Bachelor of Science in Electrical and Electronic Engineering

EEE 400 (January 2024): Thesis

Search Towards an Advanced Deep Learning Algorithm for MRI Quantitative Susceptibility Mapping (QSM) with Magnetic Anisotropy

Submitted by

Tasmin Khan
201906055

Supervised by

Dr. Maruf Ahmed
Assistant Professor

Department of Electrical and Electronic Engineering

Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

January 2024

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis, titled, "Search Towards an Advanced Deep Learning Algorithm for MRI Quantitative Susceptibility Mapping (QSM) with Magnetic Anisotropy", is the outcome of the investigation and research carried out by us under the supervision of Dr. Maruf Ahmed.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Tasmin Khan
201906055

CERTIFICATION

This thesis titled, **"Search Towards an Advanced Deep Learning Algorithm for MRI Quantitative Susceptibility Mapping (QSM) with Magnetic Anisotropy"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for EEE 400: Project/Thesis course, and as the requirements for the degree B.Sc. in Electrical and Electronic Engineering in January 2024.

Group Members:

Tasmin Khan

Supervisor:

Dr. Maruf Ahmed
Assistant Professor
Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my supervisor, Dr. Maruf Ahmed, whose exceptional patience and support have been instrumental throughout this work. He has taught me not only the smallest details, such as how to approach reading research papers effectively, but also the more complex and intricate aspects of this field. His guidance has been invaluable, and I consider him one of the most honorable individuals I have had the privilege to know.

I am also grateful to Ms. Ruimin Feng, Ph.D. candidate at Shanghai Jiao Tong University, for her significant assistance in elucidating various processing steps and critical details which have been indispensable to the progress of this work.

I extend my sincere thanks to my thesis peers as well, whose timely assistance with resources and collaborative spirit have greatly enriched this whole experience.

Tasmin Khan

Dhaka
January 2024

Contents

List of Figures

List of Tables

ABSTRACT

Quantitative Susceptibility Mapping (QSM) is an MRI imaging technique that provides insights into the distribution of tissue magnetic properties, assisting in the early diagnosis of diseases such as Multiple Sclerosis, Alzheimer's, and Parkinson's. By detecting abnormal distributions of paramagnetic or diamagnetic materials in tissue, QSM plays a vital role in understanding disease progression. However, the process of converting the acquired MRI phase map into a susceptibility map remains ill-posed, posing challenges for accurate reconstruction. Additionally, the consideration of magnetic anisotropy presents significant difficulty due to the complex nature of data acquisition.

This research explores the development of a deep learning-based solution to address the ill-posed inversion problem in QSM reconstruction. A dataset comprising multi-orientation scans from 8 different patients was used to account for the directional dependence of magnetic response. The data were processed to generate phase maps (input) and susceptibility maps (target output). Two distinct models were trained: one on a 3D dataset and the other on 2D slices. Two popular metrics and loss function was monitored to better understand the results.

By processing 2D slices instead of the entire brain volume, the model became not only computationally efficient but also more practical for clinical analysis and diagnosis. Experimental results revealed that while transformer-based architectures showed subpar performance with multi-orientation data from the same subject, the model trained on 2D slices was computationally less expensive. In conclusion, the integration of deep learning models for QSM reconstruction holds great potential for improving clinical workflows. Successful implementation will require both technical advancements and the adaptation of healthcare practices to ensure trust and effectiveness in real-world applications

Chapter 1

Introduction

1.1 Background and Motivation

Magnetic Resonance Imaging has revolutionized medical imaging by enabling noninvasive visualization of internal anatomical structures. MRI basically captures the magnetic response of the physical tissues. Among many domains within MRI,significant advancement has happened in terms of Quantitative Susceptibility Mapping (QSM), which is a technique for the visualization of the magnetic susceptibility of different tissues within the body, essentially capturing the distribution of different magnetic materials [3].

In clinical practice, QSM is able to distinguish the distribution of paramagnetic (iron, deoxyhemoglobin) from diamagnetic (calcium, myelin) substances. QSM shows positive susceptibility for paramagnetic materials and negative susceptibility for diamagnetic materials. These are critical biomarkers in various neurological disorders. For example, mapping iron accumulation and myelin damage greatly aid in diagnosing Alzheimer's [4], Parkinson's [5], stroke and Multiple Sclerosis (MS) [6]. As shown in figure 1.1, the segmented regions help identify the targeted lesions immediately. It has also been used in studies of tumor characterization, hemorrhage detection, demyelination disorders, and venous oxygenation [7]. And thus, QSM has become vital for brain pathology research.

Despite its potential, QSM reconstruction is challenging because it involves reconstructing tissue susceptibility maps from MRI phase images. Since each voxel's susceptibility is unknown, solving this is mathematically complex and presents an ill-posed problem. Traditional reconstruction methods such as iLSQR (Improved Sparse Linear Equation and Least Squares) [8], MEDI (Morphology Enabled Dipole Inversion) [9], and STAR-QSM [10] rely on mathematical regularization techniques to approximate solutions. However, these methods suffer from time consuming iterative pipeline to converge while relying on manually tuned parameters. They struggle specially in low signal regions and the performance varies across imaging protocols.

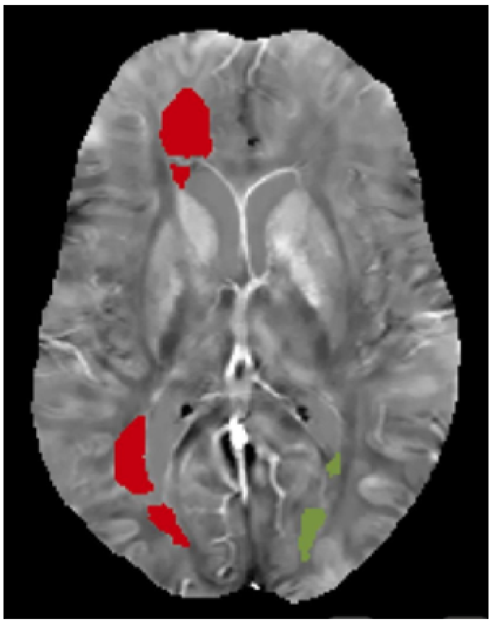

Figure 1.1: Segmentation of Iron Accumulation Regions in the brain, Red segmentation is Iron Accumulation (Rim+) and Green segmentation is Absence of Iron (Rim-) lesions [1].

One fundamental limitation of conventional QSM methods is the assumption of isotropic magnetic susceptibility, meaning that the same susceptibility value is applied uniformly in all directions. However, in reality, many biological tissues exhibit magnetic anisotropy, where susceptibility varies depending on the orientation of the tissue structure relative to the magnetic field. This effect is particularly evident in white matter fibers in the brain, where the organization of myelin sheaths causes directional variations in susceptibility. Ignoring anisotropy can lead to distortions in susceptibility maps, particularly when analyzing multiple orientations in QSM.

Recent works in deep learning have introduced a rather data driven approach for QSM reconstruction. In this way the model can learn non linear transformations between input phase image and susceptibility maps. It can improve the speed of inference by avoiding computationally expensive iterative solver. Depending on the architecture, these models rarely need manual parameter tuning once trained. By controlling what kind of data the network is exposed to, the model can be taught to reduce artifacts and noise as well. Furthermore, by carefully selecting and augmenting training data, these models can be optimized to enhance susceptibility estimation in complex tissue environments. These advancements in deep learning-based QSM reconstruction mark a significant shift from traditional methods, offering potential improvements in accuracy, speed, and robustness.

However, despite these advancements, incorporating magnetic anisotropy into deep learning-based QSM reconstruction remains a key challenge. Most deep learning models are trained under the assumption of isotropic susceptibility, which can lead to inaccuracies when applied to anisotropic tissues—particularly in regions like white matter, where susceptibility varies with orientation. Additionally, while deep learning reduces reliance on manual parameter tuning, the overall network architecture, quality of training data, and choice of regularization strategies still play a crucial role in determining model performance.

1.2 Problem Formulation

Quantitative Susceptibility Mapping (QSM) aims to reconstruct the underlying magnetic susceptibility distribution $\chi(\mathbf{r})$ from the phase images acquired through magnetic resonance imaging (MRI). The phase accumulation in MRI is a result of local variations in the magnetic field $\Delta B(\mathbf{r})$, which, in turn, is influenced by the susceptibility distribution within the tissue. The mathematical relationship between the perturbed magnetic field and the susceptibility distribution is described by the convolution:

$$\Delta B(\mathbf{r}) = D(\mathbf{r}) * \chi(\mathbf{r}), \tag{1.1}$$

where $D(\mathbf{r})$ represents the dipole kernel that characterizes the influence of magnetic susceptibility on the local field perturbation at a specific point at the space. The objective of QSM reconstruction is to solve for $\chi(\mathbf{r})$ given the measured field $\Delta B(\mathbf{r})$, which requires inverting the convolution in equation (1.1). This inversion is known as the *dipole inversion problem*.

