Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Eivind Lysheim

Analysis of Quantitative Susceptibility Mapping in Healthy Volunteers at 3T and 7T

Master's thesis in Applied Physics and Mathematics
Supervisor: Pål Erik Goa
June 2021

NTNU
Norwegian University of
Science and Technology

Eivind Lysheim

Analysis of Quantitative Susceptibility Mapping in Healthy Volunteers at 3T and 7T

Master’s thesis in Applied Physics and Mathematics
Supervisor: Pål Erik Goa
June 2021

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

NTNU
Norwegian University of
Science and Technology

Preface

This master thesis is the final project of my degree within the field of Biophysics and Medical Technology, and marks the end of a 5-year journey. The project is a continuation of the specialisation project initialised the autumn of 2020, and builds on the foundations of this project. I want to thank my supervisor, Pål Erik Goa, for offering me this project and giving me the guidance, tools and the freedom to solve this assignment. I am also grateful for the help received by Dr. Runa Unsgård. Without your segmentations, this project would not have been possible. Additionally, I would like to thank the volunteers who have been so kind to let me scan their brains in this project, and the employees at 7T MR centre who have been so kind to lend me their brand new MR scanner.

I could have spent a long time listing all the things and people I have been grateful for, these last 5 years. However, I will constrain myself to thanking my friends and closest family for making these years so memorable. I am especially thankful for companionship and motivation I have received the last year, which has been mostly influenced by the pandemic Covid-19.

Eivind Lysheim
Trondheim, 18th June, 2021

Abstract

Today, extraction of information from brain structures critical in Parkinson's Disease (PD) and Amyotrophic Lateral Sclerosis (ALS) diagnosis is usually performed by manual segmentations of the regions of interest (ROIs), which is both time consuming and prone to errors. Automating this process using Convolutional Neural Networks (CNNs) could both save hours of specialised labour and improve quality of segmentations, which in turn aids the doctor in making a diagnosis.

40 Quantitative Susceptibility Mapping (QSM) images with segmentations of the Red Nucleus (RN), Substantia Nigra (SN) and Subthalamic Nucleus (STN) were donated by a Swedish research group. Additionally, 15 volunteers were scanned on a 3T and 7T scanner at St.Olavs hospital. The images were reconstructed from Multi Gradient Echo (MGRE) sequences using the total generalised variation reconstruction pipeline. RN, SN, STN, Cerebrospinal Fluid, Primary Motor Cortex (PMC) and Primary Somatosensory Cortex (PSSC) were manually segmented on the 7T data. Intra-subject co-registration was performed to transform the 7T masks of RN, SN and STN to their corresponding 3T image.

CNNs trained on all three datasets yielded Dice Score (DS) with a minimum average of 0.77, occurring for the STN, and a maximum average of 0.94, occurring for the RN. The CNNs trained on the 7T images achieved on average the best results, and had the smallest deviations, with a DS of 0.94 ± 0.01, 0.90 ± 0.01 and 0.89 ± 0.02 for the RN, SN and STN, respectively. The PSSC was segmented with a DS of 0.80 ± 0.03 and the substructures of the PMC were segmented with a DS of 0.86 ± 0.02, 0.86 ± 0.04 and 0.86 ± 0.02 for the Arm, Face and Omega, respectively. The predicted susceptibility values from the Swedish 3T datasets and the 7T dataset gave a Mean Absolute Percentage Error (MAPE) lower than 5% for RN, SN og STN, when compared to

the ground truth. The Swedish 3T CNNs were most accurate when predicting values for RN and SN, while the 7T CNNs predicted more accurate values for the STN. Meanwhile, for the last 3T dataset, the MAPE varied between approximately 26% and 44%. For the intra-subject scanner comparison between the Norwegian 3T dataset and the 7T dataset, the different scanners yielded highly varying results, with both large deviations and very weak correlation between the measurements. Furthermore, the inter-subject variability in susceptibility values for all datasets were large.

For all three datasets, automatic segmentation of the RN and SN yielded better results than the intra-rater variability of professionals, indicating higher consistency. In the case of the 7T data, this was true for STN, PMC and PSSC as well. The segmentation accuracy of PMC and PSSC showed potential to be used in further ALS research. However, it is recommended that the CNNs are further trained on a dataset with larger variation in age. The CNNs trained on the Swedish dataset and CNNs trained on the 7T dataset yielded on average the most accurate susceptibility values. It was found that the quality of the manual segmentations obtained on the Norwegian 3T dataset were not adequate, and could not to be used as ground truth, thus the CNNs trained on this data yielded highly varying results. This was emphasised when there was found no correlation between intra-subject susceptibility values when comparing 3T and 7T susceptibility values. Inter-subject variation showed that using RN and SN as bio-markers for single-scan QSM imaging in PD diagnosis was not feasible, as the natural variations in susceptibility values of RN and SN are too large.

Sammendrag

For å anskaffe informasjon vedrørende hjernestrukturer sentrale i diagnosistisering av Parkinsons sykdom og Amyotrofisk lateralsklerose ufører man gjerne manuelle segmenteringer av de interessante regionene (ROIs). Dette er en jobb som både er tidkrevende og vanskelig. En automatisering av denne prosessen ved bruk av CNNs (Convolutional Neural Networks) kan både spare legen for timer med arbeid, men også forbedre kvaliteten på segmenteringene, og dermed forbedre diagnostisering.

40 Quantitative Susceptibilty Mapping (QSM) bilder med manuelle segmenteringer av Red Nucleus (RN), Substantia Nigra (SN) og Subthalamic Nucleus (STN) ble donert av en svensk forskningsgruppe. I tillegg ble 15 frivillige skannet på en 3T og 7T skanner lokalt på St.Olavs sykehus. QSM-bildene ble rekonstruert ved bruk rekonstruksjonsalgoritmen total generalised variation. RN, SN, STN, Cerebrospinal Fluid, Primary Motor Cortex (PMC) og Primary Somatosensory Cortex (PSSC) ble alle manuelt segmentert på 7T QSM-bildene. Intra-subjekt co-registrering ble utført mellom de norske 7T og 3T bildene, og RN, SN og STN-maskene ble transformert fra 7T bildene til deres korresponderende 3T bilde.

CNNs trent på alle tre datasettene opnådde en gjennomsnittelig Dice Score (DS) på minimum 0.77, ved segmentering av STN, men maksimum gjennomsnittlig oppnådd DS var 0.94, ved segmentering av RN. CNNs trent på 7T datasettet ga i snitt de beste resultatene med høyest DS, i tillegg til å ha de laveste avvikene med 0.94 ± 0.01, 0.90 ± 0.01 og 0.89 ± 0.02 for RN, SN og STN, respektivt. PSSC ble segmentert med en DS på 0.80 ± 0.03, og understrukturene til PMC ble segmenert med en DS på 0.86 ± 0.02, 0.86 ± 0.04 and 0.86 ± 0.02 for Arm, Face og Omega, respektivt. De predikerte suseptibilitetsverdiene oppnådd på det svenske 3T og det norske 7T datasettet ga de minste gjennomsnittlige absolutt prosentvis avvik (MAPE), mindre enn 5% for RN, SN og STN da de ble sammenlignet med faktiske verdier. CNNs trent på det svenske datasettet ga mest nøyaktige verdier for RN og SN, mens CNN trent på 7T ga mest nøyaktige verdier for STN. For det resterende datasettet så varierte MAPE mellom ca. 26% og 44%. Intra-subjekt skanning som sammenlignet datasettene fra St.Olavs avslørte sterkt varierende suseptibiltitetsverdier i korresponderende ROIs, med både store avvik og tilsynelatende ingen korrelasjon. Videre så kunne man se at inter-subjekt variasjonen i suseptibilitetsverdiene for RN, SN og STN var meget stor.

For alle datasettene så var segmenteringen av RN og SN bedre enn intra-rater variasjonen fullført av profesjonelle. For 7T dataene så gjaldt dette også for STN, PSSC og PMC. Dette indikerer at CNNs fra dette prosjektet trent på å segmentere PMC og PSSC har potensial til å bli brukt i videre forskning. Før de tas i bruk så anbefales det å trene dem på et mer variert datasett, med høyere snittalder. Det viste seg at det svenske 3T datasettet og 7T datasettet hadde de laveste MAPE. Ettersom at CNN trent på det norske 3T datasettet hadde veldig høy MAPE, så kan dette forklares med at den manuelle segmneteringen ikke var vellykket. Dette ble videre bevist da man ikke fant en sammenheng mellom suseptibilitesverdier ved intra-subjekt skanning for de norske 3T og 7T datasettene. Inter-subjekt variasjonene innad i RN og SN var så store at disse ikke kunne brukes som biomarkører i PD ved singel-skanning QSM-avbildning.

Acronyms

ALS	Amyotrophic Lateral Sclerosis
ASPIRE	A Simple Phase Image Reconstruction For Multi-Echodata
BET	Brain Extraction Tool
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSF	Cerebrospinal Fluid
DL	Deep Learning
DoF	Degrees Of Freedom
DS	Dice Score
FAST	FMRIB's Automated Segmentation Tool
FID	Free Induction Decay
FLIRT	FMRIB's Linear Image Registration Tool
FMRIB	Functional Magnetic Resonance Of The Brain
FN	False Negative
FNIRT	FMRIB's Non-Linear Image Registration Tool
FP	False Positive

FSL	FMRIB's Software Library
GPU	Graphics Processing Unit
GRE	Gradient Echo
GUI	Graphical User Interface
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MEDI	Morphology Enabled Dipole Inversion
MGRE	Multi Gradient Echo
MR	Magnetic Resonance
MRI	Magnetic Resonance Imaging
NMR	Nuclear Magnetic Resonance
NTNU	Norwegian University Of Science And Technology
PD	Parkinson's Disease
PMC	Primary Motor Cortex
PSSC	Primary Somatosensory Cortex
QSM	Quantitative Susceptibility Mapping
ReLU	Rectified Linear Unit
RESHARP	Regularised Enabled Sophisticated Harmonic Artifact Reduction For Phase
RF-Pulse	Radio Frequency Pulse
RN	Red Nucleus
SD	Standard Deviation
SHARP	Sophisticated Harmonic Artifact Reduction For Phase
SN	Substantia Nigra
STN	Subthalamic Nucleus

TGV	Total Generalised Variation
TN	True Negative
TP	True Positive
TV	Total Variation
VOI	Volume Of Interest
W.r.t.	With Respect To

Contents

Preface **i**

Abstract **iv**

Sammendrag **vi**

Acronyms **ix**

1 Introduction **1**

1.1 Motivation 1

1.2 Project Description 2

1.3 Project Goals and Research Questions 3

1.4 Contributions 4

1.5 Report structure 4

2 Theory **5**

2.1 Degenerative Nerve Diseases 5

2.1.1 Red Nucleus, Subtantia Nigra and Subthalamic Nucleus 6

2.1.2 Primary Motor Cortex 8

2.1.3 Primary Somatosensory Cortex 8

2.2 Nuclear Magnetic Resonance 9

2.2.1 Magnetisation 10

2.2.2 Rotating the Magnetisation and Relaxation 11

2.2.3 Free Induction Decay 14

2.3 Acquisition Methods 14

2.3.1 Gradient Echo and Multi Gradient Echo 15
2.4 Quantitative Susceptibility Mapping 16
2.4.1 Types of Magnetic Materials in the Brain 16
2.4.2 Phase to Field 19
2.4.3 A Simple Phase Image Reconstruction for multi-Echodata 20
2.4.4 T_2^*-Weighted Magnitude Images 21
2.4.5 Background Field Removal 22
2.4.6 Dipole Field Inversion 26
2.4.7 Morphology Enabled Dipole Inversion 28
2.4.8 Total Generalised Variation 31
2.4.9 Choosing a Reference Susceptibility Value 33
2.5 Linear and non-linear image registration 33
2.5.1 Linear Image Registration 33
2.5.2 Non-Linear Image Registration 35
2.6 Deep Learning 36
2.6.1 Artificial Neural Networks 36
2.6.2 Convolutional Neural Networks 42
2.6.3 Overfitting 45
2.6.4 U-Net 45

3 Material and Methods **47**
3.1 Setting up HUNT Cloud 47
3.2 Data Acquisition 49
3.2.1 MRI Acquisition GE 3T 49
3.2.2 MRI Acquisition Siemens 3T 49
3.2.3 MRI Acquisition 7T 50
3.3 QSM Acquisition 50
3.3.1 QSM Acquisition GE 3T 50
3.3.2 QSM Acquisition Siemens 3T 51
3.3.3 QSM Acquisition 7T 51
3.4 Manual Tracing 52
3.5 Intra-subject Label Transfer 52
3.6 Data Augmentation 55
3.7 Network Architecture 56
3.7.1 U-Net 56
3.8 Training and Evaluation 58

3.8.1 Training on the GE 3T dataset 58
3.8.2 Training on the Siemens 3T and 7T datasets 58
3.8.3 Evaluation Metrics 59

4 Results 61
4.1 Image Segmentation 61
4.1.1 RN, SN and STN Image Segmentation on GE 3T Data 61
4.1.2 RN, SN and STN Image Segmentation on Siemens 3T Data 63
4.1.3 RN, SN and STN Image Segmentation on Siemens 7T Data 64
4.1.4 PMC and PSSC Image Segmentation on Siemens 7T Data 65
4.2 Extracting Susceptibility Values 65
4.2.1 Susceptibility values: RN, SN and STN GE 3T 65
4.2.2 Susceptibility values: RN, SN and STN Siemens 3T 66
4.2.3 Susceptibility values: RN, SN and STN Siemens 7T 68
4.2.4 Comparing Susceptibility Values: Siemens 3T and 7T 69

5 Discussion 70
5.1 Image Segmentation 70
5.1.1 RN, SN and STN Image Segmentation on GE 3T Data 70
5.1.2 RN, SN and STN Image Segmentation on Siemens 3T Data 72
5.1.3 RN, SN and STN Image Segmentation on Siemens 7T Data 74
5.1.4 PMC and PSSC Image Segmentation on Siemens 7T Data 75
5.2 Extracting Susceptibility Values 76
5.2.1 Susceptibility values: RN, SN and STN GE 3T 76
5.2.2 Susceptibility values: RN, SN and STN Siemens 3T 77
5.2.3 Susceptibility values: RN, SN and STN 7T 79
5.2.4 Comparing Susceptibility Values Obtained: Siemens 3T and 7T . 80
5.2.5 Using Susceptibility Values as Bio-Marker for Neurodegenerative Diseases 81
5.3 Further Work 81

6 Conclusions 84
References 87

A Appendix A 98
A.1 Additional Results 98

B Appendix B **105**

B.1 HUNT Cloud Documentation 105

C Appendix C **112**

C.1 Pre-processing of data 112

C.2 U-Net 114

C.3 Training 119

C.4 Obtaining results and post-processing of the results 121

C.5 Helper functions 124

Chapter 1

Introduction

This chapter gives an introduction to the thesis written this spring. It will present the motivation behind the choice of the thesis and introduce the project description, goals, research question and the contributions. The last section concerns the report structure of the thesis.

1.1 Motivation

Nearly 10 million people worldwide are diagnosed with Parkinson's Disease (PD) (1) and nearly 250 thousand people are diagnosed with Amyotrophic Lateral Sclerosis (ALS) each year (2). To deliver an effective treatment, it is advantageous to uncover these diseases at an early stage, which in some cases may be accomplished using Magnetic Resonance Imaging (MRI) (3)(4). PD is tightly connected to the degradation of the iron rich mid-brain structures Red Nucleus (RN), Substantia Nigra (SN) and Subthalamic Nucleus (STN) (5). As they are all encapsulated in a molecular complex called Ferritin, they have high iron concentrations, relative to their neighbouring structures, making them rich in contrast in Quantitative Susceptibility Mapping (QSM). Thus, the degradation can be observed particularly when the SN and RN experiences increased iron depositions in their nuclei, in turn increasing contrast. ALS can at an early stage of the disease be viewed in QSM images, as one will observe that the Corticospinal Tract has a high intensity. As the disease develops, one can also observe an increased intensity and volume loss of the entire tract, ranging from the spinal cord to the motor strip (6). Particularly in QSM imaging, one can also observe loss of signal around the pre-central and post-central Gyrus due to iron deposition in the cortex (7).

In the case of PD, for a radiologist to make a conclusion purely based on QSM images, the doctor usually performs manual segmentations of the relevant brain structures to obtain a status report about the degradation of mainly RN, SN, and to some degree STN, based on their respective susceptibility values. Or in the case of ALS, to obtain important information about the Primary Somatosensory Cortex (PSSC) and the Primary Motor Cortex (PMC), could be crucial to make a diagnosis (8). However, manual segmentation is a time consuming and expensive use of specialised labour, often prone to errors. An automation of this process has the potential to be more accurate, cost effective and significantly quicker, which in turn can be a part of the radiologist's toolbox when diagnosing patients.

Currently, the most widely used research segmentation tools, such as the FMRIB's Automated Segmentation Tool (FAST), from the Functional Magnetic Resonance of the Brain (FMRIB) Software Library (FSL) (9) or segmentation tools from Freesurfer (10) do not support segmentation of the relevant brain structures for PD and ALS. Currently, the doctors at St.Olavs Hospital in Trondheim do not have any tools for automatically segmenting brain structures. The introduction of Deep Learning Deep Learning (DL) has exploited the abundance of data, and has yielded ground braking results within the field of digital image processing (11). Thus, techniques from DL, in particular Convolutional Neural Networks (CNNs), have been implemented in this thesis to perform the segmentations.

1.2 Project Description

This project is a continuation of the DL segmentation project the author initiated the autumn of 2020, and some of the data, code and report is built on this. The data in this paper originated from 3 different sources; 40 3T QSM images were donated by a Swedish research group at Karolinska Institutet. These images had manual segmentations of the RN, SN and STN. An additional 15 7T QSM images were acquired from healthy volunteers recruited by the author. The RN, SN, STN, Cerebrospinal Fluid (CSF), PMC, PSSC were manually labelled by a neurologist on the 7T dataset. 15 3T QSM images of the same volunteers were acquired and linear co-registration was performed for intra-subject mapping of the RN, SN, STN and CSF from the 7T images to the 3T QSM images. To perform the QSM reconstructions, MRI data were transferred to the cloud community, HUNT Cloud. The lab environment was set up to perform QSM reconstructions on dedicated, secure servers.

The manually labelled datasets were then used as training data for CNNs. The goal of the training was to create CNNs which could perform accurate and precise automatic segmentations. The precision of the segmentations of the CNNs were then evaluated by comparing them to the manually contoured images that had not been used as training data, and the segmentation accuracy of 3T vs 7T images were compared. Additionally, the CNNs were used to extract susceptibility values of the regions of interest (ROIs). An intra-subject susceptibility comparison was performed to compare the susceptibility values across magnetic field strengths. Lastly, an analysis regarding the feasibility of using the RN and SN as biomarkers for PD was completed.

1.3 Project Goals and Research Questions

There were multiple goals to this project. The first goal was to create a database of manually segmented 3T and 7T QSM images that could be used in training of CNNs. Automatic segmentation accuracy of the 3T and 7T images were compared, as it was of interest to see whether or not the new 7T MRI scanner at Norwegian University of Science and Technology (NTNU) showed improvement in medical imaging of brain structures critical in PD and ALS. CNNs were applied to create segmentation method which quickly could extract crucial information, such as susceptibility values, volume and position from the ROIs, which then could be implemented at St.Olavs as a helpful tool for the radiologists. The third goal was to see if the susceptibility values of the RN, SN and STN were suitable as a bio-markers for PD diagnosis, and if the segmentation accuracy of the PMC and PSSC were good enough to be applied in ALS research.

Additionally, the Magnetic Resonance (MR) physics group has recently started using the cloud community HUNT Cloud, and this will be the main neuroimaging research platform used by NTNU in the years to come. Thus, a component of this project was to initialise and setup the virtual lab, which would allow for new master- and Ph.D. students to quickly start their work in the lab, without the need to perform the tedious initialisation steps. Furthermore, a large of part of this sub-task was to create a data pipeline for QSM reconstruction methods and Graphics Processing Unit (GPU) accelerated training of neural networks.
The main research questions in this thesis are summarised below:

- **RQ1** How accurate can CNNs segment the ROIs at 3T and 7T scanners, and how accurate susceptibility values can be extracted?

- **RQ2** For intra-subject imaging, is QSM imaging reproducible across scanners, i.e. is the same susceptibility values obtained for 3T and 7T?

- **RQ3** Is it feasible to use susceptibility values as a bio-marker for diagnosis of PD?

1.4 Contributions

This thesis has made the following contributions: The MR physics group at St.Olavs has now a database with 70 QSM images with manual labelling. 55 of which are 3T images with RN, SN and STN segmentations and 15 are 7T images with segmentations of RN, SN, STN, CSF, PMC and PSSC. The HUNT Cloud lab has been set up and all the necessary software for neuroimaging reserach and accelerated training of neural networks are now installed. A documentation has also been written, making it possible for new users to start with their work in the cloud immediately. A fully automatic end-to-end segmentation software has been produced and trained, yielding high precision segmentations and has the ability to extract information about the volume, position and susceptibility values in seconds. The product from this thesis is available for researchers at NTNU and St.Olavs and can be used as foundation for further research within the field of QSM. In addition, a multi-national partnership with the MR physics and neuro science group at Karolinska institutet has been established. Data and results have been exchanged between the two institutions and a joint abstract was sent to the 2021 conference hosted by the International Society for Magnetic Resonance in Medicine.

1.5 Report structure

This report is divided into 6 chapters, in addition to preface, abstracts and appendix. Chapter 2 gives an introduction to the relevant theory behind the neuroscience, MR physics, QSM reconstruction pipeline and DL. Chapter 3 concerns the material and methods and describes the implementation of the theory, while chapter 4 presents the results obtained in this thesis. Chapter 5 displays an in depth interpretation of the results obtained in chapter 4. Chapter 6 makes some conclusions based on the information presented in chapter 4 and 5. Lastly, the appendix presents additional information about the methods and results obtained in this thesis. This thesis mainly presents new information, but some of the theory and methods are quoted from the specialisation project written during the autumn of 2020. It will be clearly stated at the start of each chapter if some information is reprinted from the specialisation project.

Chapter 2

Theory

This chapter concerns the background theory, which is crucial for understanding the methods implemented in this project. The first section focuses on Neurodegenerative and Motor Neuron Diseases. The two following section focuses on the physics behind MR and different acquisition methods. Chapter 2.4 is a large section which gives the reader a comprehensive introduction to the field of QSM and its challenges. The following section gives a brief introduction to the basics of image co-registration in MRI, and the final section concerns the theoretical foundation behind DL. The following sections are based on information presented in the specialisation project: section 2.1.1 about the RN, SN and STN, section 2.2 about MR physics, section 2.4.1 - 2.4.2 about magnetic materials and phase to field, section 2.4.5, not including RESHARP, section 2.4.6 describing the dipole field inversion problem and chapter 2.6 about DL.

2.1 Degenerative Nerve Diseases

Neurodegenerative diseases are diseases which affect the brain, spinal cord or more peripheral nerves, causing a step-wise degradation of the nervous system (12). PD is an example of such a disease. Motor neuron disease is a collection of diseases where the motoric neuron system is affected, leaving the remainder of the nervous system intact. The degradation of these nervous cells causes loss of muscles, paralysis and in most cases death. The most commonly known motor neuron disease is ALS.

PD occurs due to loss of brain cells in the brain stem. In Norway, approximately approximately 8000 people are currently diagnosed with this disease (13), with an increasing frequency at increasing age. Dopamine is a key constituent in these brain cells, and they are freed around the nerve cells' terminal in the basal ganglia. Loss of dopamine is believed to be the root of the symptoms in PD. Why brain cells die during the course of the disease is still disputed, but it is believed that accumulation of α-synuclein in the nerve cells creates Lewy Bodies which influences the behaviour of chemicals in the brain, being a destructive influence (14). PD is mainly defined by 2 out of 3 symptoms; considerable tremor while sitting still, muscle stiffness and reduced ability to perform will-controlled movements. Currently, there exists no medicines which can effectively treat PD, however, there are different drugs which may dampen the symptoms. As PD is caused by lack of dopamine, dopamine is often used to counter the symptoms. However, as dopamine can cause strong hallucinations and in some cases worsen the symptoms, this must be strictly regulated.

ALS affects both the upper and lower motor neurons resulting in fatal outcome, affecting approximately 400 people nationwide, and the cause of ALS is still not understood (15). The disease affects the nerve cells which are connected to the will-controlled motion of muscles, however, the remaining parts of the nervous system is left untouched. Muscles are controlled by the central nervous system and are connected by several neural pathways by the upper and lower motor neuron. Usually the upper and lower neuron is influenced, effectively stopping the transmission of signal from the brain to the muscles. A simple illustration of this can be seen in Figure 2.1.1. ALS develops individually, but the common factor is paralysis and death within few years. The first symptoms are partial paralysis, muscle weakness and abnormal reflexes. ALS is most commonly uncovered during clinical testing, electro-mammography or MRI. Despite advancements within the fields of genetics and molecular biology, the cause and development still remain unrecognised, meaning that there are no effective treatment methods available.

2.1.1 Red Nucleus, Subtantia Nigra and Subthalamic Nucleus

The RN, SN and the STN, depicted in Figure 2.1.2, are all pairwise midbrain structures having relatively high susceptibility values. The RN is an oval-shaped structure that received its name from the red colour it exhibits in a freshly dissected human (16). This colour is believed to be correlated with the high levels of iron pigments within the cytoplasm of its neurons. By the use of functional MRI, the RN has been associated with speech production, pain processing and sensory discrimination. An increase in iron levels has been proven to be related with Parkinson disease (17).

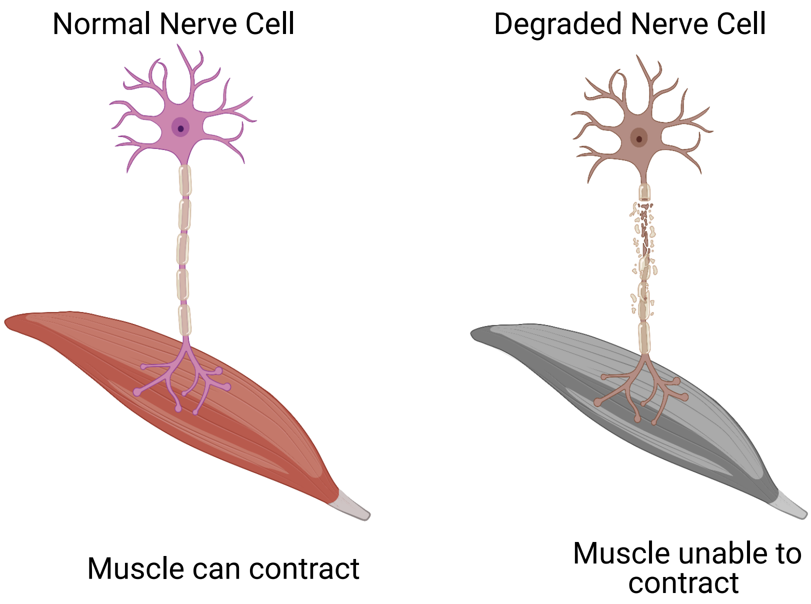


Figure 2.1.1: The left figure shows how a normal functioning nerve cell behaves. The figure on the right shows a nerve cell with sclerosis, unable to signal the muscle to contract. The illustration was created by Eivind Lysheim using BioRender.com

The SN is a midbrain dopaminergic nucleus which is crucial for motor movement and reward functions. The Nigrostriatal Pathway, the connection between SN and the Putamen, is central in the loss of motor functions during the course of Parkinson disease. It is believed that the SN is the part of the brain that suffers the most damage during the disease course. The disease causes progressive and irreversible loss of neurons in the SN. The loss of neurons are associated with symptoms such as hypokinesia, rigidity and and resting tremor (18).

The STN is a constituent of the Basal Langlia system, and is a large component of the Subthalamus. It is linked to motor control, but it also plays a role in attention, motivation and response inhibition. Furthermore, it also coordinates impulses from from Cortical and Sub-Cortical neurons, responsible for emotional pattern. As well as for the SN, during the course of Parkinson disease, the STN looses large levels of dopamine, which causes damage to central functions (19).

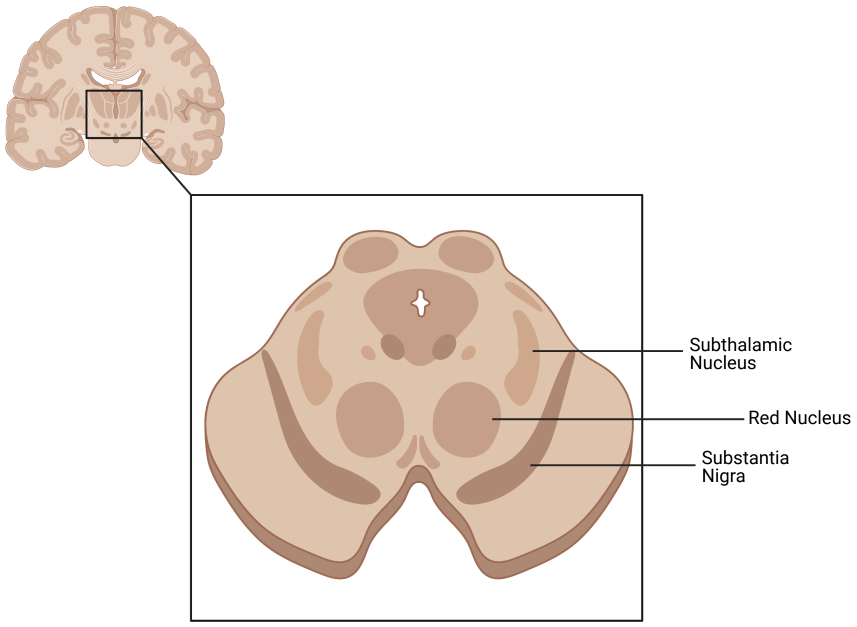


Figure 2.1.2: An illustration showing components of the midbrain including RN, SN and STN. The illustration was created by Eivind Lysheim using BioRender.com

2.1.2 Primary Motor Cortex

The PMC is situated in the Precentral Gyrus, as seen in Figure 2.1.3, and is the primary region of our motor system. This brain region is a key component in the planning and execution of movements, and works in parallel with other central motor regions, such as the premotor cortex, supplementary motor area and the posterior parietal cortex (20). It is only the region of the brain that is populated with Betz cells which anatomically is described as the PMC. The Betz cells are constituted by huge neurons which send their axons down to the spinal cord via the corticospinal tract. The axons are connected to the horn cells, via synapses, which in turn are directly linked to muscles. Lesions can results in paralysis of its connected side of the body.

2.1.3 Primary Somatosensory Cortex

The PSSC is a constitutent of the somatosensory system and is situated in the Postcentral Gyrus, as seen in Figure 2.1.3. The PSSC is the primary receptor of sensations in the body. Thalamic radiations transmit signal from muscles, tendons, skin and joints to the

PSSC (21). Lesions affecting the PSSC can cause symptoms such as loss of fine touch, vibration and proprioception.

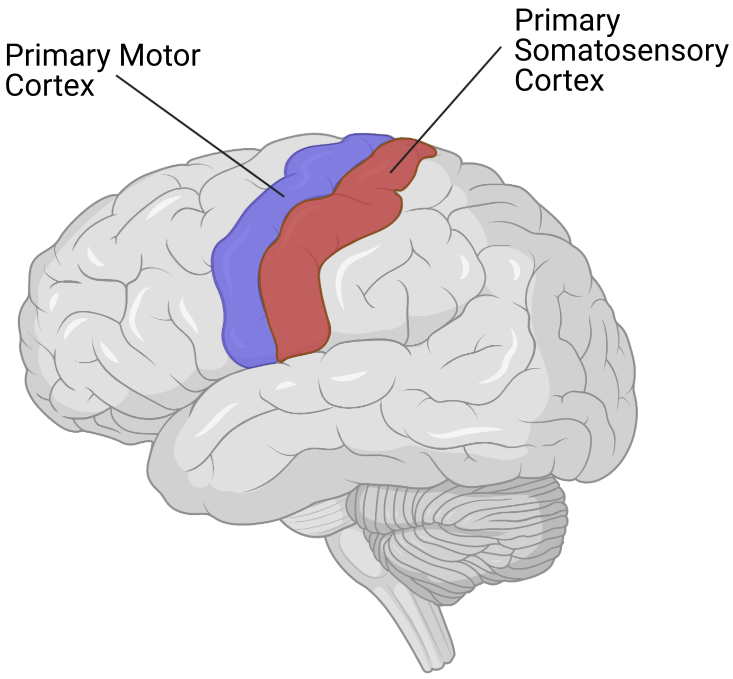


Figure 2.1.3: The PMC and the PSSC. The illustration was created by Eivind Lysheim using BioRender.com

2.2 Nuclear Magnetic Resonance

The field of Nuclear Magnetic Resonance (NMR) exploits the interactions between tissue in a strong and highly homogeneous magnetic field and non-ionising electromagnetic radiation. When a body is immersed in a static magnetic field, we will observe splitting in the nuclear spin energy of atoms and molecules, leading to discrete energy gaps. By introducing an electromagnetic pulse with an energy equal to that of the energy difference between the energy levels, the molecules in the lower energy state can be excited to a higher energy level. The system will then experience two kinds of relaxation processes, T_1 and T_2-relaxation. As the spins are subject to slightly different magnetic fields, the spins will loose its coherence and dephase, a T_2 relaxation. Simultaneously, the loss of magnetisation will induce a measurable electromotive force. Meanwhile, the magnetisation will evolve back to equilibrium, aligning itself with the magnetic field itself, a T_1-relaxation. The entire derivation of the NMR phenomenon is not shown here. For a detailed deduction, there exists many online sources and books available, such as Magnetic Resonance Imaging: Physical Principles and Sequence Design, Second Edition (22).

