume as starting position in the Z-direction.
 9:     Scale $V\_copy\_i$ in all three dimensions according to scaling matrix.
10:     Translate the body in $V\_copy\_i$ back to its original position
11:     for each slice $z\_c$ in dimension $Z$ in $V\_copy\_i$ do
12:         $K\_copy$ = Fourier transform of $z\_c$
13:         Shift zero-frequency components of $k$ to center
14:         Put $z\_c$ into $K\_copy$
15:     $mid$ = round($Y/2$)
16:     $middle\_area$ = ($mid$-$avoid\_middle$,$mid$+$avoid\_middle$),)
17:     $nmbr\_chunks$ = round($Y/chunk\_size$)
18:     if $per\_or\_prob$ > 1 then
19:         for $y$ = 0 to $nmbr\_chunks$ do
20:             if $y$ % $per\_or\_prob$ = 0 and not within $middle\_area$ then
21:                 Replace chunk $y$ in $K$ with chunk $y$ in $K_{copy}$.
22:     else
23:         for $y$ = 0 to $nmbr\_chunks$ do
24:             if $y$ is not within $middle\_area$ then
25:                 Generate $r$, a random number between 0 and 1
26:                 if $r < per\_or\_prob$ then
27:                     Replace chunk $y$ in $K$ with chunk $y$ in $K_{copy}$.
28: for each slice $z$ in dimension $Z$ in $K$ do
29:     Shift zero-frequency components back to original spectrum
30:     Inverse Fourier transform of $z$
31: return $K$
```

Appendix B

GUI for simulating motion artifacts

The graphical user interface that has been used to simulate motion artifacts was developed in Python using native packages. The GUI allows the user to upload a MR image volume in the file format *.nrrd* and simulate motion artifacts using a set of adjustable parameters. Using the scroll function the user can go through the image volume one slice at a time. Three different modes of simulation are available: *Simple*, *Advanced*, and *Very advanced*. Seven different parameters can be adjusted:

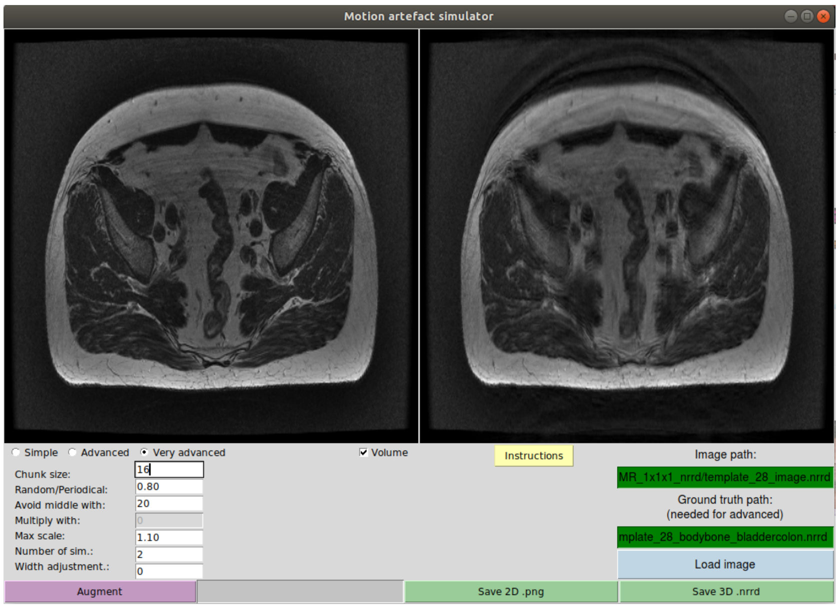
Below, the different parameters are described.

- **Chunk size:** the number of lines in k-space to be replaced for each replacement step.
- **Random or periodical:** Determines whether the lines should be replaced periodically or randomly. If the input is an integer N that is larger than 1, the lines will be replaced periodically with period 1/N. If the input is a number X between 0 and 1, the lines will be replaced randomly with the probability X
- **Avoid middle with:** The number of lines from the center of k-space not to alter.
- **Multiply with:** Used with the mode *Simply* Determines what the lines should be multiplied with, can either be an integer or the string 'random' which multiplies the lines with a random scalar.
- **Max scale:** Used with the advance modes. The maximal percentage that the original image should be scaled in the Y-direction.
- **Number of copies:** Used with the advanced modes. The number of scaled copies of the original image to create.
- **Width adjustment:** Used with the mode *Very advanced*. The number of pixels to make the scaling map in Figure 4.4 wider or narrower in the X-direction. 0 means that the scaling starts at the sides of the body.

The GUI uses a simple graphical layout with text fields where the user can set the parameters by writing a string, and buttons to activate the different functions of the program. Wrong inputs are handled with error messages. The top half part of the window is dominated by the original image volume and its corresponding augmented version.

Two different options to save the augmented images are implemented in the GUI. The first option is to save a 2D image of the current augmented slice that is visible in the file format .png. The second option is to save the whole of the augmented image volume in the file format *.nrrd*.

Below in Figure B.1, a screenshot of the GUI can be seen with default parameters set and with an artifact simulation made.