In the Fourier domain, the convolution can be expressed as a point-wise multiplication:

$$\hat{\Delta B}(\mathbf{k}) = \hat{D}(\mathbf{k}) \cdot \hat{\chi}(\mathbf{k}), \tag{1.2}$$

where $\hat{\Delta B}(\mathbf{k})$, $\hat{D}(\mathbf{k})$, and $\hat{\chi}(\mathbf{k})$ are the Fourier transforms of the local field, the dipole kernel, and the susceptibility distribution, respectively. The dipole kernel in the Fourier domain is:

$$\hat{D}(\mathbf{k}) = \frac{1}{3} - \frac{k_z^2}{k_x^2 + k_y^2 + k_z^2}. \tag{1.3}$$

Here, the $\{k_x, k_y, k_z\}$ represent the spatial frequencies in the frequency domain. Since Equation (1.2) is a simple point-wise multiplication, we can get the susceptibility map by inversion of the dipole kernel:

$$\hat{\chi}(\mathbf{k}) = \frac{\hat{\Delta B}(\mathbf{k})}{\hat{D}(\mathbf{k})}. \tag{1.4}$$

However, direct inversion is not feasible due to the presence of *zero crossings* in the dipole kernel. Specifically, from Equation (1.3), it is evident that $\hat{D}(\mathbf{k})$ approaches zero when $k_x^2 + k_y^2 = 2k_z^2$, leading to numerical instability and amplifying noise in the reconstructed susceptibility

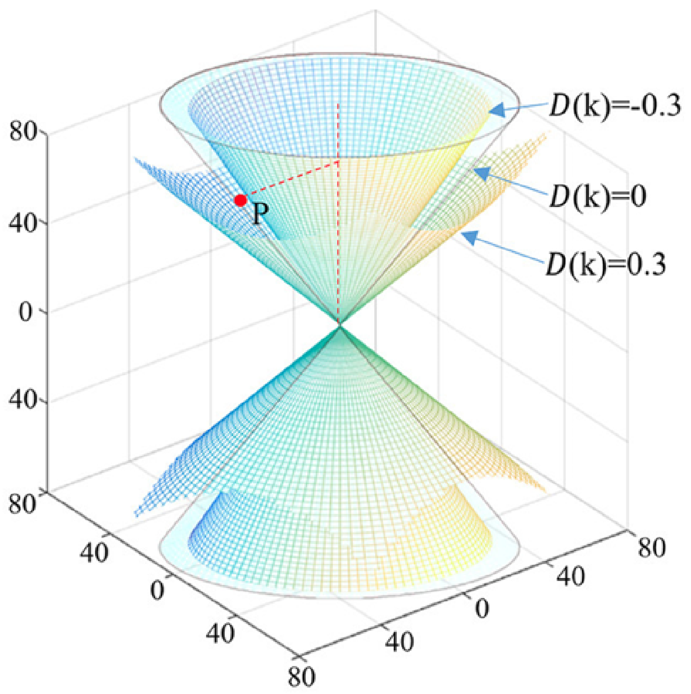

Figure 1.2: Dipole Kernel in the frequency domain, the red marker P represents a particular point in the space [2].

map. This results in an *ill-posed* problem where small perturbations in the measured field map can cause arbitrarily large variations in the computed susceptibility distribution. Since, the dipole kernel has zero crossings in frequency space, as shown in figure 1.2, certain susceptibility components cannot be recovered directly, making the problem fundamentally non-invertible without additional constraints.

To address the ill-posed nature of dipole inversion, regularization techniques like Tikhonov regularization [11] are commonly used to stabilize the inversion. This is done by introducing a penalty term:

$$\hat{\chi}_{\text{reg}} = \arg\min_{\hat{\chi}} \left\{ \|\hat{V}(\mathbf{k})\hat{\chi}(\mathbf{k}) - \hat{\Delta B}(\mathbf{k})\|^2 + \lambda \|\hat{R}(\mathbf{k})\|^2 \right\}, \tag{1.5}$$

where λ balances data fidelity and regularization, and $R(\mathbf{k})$ applies smoothness constraints like total variation (TV). However, these methods rely on iterative solvers and predefined priors. The models also need careful parameter tuning which may not generalize well across different datasets and anatomical structures.

In contrast, deep learning offers a promising solution to address these limitations. The main goal of this study is to search for a suitable Deep Learning enabled Network that directly solves the QSM inversion problem without the need for pre-imposed constraints on the phase map or parameter tuning. The model learns the underlying susceptibility distribution from the data itself, enabling a more flexible and efficient approach to QSM reconstruction.

1.3 Objective and Outline

Building a high-quality dataset is one of the most crucial and time-consuming steps in any deep learning project. It serves as the foundation for model training, as the quality and diversity of the dataset directly impact the performance and generalization of the model.

The first objective was to process raw MRI scans and prepare a 3D Quantitative Susceptibility Mapping (QSM) dataset. This involved phase image processing, including the generation of phase maps, normalization and susceptibility map reconstruction to account for magnetic anisotropy. The dataset was then fed in patch size of 48 to ensure it was suitable for deep learning model training.

The second objective was to train and analyze the deep learning model using the 3D dataset, ensuring that it could effectively learn the magnetic susceptibility distribution from phase data. The training process was designed to explore the strengths and limitations of a fully 3D-based approach, particularly in terms of model complexity, memory constraints, and generalization capability.

The third objective was to prepare a clinically relevant 2D dataset for QSM reconstruction, derived from the original 3D dataset. This involved slicing the 3D QSM volumes along the x, y, and z axes to create individual 2D slices while maintaining consistent voxel sizes across different datasets. These slices were then preprocessed, normalized, and augmented to enhance data diversity, making them suitable for training a 2D-based deep learning model. This step aligns with clinical applications, as 2D slices are commonly used in medical imaging interpretation.

Finally, the fourth objective was to train and evaluate a deep learning model using the 2D dataset to compare its performance against the 3D model. The goal was to determine whether a 2D-based approach could provide competitive results while reducing computational costs and memory requirements. Although the training did not yield fully optimized results, significant progress was made in both architecture design and dataset preparation, setting a strong foundation for future refinements. TThe objectives can thus be listed as

- Process raw MRI scans and prepare a 3D QSM dataset.
- Evaluate the feasibility of a deep learning model on the 3D QSM dataset focusing on reconstruction accuracy and computational efficiency.
- Prepare a clinically relevant 2D dataset derived from the 3D QSM volumes.
- Train and evaluate the model on the 2D dataset to reduce computational costs while maintaining accuracy.

This thesis outlines the steps taken towards achieving these objectives, highlights the challenges encountered during model training, and discusses the future steps necessary to enhance the performance and applicability of the QSM reconstruction model.

Chapter 2 reviews existing literature on traditional iterative methods and deep learning approaches for QSM reconstruction. It discusses the role of magnetic anisotropy and compares 3D and 2D reconstruction techniques.

Chapter 3 outlines the methodology used in this research, including the acquisition of raw MRI data, preprocessing steps such as phase image processing, brain mask extraction, and the formation of 2D and 3D datasets for model training. It also covers the model architecture and training systems.

Chapter 4 presents the results of the experiments conducted, highlighting the performance of both the MoDL-QSM and Transformer-based models. The evaluation focuses on reconstruction accuracy, computational efficiency, and the comparison with the SwinUNet2D architecture.

Chapter 5 summarizes the findings, assesses the issues encountered during model training, and discusses potential future directions including integrating the model into clinical workflows.

Chapter 2

Literature Review

Quantitative Susceptibility Mapping (QSM) reconstruction has evolved significantly, transitioning from traditional iterative methods to deep learning-based approaches. Iterative techniques, while effective, suffer from slow convergence, sensitivity to artifacts, and dependence on manually tuned parameters, limiting their practicality in clinical settings. In contrast, deep learning models offer a data-driven solution, improving reconstruction speed, reducing artifacts, and enhancing generalization across datasets. Among these, supervised learning has shown the highest reliability due to its access to labeled training data, enabling more accurate and consistent reconstructions. Recently, transformer-based architectures have emerged in medical imaging, demonstrating superior performance in capturing long-range dependencies. While not yet widely applied to QSM, their potential for improving reconstruction quality makes them an important area for future exploration. The following sections provide a detailed discussion of these methods and their impact on QSM reconstruction.

2.1 Traditional Iterative Methods for Reconstruction

Quantitative Susceptibility Mapping (QSM) reconstruction is an ill-posed inverse problem, requiring additional constraints to obtain accurate susceptibility estimates. Traditional methods rely on iterative optimization techniques to approximate solutions. Some widely used approaches include Truncated K-space Division (TKD) is one of the earliest approaches used in QSM reconstruction. It provides a simple and computationally efficient solution by performing direct inversion in k-space. However, TKD is known to suffer from susceptibility underestimation and streaking artifacts, particularly in regions with large susceptibility gradients [11]. Iterative Least Squares (iLSQR) improves upon direct inversion methods by employing an iterative solver to estimate the susceptibility map. This approach enhances reconstruction accuracy, but it comes at the cost of high computational expense and slow convergence, making it impractical for real-time clinical applications [8]. Morphology Enabled Dipole Inversion (MEDI) introduces anatomical priors from magnitude images to regularize the solution, thereby preserving structural consistency and improving reconstruction quality [9]. However, MEDI requires careful tuning of regularization parameters, which can vary across different datasets and acquisition protocols, limiting its adaptability in diverse clinical settings. STAR-QSM attempts to mitigate streaking artifacts by incorporating additional constraints in the inversion process. While effective in many cases, it still struggles in low-signal regions and remains sensitive to parameter selection [10].

Despite their widespread use, all these iterative methods share common limitations, including slow convergence, dependency on regularization parameters, and susceptibility to noise, which necessitate the exploration of alternative data-driven approaches.

2.2 Deep Learning-Based QSM Reconstruction

Deep learning offers a data-driven solution for QSM reconstruction by learning direct mappings between input phase images and susceptibility maps. Compared to iterative methods, deep learning reduces computational cost by eliminating the need for iterative solvers, making the reconstruction process significantly faster. Additionally, it generalizes better across different datasets, improving robustness and ensuring consistent performance across various imaging protocols. By leveraging spatial dependencies and learning noise characteristics, deep learning methods are also effective in minimizing artifacts, leading to higher-quality susceptibility maps with enhanced clinical applicability. Deep learning approaches for QSM reconstruction can be discussed under various architectural approaches as follows.