2.2.1 Magnetisation

In the case for an ensemble of nuclei, the population difference between the energy levels can be calculated using the Boltzmann equation for two populations, with an energy difference proportional to the Larmor frequency:

$$\frac{n_\alpha}{n_\beta} = exp(\frac{-\Delta E}{k_b T}) = exp(\frac{\hbar \gamma B_0}{k_b T}) \tag{2.2.1}$$

where n_α and n_β are the population of parallel and anti-parallel spins, respectively, ΔE is the energy difference between the energy levels, k_b is Boltzmann's constant , T is the absolute temperature, $\hbar$ is the reduced Planck's constant, γ is the gyromagnetic ratio which is dependent on the isotope and B_0 is the external magnetic field. We wish to maximise the magnetisation, as this will increase the signal received. The reason behind this will become apparent in the following sections. To increase the magnetisation, the population difference has to be increased. As $\Delta \mathrm{E} = \gamma \frac{h}{2\pi} B_0$, one can see that the gyromagnetic ratio and the strength of the magnetic field is directly proportional to the difference in energy levels. An increase in either will increase ΔE, and in turn increase the population difference. Changing the former means using nuclei with higher γ, while the latter means creating stronger magnets. The last point is a good argument why NTNU has recently installed a new 7T MR scanner, as this has the potential to create improved images compared to the widespread 1.5T and 3T MRI scanners. Another possibility is to use nuclei that are more abundant in the material being studied, resulting in a larger population difference, and in turn a stronger signal.

When the spins are distributed over the energy states, in the classical picture, the populations in the energy levels will presses at the same frequency, parallel or anti-parallel to the direction of B_0. As the spins are precessing out of phase, the components in the xy-plane, the transversal plane, will cancel out, and there will only be a net magnetisation component in the positive z-direction, the longitudinal direction. The magnetisation in the transversal plane and the longitudinal direction is denoted as M_{xy} and M_z, respectively. By calculating the difference in spin populations in the energy states, the magnetisation can be determined. For nuclei with spin quantum number I =

1/2, such as the $^1_1$H nucleus of water, and for a nuclei with spin I > 1/2, the equilibrium magnetisation in the z-direction is respectively given as:

$$M_0 = \frac{n\hbar^2\gamma^2}{4k_BT}B_0 \quad M_0 = \frac{n\gamma^2\hbar^2 I(I+1)}{3k_BT}B_0 \tag{2.2.2}$$

Again, notice the impact γ and B_0 has on the magnetisation.

2.2.2 Rotating the Magnetisation and Relaxation

The only way to observe the nuclear magnetisation is to detect a change in precessional motion of the spins in the transversal plane. However, at equilibrium, when there is no phase coherence and the only magnetisation vector is the static M_z, this is not possible. One can introduce a way to tip the system out of equilibrium, i.e. tip the net magnetisation vector from only being in the longitudinal plane into the transversal plane. This can be done by introducing a second oscillating electromagnetic wave, B_1, orthogonal to the B_0-field with the same angular frequency as the Larmor frequency. In other words, an electromagnetic wave that has an energy corresponding to the energy gap between the Zeeman levels, which is proportional to the γ and B_0.For a magnetic field, B_0 = 1T, the frequency of such a wave is, f = 42.6 MHz. Clearly within the radio frequency part of the electromagnetic spectrum, giving rise to the name Radio Frequency Pulse (RF-Pulse).

The B_1-field is only applied during the rotation of the magnetisation. The RF-Pulse is created by running a current through a coil for a finite amount of time. The magnetisation vector M_z experiences a torque from the RF-Pulse. During an excitation pulse, both B_1 and B_0 is acting on M_z, causing the magnetisation vector to have a perplexing motion towards the transversal plane, as seen in Figure 2.2.1. Here, both the laboratory frame and the rotating frame have been included. The difference between the two frames is that in the former, the viewer watching the magnetisation is stationary. In the latter, the viewer follows the rotating path of magnetic moment in the xy-plane of the magnetisation, making it seem like the magnetisation is only moving in the z-direction.

The angle which the B_1-field is rotated from the equilibrium magnetisation, M_0, to the transversal plane is not arbitrary, but in ideal conditions given by relation of the nutation angle:

$$\alpha = \gamma B_1 t \tag{2.2.3}$$

Where α is the angle which the magnetisation vector is tilted and t is the amount of

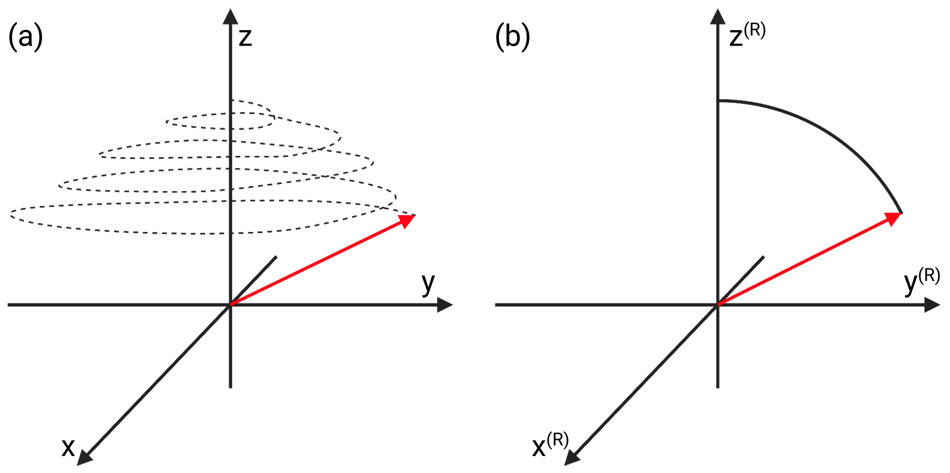


Figure 2.2.1: The figure to the left shows the complex path of the spin that has been tipped towards the transverse plane in the laboratory frame of reference. The figure to the right shows the same event, but in the rotating frame of reference. This figure is an illustration, and length of the vectors may not necessarily be in scale. Figure created by Eivind Lysheim in BioRender.com.

time the coil produces the electromagnetic pulse. By choosing an appropriate t, one can for example create a 90° pulse which rotates the entire magnetisation vector into the transversal plane. After the RF-pulse is turned off, the spins feel a slightly different magnetic field, causing the spins to loose coherence. This is known as T_2 relaxation. When the magnetisation evolves back to equilibrium, this is known as T_1 relaxation. When the transversal magnetisation is precessing around B_0 and loosing coherence, in turn loosing magnetisation, it will induce and electromotive force in the receiver coil that is placed around the sample being studied. This electromotive force can be described by Faraday's law of induction. It is this electromotive force that is the origin of the measured NMR signal.

The Transverse Relaxation Time: T_2

After an initial rotation of the equilibrium magnetisation, i.e. $M_{xy} \neq 0$, the nuclei are approximately in phase, with some deviations due to imperfections of the RF-pulse. The nuclei are all spinning with a Larmor frequency, but due to local magnetic fields created by the electromagnetic interactions between the nuclei and inhomogenities in B_0, some precess faster than others. This causes a dephasing of the rotating nuclei. The more time that passes, the greater the phase difference becomes.

T_2 is the true decay constant that describes decay to equilibrium for the transverse magnetisation in a uniform magnetic field B_0. Only spin-spin interactions are affecting the local magnetic field. However, when we talk about the transverse relaxation in non-uniform magnetic fields, the relaxation constant is commonly denoted T_2^*, the observed decay constant. Here, both spin-spin interactions and inhomognenites in B_0 is accounted for. The transverse relaxation is caused by the loss of phase coherence, but not of energy, meaning that transverse relaxation is an entropic process. After an initial transverse magnetisation at $t = 0$, the magnetisation will decay to zero, following the equation in the rotating frame:

$$\frac{dM_{xy}(t)}{dt} = -\frac{1}{T_2} M_{xy}(t) \tag{2.2.4}$$

$$\Rightarrow M_{xy}(t) = M_{xy}(0) e^{-t/T_2} \tag{2.2.5}$$

Where $M_{xy}(0)$ is the initial magnetisation in the transverse plane. A uniform magnetic field has been assumed. It follows from Equation 2.2.5 that the physical interpretation of T_2 is the time it takes to reduce the transverse magnetisation with a factor of e.

The Longitudinal Relaxation Time: T_1

Longitudinal relaxation occurs when the magnetisation vector M_z relaxes from $M_z \neq M_0$ to the equilibrium state $M_z = M_0$. The velocity of the relaxation process is governed by the time the spins use to distribute themselves on the energy levels according to the Boltzmann distribution, and is therefore proportional to M_0 - M_z. The longitudinal relaxation is related to the spin distribution among the energy levels and is effectively related to a process of energy exchange with the environment in the form of thermal energy, such as collisions, rotations or electromagnetic interactions. However, as this energy is small compared to the average kinetic energy of the molecules, this energy is quickly dispersed. As M_z evolves back to equilibrium, the total energy of the system decreases, as protons favour lower energy states. The longitudinal magnetisation is given by the Bloch equation:

$$\frac{dM_z}{dt} = -\frac{M_z - M_0}{T_1} \tag{2.2.6}$$

$$\Rightarrow M_z = M_0(1 - e^{-t/T_1}) \tag{2.2.7}$$

From the equation above it follows that T_1 is the time required to reduce the difference between M_z and its equilibrium by a factor of e.

2.2.3 Free Induction Decay

When the transversal magnetisation relaxes, i.e. exponentially evolving towards zero, the precession induces an electromotive force, which can be measured using a receiver coil. The signal is given as the induced current in the receiver coil, which is described by the Faraday's law of induction:

$$\varepsilon = -\frac{d\Phi_M(t)}{dt} \tag{2.2.8}$$

where ε is the induced electromotive force, and $\Phi_M(t)$ is the magnetic flux. When the magnetic vector precess, it induces a damped oscillating signal into the coil, the Free Induction Decay (FID).

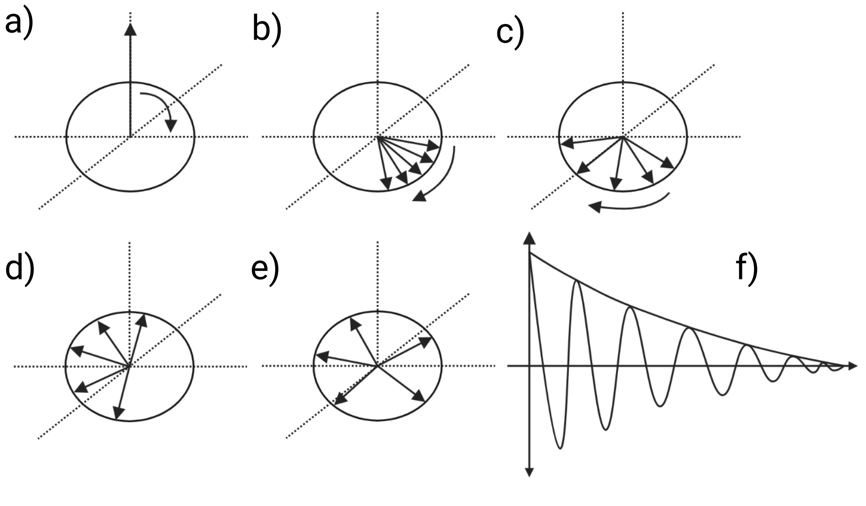


Figure 2.2.2: A figure showing the excitation and the subsequent FID originating from the dephasing of the spins in the transverse plane. (a) shows an excitation of 90°. (b), (c), (d) and (e) shows dephasing of the spins. (f) shows the observable NMR signal generated by the dephasing of spins. Figure created by Eivind Lysheim in BioRender.com

2.3 Acquisition Methods

This section will introduce and describe the relevant pulse sequences applied in this in project, i.e. sequences relevant for the QSM acquisition.

2.3.1 Gradient Echo and Multi Gradient Echo

The Gradient Echo (GRE) is a manipulation of the FID. After a 90° pulse, instead of letting the spins dephase naturally, with a decay constant T_2^*, a dephasing gradient is used to accelerate the dephasing of the spins by locally changing the magnetic field, thus altering the resonance frequency, in turn stimulating a quicker dephasing. After the decay, a rephasing gradient is applied with the same strength and opposite polarity, reversing the dephasing of the first gradient, causing an echo. A standard GRE pulse sequence can be seen in Figure 2.3.1

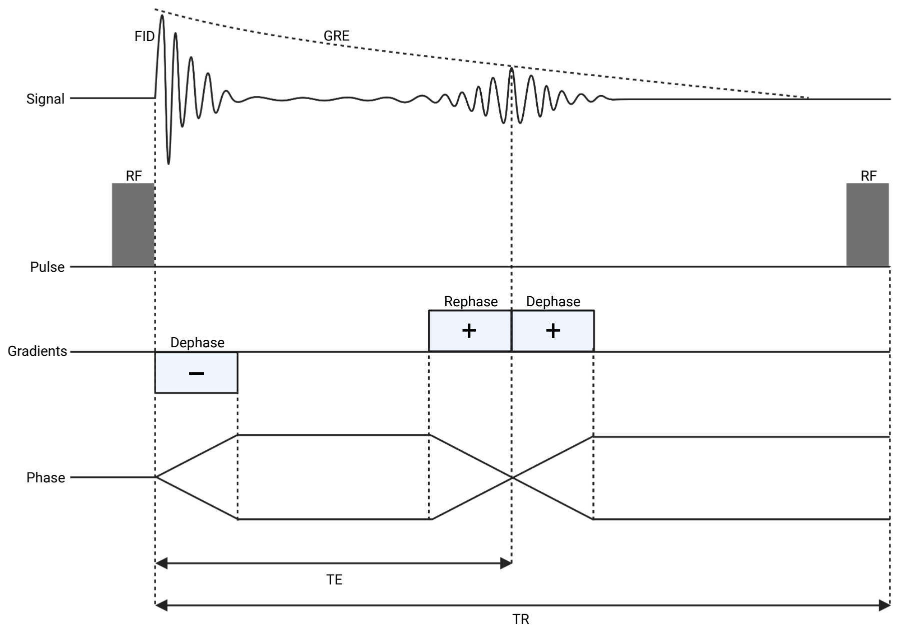


Figure 2.3.1: The schematics for a standard GRE pulse sequence. The initial RF-pulse causes a tipping of the magnetisation vector into the M_{xy} plane. The dephasing of the transveral magnetisation is accelerated by the application of magnetic gradients, causing the FID. The spins are then rephased causing a gradient echo signal before again being dephased by gradients with opposite polarity and same magnitude as the initial dephasing gradietn. The phase chart shows a graphical description of the phase, where the spins are in phase when the two lines are crossing. TE and TR are shown in the bottom of the figure. Illustration was created by Eivind Lysheim using BioRender.com

A Multi Gradient Echo (MGRE) sequence uses the same principles, but the rephasing and dephasing operations are repeated as long as the time that has passed since the initial RF-pulse is less than the T_2^* decay. A typical MGRE sequence looks identical to

Figure 2.3.1, but the dephasing and rephasing components are repeated multiple times, causing multiple echo signals.

2.4 Quantitative Susceptibility Mapping

QSM is an imaging reconstruction method that provides voxel contrast on the basis of the magnetic susceptibility, χ, making it possible to measure quantitative values of the susceptibility, an improvement from the traditional Susceptibility Weighted Imaging (23)(24). The linear relationship is useful in tissue identification and quantification of specific bio-molecules based on their content of iron, gadolinium and super paramagnetic iron oxide. To map the values of susceptibility, QSM extracts the phase and magnitude images from a MGRE sequence, removes the background field, solves the magnetic field to susceptibility inversion problem and the output is a 3D map of susceptibility values. This section will describe the theoretical background behind all the steps necessary to reconstruct complex data obtained from a MGRE sequence to QSM images. A schematic overview of the end-to-end pipelines are shown in Figure 2.4.1. Three reconstruction pipelines have been used in this project, and the difference between these approaches are described more in detail in Section 3.2 and 3.3. Additionally, a visual illustration of the reconstruction part of the QSM method can be seen in Figure 2.4.2.

2.4.1 Types of Magnetic Materials in the Brain

In magnetic theory, one mainly discuss three types of magnetic materials, which reacts differently to being immersed in a static magnetic field. A diamagnetic material is repelled by the magnetic field, and induces its own magnetic field in the direction opposite to the magnetic field (25). In general this effect happens in all materials. However, when this is the only effect a magnetic field has on a material, it is labelled a diamagnetic material. Paramagnetic materials have a weak attraction to the magnetic field. It induces a magnetic field in the direction of the magnetic field. This effect is mainly caused by the interaction between the magnetic field and an unpaired electron. The last one is ferromagnetic materials, which has a strong interaction between the magnetic field and the material itself, and is long lasting, i.e. permanent magnet (26). There exists only small levels of ferromagnetic structures in the body, such as iron, which is ferromagnetic, but is stored in the paramagnetic Ferritin (27).

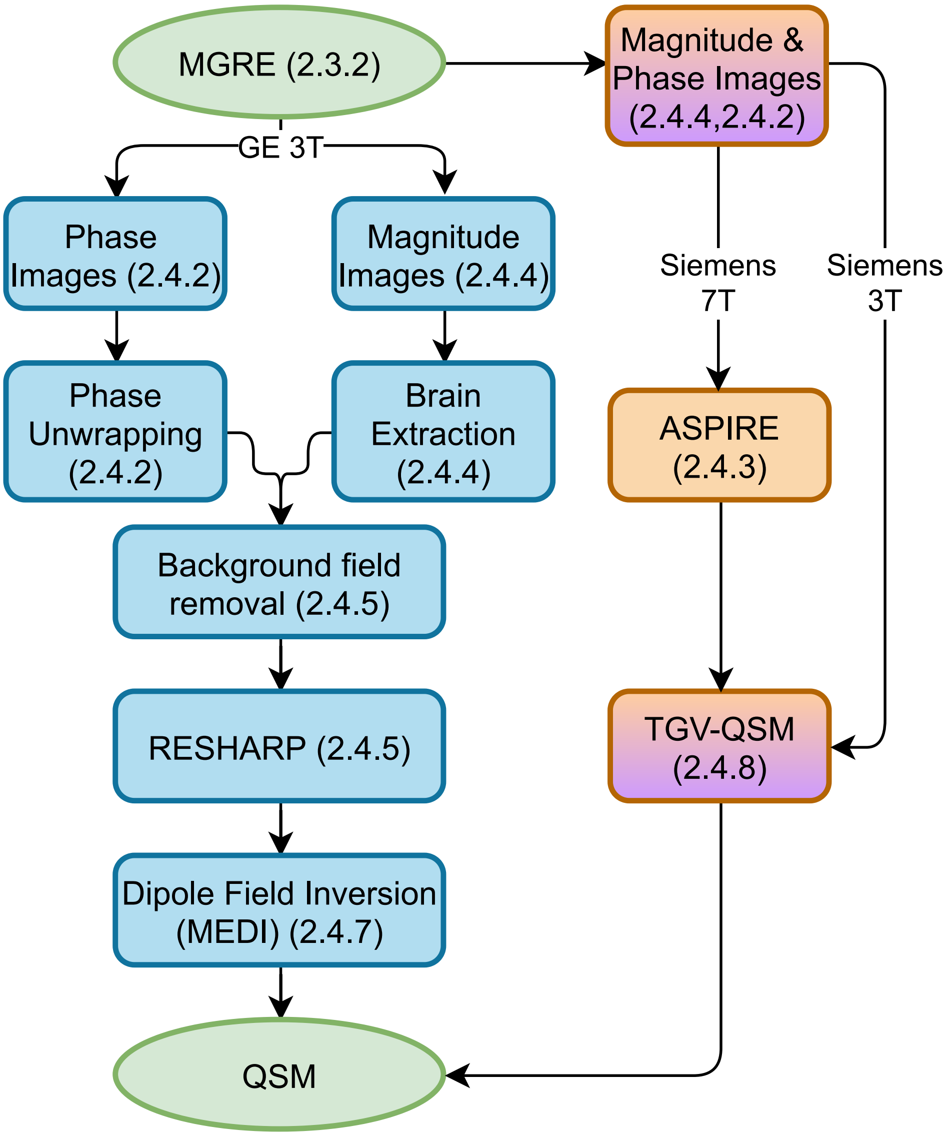


Figure 2.4.1: A flowchart showing the entire pipeline from image acquisition from a MGRE to the final reconstructed QSM image with its associated section. The green ellipses indicate start and stop and the different colours indicate different reconstruction techniques. The gradient boxes indicate that two different reconstruction techniques have an operation in common.

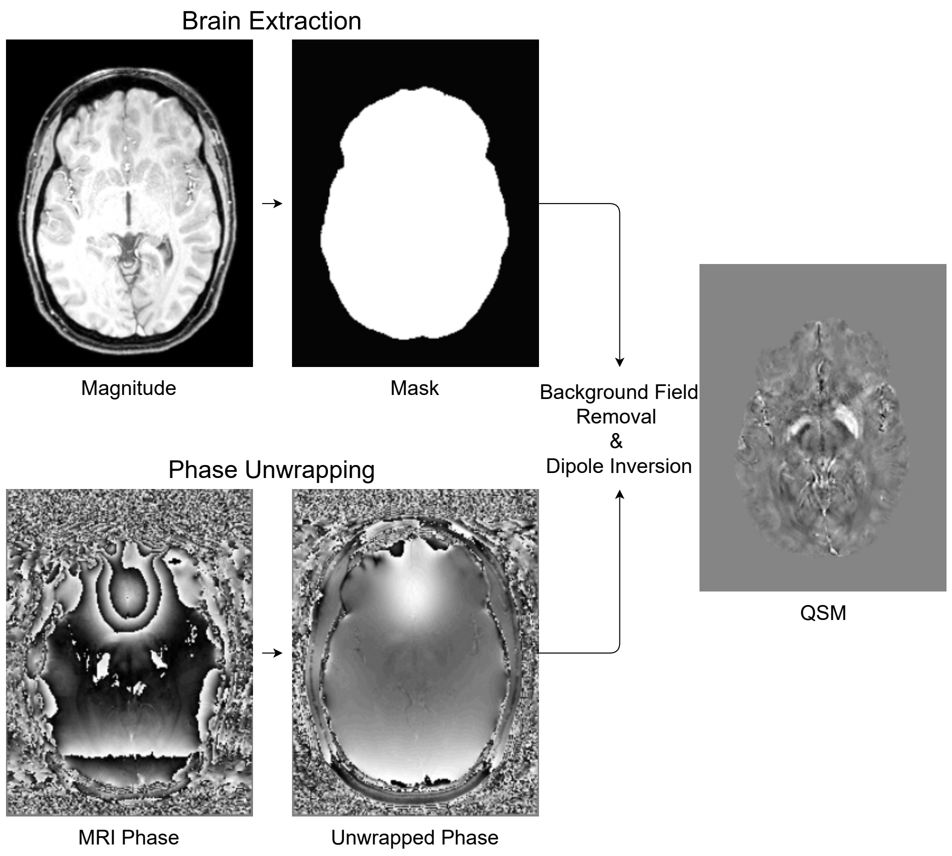


Figure 2.4.2: An overview of the QSM process. Gradient echo of MRI data creates T_2^*-maps (magnitude) and a brain mask is extracted. The phase is extracted from the MGRE and normalized between -π and π, and ΔB_0 is estimated. The contributions from the background field is then removed, yielding a map of susceptibility sources inside the ROI. Dipole field inversion is then applied to transform the ΔB_0-map to a susceptibility map. The images were obtained on a Siemens 3T scanner.

2.4.2 Phase to Field

The first step in the QSM reconstruction process is estimating magnetic field inhomogenites, ΔB, from the phase maps of the gradient echo. As the phase of the MR signal increases linearly with time, the phase of the signal is given as:

$$\phi(t) = \phi_0 - 2\pi\gamma\Delta B t \tag{2.4.1}$$

$\phi(t)$ is the accumulated phase, t is the echo time and ϕ_0 is the initial phase shift at $t = 0$. By plotting the phase as a function of the echo time, the slope will be proportional to ΔB, and can be estimated for each voxel. For a MGRE acquisition, the fitting of ΔB can be performed by using a least squares algorithm, which weights the phase shift data from the earlier echos heavier than the later ones to increase the Signal-to-Noise Ratio (28). As the phase angle can have values outside the range of [-π,π), due to large differences in susceptibility values, one can correct this by introducing an integer, n. This accounts for all the phase shifts over 2π:

$$\phi(t)_k = \phi(t) + n \cdot 2\pi \tag{2.4.2}$$

This will remove aliasing that may arise due to phase loss from Equation 2.4.1. Phase warps are removed from the phase map by adding or subtracting an additional 2π. A scaled version of ΔB is then implemented as:

$$\Delta\phi = 2\pi\gamma\Delta T E \Delta B \tag{2.4.3}$$

Where TE is the echo time. This operation is called phase unwrapping and an example of this operation is shown in Figure 2.4.3.

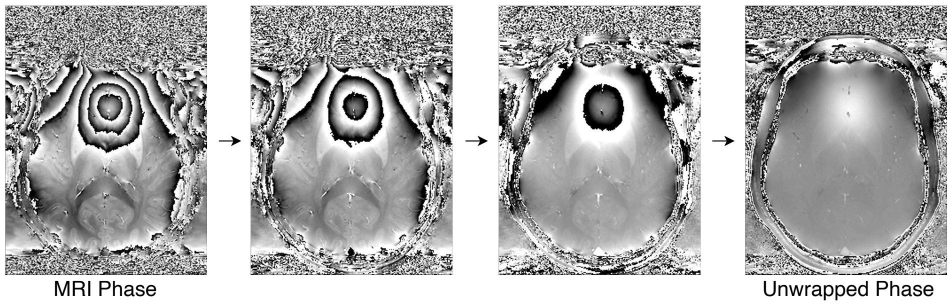


Figure 2.4.3: Phase unwrapping from a MGRE acquisition on one of the volunteers. The discontinuities are a result of phase wrapping. Images obtained using a 7T scanner.

2.4.3 A Simple Phase Image Reconstruction for multi-Echodata

A challenge with modern ultra-high field MRI systems, such as the Siemens Magnetom Terra 7T used in this project, is that it currently lacks the software to properly combine phase signal received from each individual coil. The problem is twofold; each coil has an individual phase offset, ϕ_0^c, the same offset, ϕ_0, as given in Equation 2.4.1, but now with respect to (w.r.t.) each individual coil. The offset is independent of the time echo. Additionally the warping of the phase further complicates this problem. As the measurement of ΔB_0, and in turn the reconstructed QSM image is dependent on the phase offset, we seek to remove the phase offset completely. This can be performed by applying A Simple Phase Image Reconstruction for multi-Echodata (ASPIRE) (29). An offset in the phase can be calculated by comparing the true phase at different time echos, TE. By measuring two instances of Equation 2.4.1 at different echos:

$$\phi_0^c = \frac{TE_k \cdot \phi_j^c - TE_j \cdot \phi_k^c}{TE_k - TE_j} \tag{2.4.4}$$

where ϕ^c is the phase offset, TE_k, TE_j are different time echos and the superscript c denotes the coil number receiving the signal. The phase offset is dependent on which coil receives the signal. Including the coil number and echo number, Equation 2.4.2 can be rewritten as:

$$\phi_j^c = \theta_j^c + 2\pi n_j^c \tag{2.4.5}$$

where n_j^c is the same integer used for unwrapping the phase in Equation 2.4.2, but w.r.t. echo number and coil and θ_j^c is the measured phase. Equation 2.4.4 can be rewritten:

$$\phi_0^c = \phi_j^c - \frac{TE_j}{TE_k - TE_j} \cdot (\phi_k^c - \phi_j^c) \tag{2.4.6}$$

the difference in offset term, $(\phi_k^c - \phi_j^c)$ can be substituted with $\Delta\phi_{k,j}$, a variable accounting for the coil-independent phase difference, yielding:

$$\phi_0^c = \phi_j^c - \frac{TE_j}{TE_k - TE_j} \cdot \Delta\phi_{k,j} \tag{2.4.7}$$

Inserting the true phase from Equation 2.4.5 and replacing it with the measured phase yields

$$\phi_0^c = \theta_j^c + 2\pi n_j^c - \frac{TE_j}{TE_k - TE_j} \cdot (\Delta\theta_{k,j} + 2\pi n_\Delta) \tag{2.4.8}$$

We know from Equation 2.4.5 that the phase is just an integer multiple of 2π, thus one can calculate the wrapped phase offsets the following way:

$$\theta_0^c = \theta_j^c - m \cdot \Delta\theta_{k,j} \quad mod 2\pi \tag{2.4.9}$$

Where $mod 2\pi$ represents the modulo operator of 2π. The result of this equation indicates that phase unwrapping is not necessary.

2.4.4 T_2^*-Weighted Magnitude Images

In magnitude images, the constant T_2^* is introduced to account for magnetic field inhomogenities, mostly reflecting imperfections of the external magnetic field. Their reciprocal values, i.e. relaxivities add together:

$$\frac{1}{T_2^*} = \frac{1}{T_2} + \frac{1}{T_{2,inhom}} \tag{2.4.10}$$

where $T_{2,inhom}$ accounts for the inhomogenities. Employing a MGRE, the T_2^* can be sampled at different echo times. This can then be used to calculate the relaxation time by fitting an exponential curve to the measured signals.

$$S(t) = S_0 e^{-\frac{t}{T_2^*}} \tag{2.4.11}$$

The T_2^* measurement yields a map representing the inhomogenites which causes accelerated dephasing. T_2^*-maps at different echo times are shown in Figure 2.4.4.

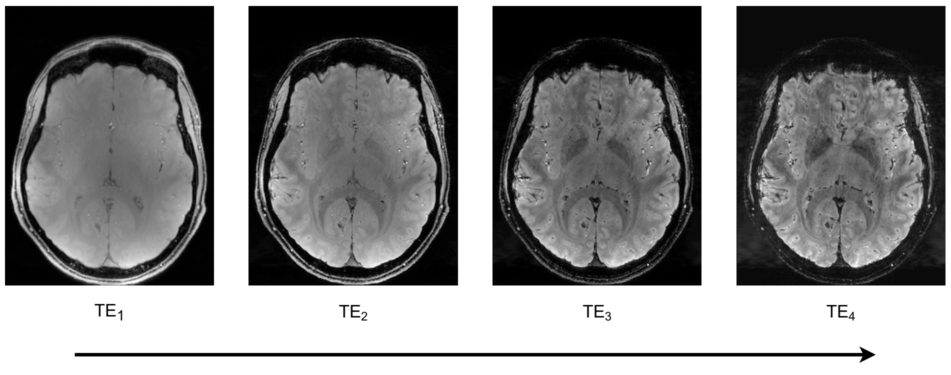


Figure 2.4.4: Different T_2^*-maps obtained at different echo times for one of the volunteers at the Siemens 7T scanner.

Brain Extraction

As seen in Figure 2.4.2, from the magnitude images, one can obtain a mask for the brain using a Brain Extraction Tool (BET) (30), which function is shown in Figure 2.4.5. The method can be divided into 7 steps. First the intensity histogram is extracted from the T_2^*-weighted image. Second, from the histogram, both the size and the centre of mass can be obtained. Third, a sphere's surface is approximated by triangular tessellations. Subsequently, the sphere is allowed to be deformed by the inflicting forces acting upon the vertexes. This process is rerun with higher order smoothness constraints if the previous brain mask is not sufficiently clean. When the image intensity histogram is used, outliers are removed as they are not representative of the intensity of the ROI in the T_2^* map. A threshold based estimation is then used to distinguish between brain matter and bone (background). The threshold between the brain and the background is then used to approximate the centre of gravity in the brain. The next step is to roughly calculate the distance between the centre of mass in the brain to the edge of the brain, a form of radius. The radius is found by counting all the voxels inside the skull, i.e. within the threshold set by the intensity of the skull. The brain is estimated as a sphere, and the sphere is centred around the centre of gravity.

The tessellated sphere is initialised with a radius half of the radius of the brain. In the main training loop, the each vertex is updated to improve surface approximation. It is performed approximately 1000 iterations per step to ensure accurate updates. The entire model is rerun if the brain mask is not clean enough, i.e. if it is self-intersecting. Then the algorithm is rerun with a higher smoothness constraint, for the initial 75% of the iterations, but the constraint has a linear drop towards the original constraint for the remaining fraction. The last term is added to prevent the surface from self-intersecting.

2.4.5 Background Field Removal

Background field removal is a vital component of the QSM reconstruction and includes the removal of all field contributions not originating from the Volume of Interest (VOI), i.e. brain tissue. The background field components are sources of susceptibility that do not arise from the local susceptibility distribution inside the VOI, such as chemical shifts, offsets in the receiver coil, eddy currents and especially air-tissue interface at the skull, paranasal sinus and the human torso (31). The last component is by far the strongest contributor as the susceptibility difference between air and brain tissues is of the order one magnitude (24). This quickly varying susceptibility distribution at the

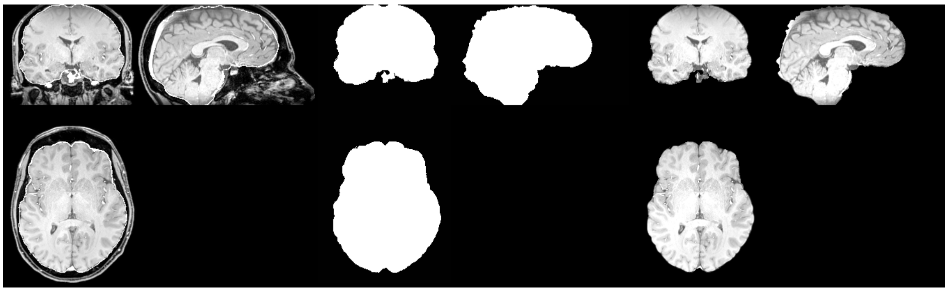


Figure 2.4.5: An example of how the BET works in practice. On the left hand side we can see the full magnitude image before skull stripping, with the brain outlined. In the middle only the obtained mask is presented. One the right hand side we can see the skull stripped magnitude image. Example shown on a volunteers using a 3T scanner.

edge of the VOI can create artifacts which may obscure the magnetic field inside the VOI, distorting the actual susceptibility values in the brain. Therefore, we only want to consider the internal magnetic field.