Simple [✓ Very advanced] Advanced
[✓ Volume]

Chunk size:	16
Random/Periodical:	0.80
Avoid middle with:	20
Multiply with:	0
Max scale:	1.10
Number of sim.:	2
Width adjustment.:	0

Image path:
MR_1x1x1_nrrd/template_28_image.nrrd

Ground truth path:
(needed for advanced)
mplate_28_bodybone_bladdercolon.nrrd

Figure B.1: Screenshot of the graphical user interface developed to simulate motion artifacts in MR images.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[2] J. Akeret, C. Chang, A. Lucchi, and A. Refregier. Full length article: Radio frequency interference mitigation using deep convolutional neural networks. *Astronomy and Computing*, 18:35 – 39, 2017.

[3] A. Asperti and C. Mastronardo. The Effectiveness of Data Augmentation for Detection of Gastrointestinal Diseases from Endoscopical Images. 2017.

[4] P. G. Batchelor, D. Atkinson, P. Irarrazaval, D. L. G. Hill, J. Hajnal, and D. Larkman. Matrix description of general motion correction applied to multishot images. *Magnetic Resonance In Medicine*, 54(5):1273 – 1280, 2005.

[5] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533, 2012.

[6] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(2):281 – 305, 2012.

[7] E. Castro, J. S. Cardoso, and J. C. Pereira. Elastic deformations for data augmentation in breast cancer mass detection. *2018 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, page 230, 2018.

[8] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification. 2018.

[9] E. Gibson, W. Li, C. Sudre, L. Fidon, D. Shakir, G. Wang, Z. Eaton-Rosen, R. Gray, T. Doel, Y. Hu, T. Whyntie, P. Nachev, D. C. Barratt, S. Ourselin, M. J. Cardoso, and T. Vercauteren. NiftyNet: a deep-learning platform for medical imaging. volume 158, pages 113–122, 2018.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[11] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. C. Courville, Y. Bengio, C. Pal, P. Jodoin, and H. Larochelle. Brain Tumor Segmentation with Deep Neural Networks. *CoRR*, abs/1505.03540, 2015.

[12] A. Jung. Imgaug. `https://github.com/aleju/imgaug`, 2015.

[13] E. K. Kim and Y. J. Lee. MRI Artifact Cancellation due to Unknown Respiratory Motion. In *The 9th International Conference on Advanced Communication Technology*, volume 1, pages 389–394, Feb 2007.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60(6):84 – 90, 2017.

[15] J. Lemley, S. Bazrafkan, and P. Corcoran. Smart Augmentation - Learning an Optimal Data Augmentation Strategy. 2017.

[16] W. Li, G. Wang, L. Fidon, S. Ourselin, M. J. Cardoso, and T. Vercauteren. On the Compactness, Efficiency, and Representation of 3D Convolutional Networks: Brain Parcellation as a Pretext Task. In M. Niethammer, M. Styner, S. Aylward, H. Zhu, I. Oguz, P.-T. Yap, and D. Shen, editors, *Information Processing in Medical Imaging*, pages 348–360, Cham, 2017. Springer International Publishing.

[17] D. Moratal, A. Valles-Luch, L. Marti-Bonmati, and M. Brummer. k-Space tutorial: an MRI educational tool for a better understanding of k-space. *Biomed Imaging Interv J*, 4(1):e15, Jan 2008.

[18] OpenStax College. Planes of body, 2017. `https://commons.wikimedia.org/wiki/File:Planes_of_Body.jpg`,; accessed on 23-05-2018.

[19] C. B. Paschal and H. D. Morris. K-space in the clinic. *Journal Of Magnetic Resonance Imaging: JMRI*, 19(2):145 – 159, 2004.

[20] L. Perez and J. Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *CoRR*, abs/1712.04621, 2017.

[21] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14(5):503–519, Oct 2017.

[22] R. A. Pooley. Fundamental Physics of MR Imaging. 2005.

[23] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015.

[24] H. Salehinejad, J. Barfett, S. Valaee, and T. Dowdell. Training Neural Networks with Very Little Data- A Draft. Aug 2017.

[25] J. Shijie, W. Ping, J. Peiyi, and H. Siping. Research on data augmentation for image classification based on convolution neural networks. In *2017 Chinese Automation Congress (CAC)*, pages 4165–4170, Oct 2017.

[26] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556, 2014.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *CoRR*, abs/1409.4842, 2014.

[29] Wikimedia Commons user Aphex34. Typical cnn., 2015. `https://commons.wikimedia.org/wiki/File:Typical_cnn.png`; accessed on 23-04-2018.

[30] Wikimedia commons user Cmglee. 2d affine transformation matrix, 2016. `https://commons.wikimedia.org/wiki/File:2D_affine_transformation_matrix.svg`; accessed on 26-05-2018.

[31] Wikimedia Commons users Dake and Mysid. A simplified view of an artifical neural network., 2006. `https://commons.wikimedia.org/wiki/File:Neural_network.svg`; accessed on 23-04-2018.

[32] M. Zaitsev, J. R. Maclaren, and M. Herbst. Motion artifacts in MRI: A complex problem with many partial solutions. *Journal of magnetic resonance imaging : JMRI*, 42 4:887–901, 2015.

[33] Y. Zhu, S. Gao, L. Cheng, and S. Bao. Review: K-space trajectory development. pages 356–360, 10 2013.