2.2.1 Conventional model Architectures

Among many deep learning models mostly consisting of convolutional neural networks (CNNs), U-Net has been one of the most frequently employed architectures due to its encoder-decoder structure with skip connections. This design effectively preserves fine details through multi-scale feature extraction, helping retain anatomical information while reducing artifacts. However, U-Net is not the only CNN-based approach used in QSM reconstruction, and several alternative architectures have been explored to address its limitations.

One such model, QSMGAN [12], integrates a generative adversarial network (GAN) with a U-Net backbone, enhancing the perceptual quality of susceptibility maps and reducing blurring artifacts. QSMnet [13], a widely used CNN-based model, employs deep learning to learn the QSM dipole inversion from paired training data, improving reconstruction accuracy while maintaining structural consistency. IR2QSM [14] introduces iterative reverse concatenations and recurrent modules within a CNN framework, optimizing latent feature utilization for better reconstruction accuracy.

Beyond these architectures, other CNN-based models have been designed to improve specific aspects of QSM reconstruction. xQSM [15] incorporates octave convolutional layers to process high- and low-frequency components separately, enhancing contrast and suppressing artifacts in low-signal regions. Meanwhile, DIAM-CNN [2] introduces multi-channel convolutional processing to adaptively handle dipole inversion, improving susceptibility estimation accuracy.

Despite their advancements, CNN-based models have inherent limitations. Their reliance on convolutional kernels restricts their ability to capture long-range dependencies, making them less effective in modeling global spatial structures crucial for accurate susceptibility estimation. Additionally, CNNs possess inductive biases that limit their adaptability to highly variable QSM datasets, requiring extensive fine-tuning for different acquisition settings. While these architectures have contributed significantly to QSM reconstruction, they remain constrained by their local receptive fields and dependence on pre-defined network structures.

2.2.2 Physics-Guided Architectures

To enhance QSM reconstruction, deep learning models have been integrated with physics-based constraints, leveraging domain knowledge to improve accuracy and robustness. These constraints make this kind of model very reliable in terms of the output generated.

Model-Based Deep Learning (MoDL) [16] introduces iterative updates within the network to enforce MRI physics constraints during training, ensuring that the learned susceptibility representations remain physically consistent. Multiple CNN networks are used to learn various parameters of the equations that represent the physical model of reconstruction. Another such unrolled architecture, Learned Proximal Convolutional Neural Networks (LPCNN) [17] incorporates proximal gradient descent principles alongside deep learning techniques, aiming to improve optimization efficiency. Variational Dipole Inversion Network (VaDNI) [18] follows a similar approach by formulating QSM reconstruction as a variational optimization problem and incorporating the dipole forward model directly into the learning process.

Despite their advantages, physics-guided deep learning models have some limitations. These architectures often require iterative optimization, increasing computational cost and slowing down inference. Most significantly, they struggle with capturing long-range spatial dependencies, as their design primarily focuses on local feature extraction. Furthermore, models like MoDL, LPCNN, and VaDNI require the dipole kernel as an explicit input, introducing additional dependencies on precomputed fields and limiting flexibility when dealing with different acquisition protocols. These constraints make physics-guided approaches less scalable and more computationally demanding, limiting their applicability in certain real-world scenarios.

2.2.3 Unsupervised Learning based Architectures

Unsupervised learning methods have been explored in QSM reconstruction to infer susceptibility maps without requiring explicit phase-QSM ground truth pairs. These models leverage self-supervised learning, adversarial training, or implicit neural representations to improve generalization across different datasets while reducing dependency on labeled data.

One such model, CycleQSM [19], applies a physics-informed CycleGAN to learn QSM reconstruction from unpaired data, ensuring domain adaptation across different imaging settings. INR-QSM [20] utilizes implicit neural representations to model susceptibility maps as continuous functions of spatial coordinates, optimizing directly on raw phase data without the need for paired training. QSMDiff [21], a diffusion-based unsupervised model, leverages patch-wise training to enhance robustness across varying scan parameters and improve generalization.

Although these models address limitations of supervised learning, they introduce new challenges. Their reliance on adversarial or implicit representations can lead to unstable training and unpredictable outputs, reducing reliability in clinical applications. Additionally, unsupervised approaches struggle with quantitative accuracy, as they lack direct supervision to ensure alignment with true susceptibility distributions [22]. While these methods contribute to QSM reconstruction, their clinical viability remains limited due to these inherent drawbacks.

2.2.4 Transformer-Based Architectures

Recent studies in medical imaging have demonstrated the effectiveness of transformers in image reconstruction tasks, particularly for MRI super-resolution and image denoising. Unlike convolutional neural networks (CNNs), which rely on local receptive fields, transformers utilize self-attention mechanisms to capture long-range dependencies, making them well-suited for tasks that require spatially aware representations [23]. This ability to model global context improves generalization making the models adaptable to variations in acquisition parameters.

Transformers have been successfully applied in computed tomography (CT) [24] to improve volumetric reconstruction from limited projections. Meanwhile, in positron emission tomography (PET) reconstruction. Diffusion Transformer models have demonstrated superior performance in low-dose imaging [25]. Transformers have also been successfully applied to MRI segmentation tasks. For instance, the TransUNet [26] model integrates transformer modules with a U-Net architecture to enhance 2D medical image segmentation. In 3D applications, the Swin UNETR [27] model employs a Swin Transformer as the encoder for volumetric segmentation of medical images, successfully capturing complex anatomical structures.

Given their effectiveness in related medical imaging tasks, transformers present a promising direction for QSM reconstruction. For this work, the Swin Transformer has been selected due to its hierarchical feature representation, efficient self-attention mechanism, and shift-based windowing approach, which collectively enhance both computational efficiency and spatial modeling. Unlike standard Vision Transformers (ViTs) [28], which apply global self-attention to the entire image and scale poorly with increasing resolution, the Swin Transformer [29] employs a hierarchical design with patch merging, reducing computational complexity while preserving critical spatial dependencies. Furthermore, its shifted window attention enables localized context refinement while maintaining the ability to capture long-range interactions, making it particularly suited for QSM, where susceptibility variations must be modeled at both fine and global scales.

While Swin Transformers provide excellent performance, a hybrid CNN-Transformer model combining local feature sensitivity from convolutions with global context modeling from transformers, ensures accurate fine anatomical details. This hybrid approach has been successfully demonstrated in medical image segmentation, where models like Swin Unet3D [30] combine CNNs and Transformers in parallel in the feature extraction module to enhance performance.

Looking forward for future work, incorporating GAN-based refinement techniques [31] could significantly improve the quality of susceptibility maps by enhancing the local consistency of the susceptibility distribution. As a post-processing step, GANs would learn the residuals of the transformer output and reduce artifacts by enforcing realistic spatial coherence. By acting as a dynamic loss function, GANs optimize the model through adversarial training, ultimately refining the transformer-based reconstructions of the susceptibility maps.

2.3 Magnetic Anisotropy

Magnetic anisotropy refers to the directional dependence of magnetic susceptibility, meaning that a material's susceptibility varies depending on the orientation of its structure relative to the applied magnetic field. This phenomenon is particularly significant in biological tissues, such as the white matter in the brain, where the highly organized myelin sheaths introduce anisotropic susceptibility effects. Unlike isotropic susceptibility, which assumes uniform magnetic properties in all directions, anisotropic susceptibility accounts for the structural alignment of tissue components, making it a crucial factor in accurate QSM reconstruction.

Traditional QSM techniques often assume isotropic susceptibility, which can introduce errors in susceptibility estimation, particularly in regions with highly ordered structures. In white matter, susceptibility varies as a function of fiber orientation relative to the main magnetic field direction (B_0), leading to systematic errors when anisotropic effects are ignored. Failing to account for anisotropic susceptibility can result in:

- Distortions in susceptibility maps, particularly in fiber-rich areas.

- Overestimation or underestimation of susceptibility values due to orientation-dependent phase shifts.
- Inconsistencies across acquisition protocols, making relative comparisons unreliable.

Several physics-based methods have been developed to mitigate these issues. Anisotropic Susceptibility Tensor Imaging (aSTI) [32] explicitly models susceptibility anisotropy using multiple orientation acquisitions, providing a voxel-wise susceptibility tensor to represent direction-dependent variations. However, aSTI requires acquiring multiple QSM scans at different orientations, significantly increasing scan time and limiting its feasibility in clinical settings.

Deep learning models have the potential to address anisotropic susceptibility estimation without requiring multiple orientations by leveraging data-driven representations. However, most existing deep learning-based QSM reconstruction methods are trained under the assumption of isotropic susceptibility, which limits their ability to generalize in anisotropic tissue regions. Potential future directions for pure deep learning mode include Developing anisotropy-aware deep learning models that integrate fiber orientation-dependent priors.

2.4 Comparison between 3D and 2D Reconstruction

Quantitative Susceptibility Mapping (QSM) is traditionally performed using three-dimensional (3D) gradient-recalled echo (GRE) sequences, capturing comprehensive spatial susceptibility variations across tissues [33]. However, 3D QSM acquisition and processing are computationally demanding, often requiring high-end hardware to handle large datasets and complex reconstruction algorithms. With the advancement of rapid imaging techniques, a growing interest has emerged in developing two-dimensional (2D) QSM techniques, particularly for applications such as functional MRI (fMRI), where fast acquisition and real-time processing are crucial [34]. While QSM is being developed for 2D applications, most available datasets are inherently 3D, making it necessary to adapt current methodologies rather than rely solely on dedicated 2D acquisitions. Various aspects can be discussed to draw a comparison.