A number of methods have been proposed to tackle this problem, with varying luck, such as high pass filtering (32), polynomial filtering (33) (34) and field forward estimation (35). Drawbacks of these methods are that they tend to attenuate the local field or leave residual background fields, both degrading the quality of the reconstruction. The Sophisticated Harmonic Artifact Reduction for Phase (SHARP) algorithm (36), which employs the mean value theorem to separate the non harmonic internal magnetic field and the harmonic background field has shown good results. Additionally, the Regularised Enabled Sophisticated Harmonic Artifact Reduction for Phase (RESHARP) (37) algorithm introduces a Tikhonov regularisation to improve the background filtering of SHARP. The principles behind SHARP and RESHARP will be explained in the following sections.

SHARP

The SHARP algorithms exploits the fact that the background field, i.e. the air surrounding the skull, is homogeneous. Thus, all the components of the magnetic field satisfies the Laplace equation (38).

$$\frac{\partial^2 B_{back}}{\partial x^2} + \frac{\partial^2 B_{back}}{\partial y^2} + \frac{\partial^2 B_{back}}{\partial z^2} = 0 \tag{2.4.12}$$

This can be seen by setting the magnetic field derivatives in the electromagnetic wave equation to zero (39). However, the field inside the ROI is not harmonic and does not satisfy this equation, as the brain has a highly inhomogeneous susceptibility distribution. The field map containing both harmonic and non-harmonic fields is then projected onto the space constructed by only the non-harmonic functions. This will directly present the internal field we are looking for. This step is done by solving the Poisson's equation of the internal field:

$$\nabla^2 B_{internal} = \nabla^2 B_0 \tag{2.4.13}$$

By exploiting the spherical-mean-value theorem of harmonic functions, this can be solved. This theorem states that the mean value of a harmonic function, $f(\overrightarrow{r})$ calculated over a sphere, S centred at $\overrightarrow{r_0}$ is equal to the harmonic function itself at $\overrightarrow{r_0}$:

$$\langle f(\overrightarrow{r})\rangle_{S(\overrightarrow{r_0})} = f(\overrightarrow{r_0}) \tag{2.4.14}$$

By using the convolution operator, the relation above can be expressed in three dimensions. Using the Fourier convolution theorem:

$$f - S * f = \bar{S} * f \quad and \quad \bar{S} = \hat{\delta} - \bar{S} \tag{2.4.15}$$

where $\hat{\delta}$ is the unit pulse, in the centre of the sphere. The harmonic background field can now be removed:

$$\bar{S} *^{-1} [\lambda_{mask}(\bar{S}B)] \approx B_{internal} \tag{2.4.16}$$

where λ_{mask} denotes an arbitrary mask that has been applied to remove artifacts created by the deconvolution.

SHARP suffer from some limitations. A drawback is that the ROI must be explicitly defined for the algorithm to work, and errors occurring due to poorly contoured ROIs will cause errors to propagate and influence the final reconstructed QSM map. This is often solved using BET. As one can see in Figure 2.4.5, the brain extraction has in this case segmented a too large volume. Furthermore, susceptibility values close to the edge of the VOI are often unreliable. The original SHARP algorithm solved this by not providing any values close the to the edge. Improvements have been proposed, such as introducing Tikhonov regularisation to remove the distortions occurring at the edge of the VOI. This regularisation algorithm will be discussed in the following section.

RESHARP

Tikhonov regularisation has formerly been applied in MR, and is widespread within the world of tomography (40)(41). The RESHARP algorithm is based on the SHARP algorithm, but adds a regularisation term. Briefly, the matrix form of the RESHARP algorithm can be derived in the following way, by exploiting the harmonic relation of the background field first shown in Equation 2.4.12:

$$M((\delta-\rho)*B_{back})=0 \tag{2.4.17}$$

where M is the binary mask, yielding a value of 1 inside the brain region and 0 outside, δ denotes the Dirac delta function, ρ is the radially symmetric, non-negative normalized convolution kernel. The brain mask is compromised by the radius of ρ as the mean value property is violated at points where ρ overlaps between 2 different regions in the brain mask. Again, transforming the problem into Fourier space to make for easy calculation of the convolution operators yields:

$$M\mathscr{F}^{-1}\{C\mathscr{F}\{B_{back}\}\}=0 \tag{2.4.18}$$

where C denotes $\mathscr{F}(\delta-\rho)$. One can then manipulate the equation by multiplying $M\mathscr{F}^{-1}\{C\mathscr{F}\}$ by the total field, B_{tot}. As the background component is 0, the only field component left is the local field, which has to be solved:

$$M\mathscr{F}^{-1}\{C\mathscr{F}\{B_{loc}\}\}=M\mathscr{F}^{-1}\{C\mathscr{F}\{B_{tot}\}\} \tag{2.4.19}$$

The system of equations given above is not determined, and we seek additional information such that this can be solved, obtaining only one unique solution. As the background field is the dominating component of the total field, the residual local field component with the best fit is then chosen as the solution. Hence, finding the least norm of Equation 2.4.19 represents a minimisation problem with constrictions. A common method for solving these kinds of challenges is to introduce Lagrange Multipliers. In the RESHARP algorithm, this is implemented by introducing Tikhonov regularisation to the B_{loc} term, in addition to a balancing operation performed by the Lagrange multiplier:

$$argmin_{B_{loc}}||M\mathscr{F}^{-1}\{C\mathscr{F}(B_{loc}-B_{tot})\}||_2^2+\lambda||B_{loc}||_2^2 \tag{2.4.20}$$

argmin$_{B_{loc}}$ expresses the different values the local magnetic field can have to minimise the function, $||...||_2^2$ represents the sum of squares. The first norm term is the data fidelity term and guarantees the harmonic behaviour of the background field. The second term is the Tikhonov regularisation term to keep the characteristics of the local field, λ is a Lagrange multiplier, for which the equation is minimised for different values of λ, where the best solution has the smallest error. In practice, this is performed by minimising Equation 2.4.20 for a number of different values of λ. The solution norm ($||B_{loc}||_2^2$) is plotted against norm of the data fidelity term. The optimal choice of λ is the value that corresponds to the point of maximal curvature on the so-called L-curve (42). An L-curve is the norm of the regularised solution vs the norm of its corresponding residual norm plotted in a log-log plot. In the log-log plot the user obtains a graphical tool for studying the fit of the data vs the regularised solution as λ varies, making it more convenient to choose between the trade offs (42).

2.4.6 Dipole Field Inversion

When an atom is immersed in a static magnetic field, the electrons belonging to the nucleus and the magnetic field will interact, causing a magnetisation of the matter. In magnetic resonance imaging, the susceptibility of a voxel is given as:

$$\chi = \frac{\overrightarrow{M}}{\overrightarrow{H}} \tag{2.4.21}$$

Where χ is the magnetic susceptibility, $\overrightarrow{M}$ is the magnetisation vector, and $\overrightarrow{H}$ is the magnetic field strength. It is related to the magnetic flux density, by the relation $\overrightarrow{B} = \mu_0\mu_r\overrightarrow{H}$. Where μ_0 and μ_r is the permeability in vacuum and the relative permeability of a material, respectively

As the body is an ensemble of different magnetic materials, where each material either opposes or aligns itself with the magnetic field; an application of a magnetic field, B_0, will create magnetic inhomogenites in tissues, $\Delta B(r)$. One can utilise the relationship between the magnetic susceptibility and the magnetic field inhomogenity, $\chi(r)-\Delta B(r)$ to obtain more information about the voxel. Each particle can be modelled as a dipole, and a voxel is considered as an ensamble of each particle in said voxel. The z-component of the local magnetic field produced by a dipole, μ_t, can be described as:

$$B_{\mu z} \propto \frac{\mu_t}{r^3}(3cos^2\theta - 1) \tag{2.4.22}$$

Where μ_t is the dipole moment of a material in tissue, r and θ denotes the position w.r.t. the distance from the nucleus and the angle relative to the direction of the magnetic field, respectively. An illustration is shown in Figure 2.4.6.

At an angle of $\theta \approx 54.7^\circ$ the field will be equal to zero. This is also known as the

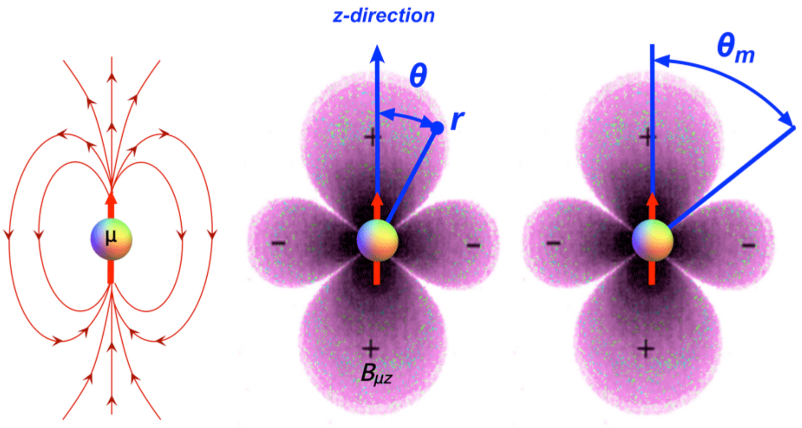


Figure 2.4.6: The leftmost figure shows the magnetic field lines originating from the magnetic dipole. The figure in the middle shows the z-component of the local field produced by the dipole μ. The figure to the right shows $B_{\mu z}$= 0 at $\theta \approx 54.7°$. Courtesy of Allen D. Elster, MRIquestions.com.

"magic angle". We will see later that this is the origin of the inversion problem. As we are only looking at the field in the z-direction, Equation 2.4.21 can be rewritten as:

$$M_z(r) = \chi(r)H_0 = \chi(r)\frac{B_0}{\mu_0\mu_r} = \chi(r)\frac{B_0}{\mu_0(1+\chi(r))} \tag{2.4.23}$$

$$M_z \underset{\chi<<1}{\approx} \chi(r)\frac{B_0}{\mu_0} \tag{2.4.24}$$

This approximation is valid as susceptibility values in human body are much smaller than 1 (43). By exploiting the expression from Equation 2.4.24, this can be more compactly written as:

$$\Delta B_z(r) = \mu_0 \cdot M_z(r) * D(r) \implies \frac{\Delta B_z(r)}{B_0} = \chi(r) * D(r) \tag{2.4.25}$$

Where $\chi(r)$ is the susceptibility and $D(r)$ is the dipole field. Trying to solve this in the spatial domain is an ill-posed problem. However, by introducing the Fourier Transform and transforming the problem from real space to k-space:

$$\frac{\mathscr{F}\{\Delta B_z(r)\}}{B_0} = \frac{\Delta B_z(k)}{B_0} = \mathscr{F}\{\chi(r)\} \cdot \mathscr{F}\{D(r)\} = \chi(k) \cdot D(k) \tag{2.4.26}$$

The expression is now only a point-wise multiplication between the expression for the dipole field, $D(k)$, and the susceptibility distribution ,$\chi(k)$, in k-space. The unit dipole in the z-direction can then be expressed as the following:

$$D(k) = -\frac{1}{3}\left[\frac{3k_z^2}{k_x^2+k_y^2+k_z^2} - 1\right] = \left[\frac{1}{3} - \frac{k_z^2}{k^2}\right] \tag{2.4.27}$$

where k_x, k_y and k_z denotes the wave number in x,y and z direction, respectively. For a small rotation of θ, w.r.t. the x-axis, an expression for the dipole field can also be obtained:

$$D(k) = -\frac{1}{3}\left[3\frac{(k_z \cdot cos\theta - k_y \cdot sin\theta)^2}{k_x^2+k_y^2+k_z^2} - 1\right] \tag{2.4.28}$$

where θ is the angle between the z-axis and the angle at which the nucleus is precessing, as shown in the two illustrations to the right in Figure 2.4.6. This expression in the Fourier domain can effectively predict a perturbation in the field, if the susceptibility distribution is known. However, at the magic angle, where the local magnetic field is 0, the field to source inverse problem has a division by 0. This divergence is the source of striking image artifacts and large distortions in the reconstructed image.

2.4.7 Morphology Enabled Dipole Inversion

The problem that arises as a result from Equation 2.4.28 is a well studied problem. These methods typically try to estimate the susceptibility map by finding the variable that minimises the distance between the estimated $B_0(\chi$(k)$*$D(k)) and the measured ΔB(k). Morphology Enabled Dipole Inversion (MEDI) aims to solve the divergence problem in image space (44).

As shown in equation 2.4.25, the local magnetic field of tissue can be expressed as the convolution between the susceptibility distribution and the dipole kernel giving the following expression (45):

$$\Delta B(r) = \frac{3cos^2(\theta) - 1}{4\pi r^3} * \chi(r) \tag{2.4.29}$$

which can be expressed on matrix form:

$$b = D\chi \tag{2.4.30}$$

where b and χ are the vector forms of the measured local field and susceptibility distribution, respectively, and D is a matrix representing the convolution of the dipole kernel, i.e. the magnetic field.

The MEDI method uses a physical prior which minimises the number of voxels that belong to the edges of the susceptibility map and not edges in the T_2^* map. The reconstruction w.r.t. the susceptibility is then a L_1-norm minimisation problem with constraints:

$$min_\chi ||MG\chi||_1 \quad \text{subject to} \quad ||W(D\chi - b)||_2^2 \leq \varepsilon \tag{2.4.31}$$

where M is the structural weighting matrix, which is extracted from the gradient of the T_2^*-weighted magnitude image, G is the gradient operator, W compensates for the noise variation inn the field measurement proportional to the image magnitude and ε is the noise and controls the faithfullness of the reconstruction. M is a binary mask that assigns 1 to voxels with small gradients and 0 to voxels with large gradients given a threshold in the magnitude image:

$$M = \begin{cases} 1 & |Gm| < \mu \\ 0 & |Gm| \geq \mu \end{cases} \tag{2.4.32}$$

where μ is a threshold associated with the noise level in the image and m is the vector form of the magnitude image.

Algorithm Implementation

Equation 2.4.31 can be solved by rewriting it using the Lagrange multiplier method into the following minimisation problem:

$$min_{\chi,\lambda} E(\chi,\lambda), \quad \Big| \quad E(\chi,\lambda) \equiv ||MG\chi||_1 + \lambda(||W(D\chi - b)||_2^2 - \varepsilon) \tag{2.4.33}$$

This equation can be numerically solved by finding the global minimas of the following functions:

$$\begin{cases} \nabla_\chi E(\chi,\lambda) = 0 \\ \nabla_\lambda E(\chi,\lambda) = 0 \end{cases} \tag{2.4.34}$$

The first step to minimising the cost function, E, is setting the gradient w.r.t. χ to 0 for a chosen value of λ:

$$0 = \nabla_\chi E(\chi,\lambda) = (MG)^H sign(MG\chi) + 2\lambda(WD)^H(WD\chi - Wb) \equiv L(\chi)\chi - \widetilde{b} \tag{2.4.35}$$

where $(...)^H$ is the conjugate transpose $L(\chi)$ is given as:

$$L(\chi) = (MG)^H \frac{1}{|MG\chi|} MG + 2\lambda(WD)^H WD$$

furthermore, $\tilde{b}$ is a shortening for:

$$\tilde{b} = 2\lambda(WD)^H Wb$$

A weak derivative function is used to find the derivative of the L$_1$ norm. A weak derivative is used in cases where the function is integrable, but not differentiable for all values:

$$(||\chi||_1)' = sign(x) = \begin{cases} x/|x| & \text{if} \quad x \neq 0 \\ 0 \quad \text{if} & x = 0 \end{cases} \tag{2.4.36}$$

where $sign(x)$ denotes the sign function, which is integrable, but not differentiable for all values of x. A method for solving Equation 2.4.35 is to rewrite it as a fixed point equation, and then solve it iteratively (46):

$$\chi_{n+1} = L^{-1}(\chi_n)\tilde{b} \tag{2.4.37}$$

by a reordering of Equation 2.4.35, $\tilde{b}$ can be expressed as:

$$\tilde{b} = L(\chi_n)\chi_n - \nabla_\chi E(\chi_n, \lambda)$$

the fixed point iteration from Equation 2.4.37 has now become a quasi-Newton problem (47). An advantage of this is that it is more robust against round-off errors. The equation above can be rewritten as:

$$\chi_{n+1} = L^{-1}(\chi_n)(L(\chi_n)\chi_n - \nabla_\chi E(\chi_n, \lambda)) = \chi_n - L^{-1}(\chi_n)\nabla_\chi E(\chi_n, \lambda) \tag{2.4.38}$$

The χ_{n+1} term can be computed by employing a conjugate gradient descent method (48).

When the derivative of the cost function, E, w.r.t. χ and λ is equal to zero it, the constraints term are equal to the expected noise power iteself:

$$0 = \nabla_\lambda E(\chi_n, \lambda) = ||||W(D\chi - b)||||_2^2 - \varepsilon||||(D\chi^* - b)||||_2^2 \approx \varepsilon \tag{2.4.39}$$

χ^* is the susceptibility solution from Equation 2.4.35 given a value of λ. Equation 2.4.39 is solved numerically for different values of λ, and a λ is chosen whose corresponding χ satisfies this equation.

2.4.8 Total Generalised Variation

The implementation of dipole inversion methods called Total Variation (TV) penalties have been used in the reconstruction of e.g. QSM-weighted images obtained from gradient echo phase data (49)(35). A drawback of this method is that TV only takes into account the first order derivative of the susceptibility distribution. This means that TV does not enforce higher order smoothness, causing distortions, known as staircase artifacts in the case of images which are not piecewise constant. Total Generalised Variation (TGV) generalises the TV method by including higher order smoothness, in turn leading to more accurate reconstruction (50).

The TV semi-norm of an image is well known , and in the case of our problem, the susceptibility distribution can be described by:

$$TV(\chi) = ||\nabla\chi||_M \tag{2.4.40}$$

here ∇ is the gradient and $||..||_M$ is the Radon norm, which is a generalisation of the L_1-norm. The TGV, which in this case is defined as a second order TGV^2 is a minimisation problem :

$$TGV^2_{\alpha_1,\alpha_0}(\chi) = \min_w \alpha_1 ||\nabla\chi - w||_M + \alpha_0 ||\varepsilon w||_M \tag{2.4.41}$$

We minimise over w, i.e. over all vector fields, and ε denotes the symmetrical derivative of w. The result of this operation creates a symmetric tensor field of second order:

$$\varepsilon w = \frac{1}{2}(\nabla w + \nabla w^T)$$

where w^T denotes the transpose of the vector fields. As an example, ε for the 2D case is defined as follows:

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \varepsilon w = \begin{bmatrix} \frac{\partial w_1}{\partial x} & \frac{1}{2}(\frac{\partial w_1}{\partial y} + \frac{\partial w_2}{\partial x}) \\ \frac{1}{2}(\frac{\partial w_1}{\partial y} + \frac{\partial w_2}{\partial x}) & \frac{\partial w_2}{\partial y} \end{bmatrix} \tag{2.4.42}$$

The ratio of the positive weights, α_0 and α_1, introduced in Equation 2.4.41, is used balance the first and second derivative of a function in TGV^2. It has been showed that the ideal ration lies between 2:1 and 3:1 (51).

Integrative QSM Reconstruction using TGV

Traditionally, QSM reconstruction has involved multiple steps, such as phase unwrapping, background field removal and dipole field inversion, meaning that errors can propagate and grow in each step, possibly creating distortions and inaccuracies in the final reconstructed image. To avoid this problem, the wrapped phase data is directly subjected to the TGV, an auxiliary variable is then created to account for the background field removal in the regularisation process. Another advantage of this method is that there is no need to introduce a threshold parameter, which is used in other background field removals, such as SHARP. QSM maps are obtained from the wrapped phase, $\phi_{wrapped}$ of the gradient echo. The Laplacian of the wrapped phase is given as the following:

$$\Delta\phi = Im(\Delta e^{i\phi_{wrapped}} \cdot e^{-i\phi_{wrapped}}) \tag{2.4.43}$$

The reconstruction method then bases itself on an optimisation procedure applied on the unwrapped version of Equation 2.4.43(52). By exploiting the Laplacian of the phase, the dipole inversion is directly implemented as:

$$\frac{1}{3}\left(\frac{\partial^2\chi}{\partial x^2} + \frac{\partial^2\chi}{\partial y^2} + \frac{\partial^2\chi}{\partial z^2}\right) = \frac{1}{2\pi T_E \gamma B_0}\Delta\phi \tag{2.4.44}$$

As stated above, the background field was implicitly incorporated by the auxiliary variable, ψ, which has a Laplacian equal to the discrepancy of the equation on the brain mask M, which can be extracted using the BET described in Section 2.4.4. A squared L^2-norm was used to penalise ψ, in other words by integrating its absolute value squared over M. The regularisation parameters $\alpha = (\alpha_0, \alpha_1)$ and the TGV of second order was used for inversion, resulting in a variational problem:

$$\begin{gathered} \min_{\chi,\psi} \int |\psi|^2 dx + TGV_\alpha^2(\chi) \quad \text{subject to} \\ \Delta\phi = \frac{1}{3}\frac{\partial^2\chi}{\partial x^2} + \frac{1}{3}\frac{\partial^2\chi}{\partial y^2} - \frac{2}{3}\frac{\partial^2\chi}{\partial z^2} = \frac{1}{2\pi T_E \gamma B_0}\Delta\phi \text{in} M \end{gathered} \tag{2.4.45}$$

However, another auxiliary variable is introduced by the TGV function itself:

$$TGV_\alpha^2(\chi) = min_w \alpha_1 ||\nabla\chi - w||_M + \alpha_0 ||\varepsilon w||_M$$

Similar to other QSM reconstruction, solving the optimisation problem yields susceptibility maps. However, the phase unwrapping and background filtering is done in a single step, by only applying the Laplacian of the unwrapped phase, ϕ in the inversion problem. Therefore, the unwrapped phase is not employed directly (53). In terms of the Laplacian operator, the discrepancy can be interpreted as the variation of the phase ϕ and χ convoluted with the dipole kernel. Minimising w.r.t. ψ, is equivalent to removing the harmonic background field. Furthermore, minimising w.r.t. χ yields the TGV-regularised dipole inversion.

2.4.9 Choosing a Reference Susceptibility Value

As the QSM images are not defined at the origin of k-space, the QSM pipeline yields only relative susceptibility values. This causes mainly two constraints; form a clinical viewpoint it is very useful to know the absolute susceptibility value of some tissues, e.g. for RN and SN in possible diagnosis of PD. Additionally, with only relative susceptibility values, the images may not be comparable to each other, making them unfit for use in automatic segmentation algorithms, as the intensity from image to image may vary from scanner to scanner. This problem is normally solved by taking the average susceptibility in a reference region and subtracting it from the susceptibility in the other voxels (54)(49). As the age of the subjects varies, one therefore seeks a reference tissue that degrades as little as possible with increasing age. Regions in the Corticospinal Tract and the CSF have been shown to be one of the brain tissues that is most resilient to degeneration due to ageing (55)(56)

2.5 Linear and non-linear image registration

The use of linear image registration is a commonly used tool within the field of MR brain image analysis. Both linear and non-linear image registration will be described in this section.

2.5.1 Linear Image Registration

Linear image registration is a tool usually applied for co-registering images of different modalities, within the same subject. However, co-registering can be performed within the same modality, but with different resolution, e.g. transforming a mask between 7T and 3T QSM images. The warps are so similar that the image to image registration can be performed only using a linear transformation with a maximum of 12 degrees of freedom Degrees of Freedom (DoF). A linear transformation is generally described in the following way:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = T \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.5.1}$$

where T is a 4x4 matrix describing a transformation. The challenge is to find the best transformation T which best aligns a reference image, I_{Ref}, with another image, I_N. Finding a suitable transformation is an optimisation problem, described as:

$$argmin\{C(I_{Ref}, I_N \cdot T)\} \tag{2.5.2}$$

where C is the cost function and I_N ·T is the image I_N after it has been transformed by T. To constrain our problem, we must consider which kind of transformations which are allowed. In linear image registration, we mainly distinguish between Rigid Body and Affine transformations, having 6 and 12 DoF, respectively. Rigid Body transformation can include 3 rotations and 3 translations while an Affine transformation includes an additional 3 scalings and 3 skews/shears.

For discrete data, in the case of MR images, the intensity is defined voxel-wise, meaning that we need to interpolate to find the intensity between these points. Classically, continuous tri-lienar, spline or windowed sinc kernels are convoluted with the discrete voxels to create a discrete interpolation (57)(58). The type of interpolator used will also affect whether or not the cost function becomes continuous or not as the parameters in the transformation changes.

To maximise the similarity between two images, one can introduce a cost function. Common cost functions are Mutual Information and Correlation Ratio. The registration problem stated in Equation 2.5.2 can be descried by the transformation, interpolator and the cost function. An optimiser must be found such that the transformation, T, minimises the cost function that is used. Furthermore, we must be sure that this is a global minimum and not a local minimum, i.e. that we have truly achieved the highest similarity between the different images. When the optimiser is performing its task, it would be foolish to try to start with comparing the two images with the highest resolution possible for each degree of freedom. Instead, the images can at first be under-sampled and the optimiser can then improve the transformation and find the global minimum and use this as a reference when the process is repeated with a higher sampling. This process is continued until the optimisation has been performed on the resolution of the original images. In addition to the drastically reduced computational time, lower resolution images are easier to align as the larger features, such as the edge of the brain is more dominant than smaller features, i.e. there usually exist less local minimums (59).

2.5.2 Non-Linear Image Registration

When one is using brain imaging data from a wide variety of subjects, it is often beneficial to register the brains to each other by using a common template. This is profitable as it creates a better foundation for the statistical analysis and makes it easier to compare subjects. Between different subjects, the warps can usually not be described by linear deformations, thus non-linear image registration must be applied.

In general, non-linear transformations can be descried as a linear transformation, but with an added term accounting for the non-linearity:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = T \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} + \begin{bmatrix} \partial_x(x,y,z) \\ \partial_y(x,y,z) \\ \partial_z(x,y,z) \\ 1 \end{bmatrix} \tag{2.5.3}$$

where T is an affine transformation described in the previous section. The matrix containing the partial derivatives are displacement fields, and gives information about the displacement in the position in the respective directions. The displacement fields have the same dimension as the image it tries to approximate. To reduce computational time, the displacement fields can be represented by a linear combination of basis function, in turn reducing the amount of fitted parameters which needs to be calculated.

The optimisation methods are similar to those used in linear image registration, as they start the optimisation process with a sub-sampled image to both reduce computational costs, reduce chance of being stuck in a local minima, but also detect the larger scale objects, before iteratively increasing the resolution. What separates the non-linear optimisation process from the linear one is the regularisation term. The non-linear optimisation process consists of finding a compromise between making the images look similar, i.e. minimising the cost functions, and creating warps which are physically plausible. Statistically, sharper warps are less common than smoother ones. Typical regularisation functions are membrane energy or bending energy. Thus, the goal is to minimise the cost-function:

$$O(q) = \sum_{i=1}^{\infty} (g(x_i'(x_i, q)) - f(x_i))^2 + \lambda \epsilon(q) \tag{2.5.4}$$

where q is a parameter which minimises O, g is the image we want to warp, x' is the transformed image, f is the image used as a reference, ϵ is a regularisation function and λ is a regularisation parameter where the value reflects the focus on similarity or the likeliness of warps.

2.6 Deep Learning

This section concerns the theory of DL and is based on the theory presented in the specialisation project. DL is a branch of machine learning that excels in solving problems involving a large amount of features, such as speech, images and text processing problems (11). The ground pillar in DL is the use of artificial neural networks, inspired by the information processing and communication in biological systems, such as the human brain (60). The name "Deep Learning" originates from the use of deep neural networks, which is composed of long chains of neurons. In addition to the fact that it exploits deep features of data, i.e. features extracted from other features.

2.6.1 Artificial Neural Networks

A simple artificial neural network, as depicted in Figure 2.6.1, is constituted by an input layer, hidden layer and an output layer, each layer being composed by a given number of neurons. Every neuron takes in multiple inputs, and delivers a single output, which in turn can be sent along to all neurons in the next layer, or be expressed as an output.

The Artifical Neuron

An artificial neuron, also known as the perceptron, is a single layer neural network. Figure 2.6.2 shows a perceptron overlaid on a neuron. The perceptron is constituted by one or multiple outputs received from the previous neural layer, which is passed to the subsequent neuron and used as an input. Each input, $x_1, x_2, ..., x_n$ is then multiplied with a weighting factor $w_1, w_2, ..., w_n$, respectively. The weighted inputs are then summed and added a bias, b:

$$\sum_{n=1}^{n} w_n x_n + b \tag{2.6.1}$$

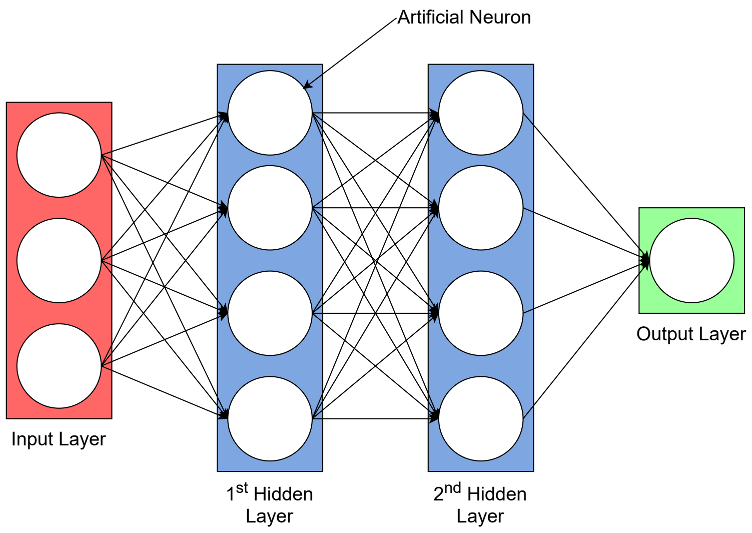


Figure 2.6.1: A model of a fully connected Neural Network, consisting of an input layer, two hidden layers and an output layer. Figure created by Eivind Lysheim in draw.io.

where the summation is over all weights and inputs. However, as very few problems in this world can be solved by simple linear models, an activation function is therefore introduced to create a non-linearity in the system. Each neuron therefore implements a logic regressor:

$$\sigma(wx + b) \tag{2.6.2}$$

Where wx symbols all the weighted inputs from the former neural layer, b is the bias and σ is the activation function. The output from Equation 2.6.2 is then sent to the subsequent layer.

Activation Functions

The activation function of an artificial neuron converts the summed inputs and biases from input to output, and can therefore be described as a mapping from input-space to output-space. There are many reasons for using activation functions, the main reason being complex tasks, as they can not be solved using linear combinations of neurons. Only using linear activation functions would result in a deep neural network that in reality would be equivalent to only having one neuron. Furthermore, as the neuron is just a summation of weighted inputs and a biases, without the use of activation functions, the range of the values of an output would have a much larger range of

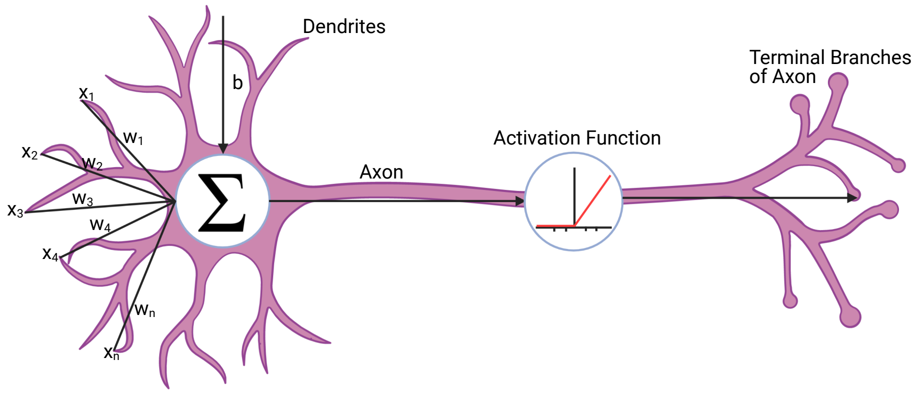


Figure 2.6.2: A neuron overlayed by an artificial neuron. The dendrites are represented by the weighted inputs before it is added together, before it is sent to the axon, represented by the axon. The illustration was created by Eivind Lysheim using BioRender.com

possible values, depending on the weighting and the output-number of neurons in the previous layer. Therefore, activation functions are also important in restricting the output values to a given range, and to ensure that smaller inputs are not neglected. Which activation functions one chooses to use are dependent on the training data and the neural network. It is especially important to choose a suiting activation function w.r.t. back-propagation, which will be explained in the following section. Each activation function has its advantageous and disadvantages.

The Sigmoid is an activation function that is a differentiable real function for real inputs and has a positive derivative everywhere. The Sigmoid is given as:

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

It is restricted to values between 0 and 1 and is therefore handy in binary classification problems where we want the network's output to be a probability floating number between 0 and 1. The Sigmoid function is easy to understand and is therefore used in introductory courses to Deep Learning, but should be avoided in the central parts of a network, mainly due to the vanishing gradient problem, where the gradient tends to zero, not being able to update weights in the earlier layers. Thus making it impossible for the network to improve its performance (61).

The Rectified Linear Unit (ReLU) activation function has proved itself to be a faster activation function than the Sigmoid, in addition to avoiding the vanishing gradient problem (62). The function is given as:

$$g(x) = max(0,x) = \begin{cases} x_i, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

As all inputs less than zero is set to zero, this eliminates the vanishing gradient problem.

Training a Neural Network

In a neural network, weights and biases are periodically updated to improve the output of the network. The goal of the training is to optimise these weights and biases to produce the best output, depending on the objective. During the training of a neural network, one has a loss function, $L = L(x,\theta)$, dependent on the input, x, and the parameters θ. Normally, when we are talking about optimisation of an algorithm, an objective function is used to evaluate the suggested solution. In the context of neural networks we seek to minimise the error between the predicted output and the actual solution, and the objective function is usually referred to a as loss function. There are mainly two losses; the loss for the training data, commonly named "loss" and loss for the validation data, commonly called "validation loss". To optimise the values of the weights and biases, the goal of the training process is to minimise the loss functions. As the input data is fixed; only the parameters θ can be updated during training. Therefore, for training purposes, the loss function is only dependent on the parameters of the model only, reducing the loss function to $\mathrm{L} = \mathrm{L}(\theta)$.