2.4.1 Computational Efficiency and Practicality

Processing 3D MRI data demands substantial computational resources, which can limit the feasibility of complex models, especially in settings with constrained hardware capabilities. Since dipole inversion is inherently local and can be performed per slice, adaptation to 2D allows for significant reductions in computational cost while retaining the integrity of phase information. A single 3D QSM scan provides only a limited number of volumetric samples, but when sliced into 2D images, it generates a substantially larger dataset, enriching training diversity. This is particularly advantageous for transformer-based architectures, which thrive on large datasets to extract complex spatial dependencies and improve reconstruction performance [28] [29]. While transformers are often associated with high data requirements, this shift to a vast 2D dataset from 3D acquisitions aligns perfectly with their strengths, enabling better generalization, robustness, and feature learning across a diverse range of slices without necessitating high-end computational infrastructure.

2.4.2 Alignment with Clinical Practices

In clinical scenarios, radiologists often interpret MRI scans on a slice-by-slice basis, focusing on individual 2D images to assess specific anatomical structures or pathologies [35]. This practice is particularly prevalent in modalities like fMRI, where rapid acquisition techniques such as echo-planar imaging (EPI) inherently produce 2D slices [36]. Aligning computational models with this practice by utilizing 2D slices can enhance the interpretability of the results, facilitating seamless integration into clinical workflows.

2.4.3 Challenges and Considerations

Transitioning from 3D to 2D QSM introduces certain challenges, primarily related to the loss of through-plane information. When switching to 2D processing, inter-slice continuity may be disrupted, leading to potential inconsistencies in reconstructed susceptibility maps. Although this thesis primarily focuses on 2D QSM reconstruction, future work will extend this research to explore 3D QSM reconstruction using transformer-based models. While 2D slices provide a practical and computationally efficient solution, transformers have shown great promise in capturing long-range dependencies and volumetric structures when applied to 3D data [30] as well. 3D transformers thus have potential to enhance susceptibility evaluation while retaining deep learning benefits despite heavier computation cost.

Chapter 3

Methodology

3.1 Data acquisition

For this study, a MRI scan dataset consisting of 8 subjects was utilized, that provide high-quality gradient-recalled echo (GRE) quantitative susceptibility mapping (QSM) acquisitions.

3.1.1 Dataset: A Multi-Orientation Gradeint Echo MRI Dataset

This dataset, introduced by Shi et al. [37], provides a comprehensive multi-orientation GRE acquisition designed to serve as a benchmark for deep learning-based QSM methods. The dataset consists of 144 scans collected from 8 healthy subjects using a 3D gradient echo (GRE) sequence. The scans were acquired with isotropic voxel sizes of 1 x 1 x 1 mm, ensuring uniformity across all datasets. The imaging protocol includes a field of view (FOV) of 210 x 224 x 160 mm^3, a matrix size of 210 x 224 x 160, and multiple echo times (TE) ranging from 7.7 ms to 38.2 ms. A flip angle of 20° and a bandwidth of 190 Hz/pixel were used for data acquisition.

The raw data includes phase images, magnitude images acquired in multiple orientations, and corresponding COSMOS-labeled and chi33-labeled susceptibility maps for the supine position. These reference susceptibility maps were used for training the model, providing ground truth for the deep learning-based QSM reconstruction. The dataset also includes deep gray matter (DGM) parcellations, which allow for the extraction of regional susceptibility values for quantitative accuracy evaluation of different QSM reconstruction methods.

In this work, all the data were preprocessed in 3D before being sliced into 2D images for model training. This preprocessing step was essential to maintain the integrity of the QSM phase processing while increasing the dataset size. The 2D slices were used to augment the data and ensure better generalization during model training.

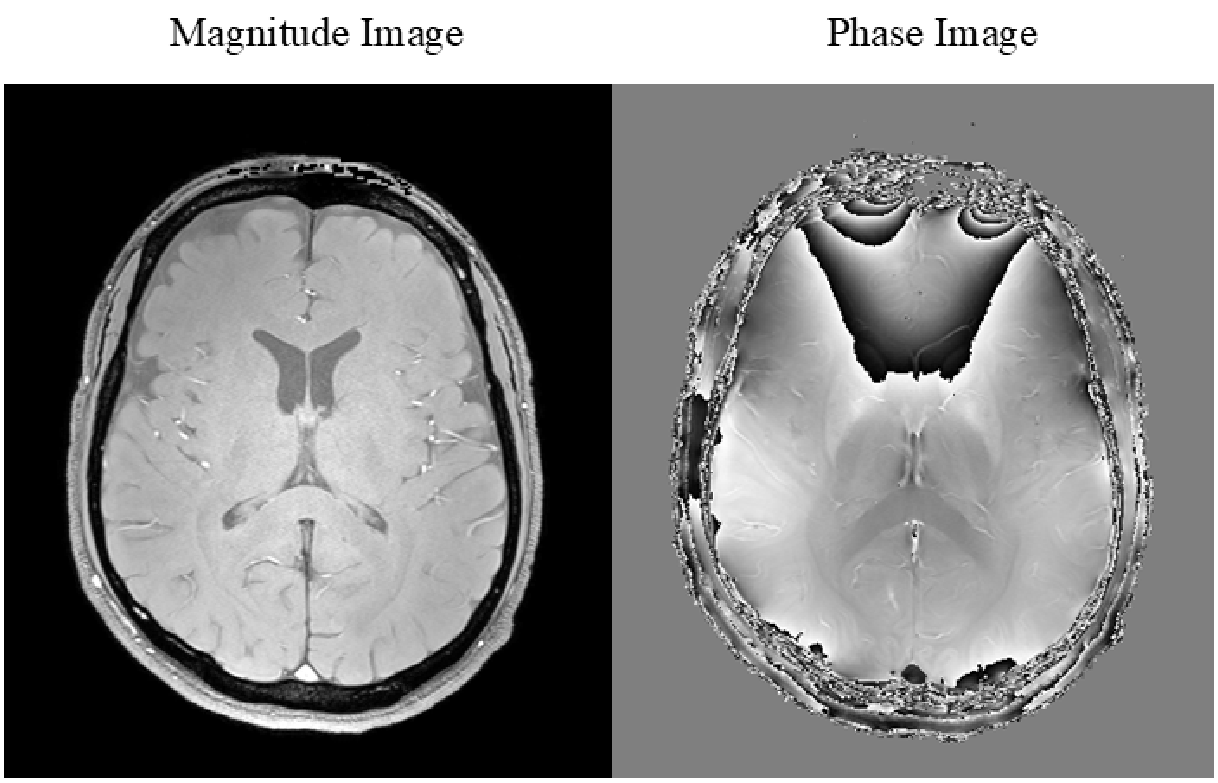

Figure 3.1: Magnitude and Phase Images Extracted from Raw Scans

3.2 Phase Image and Susceptibility Map Processing

This section outlines the processing pipeline applied to the datasets utilized in this study, encompassing phase image extraction, brain mask generation, phase unwrapping, background field removal, and susceptibility map reconstruction. The processing steps were tailored to the characteristics of each dataset to get the best results.

3.2.1 Image Extraction from Raw Scans

The initial step involved extracting phase and magnitude images from the raw gradient-recalled echo (GRE) MRI scans shown in figure 3.1. Each scan produced twice as many images as echo times.

3.2.2 Brain Mask Extraction

Accurate brain mask extraction is crucial for isolating brain tissue from non-brain elements, thereby enhancing the precision of subsequent analyses as shown in 3.2. For this purpose, the Brain Extraction Tool (BET) was employed, which delineates the brain's contour based on the intensity histogram of the MRI magnitude data [38].

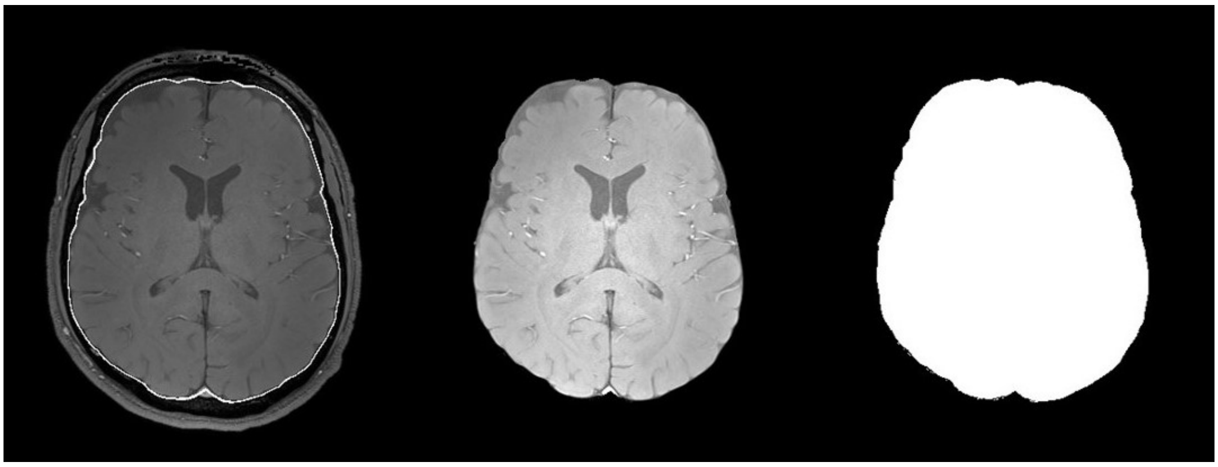

Figure 3.2: Brain Mask Extracted from Magnitude Images

3.3 Image Registration

In this step, we perform image registration to align MRI scans that were acquired in different orientations. This process transforms the images into a common frame of reference, ensuring that all images correspond to the same anatomical space. The registration involves scaling, rotation, translation, and shearing to match the images properly. The transformation is represented using an affine transformation matrix A, which is a 4x4 matrix that captures these operations. The affine transformation algorithm is applied as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here, the 3x3 submatrix R:

$$R = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

represents the **rotation matrix**, which performs the rotation and scaling of the image. The translation along the x, y, and z axes is represented by t_x, t_y, t_z, respectively. The magnitude images are used to acquire the registration matrices, as they typically contain clearer structural details that make them suitable for alignment. Once the registration matrices are computed, they are applied to align the phase images accordingly. Using the affine transformation algorithm, the matrix A is applied to the coordinates of a point in the original image (x, y, z) to obtain the transformed coordinates (x', y', z'):

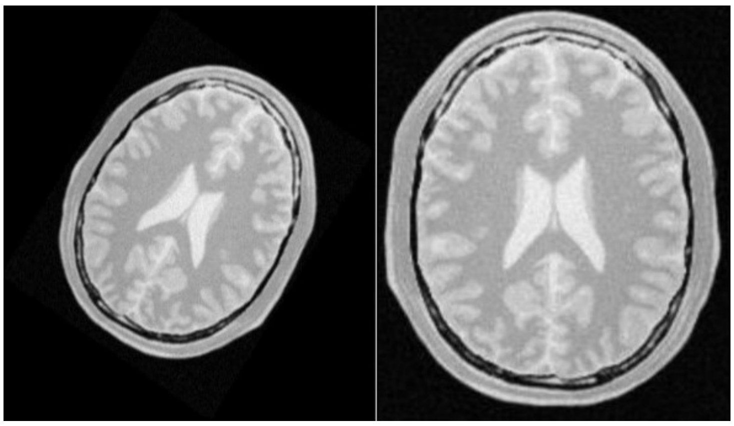
Figure 3.3: Example of Transformation Done by Applying Registration Matrix

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

Images acquired before and after transformation is shown in figure 3.3. This affine registration ensures that all images of the same subject are aligned accurately before moving on to the phase unwrapping and background phase removal steps.

3.3.1 Phase Unwrapping

Phase unwrapping addresses the issue of phase aliasing in MRI data, ensuring that the phase values accurately represent the underlying magnetic field variations. The Laplacian-based phase unwrapping method was applied as in figure 3.4, which effectively reconstructs the true phase by solving the Poisson equation, thereby mitigating phase discontinuities [39].

3.3.2 Background Field Removal

To isolate the local tissue-induced magnetic field from external sources, background field removal was performed. The Variable-kernel Sophisticated Harmonic Artifact Reduction for Phase data (V-SHARP) method was utilized as shown in 3.5, which employs variable spherical mean value kernels to effectively eliminate background fields [40]. However, a known limitation of V-SHARP is that it tends to erode brain regions during processing [41].

To mitigate this issue, additional morphological processing, binary erosion, was performed on the original extracted brain mask to generate a refined version that compensates for the tissue loss introduced by V-SHARP algorithm. Erosion by 1, 2, and 3 voxels were tested to deter-

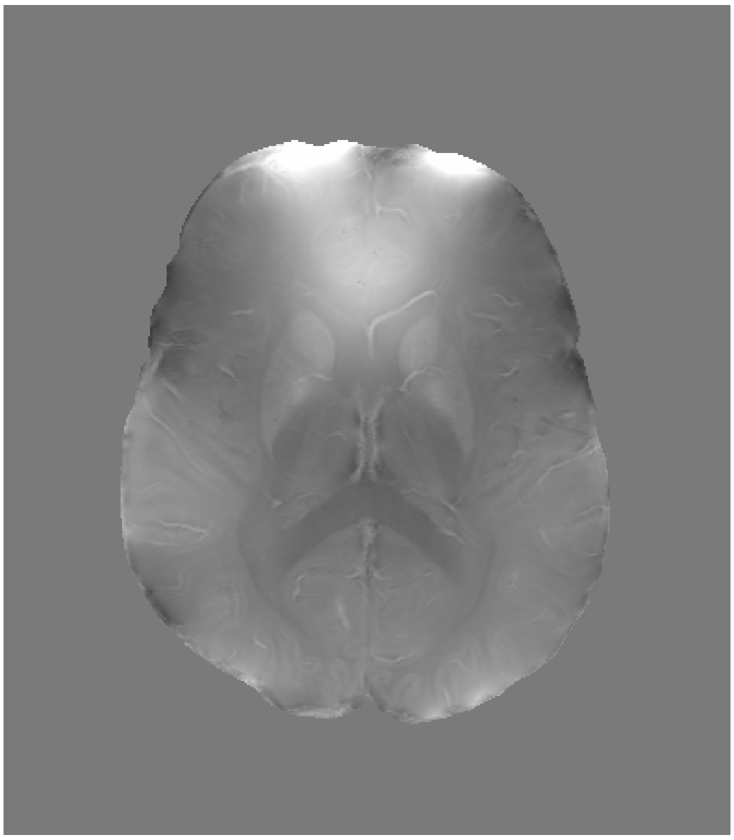

Figure 3.4: Phase Unwrapped Image using Laplacian Phase Unwrapping

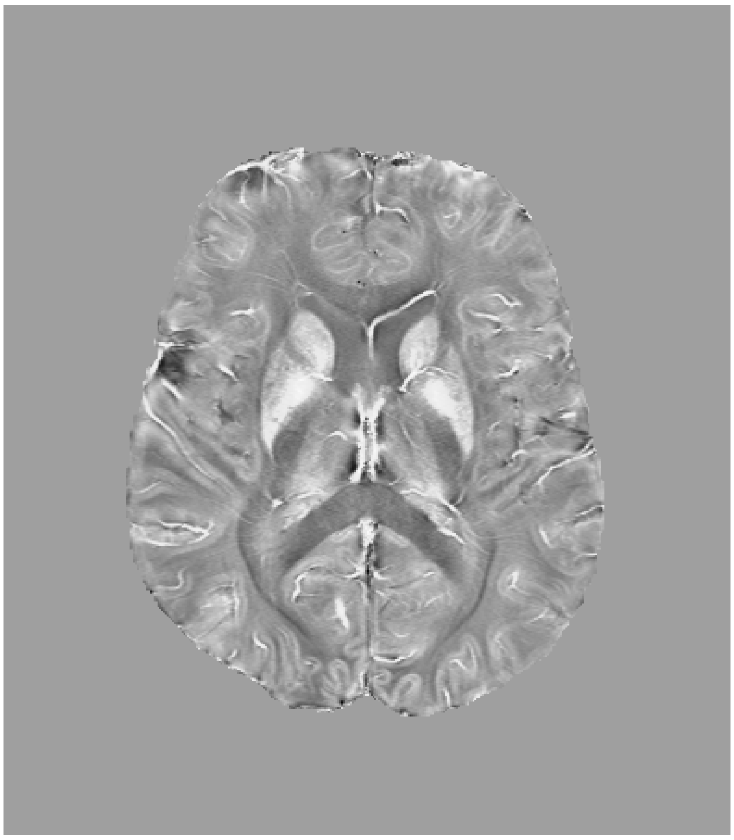

Figure 3.5: Background Phase Removed Image using VSHARP

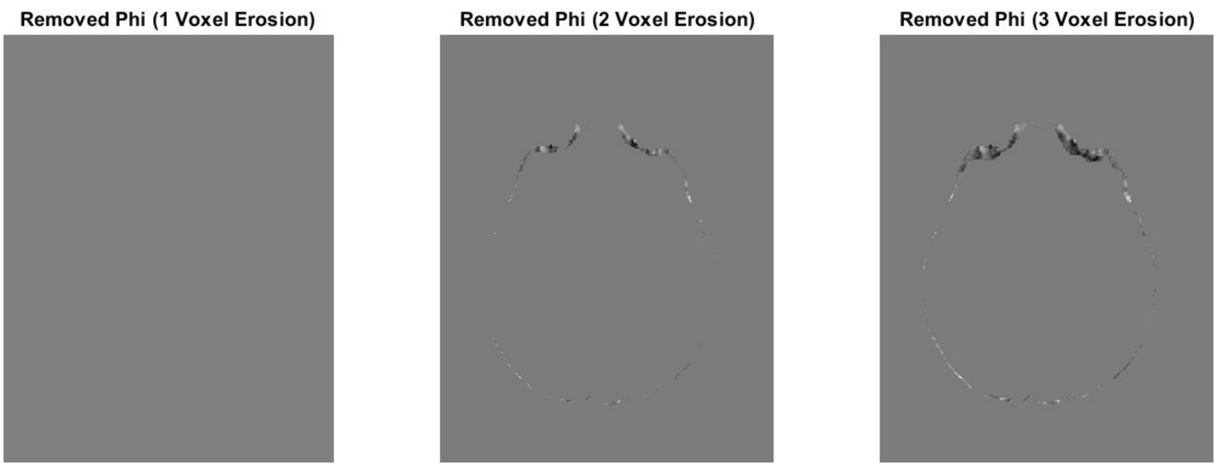
Removed Phi (1 Voxel Erosion)
Removed Phi (2 Voxel Erosion)
Removed Phi (3 Voxel Erosion)

Figure 3.6: Phase Image Erosion Map

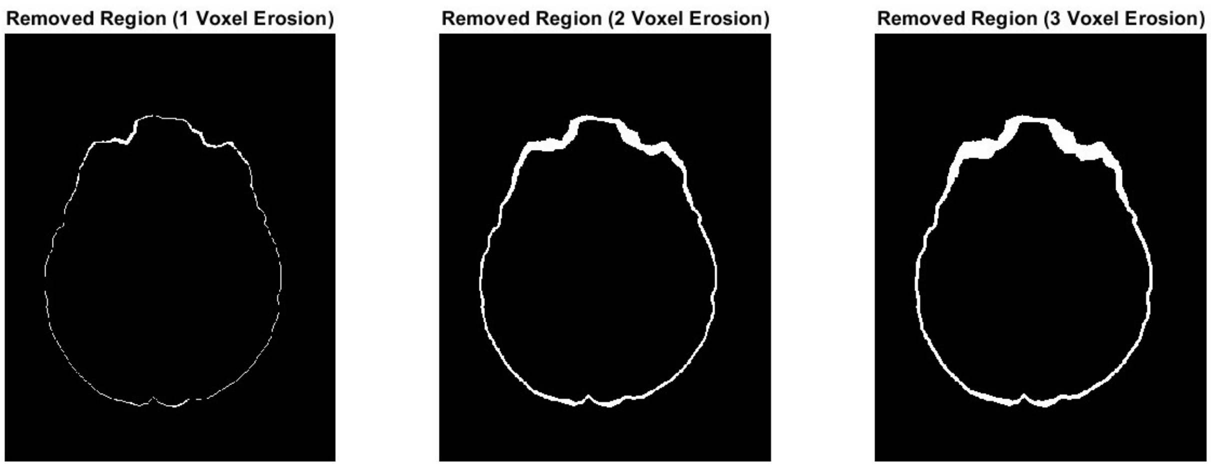
Removed Region (1 Voxel Erosion)
Removed Region (2 Voxel Erosion)
Removed Region (3 Voxel Erosion)