One method for training the network is by randomly changing the values of each weighting and bias, and check if the output is better than the former results, and only save the weights and biases if it is. However, this is an extremely inefficient method, with a high probability of making things worse. Another option is to simply predict the next adjustment that will decrease the loss for each iteration. By introducing a loss function, one can use the derivative of the loss function to calculate the direction that decreases the loss the most. The derivative is given by taking the tangent line of the angle α. If the derivative is positive, i.e. $\alpha < 90°$, the parameters, θ, must be decreased. On the other hand, if it is negative, i.e. $90° < \alpha < 180°$, the parameters must be increased.

The example above only holds for one parameter. However, neural networks tend to have a huge amount of parameters. Therefore, one needs to look at the partial derivative of all parameters, i.e. the gradient:

$$\nabla(L(w)) = (\frac{\partial L(w)}{\partial w_1}, ..., \frac{\partial L(w)}{\partial w_n}) \tag{2.6.3}$$

Where ∇ is the vectorial differential operator, $L(w)$ is the loss function in n-dimensions w.r.t. the weights, w.

This is the foundation of the gradient descent technique, which is a technique composed of 4 steps:

1. Parameters starts out with a configuration. Either random or non-random initialisation.

2. Calculate the partial derivatives of the loss function, L(θ).

3. Move in the opposite direction of the gradient.

4. Repeat step 2-4 until the loss is below a given threshold appointed by the beforehand.

Immediately, this method raises two questions; How long is the movement in the opposite direction and how large should the threshold be?

The first question is decided by the update function, which updates the weights at time (t-1):

$$w^t = w^{t-1} - \mu_l \nabla L(w) \tag{2.6.4}$$

where μ_l is the learning rate, and dictates the weighting of the gradient of the loss function.

For a simple network, such as the perceptron seen Figure 2.6.2, computing the gradient descent technique is easily done w.r.t. a given parameter. However, to improve a neural network, one needs to update the weights w.r.t. to many parameters. This is done by the implementation of the back-propagation algorithm (63). It provides iterative rules used for computing partial derivatives of the loss function w.r.t each parameter, i.e. weights and biases in the network. The basis for this method is the chain rule for derivatives. The derivation of how we obtain an expression for updating the parameters of each neuron is not shown here, but a nice derivation is shown here (64). Most importantly, for each forward pass through a network, a backward pass adjusting the weights and biases is performed. After the entire training data has been passed both forward and backwards through the network, an entire epoch has been completed. As the dataset tends to be quite large and usually too much for the memory to handle all at once, the training data can be divided into batches. The batch size is the fraction of training examples utilised in each iteration.

To perform a back propagation, the network needs to know the difference between the predicted value and the actual value, the loss. A commonly used loss function is the binary cross-entropy. It calculates the cross entropy between true and predicted labels whose output is a probability between 0 and 1 (65). Consider an objective or an underlying probability P that we want the system to mimic, and the system's approximation of this is the distribution Q. The cross entropy of Q from P will then be the number of additional bits that has to be used to represent an event using Q instead of P. When the difference between the predicted and true labels is very small, the entropy converges towards zero. When the difference increases, the entropy diverges.

As stated above, the goal is to minimise the loss function to make the predictions of the network as correct as possible. To do this, the parameters of the model need to be updated. Optimisers are used to solve this problem. They tie together the model parameters and the loss function by updating the model parameters w.r.t. the output from the loss function. In essence, the loss function has the map of the local terrain and gives feedback to the optimiser if its moving in the right direction or not. A useful optimiser is the "RMSprop". Opposed to the gradient descent where all gradients accumulated for the momentum, this optimiser divides the gradient by a running average of its recent magnitude, only using gradients accumulated inside window, creating a more robust output (66).

As one only obtains a relative performance metrics by using only the loss function, i.e. you only know if you are going in the right direction, but you do not know how far away you are from the goal, an absolute performance metrics is useful. In the case of problems like ours, where the objective is to train a network that never mislabels, but also labels all of the regions of interest, the Dice Score (DS) is a relevant and widely used metric. The DS, can be written in the following way:

$$DS = \frac{2 \cdot |A \cap M|}{|M| + |A|} \tag{2.6.5}$$

where A are the pixels that has been labelled by the automatic segmentation and M are the pixels that has been manually segmented. To make a more robust metrics, a smoothing factor, λ, can be introduced. A dice coefficient of 1.0 means perfect overlap and no miss-labelling between predictions and the ground truth, while a coefficient of 0.0 means that there has been no correct predictions. In addition to the DS, there are several other methods for evaluating the performance of a segmentation algorithm.

A well known method is looking at the True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN). As the brain structures a are small compared to the entire volume of the MR image, TN is not of interest to us, as it would always show a number between 0.99 and 1.0. Thus, only the sensitivity and precision were applied to evaluate the segmentations:

$$Sensitivity = \frac{TP}{TP + FN} \quad | \quad Precision = \frac{TP}{TP + FP} \tag{2.6.6}$$

Sensitivity is often called true positive rate or recall and is a measure of the fraction of actual positives which are correctly identified, the precision, often called positive predictive value is the fraction of relevant instances among the retrieved instances.

A method for reducing training time is to introduce the batch normalisation layer (67). A problem with deep neural networks is the continuous changing of inputs, caused by an update in the parameters in the previous layer. This leads to a lower learning rate and a more sensitive initial parameter initialisation, causing an increased training time. Performing a normalisation of each mini-batch, the learning rate can be increased and the initialisation will become less sensitive.

2.6.2 Convolutional Neural Networks

CNNs are networks that particularly excels in handling 2D images (68). One of the main traits of a CNN is its ability to extract deep features from an image by applying convolutional layers.

Imagine one has a neural network similar to the one in Figure 2.6.3. All neurons are fully connected to the next, in a dense layer. If one wants to use this kind of densely connected network to process images, one would need to collapse the dimensions of the image into one dimension, to fit as an input to the neurons. By collapsing all the information in the image into a 1D tensor, this would cause a loss of spatial information that one could have used to extract features from in the image. The spatial structure is indispensable in image processing, and one must therefore look for a solution which can retain the spatial structure of the image.

To overcome the problem described above, one can first start to look at what a digital picture really is, a matrix full of numbers between 0 and 255, either with 1 channel (grayscale) or 3 channels (RGB). Instead of connecting every single matrix component to a single neuron, which also is computationally extensive, one instead looks at patches of inputs and connect the patches to the neurons instead. By sliding the patch-window all across the pixel-matrix, by e.g. 3 units each slide, one creates

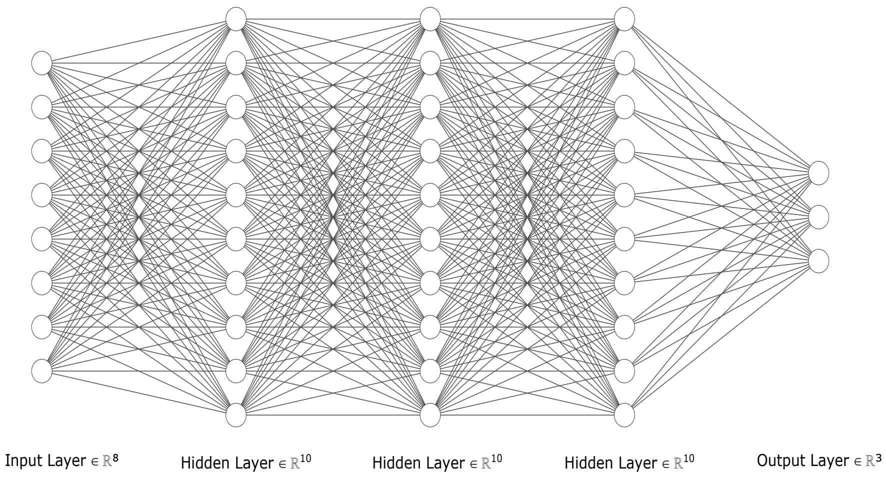


Figure 2.6.3: A dense artificial neural network consisting of one input layer, 3 hidden layers and an output layer. Image created by Eivind Lyseheim in NN-SVG.

connections between each patch and the neuron. This will keep more of the spatial structure intact, as the objective was.

Filters, Feature Maps and Max Pooling

It is not only the spatial structure that is of interest, but also the features that can be extracted from the image. The term "features" is a very broad term, but in the case of computer vision it applies to the detection of edges, curves, facial attributes etc. To extract features, a popular technique is to weigh each pixel with a filter before connecting the patch to the neuron in the next layer. This operation is simply known as convolution. In practice, for a 2D image, this operation first takes a filter, usually a 3x3 matrix, containing 9 different weights and apply it to the 3x3 patches. The dot product of the patch and the filter is then taken, i.e. the convolution of the patch and the filter, and then summed together. A feature map is then created. The output of this layer reflects the features of the patches. The convolution operation is shown in Figure 2.6.4.

We use a filter that is larger than 1x1 as features are usually local and not confined in just one pixel, meaning that we should extract information from a region, not a point. The filter size is odd because the pixels in the previous layer are symmetrically placed around the output filter. If we had implemented even sized filters on the other hand, we would need to account for distortions across the layers, making things unnecessary complicated. The reason that 3x3 filters are normally preferred, is mainly due to

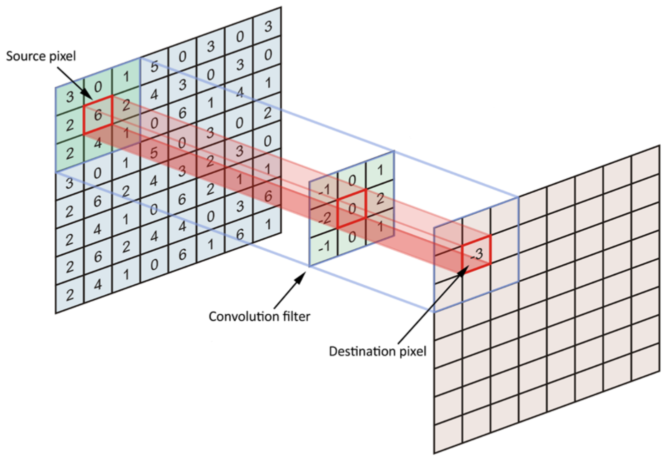


Figure 2.6.4: A figure showing the 3x3 filter operation in a convolutional layer. Courtesy of Thomas Dean (69).

computational resources, as 5x5 filters will create almost 3 times as many trainable parameters, causing a massive increase in computational cost, in turn making training very slow. The filter needs to learn how to extract the features, therefore the weights in the filter is updated through back propagation. By applying subsequent convolutional layers, each new layer has the ability to extract higher order features from the existing feature maps. E.g. the first layers can extract features such as dots and corners etc. Subsequent layers can combine those small features to extract shapes and trends etc.

Two important parameters are used to define the convolutions, stride and padding. Stride is the number of pixels which we slide our filter matrix over the input in each step. Stride of 1 means that the filters are only moved one pixel before a convolution is performed. Padding are filter elements that can be added outside our matrix, making it possible to perform convolutions at edge of the matrix.

In addition to the normal convolution that has been introduced, the transposed convolution, or the "de-convolution", does the opposite, and is very useful tool when we want to up-sample an image from low resolution to high resolution. Again, we start by taking a pixel as the input and convolute it with the filter. This is then performed with all the input pixels. The filters can then be trained to perform the best up-sampling possible.

In CNNs it is sometimes useful to reduce the dimension of the input so that we can work with fewer parameters. The max pooling operation takes the highest pixel value from sections of an image, creating a pooled feature map. In this map only the features which describe the context of an image the best can be seen, neglecting the features that are deemed not important enough.

2.6.3 Overfitting

Overfitting is one biggest challenges one meets in the field of DL (70). It is concerned with the generalisibilty of the network. If the network is overfitted, the weights and biases have most likely been tuned to "remember" the images and creates a prediction based on what it remembers is correct, rather than actually learning the features of the images. An overfit network has a high variance and low bias, and the performance varies widely with the introduction of new images, or even known images with added statistical noise.

Overfitting can be diagnosed by looking at the loss and validation loss. Normally, both losses will decrease then starts to stabilise. However, in an overfit model the validation loss will start to increase again after a while. As the model starts to remember the images, instead of learning the relevant features, its prediction on the validation set will loose its accuracy, in turn leading to a higher validation loss. A simple solution to this is implementing an early stopping mechanism that stops the training and saves the weights and biases of the model if the a criteria is met, e.g. the validation loss starts to grow over a certain threshold.

However, there exists other preventive methods for reducing overfitting. One method is to introduce dropout layers, placed after convolutional layers (71). This layer sets input units to zero, with a frequency given by the rate during training. The weights of the inputs that are not set to be 0 are scaled up by $\frac{1}{1-rate}$. Therefore, the sum over all inputs remain unchanged. A major drawback of the dropout layer is that some information go lost when the input is set to 0. Therefore, the dropout-rate should be lower at the start of the network than at the end of the network.

Another method for reducing overfitting is to increase the size of the dataset (72). By adding data, the model is unable to overfit all samples, forcing it to generalise. If it is not possible to obtain more real data, data augmentation can be performed by applying transformations to the existing dataset. These transformations can be angle rotations, sheer forces, flipping, addition of noise, contrast improvement, blur etc.

2.6.4 U-Net

The 2D U-Net was first introduced in 2015 (73). The U-Net architecture is based on a contracting path introduced to capture the context in the image and an expanding part introduced to capture the localisation of a feature. The symmetry between the contracting and the expanding parts creates the "U" structure, as seen in Figure 3.7.1. The U-Net exceeds mainly due to two different aspects; its high precision segmentation as well as its ability to train using only a fraction of the images other state of the art CNNs need (73). The left part of the network that contains the high resolution features are concatenated with the expansive part. A subsequent convolutional layer will make it easier to learn the network to produce high precision predictions.

The left side has a three dimensional input. Two dimension comes from the height and width, while the last dimension comes from the colours in the image (RGB). This is passed through a convolutional layer which extracts features, creating a feature map (65). The data is then passed through a batch normalisation layer and a dropout layer, which normalises the input values and skips the training of a fraction of the neurons, respectively (74)(75). This is repeated once before the data is both concatenated with its analogue in the expanding part, as well as being down-sampled by a max pooling layer (76)(77). A concatenation is just an addition of "List A" to the end of "List B", where the product is a list that has the length of List A + List B. The purpose of this operation is to reuse features by concatenating them to new layers, which allows for more information to be retained from previous layers. The concatenation operation comes before the max pooling operation, which only extracts the strongest features. These operations are repeated 3 times before one meets the lowest layer of the U-Net. Here the data is passed through two convolutional layers, before it is passed to a deconvolutional layer that up-samples the data (78). As the output is a high-resolution image, in which all pixels are classified as either ROI or non-ROI, the deconvolution is needed to up-sample the image. The two next steps of the expansive part of the network is identical, but it includes the data from the concatenation. The final convolutional layer maps each each n-component feature vector to the amount of classes in the system.

Chapter 3

Material and Methods

This chapter will describe the operations performed in this project, which is heavily based on theory introduced in the previous chapter. The chapter is mainly divided into 8 sections. The first section describes the necessary steps to set up HUNT Cloud. The following four sections describe the acquisition methods, QSM reconstruction and how the manual segmentations were obtained. The following section describes which operations were performed for data augmentation. The last two sections describe the network architecture and the details about the training of the CNNs.

3.1 Setting up HUNT Cloud

All the imaging data needed to be transferred to secure servers to the cloud community, HUNT Cloud, operated by NTNU and St.Olavs hospital (79). In addition to secure containment of data, the cloud community offers state of the art GPUs and Central Processing Units (CPUs) for high speed data processing. As the MR Physics group at the university recently became a user of this digital lab system, a part of the thesis was concerned with setting up and installing the relevant software on the servers. The goal of the task was twofold; firstly the relevant software needed to be installed to allow for GPU accelerated training of CNNs, as described more in detail in the subsection below. Additionally, by spending some time setting up HUNT Cloud and installing the relevant research software, new PhD. and master students can start using the HUNT Cloud immediately, without the need to perform tiresome installations and changes to the system.

A short summary of the process of setting up the Cloud is described here, and a detailed documentation can be seen in Appendix B.1. After the initial user registration was finished, a Graphical User Interface (GUI) was installed. Subsequently, all the necessary packages were installed to perform needed operations in the lab, the most important were a Conda (80), NIfTI viewers, MATLAB, FSL and Python interpreters. The Conda environment was configured to run the TGV-QSM reconstruction and GPU accelerated training of CNNs.

Tensorflow and GPU

Tensorflow is a an open source library with a particular focus on DL operated by Google, and provides an interface for simpler implementation of neural networks (81). Additionally, Tensorflow provides a vast collection of classes which can improve training, stability and analysis of neural networks. It also handles memory allocation and optimisation of Graphics Processing Unit (GPU).

As working with neural networks usually include handling huge amount of data and parameters, GPUs are exploited for acceleration of training as they allow for parallel computation. Inside the GPU a particular task is broken into different independent pieces, and calculated simultaneously. This allows for an increase in computational efficiency, leading to a decreased training time compared to conventional CPUs. Additionally, as the memory of GPUs tend to be quite large, they are suitable for handling large amounts of data. The GPU used for training in this project was a NVIDIA Tesla P100 with Pascal architecture and 16GB of memory. Despite having all the hardware ready, one can not take advantage of accelerated training without installing the necessary software offered by NVIDIA. Essentially, the NVIDIA container toolkit allows a user to run GPU accelerated containers. The containers are analogous to the containers used in shipping today. They can contain everything, but the outside is standardised. By using the Cuda toolkit, one can run everything from neural networks to FSL without the need to interfere with the GPU software each time one needs to use a different application. The Driver Version was updated to version 460.73.01, Cuda version was updated to 11.2 and Cuda compilation tools 9.1.85. These updates were chosen to be in compliance with Tensorflow-GPU 2.0.0.

3.2 Data Acquisition

The image acquisition was performed on three different scanners. The acquisition and processing of the first 3T QSM data, was performed by an external partner at Karolinska Instiutet in Sweden, and will be referred to as the GE 3T dataset. The acquisition of the second 3T QSM dataset was performed at St.Olavs university hospital, Trondheim, Norway. This will be referred to as the Siemens 3T dataset. The third dataset was obtained at the new 7T MR scanner at St.Olavs, owned by NTNU. This will be referred to as the 7T dataset. The processing of the Siemens data was done locally by the author on the Hunt Cloud machines. The data collection performed here in Trondheim was approved by the Regional Ethical Committee. The thesis is a part of the project "Kvalitetssikring og optimalisering av pasientundersøkelser på 7T MR" (Quality assurance and optimisation of patient examination on 7T MR), with project number: 108066. A total of 18 volunteers were recruited by the author, and all signed a contract consenting to that images could be used anonymously.

3.2.1 MRI Acquisition GE 3T

Data was collected from 40 healthy participants (82). 22 were a group of younger adults with a mean age = 36.18 years, SD = 4.66 years, in the range 26-42 years old. The remaining 18 were a group of older adults with a mean age = 69.89 years, SD = 3.17 years in the range 65 - 77 years. The image acquisition was performed using a Discovery MR750 3T scanner (General Electric, Milwaukee, WI, USA), using an 8-channel phased array-receiving coil. The sequence used was a MGRE, with TR = 37.52 ms, spatial resolution 0.9 x 0.9 x 1 mm^3, flip angle of 20°, FoV = 24 cm and 176 axial slices. First echo was after TE = 3.74 ms and the following 7 echos had a constant spacing of TE = 3.75 ms.

3.2.2 MRI Acquisition Siemens 3T

The MRI acquisition was performed at St.Olavs university hospital, Trondheim. The image acquisition of 18 volunteers was performed in the period November 2020 to March 2021. Due to the global pandemic caused by Covid-19, only healthy young subjects were recruited, as it was deemed unethical to encourage older adults to visit a hospital unnecessarily, possibly exposing them to the virus. The mean age = 25.63 years, SD = 3.78 years in the range 22 - 40 years. The imaging was performed using the Magnetom Skyra 3T scanner (Siemens, München, Germany), with a 32-channel phased array-receiving coil. A 3D MGRE sequence was performed, with a spatial resolution of 1 x 1 x 1 mm^3 and TR = 39 ms, FoV = 23 cm, 144 axial slices and flip angle of 20°. The MRI sequence used a MGRE, with first echo after 6.15 ms and the 5 subsequent echos had a spacing of TE = 5.48 ms.

3.2.3 MRI Acquisition 7T

The MRI acquisition was performed at the 7T MR lab at St.Olavs university hospital, Trondheim. The same 18 volunteers were imaged using the 7T MAGNETOM TERRA (Siemens, München, Germany). The image acquisition was performed in the period January 2021 to April 2021. Due to different causes, 3 of the volunteers were not able to complete more than the first few minutes of the scan, yielding no relevant data at all. Among the volunteers who managed to complete the scans, mean age = 25.81 years, SD = 4.08 years, in the range 22 - 40 years old. The MRI scanner had a 32-channel phased array-receiving coil. A MGRE sequence was performed, with a spatial resolution of 0.75 x 0.75 x 0.75 mm^3 and TR = 31 ms, FoV = 23 cm, 224 axial slices and flip angle of 12°. The first echo was after 2.54 ms and the 3 subsequent echos had a spacing of TE = 4.68 ms.

3.3 QSM Acquisition

3.3.1 QSM Acquisition GE 3T

The QSM images were obtained using the data obtained from a MGRE sequence. By employing a non-linear least squares fit, the frequency in each voxel could be determined (28). By applying a 3D best path unwrapping (83), the frequency of each voxel were unwrapped. The RESHARP algorithm was then employed to remove unwanted background inhomogenities.

The MEDI method for reconstructing the susceptibility map was employed. The images used in this project were processed using a non-linear extension (28). It imposes a data fidelity constraint, which is determined by the difference between the generated frequency map and complex exponential functions. The reconstructions were performed in MATLAB (84). Implementations of RESHARP and MEDI are open source algorithms (85).

To overcome the relative susceptibility issue, the average susceptibility of region in the Corticospinal Tract was subtracted. Applying FMRIB's Non-Linear Image Registration

Tool (FNIRT) from the FMRIB's Software Library (FSL), the obtained coordinates could be mapped to the individual coordinate system (86). The theory behind this operation is described in Section 2.5.2. A region-growing algorithm was then introduced to centre 1000 voxels around the mapped coordinates, only encompassing white matter. The region creation was performed by an in-house Python program. In the end, the FAST algorithm from the FSL library was used for white matter segmentation (9).

3.3.2 QSM Acquisition Siemens 3T

All the data obtained from the 3T scans were transferred to secure servers at HUNT Cloud. The TGV-QSM algorithm described in section 2.4.8 was implemented to reconstruct QSM images from the phase and magnitude images obtained from the Siemens 3T scanner. Before applying the TGV-QSM algorithm, some initial pre-processing was necessary. First the DICOM images were converted to NIfTI images using software available from MRIcron (87). This operation was performed as the TGV-QSM algorithm, and its components from the FSL library do not support DICOM images. FSL ROI from the FSLutils library (86) was run to extract the brain and the skull from the background. Subsequently, the BET described in section 2.4.4 was applied to separate the brain from the skull, creating a brain mask, which later was used to create a field estimation by the susceptibility sources inside the ROI. The magnitude images were then separated based on the echo time at acquisition. After this operation had been performed, the phase images were extracted using FSL ROI, unwrapped and then scaled. Now that the initial pre-processing was finished, the TGV-QSM algorithm was applied for each echo time, removing the background field and performing the dipole inversion, in this case 5 times. The median of the 5 QSM reconstructions was then calculated using FSL maths (86). In the end, if the determinant of the image was smaller than 0, the left/right orientation was flipped to present the image with the standard orientation. Finally, the average susceptibility value in the CSF region was used as reference susceptibility value for the dataset.

3.3.3 QSM Acquisition 7T

After the 7T MRI sequence was performed, the data was transferred to HUNT Cloud. The magnitude images were pre-processed using standard Siemens software, but the phase images were obtained using the ASPIRE method descried in Section 2.4.3. The data was then processed in the same manner as the Siemens 3T data. Additionally, due to the setup of the local data system at the Hospital, the susceptibility values in the

DICOM images are in integer values ranging from 0 to 4095, and not the actual floating suscpetibility values. This meant that a small conversion had to be performed when calculating the susceptibility values:

$$I_{NIfTI} = \frac{I_{DICOM}}{10000} - 0.2048 \quad | \quad I_{NIfTI} \in [-0.2048, 0.2047] \tag{3.3.1}$$

where I is the intensity of the image, i.e. the susceptibility value. If the image has intensity higher or lower than the extremes, the values are set to the closest extreme value, i.e. 0.2047 or -0.2048.

3.4 Manual Tracing

The manual delineation of RN, SN and STN for the GE 3T dataset was performed by a senior researcher (Grégoria Kalpouzos) at Karolinska institutet in Sweden. The anatomical reference that was used was the Duvernoy's atlas of the Human brain Stem and Cerebellum (88). The intensity contrast due to difference in susceptibility was used to separate tissue of interest and the remaining tissue.

The Siemens 7T dataset was manually segmented by a neurologist (Runa Geirmundsdatter Unsgård) at St.Olavs hospital. The manually segmented ROIs were RN, STN and SN, in addition to the PMC, PSSC and the CSF. The PMC was divided into three substructures: Omega, Arm and Face. All ROIs are depicted in Figure 3.4.1. The QSM images were visualised and segmented using the software application ITK-SNAP (89). Due to the limited amount of neurologists available and the limited period of this project, the Siemens 3T QSM images were not labelled. The manually labelled brain structures were then split into individual NIfTI-files using a in-house Python script before they could be co-registered to their complementary 3T image, as described in the following section.

3.5 Intra-subject Label Transfer

As only the Siemens 7T QSM images were manually labelled, a workaround was implemented to obtain segmentations for the Siemens 3T dataset. For the volunteers which managed to finish both the 3T and 7T scans one could perform a linear image registration between the 7T and the 3T images of the same subject, to transform the segmentations from 7T to 3T QSM images, as described in section 2.5.1 and depicted in Figure 3.5.1 .

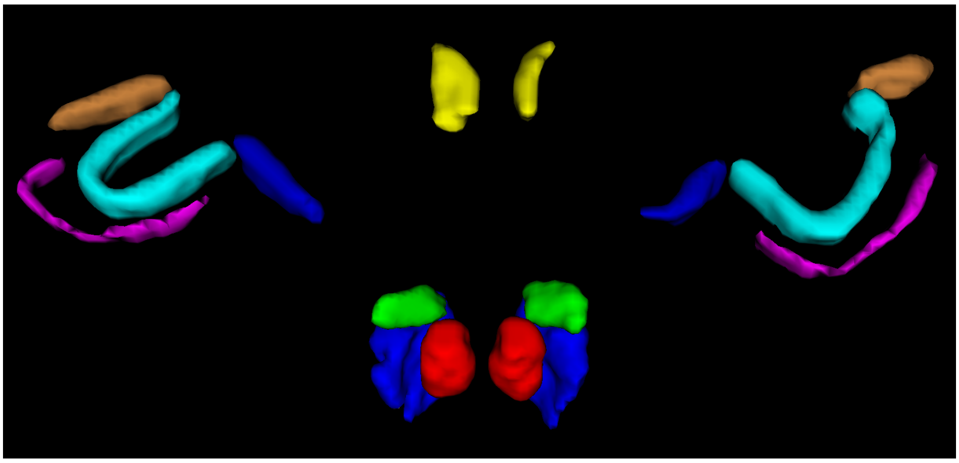


Figure 3.4.1: A figure showing the segmentations of RN (Red),SN (Blue), STN (Green), CSF (Yellow), (PSSC (Pink) and the PMC with Omega (Blue), Arm (Purple) and Face (Brown)

First the Affine transformation, allowing 12 DoF, transformed the 3T images into higher resolution, 7T images, was calculated using Correlation Ratio as cost function and a tri-linear interpolation. The inverse of the transformation matrix was then calculated and applied to the segmentations, transforming the segmentations into 3T resolution. The segmentations were now registered their corresponding coordinates in 3T space, and could then be used on the 3T dataset. As commented in the section above, to avoid thresholding complications, the segmentations of the different brain structures were divided into separate files as the different segmentations had different RGB values. The software FMRIB's Linear Image Registration Tool (FLIRT) (58)(57)(90) was used to calculate and apply the transformations and inversions and an in house python was applied to ensure correct slice registration. In the end, a threshold of 0.9 was applied to ensure minimal over-labelling (91). For each step of the method a visual inspection was performed to ensure high quality segmentations of the Siemens 3T images. See appendix B.1 for python script, installation of FSL and bash commands. This method worked for 12 of the 15 people who had both performed 3T and 7T images, for the remaining 3 volunteers, linear image registration was not successful. A visual analysis uncovered that the co-registration between the 3T image and 7T image had not been performed properly, in turn ruining the 3T segmentations. Uncovering the source of this problem is outside of the scope of this thesis, but it is believed that filters used for removing background field contributions in the QSM algorithm might have been to aggressive on either the 3T or the 7T scans, removing crucial structural information, in

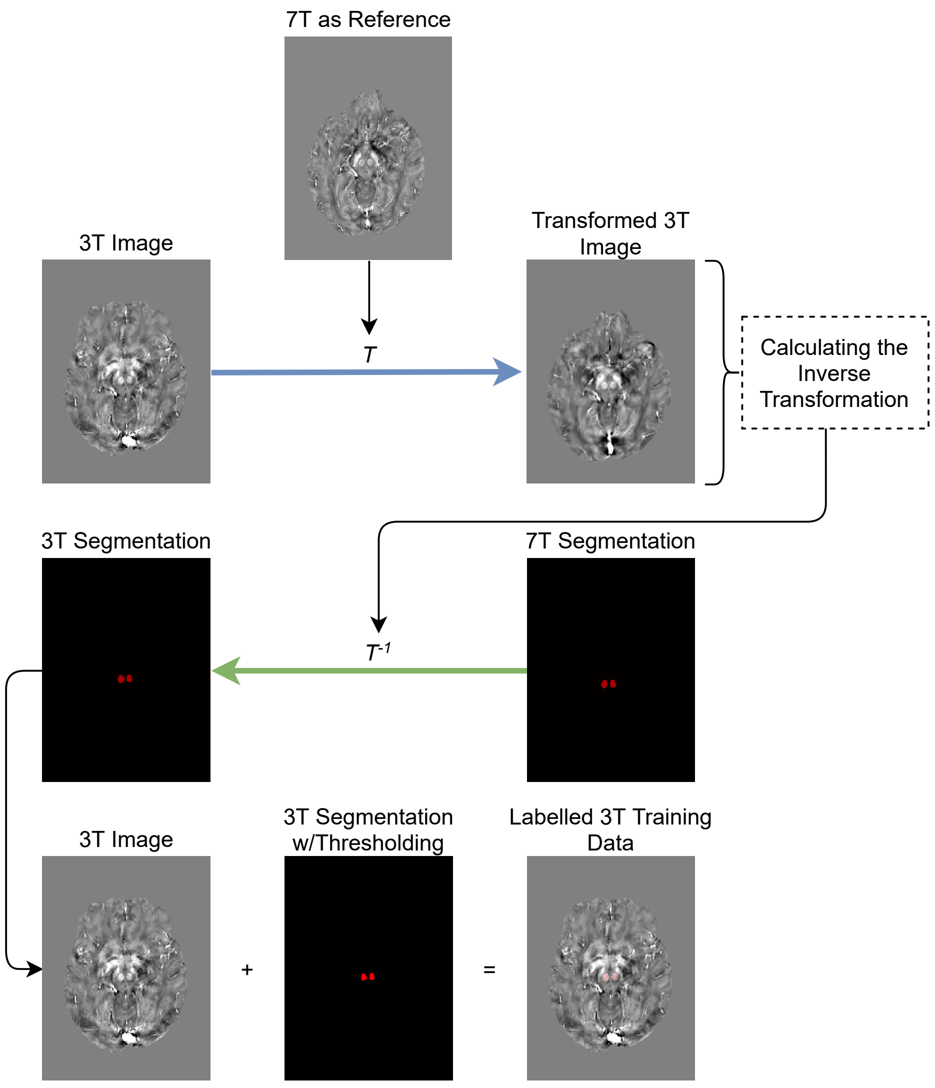
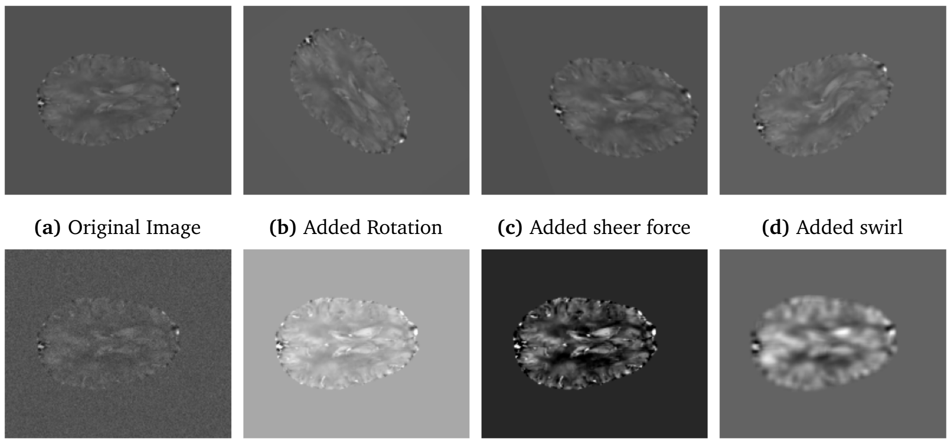

7T as Reference
3T Image
Transformed 3T Image
T
Calculating the Inverse Transformation
3T Segmentation
7T Segmentation
T^{-1}
3T Image
3T Segmentation w/Thresholding
Labelled 3T Training Data
+
=

Figure 3.5.1: An overview of the entire linear registration operation. From the top: the transformation from 3T image to 7T image is calculated using 7T as reference. The inverse transform is then calculated. Second row: the inverse transform is applied to the 7T segmentation, yielding 3T segmentations. Bottom row: the segmentations are then thresholded and applied to the original 3T image.

turn making co-registration impossible. The same filters are believed to be the reason why co-registration of PMC and PSSC did not work either, as these are situated at the edge of the brain, close to background field, creating distortions, in turn making co-registration impossible for these brain structures.

3.6 Data Augmentation

As the datasets used for training of the CNNs were sparse, there existed a possibility that the network would be prone to overfitting. Acquiring more data was not an option in this case, but one could create more data from the existing datasets. By introducing sheer-forces, random flips, addition of random noise, contrast and blur, the dataset substantially increased in volume. Examples of these modifications are shown in Figure 3.6.1.



(a) Original Image **(b)** Added Rotation **(c)** Added sheer force **(d)** Added swirl

(e) Added noise **(f)** Gamma enhancement **(g)** Enhanced contrast **(h)** Blur and contrast

Figure 3.6.1: Figures showing different methods to augment data on the GE 3T dataset. All image enhancements have been done on subfigure a. Example shown on QSM images obtained from the GE 3T dataset.

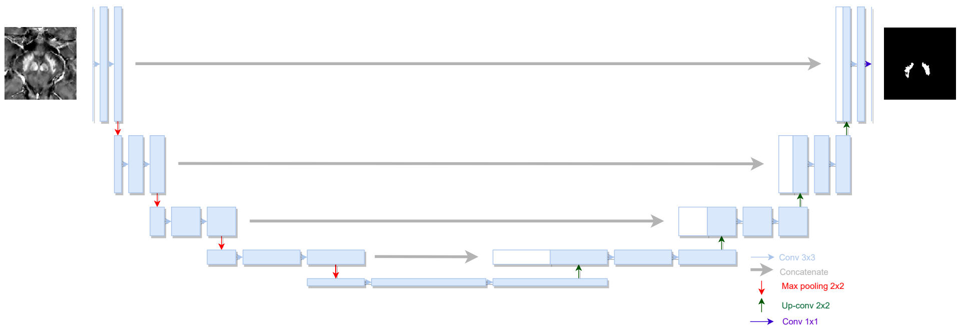


Figure 3.7.1: The general architecture of a 2D U-Net.

3.7 Network Architecture

3.7.1 U-Net

The choice of a 2D U-Net rather than a three dimensional one, was twofold; firstly, 2D U-Nets have shown itself to obtain comparable results to 3D U-Nets, and adding another dimension is computationally expensive as the amount of trainable parameters explodes. As the segmented data was received quite late in this project, we did not have the time to train multiple 3D networks. A solution to this might have been to crop the images to e.g. 80x80x80 etc. However, this would remove a lot of valuable information, especially on the 7T images. Instead, the 3D images were divided an array of 2D images and trained separately. The predictions were then performed on each individual slice, then reconstructed on top each other again, yielding fully segmented 3D images.

The general architecture of the 2D U-Net used in this project is shown in Figure 3.7.1. The difference between the U-Nets that have been applied its the amount of filters used. For less complex brain structures, or small dataset, it is typically normal to use less filters to avoid creating a too complex model, in turn causing overfitting. For the RN, SN and the substructures of the PMC, an initial 16 filters were used in the first layer and 256 filters in the lowest layer. Doubling the amount of filters after each max pooling operation. For the STN and PSSC, which are more complex, the U-Net had an initial 32 filters in the first layer and 512 in the lowest layer.

The ReLU activation function has been used in all convolutional layers except for the output, in which the Sigmoid has been used. The Sigmoid has been introduced to force the output prediction to take values between 0 and 1. The filters used in the convolutional layers were of the size 3x3, and padding was set to "same". This ensures that the image dimensions are kept constant during convolution. The dropout rate in all layers were between 0.1 and 0.3, with lower rate in the initial layers to inhibit too much informaiton loss. The max pooling layers all had a pool size of 2x2, meaning that the dimension of the image was halved every pooling. The deconvolution layers had filters of size 3x3 and strides of 2x2. The latter was used to double the size of the image. The choice of a 3x3 instead of a 5x5 filter was to reduce the trainable parameters in an already computationally extensive network. The last convolution layer had a filter of size 1x1 to maintain the dimensions of the output image equal to the input.

The "RMSprop" was chosen as the optimiser due to its robustness. In addition, the learning rate had to be tuned. Leading to a modest learning rate of 0.001. An in-house implementation of the weighted Dice coefficient was used as metrics of accuracy and precision of the predictions. The weighted version of the DS presented in Equation 2.6.5 averages all the DS of the training set in an epoch, preventing large changes from batch to batch, improving robustness. The binary cross entropy loss function was implemented (92). The reason behind this choice was twofold; its a binary model, meaning that it separates between two classes only. In our case it separated between normal tissue and labelled tissue. Additionally, the output of the neural network is a prediction map, meaning its entropy compared to the manual tracing could be easily calculated.

As one downsamples by a factor of two, four times, a drawback of this network is that one needs image dimension of the training data to be dividable by 2^4, unless one adds an overlap-tile strategy for seamless segmentation of arbitrary large images (73). Without this modification, the initial image dimension is restricted. Both by the factor of 2^4 and the memory on the computer, as larger image dimension leads to higher memory use.

Additionally, a so called "prefetch" function was implemented. The purpose of this function was to open up for dimensional expansions and data augmentation while training. Conventionally, before training, the array containing all images is appended an additional axis to account for the channels in an image, e.g. 1 channel in our case. This method is easily implemented, but not effective w.r.t. memory use, as one always has a list with dimension N + 1 loaded into memory. An alternative to this approach is to create the prefetch function. This function fetches a new batch of images before the network has finished training on the previous batch, expands the array with a channel, and sends the batch to the neural network for training, allowing for more efficient memory usage. Moreover, one can also perform randomised image augmentation on the fly while training, for a continuously varied dataset. However, despite the advantages of this, it will slow down training if the image dimensions are too large and the GPU has finished training on the batch before new images has been augmented. Therefore, after some testing, image augmentation on the fly was not implemented during training.

3.8 Training and Evaluation

3.8.1 Training on the GE 3T dataset

25% of the GE 3T dataset was randomly chosen to be a part of the test dataset, while the remaining 75% was used for training. The RN, SN and STN were trained separately, each being trained three times with validation splits of 10%, 15% and 20%. After training was performed, post-processing steps were implemented, such as the nearest neighbour algorithm where a segmentation was removed if it did not have a neighbouring segmentation in any of its neighbouring lattice sites. This was implemented based on the fact that it is not natural that the e.g. the RN has a segmentation floating in free space, separated from the rest of its body. In total 9 CNNs were trained on the GE3T dataset. The network performance was then evaluated by performing a visual analysis, calculating the DS, sensitivity, precision and comparing the predicted susceptibility to the ground truth.

3.8.2 Training on the Siemens 3T and 7T datasets

As the datasets of pure 3T and 7T images were quite small, only containing 12 and 15 images, respectively, we did not have the luxury of dividing the dataset into a training and test set. A method for solving this problem is using k-fold cross validation. This method divided the dataset into k amounts of folds, where all except one fold is used for training and the remaining fold is used to evaluate the model. For both datasets a "leave-one-out" algorithm was applied. Thus, 11 and 14 images were used for training, and the remaining image was used as validation, for the 3T and 7T dataset, respectively. The cross-validation performed on the 7T dataset can be seen in Figure 3.8.1. This is a computationally expensive algorithm, but due to limited amount of data this considered the best solution. This method is commonly used for small datasets to avoid overfitting and obtain a better statistical foundation, as one exploits the entire dataset. The same post-processing steps and evaluation methods used for the GE 3T dataset was implemented on the Siemens datasets aswell.

The 3T dataset contained segmentations of only RN, SN and STN, while the 7T dataset also contained segmentaions of the PSSC and the substructures of the PMC,

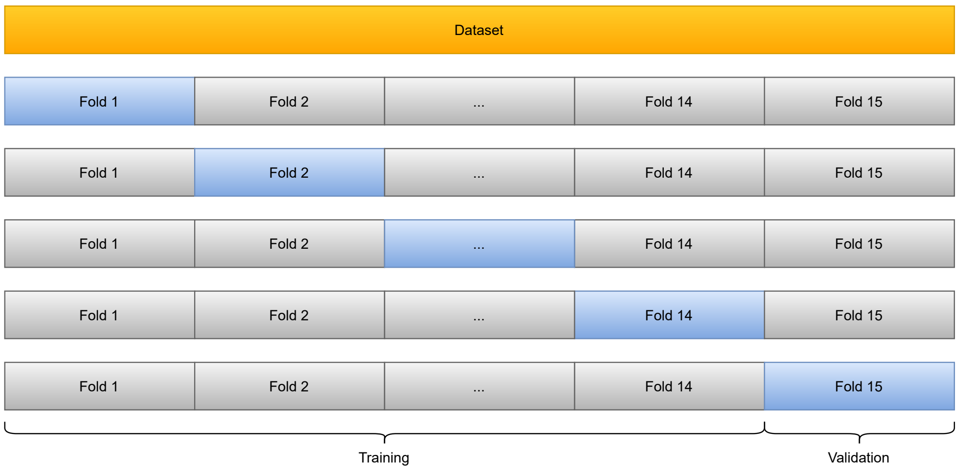


Figure 3.8.1: An example of 15-fold (leave-one-out) cross validaiton. Here the entire dataset is divided into training and test set. The training set is then divided into 15 folds, where 14 out of 15 (grey) is used for training and one is used for validation (blue). After a certain amount of training, the fold used as validation is changed.

Omega, Arm and Face. As the PMC and PSSC are not particularly magnetic, therefore not being particularly rich in contrast as seen in Figure 3.8.2, the only interest was to measure its DS, and analyse whether or not the segmentations were good enough to be implemented as a tool for the radiologist. In total, 36 CNNs were trained on the 3T dataset and 105 CNNs were trained on the 7T dataset.

3.8.3 Evaluation Metrics

The segmentation accuracy and precision was evaluated using the DS, precision and sensitivity described in Section 2.6.1. The susceptibility values obtained by the CNNs were compared with the ground truth in violin plots. The middle line in these plots represents the average susceptibility value, while the upper and lower lines represent the maximum and minimal values. The light blue "clouds" represent the density of measurements, i.e. a wider cloud represents higher density. A similar comparison was performed, but now comparing one and one value, by plotting the ground truth against the predicted values. For perfectly extracted susceptibility values, i.e. a perfect segmentation, the measurements will lie on the diagonal. If the predicted values are higher than the ground truth, the measurements will lie above the diagonal, and vica versa if the predictions are too low. Additionally the Pearson correlation, which measures the linear correlation between two datasets, Mean Absolute Error (MAE) and Mean

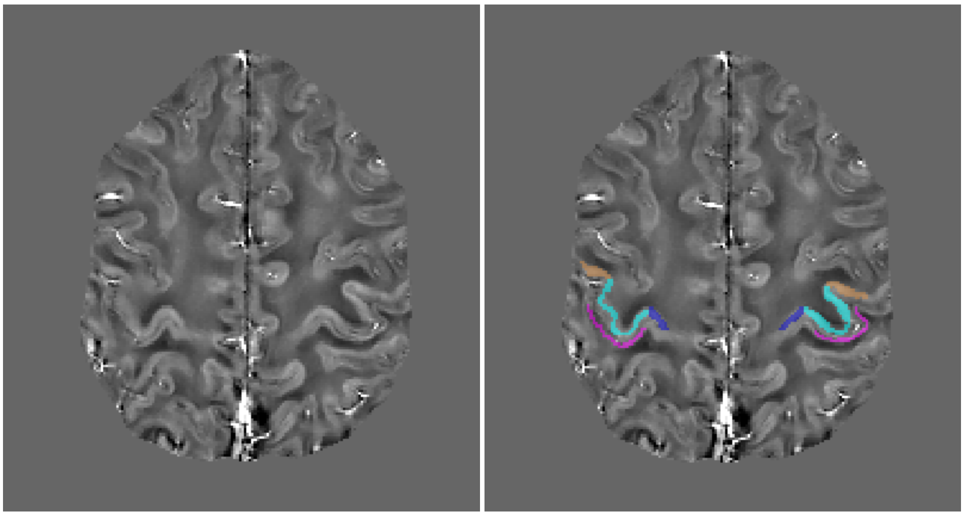


Figure 3.8.2: A figure showing the PSSC (Pink) and the PMC with Omega (Blue), Arm (Purple) and Face (Brown) on a 7T QSM image.

Absolute Percentage Error (MAPE) have been included. The latter is MAE divided by the ground truth, yielding a percentage error, which makes it possible to compare errors, independent of datasets and ROIs. MAPE* has also been included to compare two datasets with different ground truth. This is identical to the MAPE, but the instead of dividing by the ground truth, it has been divided by the average of the two values. Thus, MAPE* is only introduced to obtain a feeling about the size of the error, and not to be compared to any other MAPE measurements.

Chapter 4

Results

The result section is mainly divided into three parts, where the first part will discuss the accuracy of the segmentations based on both a visual and quantitative analysis. Segmentation accuracy and precision of networks trained on different data and using different hyperparameters will be presented and compared. Afterwards, the susceptibility values obtained using CNNs will be evaluated, comparing how well the predicted susceptibility values are compared to the ground truth. The last part of this chapter concerns the comparison of susceptibility values obtained intra-subject on the Siemens 3T and 7T scanners.

4.1 Image Segmentation

This section will present the segmentation accuracy of the CNNs for the different datasets. For the 3T datasets, the segmenations of the RN, SN and STN will be presented. Meanwhile, for the 7T dataset, the segmentations performed on the PSSC and the substructures of the PMC will be presented aswell.

4.1.1 RN, SN and STN Image Segmentation on GE 3T Data

Table 4.1 shows the mean DS with Standard Deviation (SD) obtained for the RN, SN and STN with validation split of 10%, 15% and 20%, respectively. As we can see from Table 4.1, the general trend is that the CNNs trained on the RN obtains the highest DS. The highest DS are obtained for validation splits of 10% and 20%, while showing an inferior DS for validation split 15%. The DS for SN is also quite high, but lower than for the RN. The STN lies consistently about 0.1 points below the SN. For the SN and STN, the DS do not fluctuate as much as for the RN when the validation split changes.

Table 4.1: DS for RN, SN and STN for the GE 3T dataset for different validation splits. The SD is included.

DS RN,SN, STN for the GE 3T dataset			
	Val 10%	Val 15%	Val 20%
RN	0.92 ± 0.02	0.88 ± 0.04	0.92 ± 0.01
SN	0.88 ± 0.03	0.87 ± 0.03	0.86 ± 0.06
STN	0.79 ± 0.05	0.77 ± 0.03	0.78 ± 0.06

Additionally, the sensitivity and precision for the different segmentations with a validation split of 10%, 15% and 20% are shown in Table 4.2. In this case we also observe that the segmentation of RN has obtained the best results, closely followed by the SN, while the STN is a bit behind. For different validation splits, within the same ROI, the sensitivity and precision have approximately the same value with only minor exceptions, such as precision for RN performed with CNN trained with validation split 15%. By referring to Equation 2.6.6.

Table 4.2: Sensitivity and Precision for RN,SN and STN from the GE 3T dataset for different validation splits. The SD is included.

Sensitivity and Precision for RN, SN and STN for the GE3T dataset						
	Sensitivity			Precision		
	Val 10%	Val 15%	Val 20%	Val 10%	Val 15%	Val 20%
RN	0.94 ± 0.04	0.98 ± 0.02	0.93 ± 0.04	0.91 ± 0.05	0.79 ± 0.08	0.90 ± 0.05
SN	0.90 ± 0.08	0.91 ± 0.06	0.89 ± 0.12	0.87 ± 0.04	0.85 ± 0.04	0.86 ± 0.07
STN	0.79 ± 0.09	0.80 ± 0.10	0.79 ± 0.13	0.80 ± 0.07	0.78 ± 0.06	0.73 ± 0.11

Figure 4.1.1 has been introduced as an example of how the manual and automatic segmentation appear. This example is in the case of the GE3T dataset with a validation split of 10%. In the coronal plane the automatic segmentation appears precise, only with a minor under-labelling of the STN, making it look thinner and a bit longer than the manual segmentation. In the sagittal plane, we see that the segmentation of the STN looks very similar to the ground truth, but the SN extends all the way to the STN. Additionally we see some over-labelling of the “tail” of the SN to the right of the picture. The tail is a bit thicker than its manually segmented counterpart. In the axial plane, the RN is almost perfectly segmented. The left side of the left part of the SN is to some

extend over-labelled, but the situation is reversed at the left right side of the right SN. Additionally, the STN is not labelled at all in this slice, which we can see differs from the ground truth.

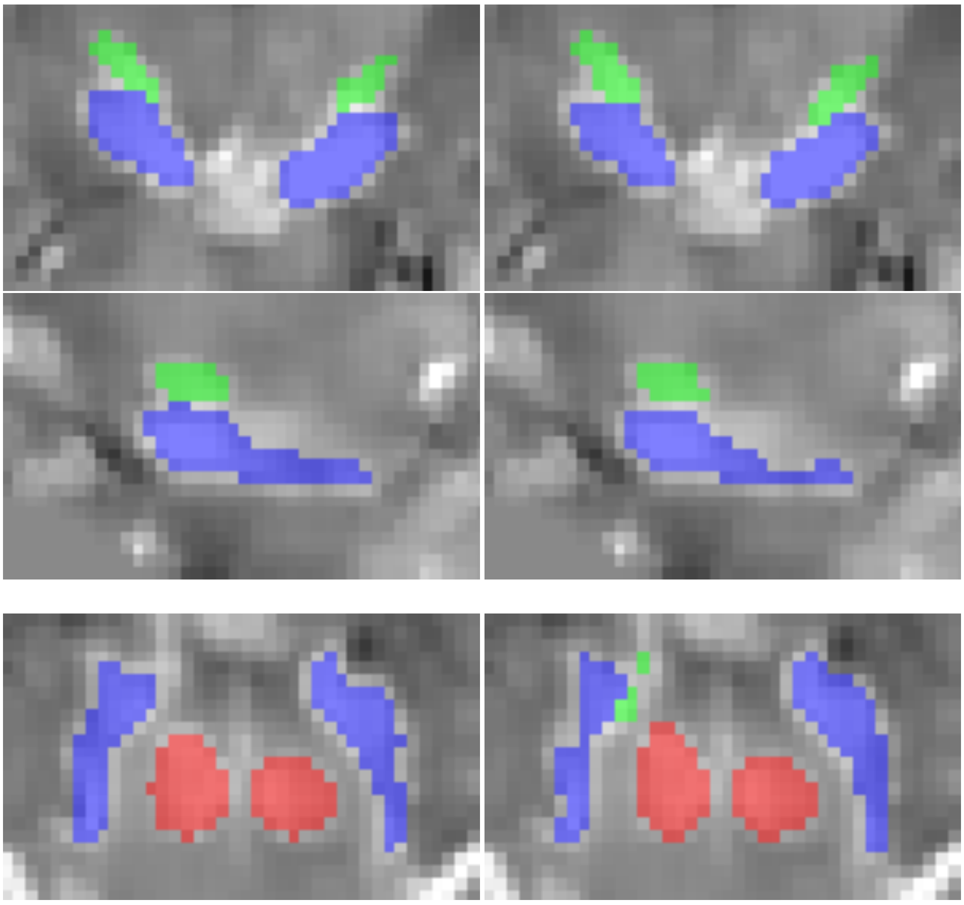


Figure 4.1.1: GE 3T automated (left) and manual segmentations (right) of the RN (Red), SN (Blue) and STN (Green) with a validation split of 10%. From top to bottom; coronal, sagittal and axial slice.

4.1.2 RN, SN and STN Image Segmentation on Siemens 3T Data

Table 4.3 shows the average DS, precision and sensitivity for the network. Again, we see that the segmentation of the RN has obtained the highest DS, followed by SN and STN. The DS for the RN is lower than most of the RN segmentations performed by the CNNs trained on the GE 3T network, but within one SD. This is also true for the SN. The DS for the STN, on the other hand, is higher than all values obtained by the GE 3T network, but still within one SD. The sensitivity for the brain structures varies between approximately 0.50 to 0.70, which is clearly lower than all the values obtained for the GE 3T segmentations. The precision, on the other hand, is converging towards 1.0. Compared to the GE 3T dataset, this is a major improvement, especially w.r.t. the segmentation of STN.

Table 4.3: Mean DS, precision, sensitivity and mean susceptibility values of the RN, SN and STN for the Siemens 3T dataset. The SD is included.

	DS	**Precision**	**Sensitivity**
RN	0.90 ± 0.08	0.99 ± 0.001	0.68 ± 0.03
SN	0.84 ± 0.01	0.99 ± 0.002	0.60 ± 0.04
STN	0.80 ± 0.02	0.99 ± 0.01	0.50 ± 0.03

4.1.3 RN, SN and STN Image Segmentation on Siemens 7T Data

The DS, sensitivity, precision for the RN, SN and STN are presented in Table 4.4. As seen in the case of both the 3T datasets, also here the DS follow the same pattern of being highest for RN and lowest for the STN. The obtained DS for the RN is the highest, but still within one SD compared to Siemens 3T and the GE 3T dataset with validation split 10%. It also has obtained highest DS for the SN, but within one SD compared to the GE 3T dataset. However, the DS for the STN is now approaching similar DS as for the SN, which is an improvement of approximately 0.1 compared to the 3T datasets. The sensitivity and precision follows the pattern of the DS, and has approximately the same value intra-ROI. Compared to the Siemens dataset, the sensitivity is much higher, but the precision is a lower.

Table 4.4: Mean DS, precision, sensitivity and mean susceptibility values of the RN, SN and STN for the 7T dataset. The SD is included

	DS	**Precision**	**Sensitivity**
RN	0.94 ± 0.01	0.93 ± 0.02	0.95 ± 0.01
SN	0.90 ± 0.01	0.89 ± 0.02	0.91 ± 0.02
STN	0.89 ± 0.02	0.89 ± 0.03	0.89 ± 0.03

4.1.4 PMC and PSSC Image Segmentation on Siemens 7T Data

The DS obtained for the PSSC was 0.80 ± 0.03, Arm 0.86 ± 0.02, Face 0.86 ± 0.04 and Omega 0.86 ± 0.02. The 3 sub-fractions of the PMC has a similar average DS, with Face having the largest SD. The PSSC obtained DS lower than the three, smaller substructures. An analysis of the obtained magnetic susceptibility can be seen in Figure A.9 and A.11 which show the error in predicted susceptibility as a function of DS for PSSC and PMC, respectively. Figure A.10 and A.12 show the ground truth plotted against the predicted susceptibility values for PSSC and PMC, respectively.

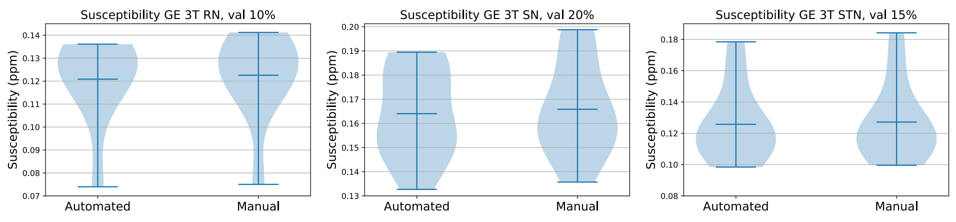
4.2 Extracting Susceptibility Values

The following part of the result section will focus on the performance of the CNNs in predicting accurate and precise susceptibility values. This part will mostly focus on the networks ability to extract correct susceptibility values and compare the values to the ground truth obtained from manual segmentations. The MAPE will then be compared across scanners and magnetic field strengths.

4.2.1 Susceptibility values: RN, SN and STN GE 3T

Violin plots showing both the predicted and ground truth susceptibility values for the GE 3T dataset are shown in Figure 4.2.1. These are not all the plots, but are representative for the different validation splits. Please see Figure A.1 for the remaining plots. The predicted susceptibility values agree quite well for RN with validation split 10% and 20%, while the network trained with validation split 15% is struggling to segment RN with higher susceptibility values. SN has similar predicted susceptibility values across all validation splits. However, also in this case, the networks are struggling to correctly predict susceptibility values with a larger magnitude. In the case of STN, validation split 10% and 15% the extracted susceptibility values agree with the ground truth. However for validation split 20%, we see that the network extracts susceptibility values which are too low.

Additionally, the difference between the ground truth and predicted susceptibility values, $\Delta\chi$, was found. For both the RN and SN, there was a correlation between increasing DS and decreased error. In 5 out of 6 plots this correlation coefficient is quite high. For the SN with a validation split of 15%, this trend is much weaker than for the prementioned 5. In the case of STN with validation splits 10% and 15%, we observe a very weak increase in error when the DS increases. With validation split 20 %, we



(a) (b) (c)

Figure 4.2.1: Violin plot comparing the average susceptibility of automatically segmented tissue and manually segmented tissue for RN, SN and STN for the GE 3T dataset.

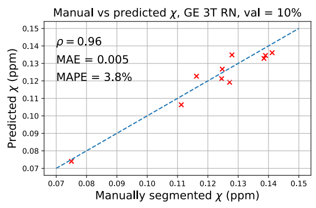
see again that the trend is decreasing. These plots can be seen in Figure A.2, with a linear fit and its Pearsson correlation coefficient, p-value and approximated functional expression.

The ground truth and susceptibility values extracted using CNNs were plotted against each other. Three plots, representative for the end results are shown in Figure 4.2.2 a,d and g, for the RN, SN and STN, respectively. For the RN, the predictions based on networks trained with validation split 10% and 20% yield similar results, while for validation split 15% we see that the MAPE is over twice as high, indicating that it predicts less accurate susceptibility values. For the SN the behaviour is similar for the different validation splits, with validation split 10% giving the best results. In the case of STN, validation split 10% and 15% show the best results, but for validation split 20%, the MAPE is almost doubled. Please see Figure A.3 for plots for each validation split for all three structures.

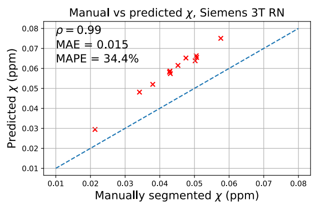
4.2.2 Susceptibility values: RN, SN and STN Siemens 3T

The extracted susceptibility values using CNNs compared to the ground truth for RN, SN and STN for the Siemens 3T dataset are shown in Figure 4.2.3. There are some clear deviations between the predicted values and the ground truth. Both the RN and STN are shifted with approximately 0.02 ppm, i.e. its maximal predicted values are too low. However, their density distributions are similar compared to the ground truth. For the SN, there is also an offset, but no striking similarities in the density distribution.

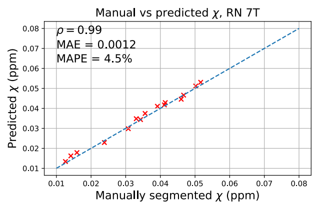
Looking at the error in predicted susceptibility values as a function of the DS, the trend that the error is increasing as the DS improves in the case of RN and STN. In the case of the SN, the trend is opposite. When the DS increases, the error in predicted susceptibility is decreasing. For complementary information, see Figure A.4.



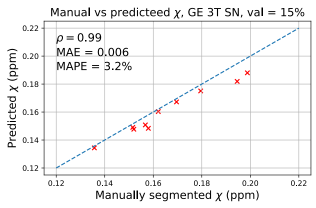
(a)



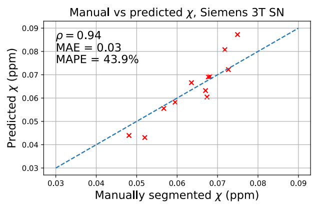
(b)



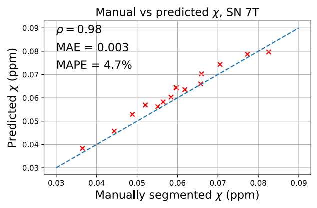
(c)



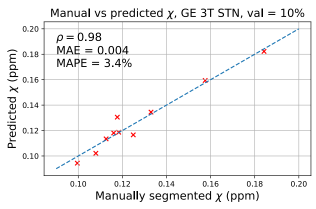
(d)



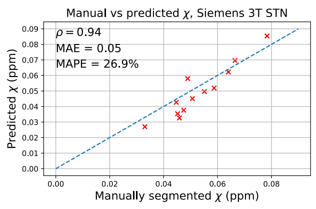
(e)



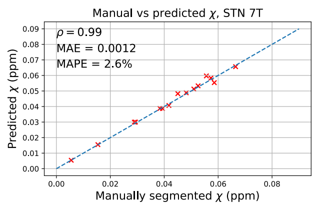
(f)



(g)

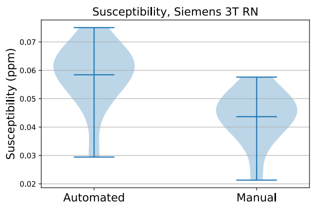


(h)

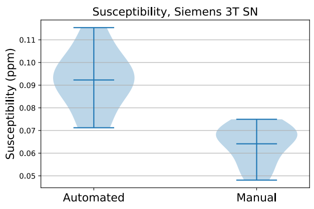


(i)

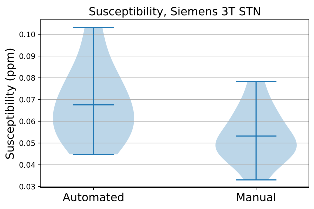
Figure 4.2.2: Susceptibility values extracted using the ground truth manual segmentation, plotted against the predicted susceptibility values extracted from the segmentation performed by the CNNs. From left to right: GE3T, Siemens 3T and 7T. From top to bottom: RN, SN and STN. The blue, dashed, line is the diagonal. Pearson Correlation coefficient, MAE and MAPE is presented. The x and y-axis do not have the same range for the GE 3T dataset and the Siemens datasets.



(a)



(b)



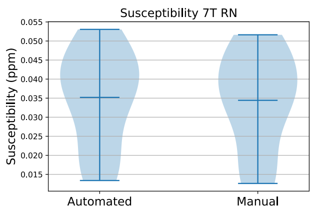
(c)

Figure 4.2.3: Violin plot comparing the average susceptibility of automatically segmented tissue and manually segmented tissue for RN, SN and STN for the Siemens 3T dataset.

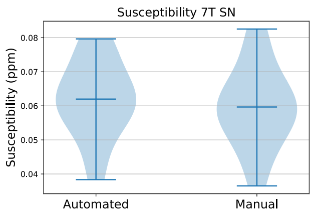
The three plots shown in 4.2.2 b,e and h show the susceptibility values extracted by the CNN, plotted against the ground truth. In all three plots we see substantial deviations between the predicted values and the ground truth. For the RN, we observe that the CNN is under-estimating all susceptibility values. For the SN and STN, the CNNs both under and over-estimates susceptibility, with no particular bias. The MAPE fluctuates between approximately 25% and 45%, indicating large errors.

4.2.3 Susceptibility values: RN, SN and STN Siemens 7T

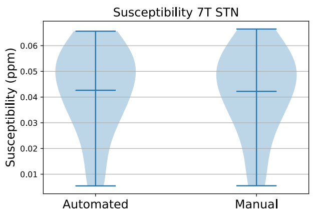
Violin plots comparing the susceptibility values of RN, SN and STN obtained by the automatic and manual segmentations are shown in Figure 4.2.4. For both the RN and STN we see that the predicted values are similar to the ground truth, with no major deviations for the RN and STN. However, for the SN, the predicted values have similar density to the ground truth, but misses to accurately predict both the highest and lowest susceptibility values.



(a)



(b)



(c)

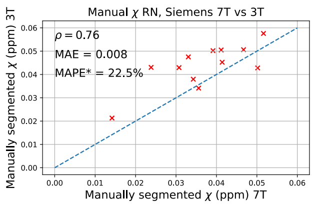
Figure 4.2.4: Violin plot comparing the average susceptibility of automatically segmented tissue and manually segmented tissue for RN, SN and STN for the 7T dataset.

For the error in the predicted susceptibility values, the trend is that the errors decrease when the DS increases. This is clearly seen for the RN and STN, with a stronger correlation coefficient. Meanwhile, the for the SN we see weaker trend. The plots are not shown here, but can be seen in Figure A.7.

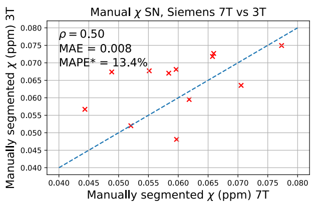
The plots in Figure 4.2.2 c,f and i show the predicted susceptibility values plotted against the ground truth. The RN and SN yields similar results with approximately equal MAPE at around 4.5%, while the STN has a MAPE which is almost half that of RN and SN. In the case of the RN and STN, we see that the network predicts both too high and too low susceptibility values, with no particular bias. However, the trend is that for the SN, it overestimates the susceptibility values.

4.2.4 Comparing Susceptibility Values: Siemens 3T and 7T

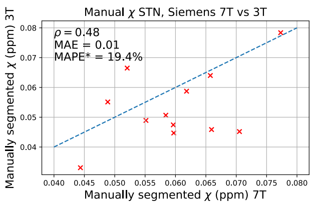
The manually obtained susceptibility values for the Siemens 3T and 7T dataset were plotted against each other, as seen in Figure 4.2.5. We see that there are some major deviations from the diagonal. The RN has the highest deviation between the 3T and 7T values, and the highest correlation coefficient. Meanwhile the SN and STN have comparable correlation coefficients. In the case of the RN, the 3T scanners yield in the majority of cases higher susceptibility values than the 7T scanner. This trend is opposite for the STN. In the case of the SN, the points are approximately distributed equally on both sides of the diagonal.



(a)



(b)



(c)

Figure 4.2.5: Manually obtained susceptibility for the Siemens 3T and 7T dataset plotted against each other for RN, SN and STN. The blue, dashed, line is the diagonal. Pearson Correlation coefficient, MAE and MAPE* is presented. From left to right: RN, SN and STN

Chapter 5

Discussion

This chapter will discuss the results presented in the previous chapter. It will follow the same structure as the results section, but three sections are appended at the end of this chapter. Section 5.1 and 5.2 will discuss the results tied to **RQ1**, while **RQ2** will be elaborated on in Section 5.2.4 and **RQ3** will be commented on in Section 5.2.5. The last section will evaluate and propose work which can be performed to improve the results obtained in this project.

5.1 Image Segmentation

This section will discuss the results presented in section 4.1, and will follow the same outline.