Figure 3.7: Brain Mask Erosion Map

mine the optimal configuration. A 3-voxel erosion was found to be too aggressive, leading to excessive removal of brain regions, while a 1-voxel erosion was insufficient to account for the changes induced by V-SHARP shown in figure 3.6. Consequently, a 2-voxel erosion was selected as the optimal choice, balancing the need for sufficient background removal without excessive tissue loss. This refined brain mask was then used for subsequent processing steps. The brain masks are shown in figure 3.7

3.3.3 Normalizing over Multiple Echoes

To obtain an MRI scan, radiofrequency (RF) pulses are first applied to the tissues. These pulses excite the hydrogen nuclei in the body, causing them to emit complex tissue signals, which consist of both magnitude and phase information. The echo time (TE) is the time interval between the application of the RF pulse and the peak of the received signal. Multiple echo times capture the phase changed at different times between two consecutive RF pulse as shown in figure 3.8. Shorter echo times give clearer magnitude images. Longer echo times give more

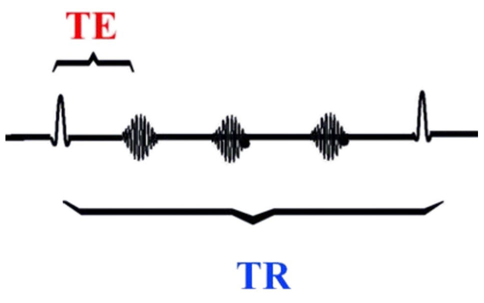

Figure 3.8: Radio Frequency (RF) pulse and Echo Time (TE) Diagram

detailed phase image.

In this study, the field maps from multiple echoes were normalized to account for variations in echo time. After background field removal using the V-SHARP method, each echo's field map was normalized by the factor $2\pi\gamma\text{TE} \cdot B_0$, where γ is the gyromagnetic ratio, TE is the echo time for the respective echo, and B_0 is the magnetic field strength. This normalization ensures that the field maps from each echo are comparable despite differences in echo times.

Subsequently, the normalized field maps were averaged across the different echoes to create a single, combined field map. Figure 3.9 present such five representative echo images and the combined phase map. This combined field map represents a more accurate and stable field estimate by leveraging information from multiple echoes.

3.3.4 Susceptibility Map Reconstruction

In Quantitative Susceptibility Mapping (QSM), the magnetic susceptibility of tissues is often represented as a tensor, allowing for the characterization of directional variations in magnetic susceptibility. The susceptibility tensor χ describes the relationship between the applied magnetic field and the induced magnetization, and it can vary depending on the tissue structure and orientation.

In the case of isotropic susceptibility, the material's susceptibility is uniform in all directions. This can be represented by a simple scalar value, where the susceptibility is the same along all axes:

$$\chi_{\text{iso}} = \begin{bmatrix} \chi & 0 & 0 \\ 0 & \chi & 0 \\ 0 & 0 & \chi \end{bmatrix}$$

where χ is the isotropic susceptibility value.

For anisotropic susceptibility, the material's susceptibility varies depending on the orientation relative to the applied magnetic field. This is particularly relevant in tissues like white matter,

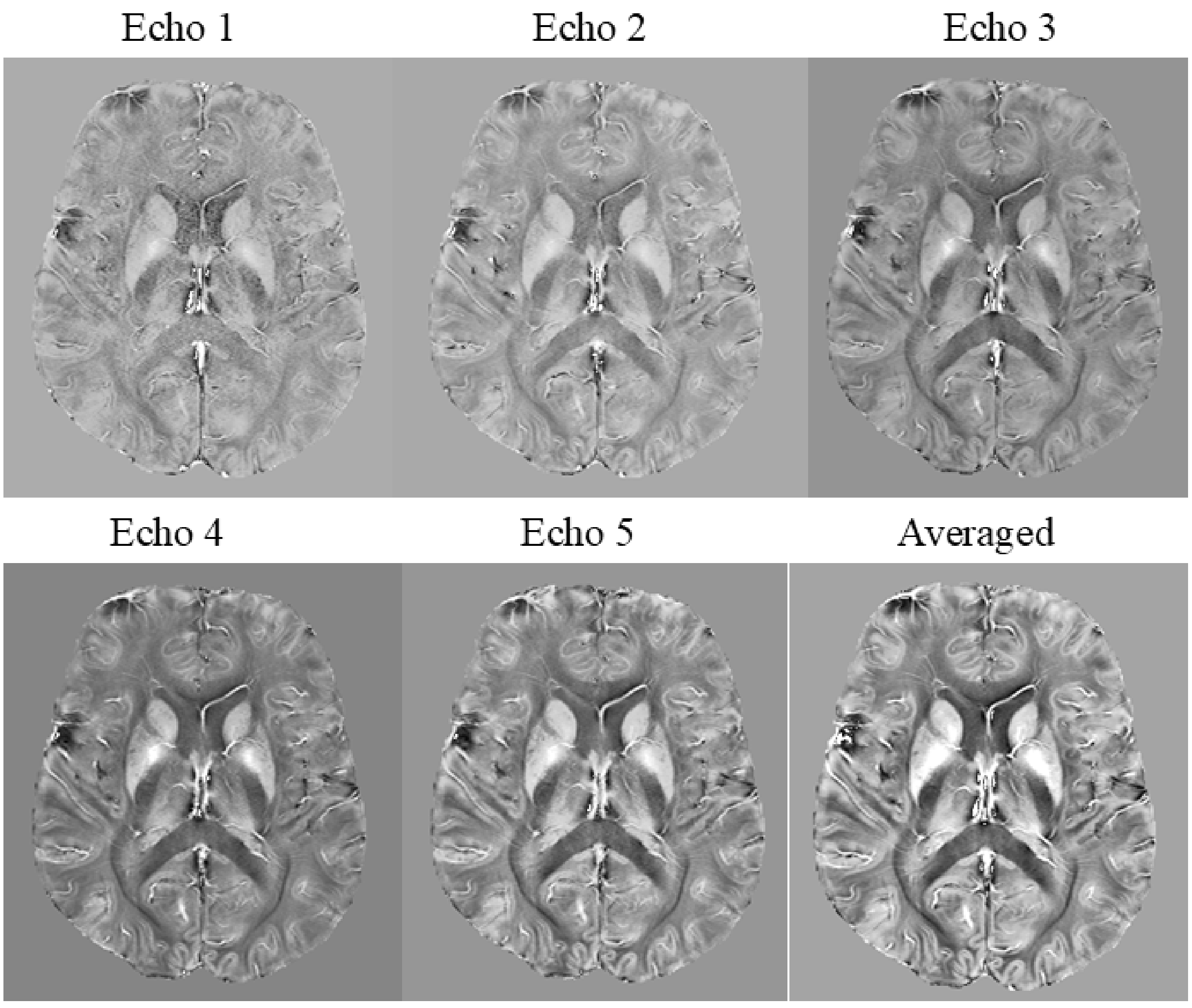

Figure 3.9: Phase Map Corresponding to Different Echo Times and the Final Phase Map

where the alignment of myelin sheaths causes direction-dependent variations. The susceptibility tensor in this case is represented as:

$$\chi_{\text{ani}} = \begin{bmatrix} \chi_{11} & \chi_{12} & \chi_{13} \\ \chi_{21} & \chi_{22} & \chi_{23} \\ \chi_{31} & \chi_{32} & \chi_{33} \end{bmatrix}$$

where χ_{11}, χ_{22}, and χ_{33} represent the diagonal components (typically aligned with the principal axes of the material), and the off-diagonal terms ($\chi_{12}, \chi_{13}, \chi_{23}$) account for the interaction between different directions.

For the anisotropic dataset used in this work, the target susceptibility map was chosen to be the χ_{33} component, which represents the susceptibility along the primary magnetic field direction (B_0). This component is particularly relevant in QSM, as it reflects the susceptibility contributions aligned with the main field, providing key information for tissue characterization in neuroimaging. The relationship of this component with phase map can be expressed as

$$\delta B = \mathcal{F}\left(\frac{1}{3} - \frac{(k_z)^2}{||k||^2}\right)\mathcal{F}\left[\chi_{33}\right] \tag{3.1}$$

Here, (δB) represents the magnetic field perturbation, $\mathcal{F}$ represents the Fourier transform, and k_z and $||k||$ are the components of the dipole kernel in the frequency domain. The χ_{33} component specifically corresponds to susceptibility along the B_0 direction, which is the key information needed for QSM reconstruction. The susceptibility tensor was computed using the Susceptibility Tensor Imaging (STI) [32] approach implemented in the STISuite toolbox [42].