5.1.1 RN, SN and STN Image Segmentation on GE 3T Data

In Figure 4.1.1 we could see in the axial plane that the STN is not segmented at all. This might be related to the overlabelling of the SN, as the SN mostly covers the area which is actually the STN. Furthermore, the automatic segmentation algorithm has not covered the uppermost patch of the STN, which might reflect on the fact that this CNN is not 3D based, loosing some spatial information by not including information from the slices above and below when making a prediction. This will in turn make it more challenging to segment some smaller areas at the edge of the ROIs, as seen here. This challenge concerns all the datasets in general, as they are all segmented with 2D CNNs.

As identified partly from the segmentation of SN in the sagittal and especially in the axial plane, the segmentation of SN is not perfect. It appears like the CNN has some challenges when it is required to make a verdict between intensity based image segmentation and the regularisation of brain structures. An explanation for this ambiguity might lie in the processing steps of the GE 3T QSM images, as they were non-linearly registered to a common template, before the ROIs were drawn in. As discussed in section 2.5.2, when the images are registered to a common template, a regularisation process is initialised. The non-linear registration compromises between making the images look as similar as possible, but also making the warps look "natural” and "reasonable" (86). A smoother warp is more probable than a sharper one. Statistically, the dataset has a majority of smoother brain structures, which in turn has trained the CNNs to assume smoother brain structures are more reasonable, in turn prioritising this above intensity. However, for the right component of the SN, in the axial plane, we can see a tendency of the opposite, indicating that the earlier arguments may not reflect all situations.

As seen in Table 4.1, the segmentation of the RN has obtained the highest DS, which is as expected in general for all datasets. The nearest neighbours of RN have a noticeably different image intensity, and the border between ROI and non-ROI is clearly identified. Additionally, the form is quite regular, making it easy for the CNN to learn how to perform an accurate segmentation. One explanation why the RN is not segmented with a DS closer to 1.0, is due to individual differences in the manual segmentation, such as where the edge of the RN is defined. This is based on research that indicates that when an expert segments the volume twice, the expert only achieves a "self DS" between 0.82 and 0.86 (93). Evidently, there must be individual differences in the training data which is used as the ground truth. In Figure 4.1.1, we can see that the RN is manually segmented all the way to the edge some places, but other places not. Even for a professional, it is in challenging to keep consistent while performing the same operation multiple times. Due to the low complexity of the RN, the errors are most likely caused by this effect. This argument is also valid for the segmentation of the RN in the other datasets. It is interesting to see that with a validation split of 15%, the DS falls to the same level as for the SN. An explanation for this drastic change might be due to the choice of training, validation and test sets. As the data is anonymous, and the GE 3T data includes volunteers from a variety of ages, we do not know which subjects who are old and young. Let us say the test data has an overweight of older volunteers, and the validation set also has an overweight of older volunteers, causing the training set to be left with a bias of younger people, which have a different brain anatomy, e.g. susceptibility values (82). If the majority of training data is young adults, the CNN will struggle with segmenting QSM images of older volunteers. In hindsight, this could have been solved applying k-fold cross validation. E.g. training on 35 images and testing on 5 images at the time, and repeating this 7 more times. This would also have increased the statistical foundation for the analysis. Another likely explanation to the poor labelling of the SN, which also applies to neural networks in general, is that the weights are randomly initialised, and if we are unlucky with the initialisation, we might end up in a large local minimia, which the optimiser are not able help us out of. As the other networks are segmented with a better DS, this seems like a feasible explanation.

The STN is by far the hardest for the CNN to label. In addition to it being small, which means that under or overlabelling will have a stronger impact on the DS, than for the RN and SN. The border between the SN and STN is quite hard to identify, and varies from subject to subject. A manual intensity inspection of the volunteers uncovered that the pixels on the border between the SN and STN, had in some places a intensity difference as low as under 1%. Thus, the CNN will struggle with pinpointing the actual border between SN and STN, based solely on image intensity. Furthermore, for a doctor to decide where the border between SN and STN is based on such small differences, leads to uncertainties in the training data. The CNNs are then forced to rely on other features such as morphology and geography of the local brain structures to obtain an accurate segmentation. An overlabelling of the SN and in turn an underlabelling of the STN, or vica-versa, will yield large effects in the resulting DS. The same arguments apply for the Siemens datasets as well.

5.1.2 RN, SN and STN Image Segmentation on Siemens 3T Data

As seen in Table 4.3, the DS of the RN fluctuates vastly compared to the segmentations of the RN performed on the other datasets. An explanation may be due to fact that the co-registration of images has not been completely successful, yielding varying quality of the masks. As the mask were first transformed, then thresholded, one can ask questions about whether too many or too few voxel have been segmented. As seen in Figure 5.1.1, we see in the 7T case that there is a gap between the RN and the SN, but not in the 3T case. This indicates that the 3T segmentations are too coarse to be used as a ground truth when training a CNN. Furthermore, one can also see that the "arms" pointing out from the main body of the SN, in the 7T image, is not seen in the 3T case. The fact that these small details have been removed by the co-registration may not be a problem, as the lower resolution makes is hard to separate the smaller details on the 3T scanner.

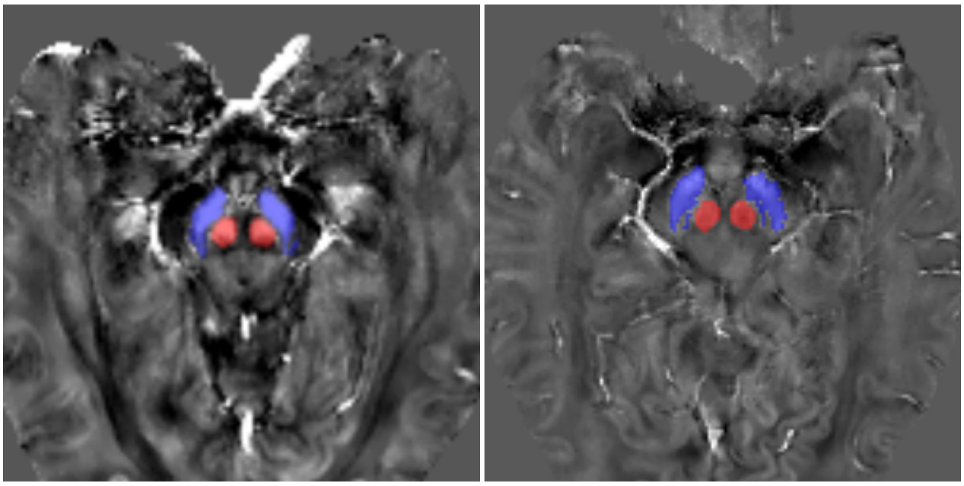


Figure 5.1.1: An example of segmentation of RN (Red) and SN (Blue) on the Siemens 3T data (left) and 7T data (right) for corresponding slices on the same volunteer.

On the other hand, in the case of the SN and STN both have small deviations in the DS, indicating that the segmentations may have been more consistent than for the RN. The effects of these contributions will become more apparent when the validity of the co-registration will be further elaborated on in Section 5.2.2 and 5.2.4.

In general, the CNNs trained on Siemens 3T data have obtained a similar DS to the CNNs trained on the GE 3T dataset. However, this has been performed with a substantially smaller training dataset. It is therefore natural to discuss the problem of overfitting. Even though the losses in training did not indicate overfitting, there is a chance that this is the case, due to the fact that we have a small highly homogeneous dataset which has achieved good results. Currently we do not know how well the CNNs trained on this group generalises to a more heterogeneous population. In the case of both Siemens datasets, it is recommended that older volunteers are recruited to broaden the dataset. This operation was not performed in this project, as it was deemed unethical to expose older volunteers to unnecessary risks of Covid-19 infections.

Another interesting observation is that the precision of the segmentations are the highest among all, while the sensitivity is lowest among all. As a high precision indicates very few FPs, this may be another indication that the mask transformation has not been completely successful and labelled to many voxels. If the ground truth has labelled to many voxels, the CNNs may label more area than the actual ROI, but it will still be counted as being correctly segmented. This can also be an indication that the DS is unnaturally high. Furthermore, the sensitivity is low, indicating a large numuber of FNs, meaning that the CNNs label too few voxels compared to the ground truth. As this number is noticeably smaller than for the GE 3T and 7T datasets, this may be another indicator that the segmenations are covering too much area.

5.1.3 RN, SN and STN Image Segmentation on Siemens 7T Data

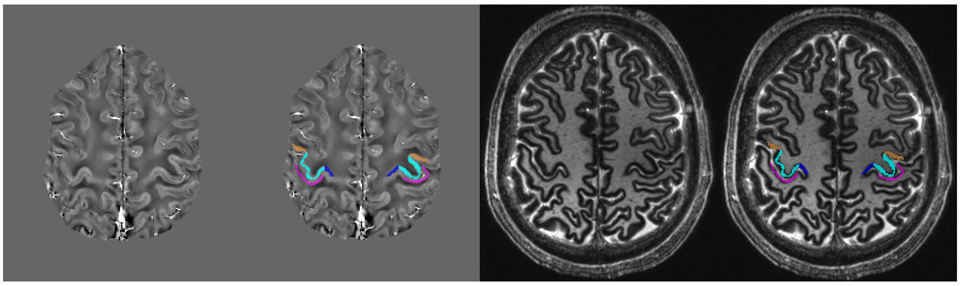
Table 4.4 shows that it is quite obvious that 7T segmentations are an improvement from both the 3T datasets with the highest average DS and lowest deviations, which is as expected. The most obvious reason for this improvement is that the voxel size has been reduced by approximately 25% compared to the 3T images. This has the potential to yield more accurate manual segmentations, as the different brain structures which are bordering each other are easier to separate, in turn improving training data. This will potentially create better segmentations. The most clear improvement can be seen in the case of the SN, in Figure 5.1.1. Earlier, we saw that in the 3T case, the "arms" of the SN are too small to be contoured due to too low resolution. In the 7T case, the voxel size is so small that we experience less fusion of signal between SN and non-SN tissue at the borders.

Additionally, neither the Siemens 3T or 7T data have been registered to a common template, sustaining all of the inter-subject variety in the dataset. A varied dataset is preferred in DL as it will improve training and reduce overfitting. We saw in Figure 4.1.1 in that particular case, that the network preferred a smoother shape of the SN, causing some overlabelling when the warps were not smooth, in turn yielding a poorer segmentation. However, this is not the case of the non-enhanced 7T dataset. When the dataset is varied, the networks tend to generalises more, in turn creating better networks and reducing overfitting. Furthermore, by keeping the sharper warps in e.g. the SN, this may work as a landmark, making it easier for the CNN to navigate and separate between brain structures.

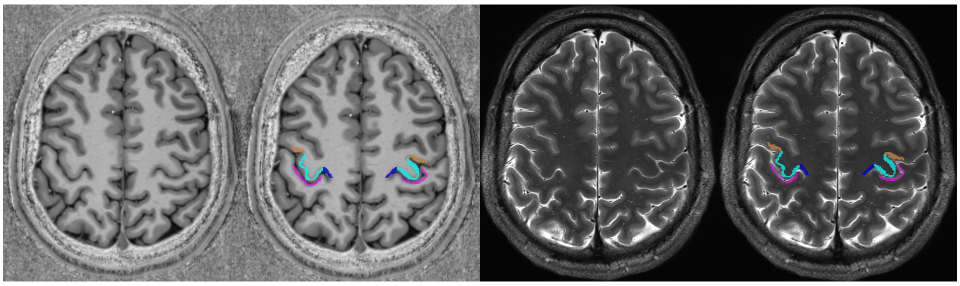
However, the improved segmentations may not only be due to the arguments mentioned in the paragraphs above. An explanation to why the 7T segmentations are better than the GE 3T segmentations is that the Siemens datasets have a very homogeneous group of young and healthy volunteers, which may make it easier for the CNNs to learn the patterns which causes accurate segmentations. In the case of the Siemens 3T dataset, the 7T obtains better results because we have more 7T images than 3T images, in turn improving the training of the models. Additionally, the 7T images have an optimised MGRE sequence and shimming. This will improve the quality of the obtained signal, and in turn reduce error propagation in the TGV-QSM algorithm. It is not just the 7T scanner itself that is the only contributing factor to the improved segmentations, but a part of it, combined with an optimised end-to-end reconstruction procedure.

5.1.4 PMC and PSSC Image Segmentation on Siemens 7T Data

As we saw in Section 4.1.4, the DS of the PSSC is somewhat lower than the different substructures of the PMC. A simple explanation for this, is that the PMC is divided into three smaller substructures, making it easier to segment each structure. Looking at Figure 3.4.1 and 3.8.2, this does not only simplify the structure, but also shrinks the range of the susceptibility values of surrounding brain structures, which in turn can make segmentation easier due less variations. However, by making the brain structures easier for the CNNs to learn, combined with a sparse dataset, the chance of overfitting is also increased. On the other hand, the average DS for the structures are not noticeably high, which is not an indication of overfitting. The reason for this mediocre DS is twofold. Firstly, the structures are quite complex. Additionally, the PMC and PSSC are not particularly magnetic, making them low on contrast in QSM images, seen in Figure 3.8.2. This will in turn make it harder for the CNN to train on and segment the structures compared to the RN, SN and STN, which are richer in contrast. However, obtained DS are for the substructures of PMC as good, and on the limit to being better than the intra-rater variability discussed in section 5.1.1. The PSSC, on the other hand, is in the lower layer. At least in the case of the PMC, the CNNs trained on this cohort have the potential to be accurate enough to be used as a tool for the radiologist. However, the DS can be improved by training the CNNs on other MRI sequences which yields better contrast w.r.t. the PMC and PSSC. Figure 5.1.2 shows the PMC and the PSSC on three different MRI sequences; QSM, T_1 and T_2-weighted images. A visual inspection shows that T_1-weighted image c and d show a clear improvement in contrast w.r.t. the PMC and PSSC in QSM imaging. In the case of the T_2-weighted image, there is a decrease in contrast. If the contrast is increased, making it easier to separate between ROIs and non-ROIs, this can improve the accuracy of the segmentations. Not to mention, another advantage of not using QSM images, as one can see from the comparison, the QSM reconstruction algorithm performs a removal of voxels at the border between the brain and the skull, where the magnetic field is ambiguous due to large differences in susceptibility values. This may lead to loss of spatial information, which makes it harder for the CNNs to learn the pattern of the ROIs.



(a) QSM **(b)** QSM with labelling **(c)** T_1 **(d)** T_1 with labelling



(e) T_1 **(f)** T_1 with labelling **(g)** T_2 **(h)** T_2 with labelling

Figure 5.1.2: Figures showing difference in contrast of the PMC and PSSC for in 7T images. Here QSM, two T_1-weighted images with different contrast and a T_2-weighted image are shown.

5.2 Extracting Susceptibility Values

This section will discuss the results presented in section 4.2 and will follow the same structure. An additional section is appended to discuss the feasibility of using RN and SN as bio-markers in PD.

5.2.1 Susceptibility values: RN, SN and STN GE 3T

As we saw in Figure 4.2.1, the general pattern is that for some validation splits the extreme values of the susceptibility are either under or overestimated. Table 4.2 and Equation 2.6.6 indicate that the amount of FNs and FPs is of the same size, which can influence the predicted susceptibility values. In general, for the Siemens datasets as well, the cases where the highest susceptibility values have not been included, is either because some central parts of the ROI is not segmented due to FNs, or because area outside the ROI is segmented due to FPs. As the sensitivity and precision decreases, we see that the similarity between the predicted and ground truth susceptibility values decreases.

For the RN and STN, as seen in Figure 4.2.2, the CNNs tends to both over- and understimate the susceptibility values. However, the predicted susceptibility values of the STN at higher concentrations yield very accurate values. This might be due to either that the error appears smaller as the concentration is relatively high, or due to the fact that segmentations are more accurate for STN with increased susceptibility concentrations, due to increased contrast. We see a small tendency to this for the three highest values for the RN as well. For the SN, the CNN continuously underestimates the susceptibility value. This is reflected in the relatively low precision of the segementation. This means that we have quite a large amount of FP, indicating that we segment values outside the ROI, which has a lower susceptibility value. Thus, it is expected that the CNN would yield too low susceptibility values.

When the error in predicted susceptibility was plotted against the DS, the trend was that the error shrunk. Intuitively, this makes sense, as one would expect lower errors in the predicted susceptibility when the segmentations are more accurate. However, for STN 10% and 15%, this is not the case. For an increasing DS, the trend is that the error increases weakly. This may indicate that only the central parts of the STN is labelled. If the central parts of the STN has a susceptibility value representative to that of the entire ROI, a low DS might not contribute to an increased error. Furthermore, the SN and STN have quite similar susceptibility values, meaning that an overlabelling into SN, does not cause an increase in error. The RN and SN do not have this luxury, as an overlabelling of the RN will mean labelling tissue with lower susceptibility values. As the STN is much smaller than the SN, an overlabelling of the SN will most likely not label the STN, but other tissue with a lower susceptibility value.

5.2.2 Susceptibility values: RN, SN and STN Siemens 3T

It is clearly shown in Figure 4.2.3 that there is a clear offset between the predicted and ground truth susceptibility values, while the density, for the RN and STN is similar. Looking at Figure 4.2.2, the Pearson Correlation coefficient is close to 1.0 for all three ROIs, but the MAPE is high, reaching almost 45 % in the case of the SN. These are very high values compared to the results obtained by the two other datasets. There may be a number of reasons for this error, but the most likely cause is the poor quality of the manual segmentation, which was discussed in section 5.1.2. The author, which has no medical background, performed a visual quality control of all the masks, and concluded that the masks were adequate. However, in hindsight this control does not seem to have been very successful. This becomes clear when looking at Figure 5.2.1, where the

SD divided by the average susceptibility value of the ROIs have been plotted for the Siemens datasets. The SD of the average susceptibility of the manually extracted ROIs are very high compared to the corresponding ROIs from the 7T dataset. This is also true for the GE 3T dataset, which has not been plotted to keep the figure simple. The high volatility presents two problems. Firstly, it indicates that area outside the ROIs are segmented by the masks, as they will contribute to much lower susceptibility values, thus increasing SD. This will also destroy the concept of the "ground truth", meaning that the metrics used for the CNNs are useless. Furthermore, the very high SD will mean that the segmentations performed by the CNN must be more or less perfect to obtain susceptibility values which are similar to the "ground truth". In other words the CNNs trained on the 3T should not be used in further research, until they are trained on manually segmented 3T data. Preferably with a less homogeneous and increased dataset.

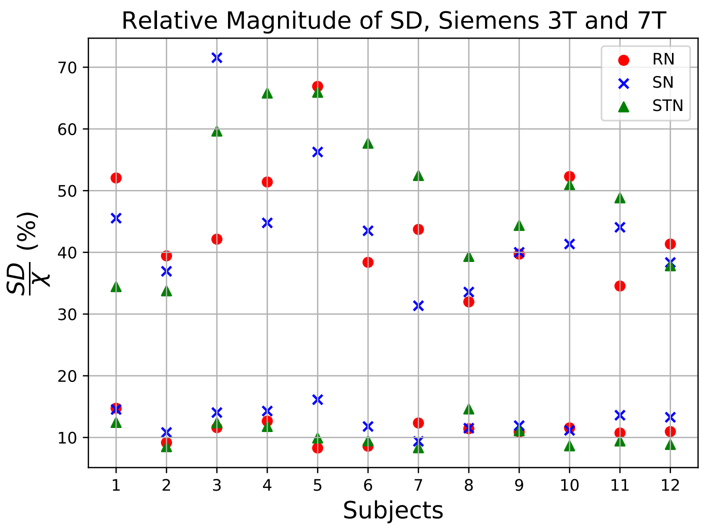
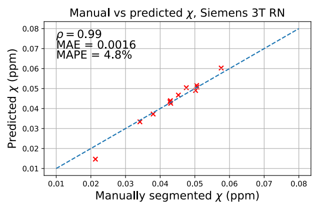
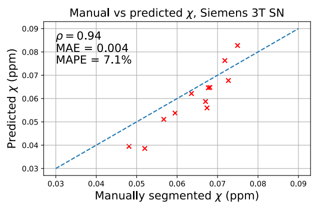


Figure 5.2.1: Plot showing the SD divided by the average susceptibility value for RN, SN and STN for Siemens 3T and 7T. The lower group are from the 7T data and the upper group is from the 3T data.

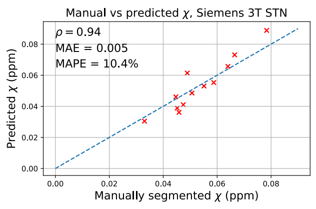
However, if we look away from this major flaw, we can discuss the performance of the CNNs. Firstly, all the susceptibility values extracted by the CNNs are within the SD of the of the average susceptibility of the manually segmented values. This indicates that even with a sparse dataset, there is potential to create quite good segmentations on a 3T dataset that has not been post-processed like the GE 3T has. This is also reflected in the DS. Furthermore, the Pearson correlation coefficients are all in the mid to high 0.90s, indicating a very strong linear correlation. By adjusting for the offset, as done in Figure 5.2.2, we see that the MAPE is drastically reduced. With the RN being reduced from 34.4% to 4.8%, SN from 43.3% to 7.0% and STN from 26.9% to 10.4%. With this offset correction we see that the MAPE for the predicted susceptibility values for the Siemens 3T RN is quite similar to the predicted values for the Siemens 7T. However, the error is a bit larger in the case of the SN and STN. We see that the offset reduction of the STN is not as successful as for the RN and SN. This is due to fact that the MAPE was initially not as large as for the other ROIs. Furthermore, the correlation is smaller than for the RN, meaning that a linear offset correction will not have as big impact as for the RN.



(a)



(b)



(c)

Figure 5.2.2: Susceptibility values extracted using the ground truth plotted against the offset corrected predicted susceptibility values for RN, SN and STN on the Siemens 3T dataset. The blue, dashed, line is the diagonal. Pearson Correlation coefficient, MAE and MAPE is presented.

5.2.3 Susceptibility values: RN, SN and STN 7T

Even though the DS and sensitivity are the highest for the CNNs trained on the 7T dataset, there are still some deviations between the predicted and ground truth susceptibility values, as seen in Figure 4.2.2. Especially in the case of the SN. This indicates that the CNN has not successfully segmented the smaller parts of the SN, which has a lower susceptibility value than the rest of the SN. This might be due to the small "arms" which have been included in the segmentation of the SN, and can be seen in Figure 3.4.1 and 5.1.1. They are hard to segment due to their size, but also due to the fact that their susceptibility value is similar to that of the surrounding tissue, in turn being low on contrast. As discussed earlier, it is here the 3D CNNs could have excelled by exploiting increased spatial information. By not segmenting the "arms", this will lead to an increased average susceptibility value, as their average susceptibility value is smaller than the main body of the SN. This argument is supported by the fact that the RN and

STN has similar DS values, but does not over estimate the susceptibility values as much. Another interesting result, which we saw tendencies of in the 3T datasets, is that the CNN trained on the STN data yields very accurate and precise susceptibility values, despite having the lowest accuracy metrics. As mentioned in section 5.2.1, this may be connected to the fact that an overlabelling of the STN into the SN does not change the average susceptibility value as the SN has comparable values. Furthermore, as one can see in Figure 5.2.1,the average relative SD for the STN is the lowest among the three ROIs. As the volatility of the susceptibility values within the ROI is not that low, FNs are not contributing that much to increased errors in predicted susceptibility values.

5.2.4 Comparing Susceptibility Values Obtained: Siemens 3T and 7T

It is expected that intra-subject imaging across field strengths would yield only small deviation in the measured susceptibility values in the ROIs (94). However, as seen in Figure 4.2.5, the plots do not indicate this. If it had only been an offset, it could be explained by a unsuccessful co-registration of the CSF in the Siemens 3T data, but the low correlation coefficient indicate that the reproducibility across scanners is almost non existing. Again, this is another indicator that the co-registration of the masks between 7T and 3T images were not successful. As the deviations between 3T and 7T are so large, likely due to poor segmentations of the 3T dataset, the susceptibility comparison between Siemens 3T and 7T dataset is not relevant anymore, only the obtained accuracy metrics. The foundation that the 3T dataset is built on is simply not good enough to conclude whether or not the extraction of susceptibility values is better on the 3T or 7T scanner. However, as most precision metrics for the CNNs trained on the 7T data are higher, this will most likely lead to more accurate prediction of the susceptibility values inside the ROIs. One can not conclude, but the results indicate that both the manual segmentations and CNNs trained on the 7T data are more accurate than that of the Siemens 3T. Additionally, I would like to emphasize that the reason why a similar comparison between the GE 3T and the 7T dataset was not performed, is due to the different processing steps and difference in volunteers of the two datasets.

5.2.5 Using Susceptibility Values as Bio-Marker for Neurodegenerative Diseases

Elevated susceptibility values, especially in the SN and RN are associated with PD (95)(96). We are investigating whether or not SN and RN can be used as a bio-markers for PD for only a single MRI scan. Research indicate that the SN experiences the largest relative increase in susceptibility values, as much as a doubling in the susceptibility values, meanwhile, for RN the increase is approximately as much as 50% (97). However, the uncertainty is comparable to the average increase in susceptibility values due to the large variety in susceptibility between individuals, as seen in Figure 4.2.2. This is also apparent the GE 3T dataset, which has a large variety of ages, but also in the Siemens datasets, which are constituent of volunteers mainly in their 20s. The inter-subject deviations are so large, that for an average person, an increase in susceptibility values due to PD would be interpreted as being within the interval of what is normal. In practice, this means that using the RN or SN as bio-markers for diagnosis of PD do not seem feasible, as elevated susceptibility values for one patient may be what is normal for another patient. However, for patients with naturally high susceptibility values, elevated values originating from PD could be detected and interpreted as a sign of PD, as these values would lie outside of interval of what is normal.

5.3 Further Work

Even though this thesis presents promising results for segmentation and susceptibility prediction on 3T and 7T QSM images, there are still a number of components which can be improved to obtain better results. One of the most obvious drawbacks of this project is that cohort that has been imaged has a relative low average age, meanwhile diseases such as PD and ALS tend to affect the older population. When the older population are vaccinated and the current Covid-19 restrictions are lifted, I recommend starting the imaging of an older cohort. This will reduce the chance of overfitting and create CNNs which have the ability to generalise to the entire population, not just a heterogeneous group of volunteers and actually train on a population which is at risk. A bi-product of this thesis has been that we have started to share QSM data with a group in Sweden. They are currently working on imaging cohorts in a wide variety of ages. If they are willing to continue shearing their data with us, this will improve the dataset and the foundation of this project substantially. As the information obtained about PD patients is purely based on research performed by others, it would be an improvement to the dataset if we could scan patients suffering from PD. Firstly, we could see for ourselves if there is a noticeable difference susceptibility values of the RN and STN between healthy and PD patients. Furthermore, it would also teach the CNNs how to generalise towards PD patietns as well. The last argument explains why ALS patients should be included in the dataset as well. Additionally, to properly perform comparison between 3T and 7T QSM segmentation, it is advised that the 3T images are segmented manually, and not just co-registered from 7T to 3T space, as this was not successful. Also in the 3T case, it is advised that the dataset is substantially increased. Furthermore, to improve the quality of the segmentations, more than one radiologist should perform the segmentations of the ROIs, at least two times each. This will not just improve segmentation, but also yield a wider statistical foundation when one is comparing weather or not the 7T segmentation are better than the 3T segmentations.

As the machines on the HUNT Cloud have been set up and labelled imaging data is already available, new master students now have the ability to initiate training of neural networks at once. This means that they have the time to train 3D CNNs. The 3D approach has the potential to improve segmentation quality due to increased spatial information, and in turn improve susceptibility prediction. In addition, DL is still in its youth and is developing rapidly, meaning that segmentation methods using CNNs are improving. CNNs such as Capsule Networks have shown very promising results within the field of medical image segmentation, and might yield better results than the U-Net applied in this project (98).

An interesting extension of this project, if enough data is available, that can improve the quality of QSM data, is to train a CNN to perform the reconstruction process. This will exchange the dipole inversion step that is prone to error propagation with a CNN. Recent developments within the field has shown potential, such as the SHARQnet (99) and the DeepQSM (100), which both exploits the U-Net for the dipole inversion. As a U-Net already has been implemented, only minor modification would be needed before training could commence.

Additionally, an improvement to the segmentation pipeline would be to train a CNN to automatically segment and extract susceptibility values from the CSF. As the pipeline is already built, it is only a manner of making a few small changes and train the CNN. By automatically recognising the CSF, the segmentation pipeline can yield absolute susceptibility values and not relative values, without the need of a doctor to make segmenations.

In the case of improving segmentations of PMC and PSSC, as they are not particularly rich in contrast in QSM, I recommend to move the training from QSM images to T_1-weighed images. As the ROIs would have stronger contrast, the training and segmentations could be improved.

Chapter 6

Conclusions

The results showed that the CNNs trained on the 7T dataset obtained the highest average DS, in addition to having the lowest deviations from the average value. For all three datasets, automatic segmentation of the RN and SN yielded better results than the intra-rater variability of professionals. In the case of the 7T data, this was true for STN, PMC and PSSC as well. The segmentation accuracy of PMC and PSSC showed potential to be used in further ALS research. However, before the CNNs trained on 7T data are applied, it is recommended that the neural networks are trained on a more varied dataset, as the generalisation of the networks have not been tested. Further segmentations of the PMC and PSSC should be performed on T_1-weighted images, which are richer in contrast w.r.t. these brain structures. As the manual segmentations of the 3T QSM images obtained at St.Olavs were lacking, the CNNs should be retrained on higher quality manually labelled data.

CNNs trained on the GE 3T dataset and CNNs trained on the 7T dataset obtained the most accurate susceptibility values. The high accuracy of the GE 3T dataset is likely due the post-processing steps, where non-linear image registration was applied. Meanwhile, the Norwegian 3T dataset, expressed low accuracy. The inadequate results seem to originate from the poor quality of the manual segmentations. Thus, a conclusion could not be made, but the results give a small indication that data obtained on the Siemens 7T scanner is better than the Siemens 3T dataset. In addition to the 7T CNNs obtaining better evaluation metrics, this is also based on the argument that CNNs trained on 14 7T QSM images, obtained nearly as good results as CNNs trained on a regularised dataset of twice the size. The CNNs trained on the GE 3T dataset showed very good results, but they are not directly comparable to the other datasets due to the post-processing steps. However it proved that regularisation of data is a powerful tool which can be used to extract highly accurate susceptibility values.

Further analysis showed that the susceptibility values in the SN and RN, have a high inter-subject variance in the case of all the datasets. Due to large variations in susceptibility value, an increased value caused by PD, would for the majority of patients not be interpreted as an abnormality. It is therefore advised that the patients undergo two QSM scans, at different times, to identify the development in the susceptibility values of the RN and SN.

Lastly, the intra-subject susceptibility comparison on the Siemens 3T and 7T scanner indicated that the scanners produce vastly different susceptibility values. However, it is believed that the error originates not purely from the inter-scanner and intra-subject differences, but mainly due to poor the segmentation of the 3T dataset, yielding incorrect susceptibility values.

Bibliography

[1] P. Foundation, “Parkinson Disease Statistics,” 2021. [Online]. Available: https://www.parkinson.org/Understanding-Parkinsons/Statistics

[2] K. C. Arthur, A. Calvo, T. R. Price, J. T. Geiger, A. Chiò, and B. J. Traynor, “Projected increase in amyotrophic lateral sclerosis from 2015 to 2040,” *Nature Communications*, vol. 7, no. 1, 11 2016.

[3] B. Heim, F. Krismer, R. De Marzi, and K. Seppi, “Magnetic resonance imaging for the diagnosis of Parkinson’s disease,” *Journal of Neural Transmission*, vol. 124, no. 8, 8 2017.

[4] M. R. Turner and M. Modo, “Advances in the application of MRI to amyotrophic lateral sclerosis,” *Expert Opinion on Medical Diagnostics*, vol. 4, no. 6, 11 2010.

[5] I. H. C. H. M. Philippens, J. A. Wubben, S. K. Franke, S. Hofman, and J. A. M. Langermans, “Involvement of the Red Nucleus in the Compensation of Parkinsonism may Explain why Primates can develop Stable Parkinson’s Disease,” *Scientific Reports*, vol. 9, no. 1, 12 2019.

[6] J. D. S. W. C. Ellis, A Simmons, “Distinct hyperintense MRI signal changes in the corticospinal tracts of a patient with motor neuron disease,” *Amyotrophic Lateral Sclerosis and Other Motor Neuron Disorders*, vol. 1, no. 1, 1 2000.

[7] Z. Dai, Y. Chen, G. Yan, Gang xiao, Z. Shen, and R. Wu, “Progress of magnetic resonance imaging in amyotrophic lateral sclerosis,” *Radiology of Infectious Diseases*, vol. 6, no. 1, 3 2019.

[8] A. Eisen and M. Weber, “The motor cortex and amyotrophic lateral sclerosis,” *Muscle & Nerve*, vol. 24, no. 4, 4 2001.

[9] Y. Zhang, M. Brady, and S. Smith, “Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm,” *IEEE Transactions on Medical Imaging*, vol. 20, no. 1, 2001.

[10] A. M. Dale, B. Fischl, and M. I. Sereno, “Cortical Surface-Based Analysis,” *NeuroImage*, vol. 9, no. 2, 2 1999.

[11] N. O’ Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, “Deep learning vs. traditional computer vision,” *arXiv*, no. April, 2019.

[12] “Neurodegenerative Diseases - National Institute of Envoironmental Health Sciences,” 2021. [Online]. Available: https://www.niehs.nih.gov/research/supported/health/neurodegenerative/index.cfm

[13] “Prevalence of Parkinson’s Disease in Norway, Helse Stavanger,” 2016. [Online]. Available: https://helse-stavanger.no/fag-og-forskning/kompetansetjenester/nasjonal-kompetansetjeneste-for-bevegelsesforstyrrelser/bevegelsesforstyrrelser/parkinsons-sykdom

[14] K. Wakabayashi, K. Tanji, F. Mori, and H. Takahashi, “The Lewy body in Parkinson’s disease: Molecules implicated in the formation and degradation of α-synuclein aggregates,” *Neuropathology*, vol. 27, no. 5, 10 2007.

[15] HelseNorge, “ALS prevalence Norway,” 2021. [Online]. Available: https://www.helsenorge.no/sykdom/hjerne-og-nerver/als/

[16] B. Drayer, P. Burger, R. Darwin, S. Riederer, R. Herfkens, and G. Johnson, “MRI of brain iron,” *American Journal of Roentgenology*, vol. 147, no. 1, pp. 103–110, 7 1986. [Online]. Available: http://www.ajronline.org/doi/10.2214/ajr.147.1.103

[17] M. M. Lewis, G. Du, M. Kidacki, N. Patel, M. L. Shaffer, R. B. Mailman, and X. Huang, “Higher iron in the red nucleus marks Parkinson’s dyskinesia,” *Neurobiology of Aging*, vol. 34, no. 5, 5 2013.

[18] R. Balestrino and A. Schapira, “Parkinson disease,” *European Journal of Neurology*, vol. 27, no. 1, 1 2020.

[19] C. Lambert, L. Zrinzo, Z. Nagy, A. Lutti, M. Hariz, T. Foltynie, B. Draganski, J. Ashburner, and R. Frackowiak, “Confirmation of functional zones within the human subthalamic nucleus: Patterns of connectivity and sub-parcellation using diffusion weighted imaging,” *NeuroImage*, vol. 60, no. 1, 3 2012.

[20] M. Aminoff and R. Daroff, *Encyclopedia of the Neurological Sciences*, 2nd ed. Elsevier Ltd¨, 2014. [Online]. Available: https://www.sciencedirect.com/referencework/9780123851581/encyclopedia-of-the-neurological-sciences#book-description

[21] W. Webb, *Neurology for the Speech-Language Pathologist*. Elsevier Ltd, 2017. [Online]. Available: https://www.sciencedirect.com/book/9780323100274/neurology-for-the-speech-language-pathologist

[22] R. W. Brown, Y.-C. N. Cheng, E. M. Haacke, M. R. Thompson, and R. Venkatesan, Eds., *Magnetic Resonance Imaging*. Chichester, UK: John Wiley & Sons Ltd, 4 2014.

[23] E. M. Haacke, Y. Xu, Y. C. N. Cheng, and J. R. Reichenbach, “Susceptibility weighted imaging (SWI),” *Magnetic Resonance in Medicine*, vol. 52, no. 3, pp. 612–618, 2004.

[24] Y. Wang and T. Liu, “Quantitative susceptibility mapping (QSM): Decoding <scp>MRI</scp> data for a tissue magnetic biomarker,” *Magnetic Resonance in Medicine*, vol. 73, no. 1, pp. 82–101, 1 2015.

[25] “Chapter 7 Types of magnetic materials,” in *Current Methods in Inorganic Chemistry*. Elsevier, 1 1999, vol. 1, no. C, pp. 345–370.

[26] R. K. Kotnala and J. Shah, “Chapter 4 - Ferrite Materials: Nano to Spintronics Regime,” ser. Handbook of Magnetic Materials, K. H. J. Buschow, Ed. Elsevier, 2015, vol. 23, pp. 291–379. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444635280000048

[27] J. J. Winzerling and D. Q. Pham, “Ferritin,” in *Comprehensive Molecular Insect Science*. Elsevier, 1 2005, vol. 4-6, pp. 341–356.

[28] T. Liu, C. Wisnieff, M. Lou, W. Chen, P. Spincemaille, and Y. Wang, “Nonlinear formulation of the magnetic field to source relationship for robust quantitative susceptibility mapping,” *Magnetic Resonance in Medicine*, vol. 69, no. 2, pp. 467–476, 2 2013. [Online]. Available: http://doi.wiley.com/10.1002/mrm.24272

[29] K. Eckstein, B. Dymerska, B. Bachrata, W. Bogner, K. Poljanc, S. Trattnig, and S. D. Robinson, "Computationally Efficient Combination of Multi-channel Phase Data From Multi-echo Acquisitions (ASPIRE)," *Magn.~Reson.~Med.*, vol. 79, no. 6, pp. 2996–3006, 6 2018.

[30] S. M. Smith, "Fast robust automated brain extraction," *Human Brain Mapping*, vol. 17, no. 3, 11 2002.

[31] A. Deistung, F. Schweser, and J. R. Reichenbach, "Overview of quantitative susceptibility mapping," *NMR in Biomedicine*, vol. 30, no. 4, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/nbm.3569

[32] A. Rauscher, M. Barth, K.-H. Herrmann, S. Witoszynskyj, A. Deistung, and J. R. Reichenbach, "Improved elimination of phase effects from background field inhomogeneities for susceptibility weighted imaging at high magnetic field strengths," *Magnetic Resonance Imaging*, vol. 26, no. 8, 10 2008.

[33] J. H. Duyn, P. van Gelderen, T.-Q. Li, J. A. de Zwart, A. P. Koretsky, and M. Fukunaga, "High-field MRI of brain cortical substructure based on signal phase," *Proceedings of the National Academy of Sciences*, vol. 104, no. 28, 7 2007.

[34] A. J. Walsh, A. Eissa, G. Blevins, and A. H. Wilman, "Susceptibility phase imaging with improved image contrast using moving window phase gradient fitting and minimal filtering," *Journal of Magnetic Resonance Imaging*, vol. 36, no. 6, 12 2012.

[35] F. Schweser, A. Deistung, B. W. Lehr, and J. R. Reichenbach, "Quantitative imaging of intrinsic magnetic tissue properties using MRI signal phase: an approach to in vivo brain iron metabolism?" *NeuroImage*, vol. 54, no. 4, pp. 2789–2807, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.neuroimage.2010.10.070

[36] F. Schweser, A. Deistung, and J. R. Reichenbach, "Foundations of MRI phase imaging and processing for Quantitative Susceptibility Mapping (QSM)," *Zeitschrift fur Medizinische Physik*, vol. 26, no. 1, pp. 6–34, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.zemedi.2015.10.002

[37] H. Sun and A. H. Wilman, "Background field removal using spherical mean value filtering and Tikhonov regularization," *Magnetic Resonance in Medicine*, vol. 71, no. 3, pp. 1151–1157, 2014.

[38] L. Li and J. S. Leigh, "High-precision mapping of the magnetic field utilizing the harmonic function mean value property," *Journal of Magnetic Resonance*, vol. 148, no. 2, pp. 442–448, 2001.

[39] Feynmann and L. Sands, "The Feynmann Lectures on Physics, Volume II - Electrodynamics," 1963. [Online]. Available: https://www.feynmanlectures.caltech.edu/II_toc.html

[40] B. Kressler, L. de Rochefort, Tian Liu, P. Spincemaille, Quan Jiang, and Yi Wang, "Nonlinear Regularization for Per Voxel Estimation of Magnetic Susceptibility Distributions From MRI Field Maps," *IEEE Transactions on Medical Imaging*, vol. 29, no. 2, 2 2010.

[41] X. Zhang, E. Y. Lam, E. X. Wu, and K. K. Y. Wong, "Application of Tikhonov Regularization to Super-Resolution Reconstruction of Brain MRI Images," in *Medical Imaging and Informatics*. Berlin, Heidelberg: Springer Berlin Heidelberg.

[42] P. Hansen, "The L-curve and its use in the numerical treatment of inverse problems," Department of Mathematical Modelling, Technical University of Denmark,, Lyngby, Tech. Rep., 1999. [Online]. Available: https://www.sintef.no/globalassets/project/evitameeting/2005/lcurve.pdf

[43] K. Shmueli, J. A. De Zwart, P. Van Gelderen, T. Q. Li, S. J. Dodd, and J. H. Duyn, "Magnetic susceptibility mapping of brain tissue in vivo using MRI phase data," *Magnetic Resonance in Medicine*, vol. 62, no. 6, pp. 1510–1522, 2009.

[44] T. Liu, J. Liu, L. De Rochefort, P. Spincemaille, I. Khalidov, J. R. Ledoux, and Y. Wang, "Morphology enabled dipole inversion (MEDI) from a single-angle acquisition: Comparison with COSMOS in human brain imaging," *Magnetic Resonance in Medicine*, vol. 66, no. 3, pp. 777–783, 2011.

[45] J. Jackson, *Classical Electrodynamics*, 3rd ed., 1998. [Online]. Available: https://www.wiley.com/en-gb/Classical+Electrodynamics%2C+3rd+Edition-p-9780471309321

[46] C. Vogel, *Computational Methods for Inverse Problems*, 1st ed. Frontiers in Applied Mathematics, 2002.

[47] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2006. [Online]. Available: https://link.springer.com/book/10.1007/978-0-387-40065-5

[48] M. Hestenes and E. Stiefel, “Methods of conjugate gradients for solving linear systems,” *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, p. 409, 1952.

[49] B. Bilgic, A. Pfefferbaum, T. Rohlfing, E. V. Sullivan, and E. Adalsteinsson, “MRI estimates of brain iron concentration in normal aging using quantitative susceptibility mapping,” *NeuroImage*, vol. 59, no. 3, pp. 2625–2635, 2012.

[50] F. Yanez, A. Fan, B. Bilgic, C. Milovic, E. Adalsteinsson, and P. Irarrazaval, “Quantitative Susceptibility Map Reconstruction via a Total Generalized Variation Regularization,” in *2013 International Workshop on Pattern Recognition in Neuroimaging*. IEEE, 6 2013.

[51] F. Knoll, K. Bredies, T. Pock, and R. Stollberger, “Second order total generalized variation (TGV) for MRI,” *Magnetic Resonance in Medicine*, vol. 65, no. 2, 2 2011.

[52] W. Li, B. Wu, and C. Liu, “Quantitative susceptibility mapping of human brain reflects spatial variation in tissue composition,” *NeuroImage*, vol. 55, no. 4, 4 2011.

[53] W. Li, A. V. Avram, B. Wu, X. Xiao, and C. Liu, “Integrated Laplacian-based phase unwrapping and background phase removal for quantitative susceptibility mapping,” *NMR in Biomedicine*, vol. 27, no. 2, 2 2014.

[54] J. Acosta-Cabronero, G. B. Williams, A. Cardenas-Blanco, R. J. Arnold, V. Lupson, and P. J. Nestor, “In vivo quantitative susceptibility mapping (QSM) in Alzheimer’s disease,” *PLoS ONE*, vol. 8, no. 11, 2013.

[55] M. De Groot, M. A. Ikram, S. Akoudad, G. P. Krestin, A. Hofman, A. Van Der Lugt, W. J. Niessen, and M. W. Vernooij, “Tract-specific white matter degeneration in aging: The Rotterdam Study,” *Alzheimer’s and Dementia*, vol. 11, no. 3, pp. 321–330, 2015.

[56] S. Straub, T. M. Schneider, J. Emmerich, M. T. Freitag, C. H. Ziener, H.-P. Schlemmer, M. E. Ladd, and F. B. Laun, “Suitable reference tissues for quantitative susceptibility mapping of the brain,” *Magnetic Resonance in Medicine*, vol. 78, no. 1, 7 2017.

[57] M. Jenkinson and S. Smith, “A global optimisation method for robust affine registration of brain images,” *Medical Image Analysis*, vol. 5, no. 2, 6 2001.

[58] M. Jenkinson, “Improved Optimization for the Robust and Accurate Linear Registration and Motion Correction of Brain Images,” *NeuroImage*, vol. 17, no. 2, 10 2002.

[59] J. P. Pluim, J. Antoine Maintz, and M. A. Viergever, “Interpolation Artefacts in Mutual Information-Based Image Registration,” *Computer Vision and Image Understanding*, vol. 77, no. 2, 2 2000.

[60] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[61] H. H. Tan and K. H. Lim, “Vanishing Gradient Mitigation with Deep Learning Neural Network Optimization,” *2019 7th International Conference on Smart Computing and Communications, ICSCC 2019*, pp. 2019–2022, 2019.

[62] A. F. M. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” *arXiv*, no. 1, pp. 2–8, 2018.

[63] S. Linnainmaa, “Taylor Expansion of the Accumulated Rounding Error Seppo Linnainmaa,” vol. 16, pp. 146–160, 1976. [Online]. Available: https://twin.sci-hub.tw/6687/1b259b2bbfd4e48014c776b4498d0b41/linnainmaa1976.pdf#view=FitH

[64] M. A. Nielsen, “Neural Networks and Deep Learning,” 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com

[65] Tensorflow/Keras, “The 2D Convolution Layer,” 2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

[66] G. Hinton, N. Srivastava, and K. Swersky, “Neural Networks for Machine Learning: Lecture 6a - Overview of mini-batch gradient descent,” 2021. [Online]. Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[67] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2015. [Online]. Available: https://arxiv.org/abs/1502.03167

[68] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, “Recent advances in convolutional neural networks,” *Pattern Recognition*, vol. 77, 5 2018.

[69] T. Dean, “Inferring Mesoscale Models of Neural Computation,” Tech. Rep., 2017.

[70] X. Ying, “An Overview of Overfitting and its Solutions,” *Journal of Physics: Conference Series*, vol. 1168, no. 2, 2019.

[71] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” Tech. Rep., 2014.

[72] S. Salman and X. Liu, “Overfitting mechanism and avoidance in deep neural networks,” *arXiv*, 2019.

[73] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” *CoRR*, vol. abs/1505.0, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[74] Tensorflow/Keras, “The Batch Normalization Layer,” 2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization

[75] ——, “The Dropout Layer,” 2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

[76] Tensorflow, “The Concatenation Layer,” 2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/concat

[77] Tensorflow/Keras, “The Max Pooling Layer,” 2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool3D

[78] ——, “Convolution 2D Transpose Layer,” 2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTranspose

[79] “NTNU | HUNT Cloud,” 2021. [Online]. Available: https://www.ntnu.edu/mh/huntcloud

[80] “Anaconda Software Distribution, Vers 2-2.4.0.” [Online]. Available: https://www.anaconda.com/products/individual

[81] 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. Mane, 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. Viegas, O. Vinyals,

P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," 3 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

[82] B. Garzón, R. Sitnikov, L. Bäckman, and G. Kalpouzos, "Automated segmentation of midbrain structures with high iron content," *NeuroImage*, vol. 170, pp. 199–209, 4 2018.

[83] H. Abdul-Rahman, M. Arevalillo-Herráez, M. Gdeisat, D. Burton, M. Lalor, F. Lilley, C. Moore, D. Sheltraw, and M. Qudeisat, "Robust three-dimensional best-path phase-unwrapping algorithm that avoids singularity loops." *Applied optics*, vol. 48, no. 23, pp. 4582–4596, 8 2009.

[84] I. The Mathworks, "MATLAB version 9.3.0.713579 (R2017b)," Natick, Massachusetts, 2017. [Online]. Available: https://www.mathworks.com/products/matlab.html

[85] Y. Wang, "Quantitative Susceptibility Mapping," 2020. [Online]. Available: http://pre.weill.cornell.edu/mri/pages/qsm.html

[86] M. Jenkinson, C. F. Beckmann, T. E. Behrens, M. W. Woolrich, and S. M. Smith, "FSL," *NeuroImage*, vol. 62, no. 2, 8 2012. [Online]. Available: https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FLIRT

[87] C. Rorden and M. Brett, "Stereotaxic Display of Brain Lesions," *Behavioural Neurology*, vol. 12, no. 4, 2000.

[88] "Duvernoy's Atlas of the Human Brain Stem and Cerebellum." p. e75, 5 2009. [Online]. Available: https://www.springer.com/gp/book/9783211739716

[89] P. A. Yushkevich, J. Piven, H. C. Hazlett, R. G. Smith, S. Ho, J. C. Gee, and G. Gerig, "User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability," *NeuroImage*, vol. 31, no. 3, 7 2006.

[90] D. N. Greve and B. Fischl, "Accurate and robust brain image alignment using boundary-based registration," *NeuroImage*, vol. 48, no. 1, 10 2009.

[91] M. Jenkinson, "FLIRT (FMRIB's Linear Image Registration Tool) ." [Online]. Available: https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FLIRT/FAQ

[92] Tensorflow/Keras, "The Binary Crossentropy Class," 2021.

[93] Y. Zhang, H. Wei, M. J. Cronin, N. He, F. Yan, and C. Liu, "Longitudinal atlas for normative human brain development and aging over the lifespan using quantitative susceptibility mapping," *NeuroImage*, vol. 171, pp. 176–189, 2018.

[94] P. Spincemaille, J. Anderson, G. Wu, B. Yang, M. Fung, K. Li, S. Li, I. Kovanlikaya, A. Gupta, D. Kelley, N. Benhamo, and Y. Wang, "Quantitative Susceptibility Mapping: MRI at 7T versus 3T," *Journal of Neuroimaging*, vol. 30, no. 1, 1 2020.

[95] Q. Chen, Y. Chen, Y. Zhang, F. Wang, H. Yu, C. Zhang, Z. Jiang, and W. Luo, "Iron deposition in Parkinson's disease by quantitative susceptibility mapping," *BMC Neuroscience*, vol. 20, no. 1, 12 2019.

[96] K. Ghassaban, N. He, S. K. Sethi, P. Huang, S. Chen, F. Yan, and E. M. Haacke, "Regional High Iron in the Substantia Nigra Differentiates Parkinson's Disease Patients From Healthy Controls," *Frontiers in Aging Neuroscience*, vol. 11, 5 2019.

[97] A. K. Lotfipour, S. Wharton, S. T. Schwarz, V. Gontu, A. Schäfer, A. M. Peters, R. W. Bowtell, D. P. Auer, P. A. Gowland, and N. P. Bajaj, "High resolution magnetic susceptibility mapping of the substantia nigra in Parkinson's disease," *Journal of Magnetic Resonance Imaging*, vol. 35, no. 1, 1 2012.

[98] A. Jiménez-Sánchez, S. Albarqouni, and D. Mateus, "Capsule Networks Against Medical Imaging Data Challenges," 2018.

[99] S. Bollmann, M. H. Kristensen, M. S. Larsen, M. V. Olsen, M. J. Pedersen, L. R. Østergaard, K. O'Brien, C. Langkammer, A. Fazlollahi, and M. Barth, "SHARQnet – Sophisticated harmonic artifact reduction in quantitative susceptibility mapping using a deep convolutional neural network," *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, 5 2019.

[100] S. Bollmann, K. G. B. Rasmussen, M. Kristensen, R. G. Blendal, L. R. Østergaard, M. Plocharski, K. O'Brien, C. Langkammer, A. Janke, and M. Barth, "DeepQSM - using deep learning to solve the dipole inversion for quantitative susceptibility mapping," *NeuroImage*, vol. 195, 7 2019.

Appendices

Appendix **A**

Appendix A

A.1 Additional Results

This appendix will present some additional plots that were not included in the result section.

Figure A.1 shows all the violin plots for the GE3T data.

Figure A.2 shows how the error in measured susceptibility changes as a function of DS for the GE3T dataset.

Figure A.3 shows the susceptibility values obtained from manual segmentation plotted against the suscpetibility values obtained by the Convolutional Neural Network (CNN).

Figure A.4 shows how the error in predicted susceptibility changes as a function of DS for the Siemens 3T dataset.

Figure A.5 shows how the error in predicted susceptibility values changes as a function of DS for the 7T dataset for RN, SN and STN.

Figure A.6 shows the violin plot for PSSC.

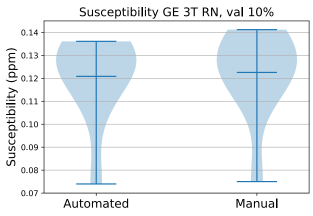
Figure A.7 shows the error in predicted susceptibility values as a function of DS.

Figure A.8 shows the violin plot for the PMC.

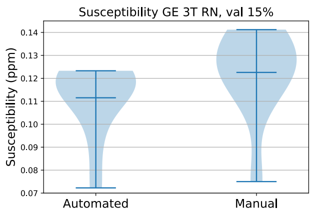
Figure A.9 shows how the error in predicted susceptibility values changes as a function of DS for the 7T dataset for the PSSC.

Figure A.10 shows how ground truth susceptibility values plotted against the predicted values for PSSC.

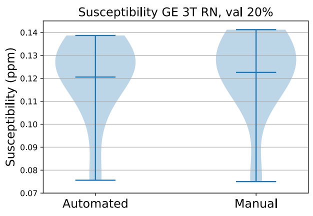
Figure A.11 shows how the error in predicted susceptibility values changes as a



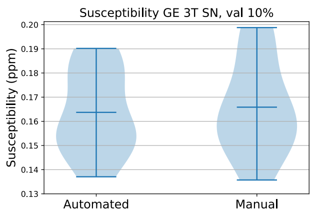
(a)



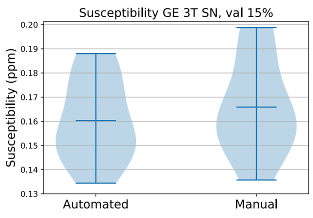
(b)



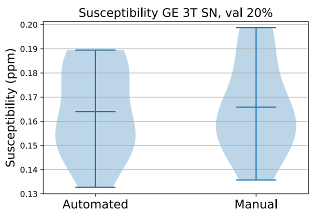
(c)



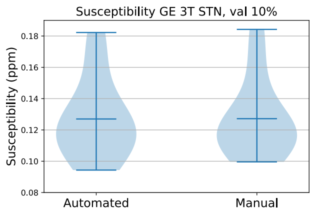
(d)



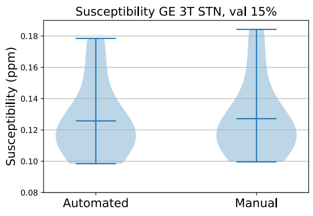
(e)



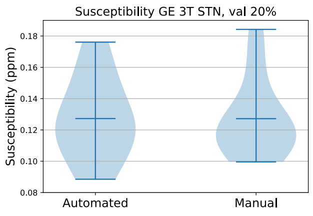
(f)



(g)

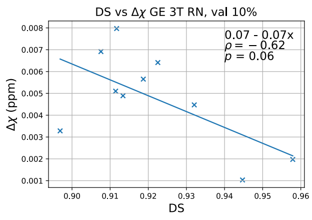


(h)

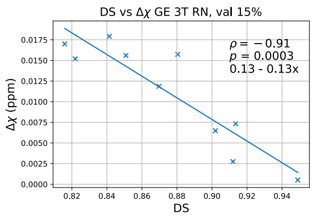


(i)

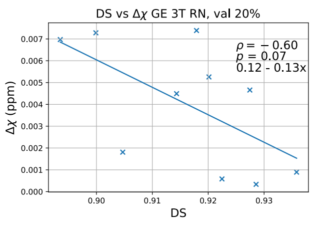
Figure A.1: Violin plot comparing the average susceptibility of automatically segmented tissue and manually segmented tissue for RN, SN and STN for the validation split 10%, 15% and 20%.



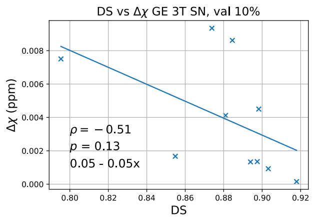
(a)



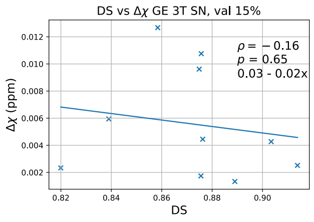
(b)



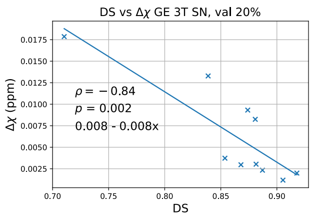
(c)



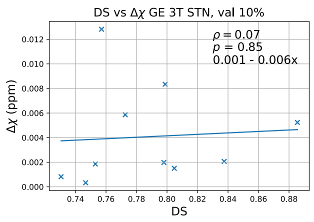
(d)



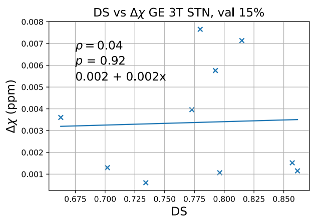
(e)



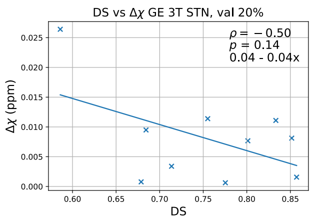
(f)



(g)

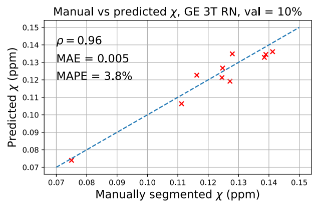


(h)

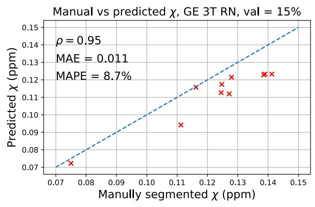


(i)

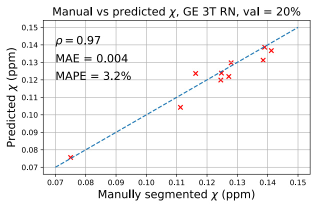
Figure A.2: Plots showing the error in predicted susceptibility values as as function of measured DS for RN, SN and STN for the GE 3T dataset. The plots have been fitted with a straight line approximated using a least squares error algorithm. Pearson Correlation coefficient, p-value and approximated functional expression is presented. From left to right: validation split: 10 %, 15 % and 20 %. From top to bottom: RN, SN and STN.



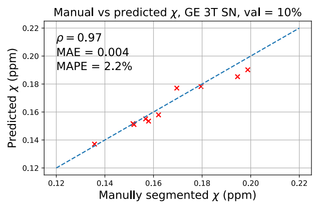
(a)



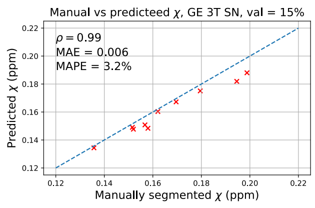
(b)



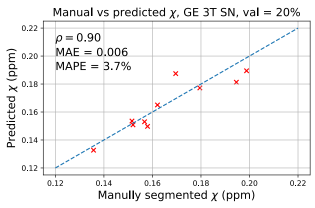
(c)



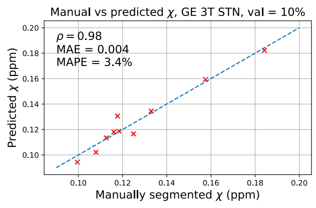
(d)



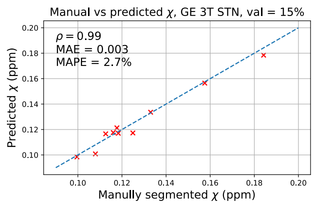
(e)



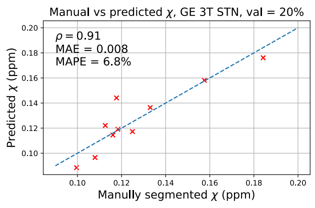
(f)



(g)

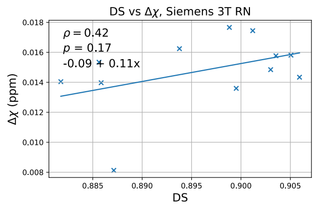


(h)

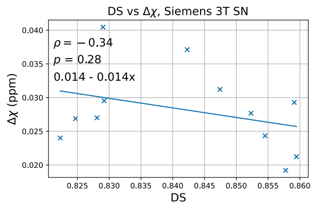


(i)

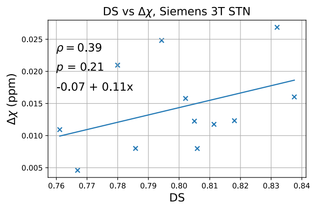
Figure A.3: Susceptibility values extracted using the ground truth manual segmentation plotted vs the predicted susceptibility values extracted from the segmentation performed by the neural network for RN, SN and STN for the GE 3T dataset. The blue, dashed, line is the diagonal.Pearson Correlation coefficient, MAE and MAPE is presented. From left to right: validation split 10 %, 15 % and 20 %. From top to bottom: RN, SN and STN.



(a)

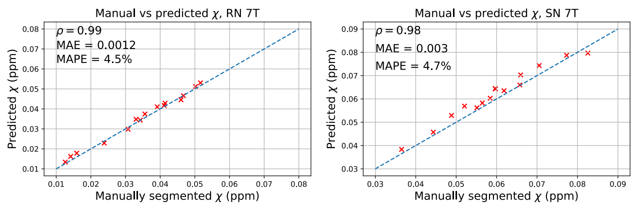


(b)



(c)

Figure A.4: Plots showing the error in predicted susceptiility value as a function of DS for RN, SN and STN for the Siemens 3T dataset. The plots have been fitted with a straight line approximated using a least squares error algorithm. Pearson Correlation coefficient, p-value and approximated functional expression is presented. Plotsfrom left to right: RN, SN and STN



(a) (b) (c)

Figure A.5: Susceptibility values extracted using the ground truth manual segmentation plotted vs the predicted susceptibility values extracted from the segmentation performed by the neural network for RN, SN and STN for the 7T dataset. The blue, dashed, line is the diagonal.Pearson Correlation coefficient, MAE and MAPE is presented. From left to right: RN, SN and STN.

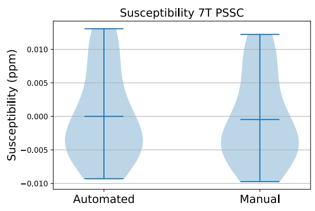
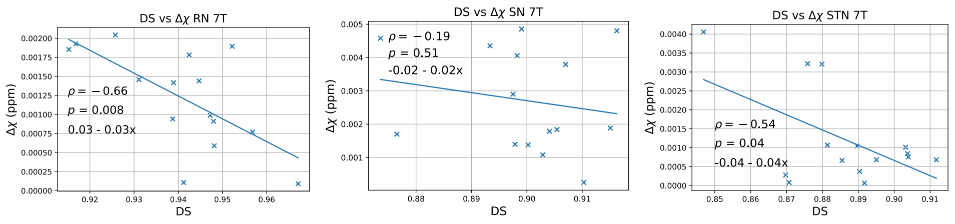


Figure A.6: Violin plot comparing the average susceptibility of automatically segmented tissue and manually segmented tissue for the PSSC for the 7T dataset.



(a) (b) (c)

Figure A.7: Plots showing the error in predicted susceptiility value as a function of DS for RN, SN and STN for the 7T dataset. The plots have been fitted with a straight line approximated using a least squares error algorithm. Pearson Correlation coefficient, p-value and approximated functional expression is presented. Plots from left to right: RN, SN and STN.

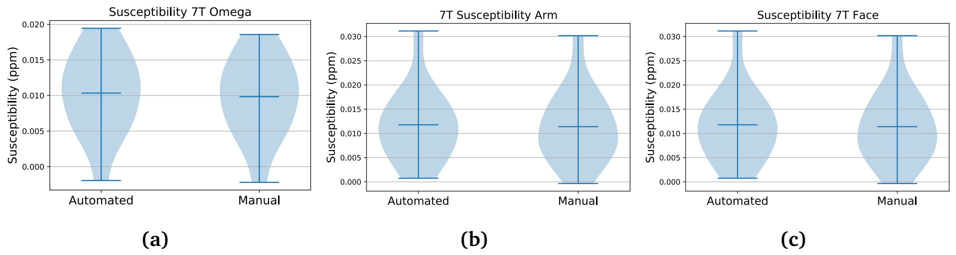


Figure A.8: Violin plots comparing the average susceptibility of automatically segmented tissue and manually segmented tissue for the substructures of the PMC: Omega, Arm and Face for the 7T dataset

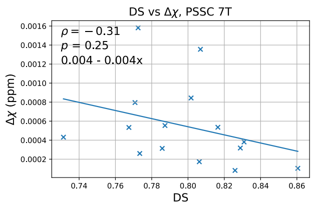


Figure A.9: Plot showing the error in predicted susceptibility value as a function of DS for the PSSC for the 7T dataset. The plot has been fitted with a straight line approximated using a least squares error algorithm. Pearson Correlation coefficient, p-value and approximated functional expression is presented.

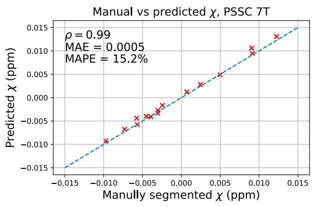


Figure A.10: Susceptibility values extracted using the ground truth manual segmentation plotted vs the predicted susceptibility values extracted from the segmentation performed by the neural network for the PSSC for the 7T dataset. The blue, dashed, line is the diagonal.Pearson Correlation coefficient, MAE and MAPE is presented.

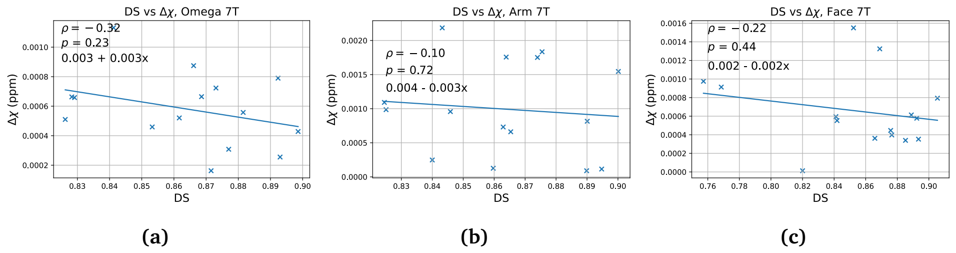


Figure A.11: Plots showing the error in predicted susceptiility value as a function of DS for the substructures of the PMC; Omega, Arm and Face for the 7T dataset. The plots have been fitted with a straight line approximated using a least squares error algorithm. Pearson Correlation coefficient, p-value and approximated functional expression is presented. Plotsfrom left to right: Omega, Arm and Face.

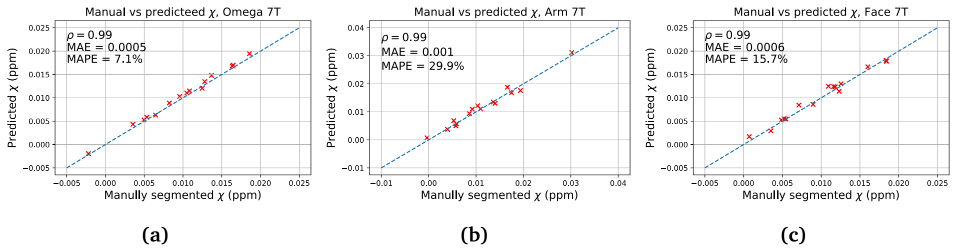


Figure A.12: Susceptibility values extracted using the ground truth manual segmentation plotted vs the predicted susceptibility values extracted from the segmentation performed by the neural network for Omega, Face and Arm for the 7T dataset. The blue, dashed, line is the diagonal. Pearson Correlation coefficient, MAE and MAPE is presented. From left to right: Omega, Arm and Face

function of DS for the 7T dataset for the PSSC.