Since a single orientation provides only a partial projection of the susceptibility distribution, multiple orientations were used to fully reconstruct the susceptibility tensor. Phase images were acquired at different head orientations relative to the scanner's B_0 field to mitigate the ill-posed nature of the inversion problem. However, before computing χ_{33}, all orientations had to be spatially aligned to a common reference frame. In this study, the supine position was chosen as the reference. The B_0 field direction was adjusted accordingly for each scan to maintain the correct directional relationship. The transformation matrix A and rotation matrix R acquired earlier in section 3.3 was used in this stage.

Once χ_{33} was computed, susceptibility maps were adjusted for anisotropy. Magnetic susceptibility in structured tissues like white matter is inherently anisotropic, meaning it varies based on the subject orientation [43] relative to B_0. To account for this correction, the reconstructed susceptibility maps were multiplied voxel by voxel with the rotation matrix R, derived from the registration process:

$$\chi_{adjusted} = R \cdot \chi_{33} \cdot R^T \tag{3.2}$$

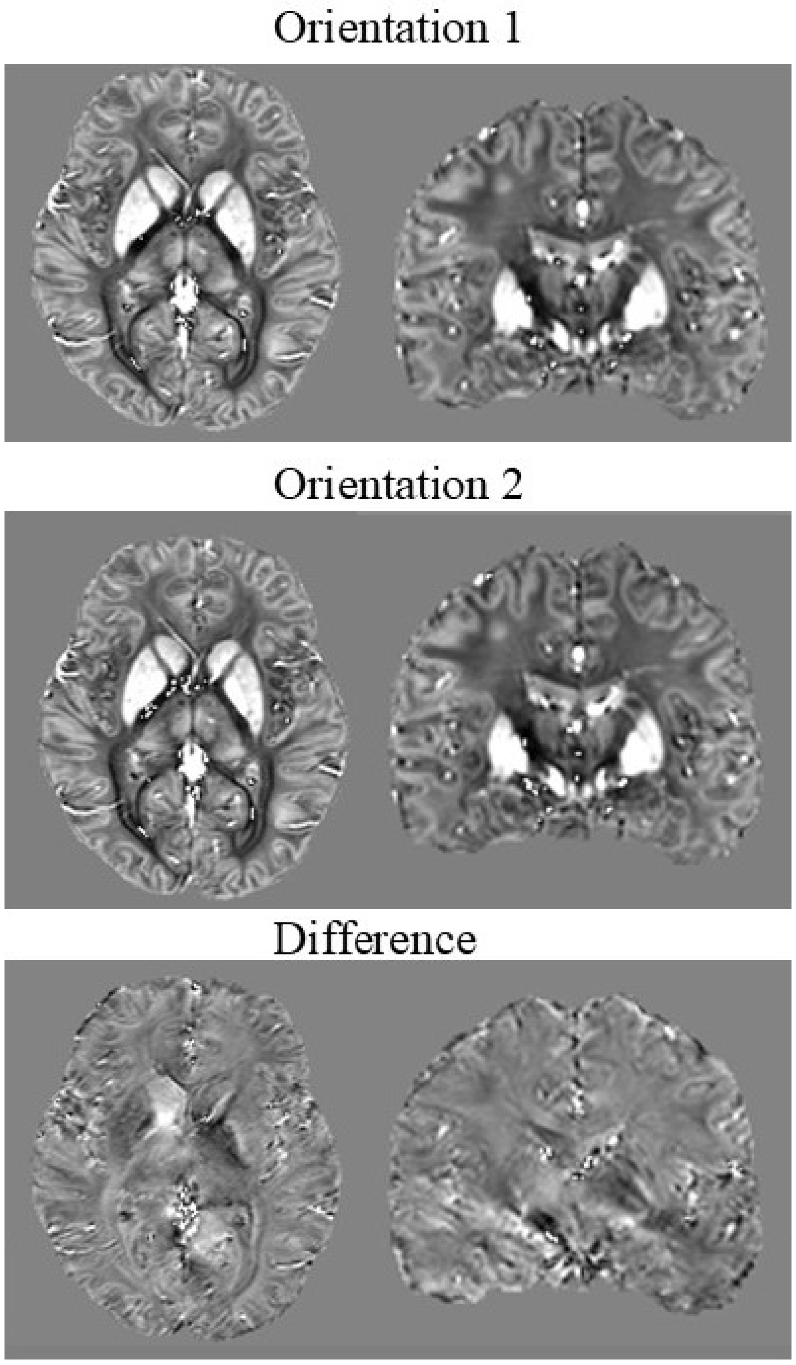

Figure 3.10: Susceptibility Map Gained from Different Orientation of the Same Subject

This transformation ensured that the susceptibility tensor was correctly aligned with the subject reference frame [32], preserving directional accuracy across different orientations.

3.4 2D dataset formation

The dataset used in this work consisted of multi-orientation scans from 8 healthy subjects, amounting to a total of 144 scans. These scans had isotropic voxel sizes of 1 x 1 x 1 mm, meaning no resizing was required.

For each scan, random slicing was performed along the x, y, and z axes as shown in Figure 3.11, followed by a cropping step to 128 x 128 pixels. To ensure data quality, patches with more than 70 to 80 percent invalid (zero in brain mask) regions were discarded. This step was essential for maintaining consistency in the batch sizes fed to the model and avoiding the inclusion of mostly invalid regions. A threshold of 3 to 5 was applied to avoid consecutive slices being selected, ensuring greater variability in the data. While padding could have been used to standardize batch sizes, this would have resulted in patches with excessive invalid regions, potentially reducing model performance. Since the transformer model downsamples data in multiple stages, excessive zero regions would lead to nearly all-zero values after downsampling, further harming model training.

Additionally, 40 percent of the slices were randomly flipped, either horizontally or vertically, to augment the dataset and eliminate model bias related to the particular orientation of the brain. The dataset was then split into a training set containing around 10,000 slices and a validation set consisting of 4,000 slices. During splitting, it was ensured that one subject did not contribute to both the training and validation datasets.

3.5 Model Architecture

This work employs two models for QSM reconstruction: the MoDL-QSM model for training on 3D data and the Swin Transformer model for training on 2D data. Both models aim to solve the dipole inversion problem for accurate QSM reconstruction but follow different approaches based on their architecture and training datasets.

3.5.1 MoDL-QSM Model for 3D reconstruction

The first model employed is based on the Model-based Deep Learning (MoDL) framework [16], which integrates a physical model into the deep learning architecture. The MoDL-QSM model follows the principles of Susceptibility Tensor Imaging (STI) and is designed to work with 3D