Figure A.12 shows how ground truth susceptibility values plotted against the predicted values for PMC.

Appendix **B**

Appendix B

B.1 HUNT Cloud Documentation

This subsection covers the practical aspect of using HUNT Cloud. 1 - 9 covers the basic setup, 10 and 11 covers QSM reconstruction, 12 covers tensorflow and GPU implementation and 13 covers general tips when using HUNT Cloud.

1) There are 4 initial steps before you can start using HUNT Cloud. They involve contacting you lab leader so you can get permission for the HUNT Cloud group to create a user for you. They are very well documented on their homepage (https://docs.hdc.ntnu.no/getting-started/)

2) Working in your lab. Now that you are set up on the machines with Ubuntu 18.04, you need to perform some installations to exploit the full capacity of HUNT Cloud. Except for Python, which is already installed, the machines are empty.

3) How to move inside the lab:

   (a) ssh <username>@<your-lab-IP> -XY # -XY to activate Graphics User Interface

4) ssh to "home". This is a personal machine type, only available for you. You can Download packages and programs, but without root access. As we are encouraged to share our data and results this machine only have 100GB of storage. Therefore, we should perform our calculations on the Iaas1, GPU and blue machines.

(a) ssh home -XY

5) ssh to the machine "iaas1". If you need root access, you can ask for permission from your lab leader to access the iaas machine. Having root access, you can download all the programs and packages as you wish

   (a) ssh ubuntu@mrphys-iaas1 -XY

6) To exit a machine (will take you back to the machine you ssh'ed from):

   (a) exit

7) Access the GPU machine. Currently, the CPU has 16 GB of memory, making it possible to perform extensive computations. This machine type gives you the possibility to accelerate your work, in addition to the root access. In particular, the GPU drastically decreases DL training time and drastically speeds up some of the FSL applications

   (a) ssh ubuntu@mrphys-gpu1 -XY

8) Installing GUI. To access graphical tools, you need the software X2GO.

   (a) Brew install —cash quartz # (MAC)
   Brew install —cask x2goclient # (MAC)

   (b) Follow steps here: https://docs.hdc.ntnu.no/working-in-your-lab/technical-tools/x2go/#set-up-your-local-machine

9) Installing the necessary packages. As an example, this step will be described on the gpu1 machine, but the procedure is identical on iaas1 and home. Remember that you do not have root access on home machines, which means that you might need to do some alterations...:

   (a) Installing Rsync (https://linux.die.net/man/1/rsync). An effective way of transferring data between machines, both from your own machine to HUNT Cloud and between machines in HUNT Cloud. If some files already exists on the receiving machine, the files existing files will not be updated

      i. Sudo apt install rsync (UBUNTU)
      Brew install rsync (MAC)
      rsync directory/on/Mac/filename <username>@<your-lab-IP>:$\tilde{/}$ #

Example use from MAC to HC
rsync directory/on/HC/filename ubuntu@mrphys-gpu1:~/ # Example transfer between home and gpu1

(b) Installing 7z. Medical imaging data often tends to become quite large. It is therefore recommended that you zip large files before transfer

   i. sudo apt-get install p7zip-full # (UBUNTU)
   ii. brew install p7zip # (MAC)

(c) Installing Conda. It is very convenient to install a Conda environment if you do not want to struggle with installation paths etc.

   i. wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh # Downloads Conda
   bash Miniconda3-latest-Linux-x86_64.sh # Installs Conda

(d) Jupyter Lab. To get a better Graphics user interface (GUI) when programming in your lab, you can install Jupyter.

   i. conda install -c conda-forge jupyterlab
   jupyter notebook # In X2GO

(e) Spyder

   i. conda install spyder
   spyder # Open in X2GO

(f) MATLAB

   i. Download Matlab to your own PC (https://software.ntnu.no/)
   rsync directory/MATLAB username@user-name
   rsync MATLAB mrphys-gpu1:~/ sudo apt install unzip libnss3 #(install dependencies)
   unzip <matlabfile.zip> chmod -R 777 <matlab-folder> # make sure it is executable
   # Activate license
   Matlab # X2GO

(g) PyCharm

   i. sudo apt get update
   sudo apt install snaps
   sudo snap install pycharm-community –classic
   /snap/bin/pycharm-community #(Running in x2go)

   (h) MRIcron for viewing nifti images
      i. sudo apt-get install mricron

10) Converting from DICOM to Nifti. When working with medical imaging it is very useful to convert the images to Nifti.

   (a) rsync path/to/MRimages ubuntu@mrphys-gpu1:~/
   rsync gen_nii.sh ubuntu@mrphys-gpu1:~/
   rsync extract_prot_from_dicom ubuntu@mrphys-gpu1:~/
   rsync xzdirauto.sh ubuntu@mrphys-gpu1:~/
   ssh ubuntu@mrphys-gpu1 -XY
   chmod 744 * # make sure it is excecutable
   mkdir dicom
   mv *005* dicom # These are T2* magnitiude images
   mv *006* dicom # These are phase images
   gen_nii.sh dicom # creates nifti files of phase and magnitude + protocol. Also the original dicom file is zipped

11) QSM reconstruction using TGV-QSM algorithm (Langkammer et al. 2015). Now that you have nifti images you can install TGVQSM package to reconstruct QSM from phase and magnitude images. As you need Python 2.7 I recommend creating a new environment in Conda where you can install Python2.7, as you will most likely use python3 for other task. The manual below is a small modification from https://github.com/CAIsr/qsm#using-tgvqsm-in-windows-subsystem-for-linux-wsl-10

   (a) conda create -n QSM python ==2.7 # creates a separate environment with Python 2.7. The "base" environment will not be affected
   conda activate QSM # Activating your environment with Pyhton 2.7
   conda install numpy #installs numpy in this environment
   conda install pyparsing
   # (make sure pip is not your system pip, but the one in miniconda: which pip)
   pip install scipy==0.17.1 nibabel==2.1.0 Cython==0.19.2
   wget http://www.neuroimaging.at/media/qsm/TGVQSM-plus.zip #Downloads scripts
   unzip TGVQSM-plus.zip
   cd *TGVQSM-master-*
   python setup.py install # installs TGVQSM

cd test_data
tgv_qsm -p epi3d_test_phase.nii.gz -m epi3d_test_mask.nii.gz -f 2.89 -t 0.027 -o epi3d_test_QSM #testing on test images. Takes approx 2 minutes on our machine
cd nifti # go to the nifti directory with phase and magnitude 3t_qsm_by_tgv_qsm.sh
#Do this in X2GO - you will be viewing images
# Choose **one** of the protocols (does not matter which) -> press "i" # Choose one magnitude image -> press "m" # Choose one phase image -> press "p"
# Type "yes" when you are asked

12) Run DL on ubuntu@mrphys-gpu1. GPU will drastically decrease training time as you exploit the parallellisation a GPU offer. However, to exploit this you must first make sure that you have the required drivers and libraries to exploit the GPU for that specific Tensorflow GPU version. Tensorflow-GPU 2.0.0. was used in this project. An easy and organised method for installing all the packages, without interfering with other packages and libraries is to create a new environment that you activate each time you do DL. Driver version: 460.32.03, CUDA version 11.2

    (a) ssh ubuntu@mrphys-gpu1 -XY
    conda create -n DL python=3.7 #Change the name "DL" to whatever you want
    conda activate DL conda install tensorflow-gpu=2.0.0 #Not necessarily the smartest thing to install the newest tf, as this requires newer drivers
    # Add NVIDIA package repositories (from https://www.tensorflow.org/install/gpu):

        i. wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin
        sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/ cuda-repository-pin-600
        sudo apt-key adv –fetch-keys https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/7fa2af80.pub
        sudo add-apt-repository "deb https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/ /"
        sudo apt-get update
        wget http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64/nvidia-machine-learning-repo-ubuntu1804_1.0.0-1_amd64.deb

sudo apt install
./nvidia-machine-learning-repo-ubuntu1804_1.0.0-1_amd64.deb sudo
apt-get update
wget https://developer.download.nvidia.com/compute/machine-learning/
repos/ubuntu1804/x86_64/libnvinfer7_7.1.3-1+cuda11.0_amd64.deb
sudo apt install ./libnvinfer7_7.1.3-1+cuda11.0_amd64.deb
sudo apt-get update
sudo apt-get install –no-install-recommends cuda-11-0 libcudnn8=8.0.4.30-1+cuda11.0| libcudnn8-dev=8.0.4.30-1+cuda11.0
sudo reboot # migth take a couple of minutes nvidia-smi # run management interface to verify all things are working
sudo apt-get install -y –no-install-recommends
libnvinfer7=7.1.3-1+cuda11.0
libnvinfer-dev=7.1.3-1+cuda11.2
libnvinfer-plugin7=7.1.3-1+cuda11.2

(b) Install necessary packages for DL. In the Conda DL environment, you most likely will need packages such as numpy etc. This is easily installed with Conda

i. conda activate DL
conda install skimage
conda install scikit-image
conda install scikit-learn
conda install numpy
conda install scipy
conda install glob
conda install opencv
conda install pillow
conda install tqdm

13) Install FSL through their webpage: https://web.mit.edu/fsl_v5.0.10/fsl/doc/wiki/FslInstallation(2f)Linux.html#Debian.2FUbuntu_users. The following commands will run the operation described in Figure 3.5.1.

(a) flirt -in /home/ubuntu/invol.nii.gz -ref /home/ubuntu/refvol_7T.nii.gz -out /home/ubuntu/outvol3Tto7T.nii.gz -omat outvol3Tto7TtranaformMatrixInverse.mat

-dof 12 #Transforms input (3T) image to reference (7T) space and saves nifti an transformation matrix
convert_xfm -omat outvol3Tto7TtranaformMatrix.mat -inverse outvol3Tto7TtranaformMatrix_inverse.mat #Find the inverse transformation

(b) flirt -in 7Tsegmentations.nii.gz -ref /home/ubuntu/invol.nii.gz -out /home/ubuntu/3Tsegmentations.nii.gz -init outvol3Tto7TtranaformMatrixInverse.mat -applyxfm #Transforms input (7T segmentations) to output (3T segmentations) using the inverse transformation calculated above

14) General tips when working in your lab,

(a) sudo apt-get install nohup
touch output-file.txt #Creates .txt file
nohup python file.py > output-file.txt # Runs script in background and saves output (verbose) to .txt file

(b) tail -f output-file.txt # displays the output in real time

(c) vim pyhonFile.py # quick way to do changes to scripts (or text documents)

(d) conda info –envs # list of installed conda environments

(e) history # list of last commands performed in terminal

(f) ls -rtl # list read and write permissions of files
chmod 744 filename.py # Approve read and write permission of file. Usually the case if script is not running properly

Appendix C

Appendix C

This chapter will present the most central parts of the code structures that were implemented in this thesis.

C.1 Pre-processing of data

This section presents the pre-processing of data. Only the code for the 7T images is shown, but it is similar to the 3T pre-processing.

```
X_TRAIN = []
Y_TRAIN = []
#From file paths
for MR,RN in zip(MR_7T_TRAIN,RN_7T_TRAIN):
    #Open Nifti Images
    img = read_nifti(MR)
    mask = read_nifti(RN)
    img = normalize(img)
    #Crop to avoid some unecessary background
    new_img,new_mask = crop7T(img,mask)

    #Add unaltered brain image to list#
    add_slice(new_img,X_TRAIN)
    add_slice(new_mask,Y_TRAIN)
```

```
#Rotation#
for x in range(0,1):
    new_img1,new_mask1 = random_rotate(new_img,new_mask)
    add_slice(new_img1,X_TRAIN)
    add_slice(new_mask1,Y_TRAIN)

#Flip up down
img_updown = np.flipud(new_img)
mask_updown = np.flipud(new_mask)
add_slice(img_updown,X_TRAIN)
add_slice(mask_updown,Y_TRAIN)

#Noise#
for x in range(0,1):
    new_img1 = add_random_noise(new_img)
    add_slice(new_img1,X_TRAIN)
    add_slice(new_mask,Y_TRAIN)
#Sheer#
for x in range(0,1):
    new_img1,new_mask1 = sheer_forces(new_img,new_mask)
    add_slice(new_img1,X_TRAIN)
    add_slice(new_mask1,Y_TRAIN)
#Elastic transformation (not used)
#new_img1,new_mask1 = elast

#Blur
for x in range(0,1):
    new_img1 = image_blur(new_img)
    add_slice(new_img1,X_TRAIN)
    add_slice(new_mask,Y_TRAIN)
#Contrast
for x in range(0,1):
    new_img1 = improve_contrast(new_img)
    add_slice(new_img1,X_TRAIN)
    add_slice(new_mask,Y_TRAIN)
```

```
#Create numpy array (for tf)
X_TRAIN = np.asarray(X_TRAIN, dtype = np.float32)
Y_TRAIN = np.asarray(Y_TRAIN, dtype = np.float32)

#Delete unnecessary data to free memory
del new_img
del new_mask
del new_img1
del new_mask1
gc.collect()

############################################
#Functions for expand dims (for greyscale)
def train_preprocessing(volume,label):
    volume = tf.expand_dims(volume, axis=2)
    label = tf.expand_dims(label,axis=2)
    return volume, label

def validation_preprocessing(volume,label):
    volume = tf.expand_dims(volume, axis=2)
    label = tf.expand_dims(label, axis=2)
    return volume, label
```

C.2 U-Net

This section presents a generic version of the U-Net which was implemented. Changes between the datasets and ROIs occur.

```
###############################################################
### Different metrics ###
# Dice
from tensorflow.keras import backend as K
def dice_coeff(y_true, y_pred, smooth=1.):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) /
    (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

def dice_coeff_loss(y_true, y_pred):
    return 1-dice_coeff(y_true, y_pred)

def CE_DL_loss(y_true, y_pred):
    def dice_loss(y_true, y_pred):
      y_pred = tf.math.sigmoid(y_pred)
      numerator = 2 * tf.reduce_sum(y_true * y_pred)
      denominator = tf.reduce_sum(y_true + y_pred)

      return 1 - numerator / denominator

    y_true = tf.cast(y_true, tf.float32)
    o = tf.nn.sigmoid_cross_entropy_with_logits(y_true, y_pred)
    + dice_loss(y_true, y_pred)
    return tf.reduce_mean(o)
####################################################################
#U-NET
import tensorflow as tf
from tensorflow.keras import metrics
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras import optimizers
from tensorflow.keras import callbacks
from tensorflow.keras import metrics
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger,
TensorBoard


def U_NET(filter1,x,y,channels):
    inputs = keras.Input(shape=(x,y,channels))

    conv1 = layers.Conv2D(filter1, (3, 3), activation='relu',
    padding='same')(inputs)
    conv1 = layers.BatchNormalization()(conv1)
    conv1 = layers.Dropout(0.1)(conv1)
    conv1 = layers.Conv2D(filter1, (3, 3), activation='relu',
    padding='same')(conv1)
    conv1 = layers.BatchNormalization()(conv1)
    pool1 = layers.MaxPooling2D(pool_size=(2, 2))(conv1)
    pool1 = layers.Dropout(0.10)(pool1)

#filter2 = 32
    conv2 = layers.Conv2D(2*filter1, (3, 3), activation='relu',
    padding='same')(pool1)
    conv2 = layers.BatchNormalization()(conv2)
    conv2 = layers.Dropout(0.1)(conv2)
    conv2 = layers.Conv2D(2*filter1, (3, 3), activation='relu',
    padding='same')(conv2)
    conv2 = layers.BatchNormalization()(conv2)
    pool2 = layers.MaxPooling2D(pool_size=(2, 2))(conv2)
    pool2 = layers.Dropout(0.10)(pool2)

#filter3 = 64
    conv3 = layers.Conv2D(2*2*filter1, (3, 3), activation='relu',
    padding='same')(pool2)
    conv3 = layers.BatchNormalization()(conv3)
    conv3 = layers.Dropout(0.2)(conv3)
    conv3 = layers.Conv2D(2*2*filter1, (3,3), activation='relu',
    padding='same')(conv3)
    conv3 = layers.BatchNormalization()(conv3)
    pool3 = layers.MaxPooling2D(pool_size=(2, 2))(conv3)
    pool3 = layers.Dropout(0.2)(pool3)

#filter4 = 128
    conv4 = layers.Conv2D(2*2*2*filter1,(3, 3), activation='relu',
    padding='same')(pool3)
    conv4 = layers.BatchNormalization()(conv4)
    pool4 = layers.Dropout(0.2)(conv4)
    conv4 = layers.Conv2D(2*2*2*filter1, (3, 3), activation='relu',
    padding='same')(conv4)
    conv4 = layers.BatchNormalization()(conv4)
    conv4 = layers.Dropout(0.3)(conv4)
    pool4 = layers.MaxPooling2D(pool_size=(2, 2))(conv4)
    pool4 = layers.Dropout(0.20)(pool4)

#filter5 = 256
    conv5 = layers.Conv2D(2*2*2*2*filter1, (3, 3), activation='relu',
    padding='same')(pool4)
    conv5 = layers.BatchNormalization()(conv5)
    conv5 = layers.Dropout(0.3)(conv5)
    conv5 = layers.Conv2D(2*2*2*2*filter1, (3, 3), activation='relu',
    padding='same')(conv5)
    conv4 = layers.BatchNormalization()(conv4)
    conv5 = layers.Dropout(0.2)(conv5)

##########################################
#EXPANSIVE PATH
##########################################
#filter6 = 128
    up6 = layers.Conv2DTranspose(2*2*2*filter1, (3, 3),
    activation="relu",strides = (2,2), padding="same")(conv5)
    up6 = layers.BatchNormalization()(up6)
    up6 = layers.concatenate([up6,conv4])
    conv6 = layers.Conv2D(2*2*2*filter1, (3, 3), activation='relu',
    padding='same')(up6)
    conv6 = layers.BatchNormalization()(conv6)
    conv6 = layers.Dropout(0.2)(conv6)
    conv6 = layers.Conv2D(2*2*2*filter1, (3, 3), activation='relu',
    padding='same')(conv6)
    conv6 = layers.BatchNormalization()(conv6)

#filter7 = 64
    up7 = layers.Conv2DTranspose(2*2*filter1, (3, 3),
    activation="relu",strides = (2,2), padding="same")(conv6)
    up7 = layers.BatchNormalization()(up7)
    up7 = layers.concatenate([up7,conv3])
    conv7 = layers.Conv2D(2*2*filter1, (3, 3), activation='relu',
    padding='same')(up7)
    conv7 = layers.BatchNormalization()(conv7)
    conv7 = layers.Dropout(0.2)(conv7)
    conv7 = layers.Conv2D(2*2*filter1, (3, 3), activation='relu',
    padding='same')(conv7)
    conv7 = layers.BatchNormalization()(conv7)

#filter8 = 32
    up8 = layers.Conv2DTranspose(2*filter1, (3, 3),
    activation="relu",strides = (2,2), padding="same")(conv7)
    up8 = layers.BatchNormalization()(up8)
    up8 = layers.concatenate([up8,conv2])
    conv8 = layers.Conv2D(2*filter1, (3, 3), activation='relu',
    padding='same')(up8)
    conv8 = layers.BatchNormalization()(conv8)
    conv8 = layers.Dropout(0.1)(conv8)
    conv8 = layers.Conv2D(2*filter1, (3, 3), activation='relu',
    padding='same')(conv8)
    conv8 = layers.BatchNormalization()(conv8)

#filter9 = 16
    up9 = layers.Conv2DTranspose(filter1, (3, 3),
    activation="relu",strides = (2,2), padding="same")(conv8)
    up9 = layers.BatchNormalization()(up9)
    up9 = layers.concatenate([up9,conv1], axis=3)
    up9 = layers.Dropout(0.2)(up9)
    conv9 = layers.Conv2D(filter1, (3, 3), activation='relu',
    padding='same')(up9)
    conv9 = layers.BatchNormalization()(conv9)
    conv9 = layers.Conv2D(filter1, (3, 3), activation='relu',
    padding='same')(conv9)
    conv9 = layers.BatchNormalization()(conv9)
    outputs = layers.Conv2D(1, ( 1, 1), activation='sigmoid')(conv9)

    model = tf.keras.Model(inputs=[inputs], outputs=[outputs])
    return model
    #model.summary(line_length=120)
    #Resetting model weigths
    #model.save_weights('reset.h5')
    #model.load_weights('reset.h5')
```

C.3 Training

This section presents the code that was used to train on the 7T dataset. The code has similarities to the code used for training of the 3T CNNs.

```
model = U_NET(16,176,224,1)
model.compile(optimizer="rmsprop", loss="binary_crossentropy",
metrics=[dice_coeff])
#model.summary(line_length=120)
#Retting model weigths
model.save_weights('reset.h5')
#################################################
factor = int(X_TRAIN.shape[0]/(len(MR_7T_TRAIN*150)))
for x in range(0,len(MR_7T_TRAIN)):
    print("-----Training for for fold ", x)
    model.load_weights('reset.h5')
    print("loop: ",x)
    pic_ = int(x * factor*150)
    #pic = int((x+1)*factor)
    #pic = int((x+1)*factor + 150)
    pic = pic_ + 150
    #plt.imshow(X_TRAIN[pic_:pic,:,69])
    #plt.show()

    x_val = X_TRAIN[pic_:pic]
    print("xval ",x_val.shape)
    y_val = Y_TRAIN[pic_:pic]
```

```
print("yval ",y_val.shape)
x_train = np.delete(X_TRAIN,slice(pic_,pic),axis=0)
print("xtrain ",x_train.shape)
y_train = np.delete(Y_TRAIN,slice(pic_,pic),axis=0)
print("y_train ",y_train.shape)

#Some of the data loaders code is borrowed from Keras
# Define data loaders.
train_loader = tf.data.Dataset.
from_tensor_slices((x_train, y_train))
validation_loader = tf.data.Dataset.
from_tensor_slices((x_val, y_val))
print("train dataset")
batch_size = 30
# Augment the on the fly during training.
train_dataset = (
    train_loader.shuffle(len(x_train))
    .map(train_preprocessing)
    .batch(batch_size)
    .prefetch(30)
    )

print("Validation dataset")
# Only rescale.
validation_dataset = (
    validation_loader.shuffle(len(x_val))
    .map(validation_preprocessing)
    .batch(batch_size)
    .prefetch(30)
    )


#### Training #####
import os
output_directory = "/home/ubuntu/eivind/weights/7TFACE/2DCV/"
# Train the model, doing validation at the end of each epoch
```

```
    #log_dir = "/home/ubuntu/eivind/weights/7TRN/2DCV/" +
    datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    #tensorboard_callback =
    tf.keras.callbacks.TensorBoard(log_dir=log_dir,
    histogram_freq=1)
    mcp_save = ModelCheckpoint(os.path.join(output_directory,
    "model_best"+
    str(x)+".h5"), save_best_only=True)

    epochs = 35
    model.fit(train_dataset,validation_data=validation_dataset,
    epochs=epochs,shuffle=True,verbose=2,
    callbacks=[mcp_save]) #
    callbacks=[mcp_save,tensorboard_callback,checkpoint]
```

C.4 Obtaining results and post-processing of the results

```
vekter = sorted(glob("/home/ubuntu/eivind/weights/7TRN/2DCV/*.h5"))

dice_score = []
suscM = []
suscA = []
sensitivity = []
precision = []
for i in range(0,len(vekter)):
    print("----------new brain----------")
    print("Round ", i)
    print(vekter[i])
    height_ = 150*i
    height = 150 * (i+1)
    model = U_NET(16,176,224,1) #16 for RN,SN, 32 for STN
    model.compile(optimizer="rmsprop", loss="binary_crossentropy",
    metrics=[dice_coeff])
    model.load_weights(vekter[i])
    X_TEST__ = X_TEST[height_:height,:,:,:]
    preds_test = model.predict(X_TEST__, batch_size = 1, verbose = 2)
    #suscM = []
    #suscA = []
    #dice_score = []
    preds_test_t = (preds_test > 0.5).astype(np.float32)
    preds_test_t = preds_test_t[:,:,:,0]
    #plt.subplot(2,2,1)
    #plt.hist(preds_test_t[:,:,:],bins=20)
    #plt.subplot(2,2,2)
    #plt.hist(Y_TEST[:,:,:,0])
    #plt.show()
        #preds_test_t_SN = (preds_test_SN > 0.5).astype(np.float32)
        #preds_test_t_SN = preds_test_t_SN[height_:height,:,:,0]

        #preds_test_t_RN = (preds_test_RN > 0.5).astype(np.float32)
        #preds_test_t_RN = preds_test_t_RN[height_:height,:,:,0]

    Y_TEST_ = Y_TEST[height_:height,:,:,0]
    X_TEST_ = X_TEST[height_:height,:,:,0]
    X_TEST_unormalized_ = X_TEST_unormalized[height_:height,:,:]
    dice = dice2D(Y_TEST_,preds_test_t)
    dice_score.append(dice)
    #sensitivity.append(sensitivity_)
    #precision.append(precision_)
    #susceptibility
 # Susceptibility wrt CSF has been accounted for
    manual,preds = susceptibility(Y_TEST_,preds_test_t,
    X_TEST_unormalized_)

    manual_average = (np.average(manual))
    manual_average -= (average_values_csf[i])

    preds_average = (np.average(preds))
    preds_average -= ((average_values_csf[i]))
    suscM.append(manual_average)
    suscA.append(preds_average)
    print("Susceptibility")
    print("manual")
    #print("max ", max(manual))
    #print("min ", min(manual))
    print("preds")
    #print("max ", max(preds))
    #print("min ", min(preds))
    print("average")
    print("average true ", np.average(manual))
    print("average preds ", np.average(preds))

    """
    # for saving numpy arrays as nifti
    if x == 1:

        new_preds = nib.Nifti1Image(array,affine=np.eye(4,4))
        nib.save(new_preds,"3TSN_array.nii.gz")

        new_preds = nib.Nifti1Image(preds_test_t,affine=np.eye(4,4))
        nib.save(new_preds,"3TSN_preds.nii.gz")

        new_preds = nib.Nifti1Image(Y_TEST_,affine=np.eye(4,4))
        nib.save(new_preds,"3TSN_manual.nii.gz")

        new_preds = nib.Nifti1Image(X_TEST_,affine=np.eye(4,4))
        nib.save(new_preds,"3TSN_MR.nii.gz")

        new_preds = nib.Nifti1Image(True_pos,affine=np.eye(4,4))
        nib.save(new_preds,"3TSTN_True_pos.nii.gz")
        new_manual = nib.Nifti1Image(False_neg,affine=np.eye(4,4))
        nib.save(new_manual, "3TSTN_False_neg.nii.gz")
        new_manual = nib.Nifti1Image(False_pos,affine=np.eye(4,4))
        nib.save(new_manual, "3TSTN_False_pos.nii.gz")
        new_image = nib.Nifti1Image(X_TEST_,affine=np.eye(4,4))
        nib.save(new_image,"3TMR.nii.gz")
    """
print("---------Summary---------")
print("dicescore= ", dice_score)
print("diceaverage= ", np.average(dice_score))
print("suscM= ", suscM)
print("averageM= ", np.average(suscM))
print("sucsA= ",suscA)
print("averageA= ", np.average(suscA))
print("sensitivity= ", sensitivity)
print("precision= ", precision)
```

C.5 Helper functions

```
import nibabel as nib
import numpy as np
import sklearn
import scipy

def read_nifti(file):
    img = (nib.load(file)).get_data()
    return img

def padding(file,size1,size2,size3,old_size):
    ref_pixel_val = file[0,0,0]
    new_img = np.zeros([size1,size2,size3]) #256,256,256
    new_img[:,:,0:old_size] += file[:,:,0:old_size]
    new_img[:,:,old_size:size3] += ref_pixel_val
    return new_img

def plot_images(file_,slice_):
    plt.imshow(file_[:,:,slice_],cmap="gray")
    plt.show()

def crop(image,label):
    #resized_image = image.crop((16,32,400,300))
    resized_image = image[16:240,16:240,15:159]
    resized_label = label[16:240,16:240,15:159]
    return resized_image,resized_label
```

```
def crop7T(image,label):
    #resized_image = image.crop((16,32,400,300))
    resized_image = image[30:206,32:256,45:195]
    resized_label = label[30:206,32:256,45:195]
    return resized_image,resized_label

def crop7T3D(image,label):
    resized_image = image[30:206,32:256,35:185]
    resized_label = label[30:206,32:256,35:185]
    return resized_image,resized_label

def crop_siemens3T(image,label):
    resized_image = image[14:174,32:208,:]
    resized_label = label[14:174,32:208,:]
    return resized_image,resized_label

def crop_CSF3T(image):
    resized_image = image[14:174,32:208,:]
    #resized_label = label[14:174,32:208,:]
    #print(resized_image.shape)
    return resized_image

def normalize(MR):
    #DICOM to nifti convertion
    norm = (MR/10000) - 0.2048
    norm[norm > 0.2047] = 0.2047
    norm[norm < -0.2048] = -0.2048
    return norm

def normalize_scan3T(MR,CSF):
    #Where CSF is a list of manually labelled CSF
    coord_csf = []
    csf_x,csf_y,csf_z = np.where(CSF > np.average(CSF)) #Coordinates
    print("coordinates")
    print(len(csf_x))
    print(len(csf_y))
    print(len(csf_z))
    for x in range(0,len(csf_x)):
        coord_csf.append(MR[csf_x[x],csf_y[x],csf_z[x]])
    average_csf = np.mean(coord_csf)
    averageMR = np.average(MR)
    print("average_csf: ", average_csf)

    MR = np.asarray(MR,dtype=np.float32)
    MR -= average_csf
    return MR

def normalize_scan7T(MR,CSF):
    #Where CSF is a list of manually labelled CSF
    coord_csf = []
    csf_x,csf_y,csf_z = np.where(CSF > 0) #Coordinates
    for x in range(0,len(csf_x)):
        coord_csf.append(MR[csf_x[x],csf_y[x],csf_z[x]])
    average_csf = np.average(coord_csf)
    averageMR = np.average(MR)
    print("average_csf: ", average_csf)
    print("average RM: ", averageMR)
    MR = np.asarray(MR,dtype=np.float32)
    MR -= average_csf
    return MR

def find_csf_values(MR,CSF):
    coord_csf = []
    csf_x,csf_y,csf_z = np.where(CSF > 0.9*np.max(CSF)) #Coordinates
    for x in range(0,len(csf_x)):
        coord_csf.append(MR[csf_x[x],csf_y[x],csf_z[x]])
    average_csf = np.average(coord_csf)
    print("average_csf: ", average_csf)
    return average_csf

def unormalize(img):
    #img = np.asarray(img,dtype=np.float32)
    dicom = (img + 0.2048) * 10000
    dicom[dicom > 4095] = 4095
    dicom[dicom < 0] = 0
    return dicom

def reverse_unormalize(img):
    #img = np.asarray(img,dtype=np.float32)
    norm = (img/10000) - 0.2048
    norm[norm > 0.2047] = 0.2047
    norm[norm < -0.2048] = -0.2048
    return norm

def add_slice(input_,list_):
    for slice_ in range(0,input_.shape[2]):
        list_.append(input_[:,:,slice_])
    return list_

def random_rotate(image,label):
    angle = random.randint(0,360)
    image = ndimage.rotate(image,angle,reshape=False)
    label = ndimage.rotate(label,angle,reshape=False)
    return image, label

def add_random_noise(image):
    noise = random.randint(0,5) / 100
    img_noise = random_noise(image, var=noise**2)
    return img_noise

def improve_contrast(image):
    x = random.randint(0,10) / 10
    v_min, v_max = np.percentile(image, (x, 99.8))
    img_contrast = exposure.rescale_intensity(
    image, in_range=(v_min, v_max))
    return img_contrast
```

```
def sheer_forces(image,label):
    random_sheer = random.randint(0,4) / 10
    tf = transform.AffineTransform(shear = random_sheer)
    img_sheer = transform.warp(image, tf, order=1,
    preserve_range=True,mode='constant')
    mask_sheer = transform.warp(label, tf,order=1,
    preserve_range=True,mode='constant')
    return img_sheer, mask_sheer

def image_blur(image):
    img_blur = ndimage.uniform_filter(image, size=(3,3,3))
    return img_blur

def elastic_deformation(image,label):
    [X_deformed, Y_deformed] = elasticdeform.deform_random_grid(
    [image,label],sigma=5, points=3)
    return X_deformed,Y_deformed

def plot2D(img,label,slice_):
    plt.figure()
    plt.subplot(2,2,1)
    plt.imshow(img[:,:,slice_])
    plt.subplot(2,2,2)
    plt.imshow(label[:,:,slice_])
    plt.show()

def dice2D(Y_test_,preds_test_t):
    dice1 = np.sum(Y_test_[preds_test_t == 1]) * 2.0 /
    (np.sum(Y_test_) + np.sum(preds_test_t))
    return dice1

def susceptibility(Y_test,Y_pred,X_test):
    #remember preds_test_t
    susc_true = []
    susc_pred = []
    print("Ytest", Y_test.shape)
```

```
list_true_x,list_true_y,list_true_z =
np.where(Y_test > 0) #Coordinates
print("list x true shape",list_true_x.shape)
print("len list",len(list_true_x))
print("list true 0",list_true_x[0])
list_pred_x,list_pred_y,list_pred_z =
np.where(Y_pred > 0) #Coordinates
for x in range(0,len(list_true_x)):
    susc_true.append(X_test[list_true_x[x],list_true_y[x],
    list_true_z[x]])
for x in range(0,len(list_pred_x)):
    susc_pred.append(X_test[list_pred_x[x],list_pred_y[x],
    list_pred_z[x]])
return susc_true,susc_pred
```

Eivind Lysheim

Master's Thesis

NTNU
Norwegian University of
Science and Technology