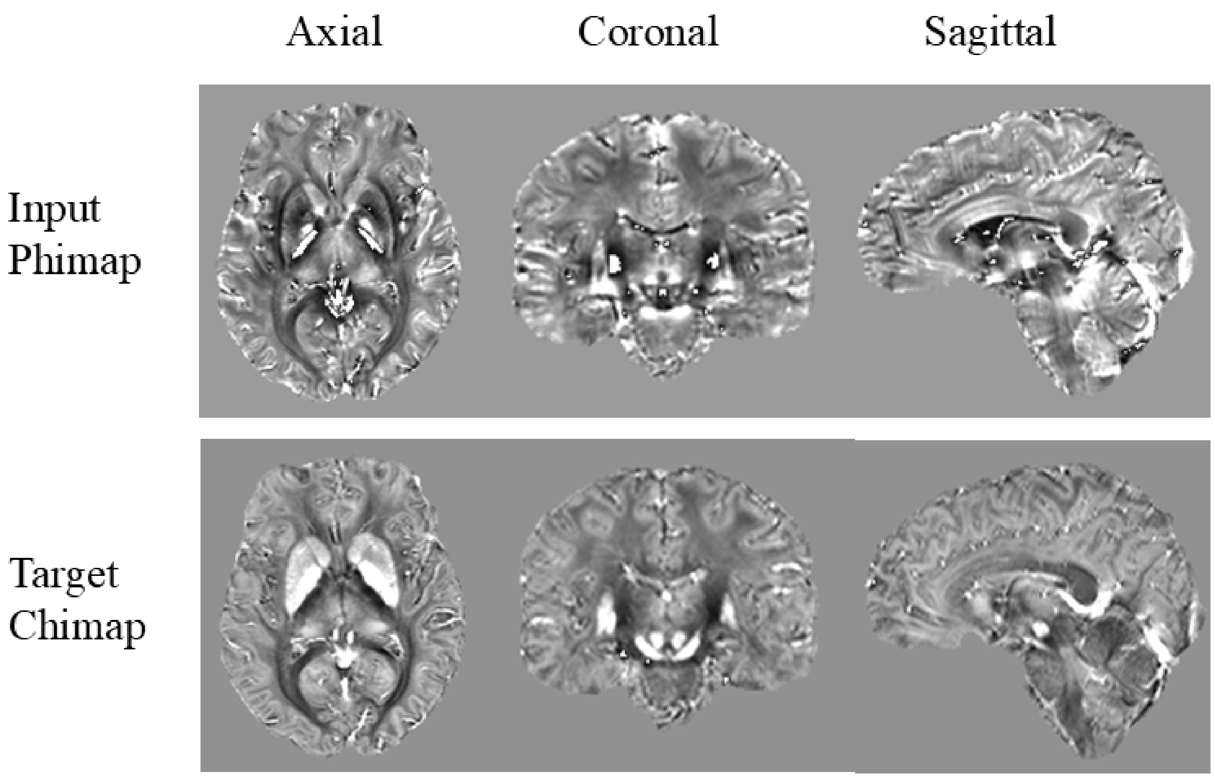

Figure 3.11: Slices Across X,Y and Z Axes

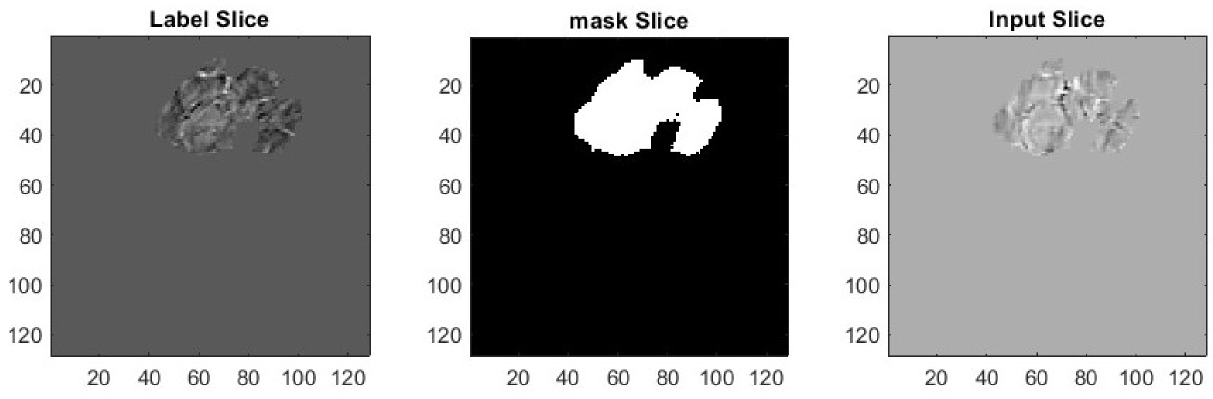

Figure 3.12: Example of an Invalid Slice

data. In this model, the relationship between the field perturbation induced by the susceptibility components and the acquired phase is learned through a deep convolutional neural network (CNN) embedded within a physical model.

The MoDL-QSM model is structured in iterations as evident in the figure 3.13 to, where each iteration refines the susceptibility map by incorporating both physical constraints and learned features. The network is unrolled into multiple iterations, with each CNN block learning the regularization term to minimize errors in the dipole inversion process. The physical model, consisting of the susceptibility tensor components, provides a regularization term that ensures the network's outputs remain consistent with the physical properties of the tissue.

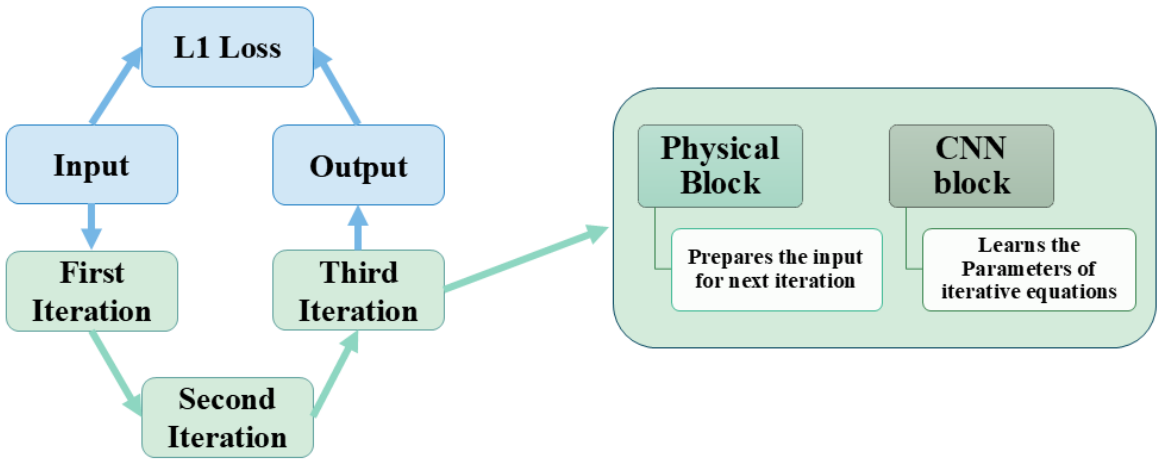

Figure 3.13: Flowchart of the MoDL-QSM Model Architecture. The input phase image undergoes three iterations in the network, with CNNs learning the regularization terms at each step. The final output consists of the susceptibility map (χ_{33}).

The model operates by minimizing the L1 loss between the output and the ground truth susceptibility map (χ_{33}), ensuring that the model preserves the anisotropic nature of susceptibility in tissues such as white matter. The model also uses the other two diagonal elements (χ_{11}) and (χ_{22}) but the evaluation of performance and preceding discussions will be limited to only (χ_{33}). The MoDL-QSM model operates on 3D data, which is crucial for capturing the full spatial context of the brain and ensuring accurate susceptibility estimations across all regions.

3.5.2 Transformer Model for 2D Reconstruction

In contrast, the second model, based on a Swin Transformer architecture, is designed for 2D QSM reconstruction. This model is specifically trained on 2D slices extracted from 3D scans to increase dataset size and allow for faster processing. The 2D slices are downsampled from the original 3D data, maintaining consistency in anatomical structure while enabling efficient training.

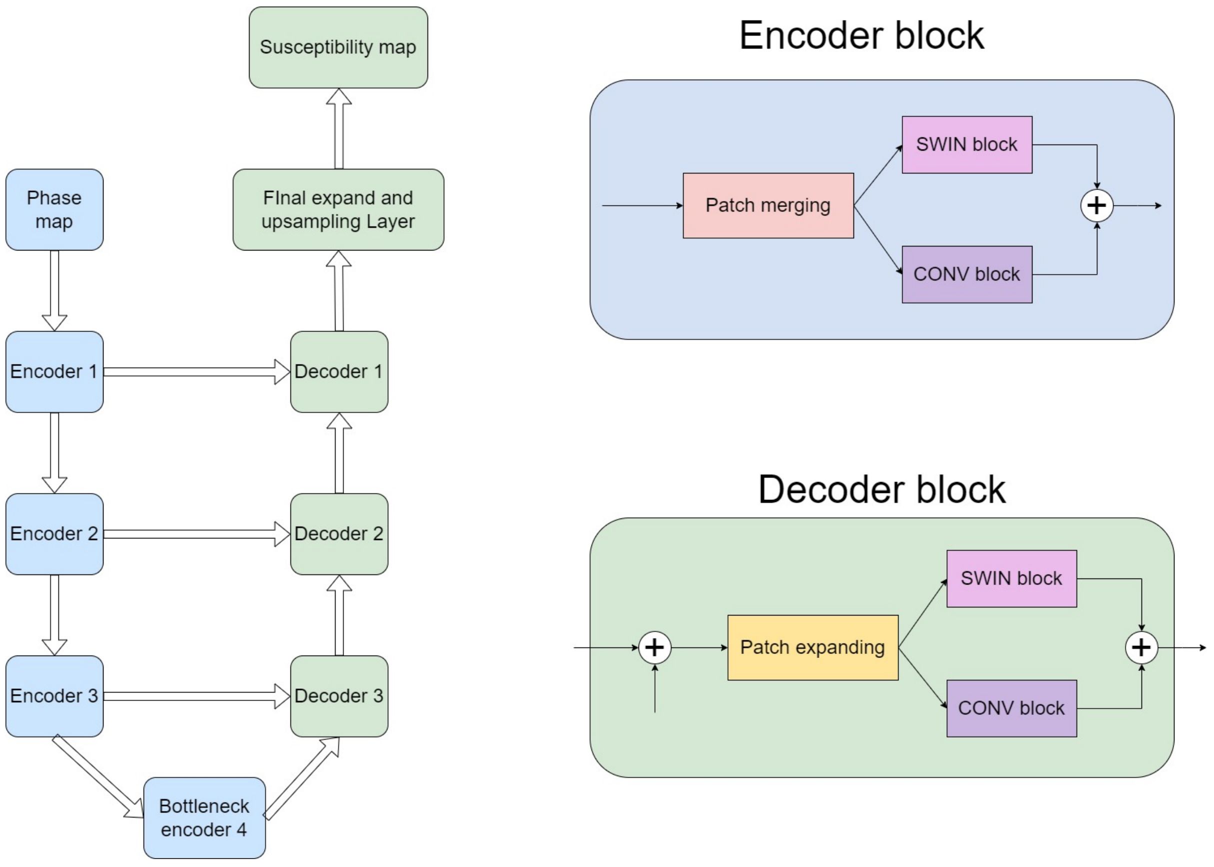

Figure 3.14: SwinUnet2D architecture with hybrid convolution blocks.

The Swin Transformer model utilizes self-attention mechanisms to capture long-range dependencies between different regions of the input image. The architecture utilized for QSM reconstruction is inspired from the Swin Transformer 3D UNet [30], originally developed for medical image segmentation tasks such as brain tumor segmentation. This model combines the strength of Swin Transformers with convolutional layers to capture both local and global dependencies effectively, ensuring high-quality reconstructions from MRI data. The model developed for training is in 3.14. The building blocks are described as follows.

Encoder

The encoder consists of 4 stages, with each stage progressively downsampling the input data. At each stage, the spatial resolution of the feature map is reduced by a downsampling factor, resulting in a total downsampling factor of 32× by the final stage after bottleneck. The encoder's primary function is to capture high-level features and long-range dependencies in the image using the Swin Transformer blocks. The window size used for self-attention within the Swin Transformer is (8, 8). In addition, the shifted window mechanism is employed [29], allowing for better interaction between neighboring windows and enabling the model to capture fine-grained details along with global dependencies.

Decoder The decoder consists of 3 stages, where each stage performs 2× upsampling to restore the spatial resolution of the feature maps. The decoder layers aim to progressively reconstruct the image with fine-grained details. The final stage of the decoder employs an additional 1×1 convolutional layer, which produces the final reconstructed susceptibility map.

Skip Connections

Skip connections are used between the corresponding encoder and decoder stages to preserve fine spatial details that may be lost during the downsampling process. These connections ensure that critical local features are maintained throughout the reconstruction process, enhancing the accuracy of the final susceptibility map.

Convolutional Blocks

In addition to the Swin Transformer blocks, convolutional blocks are used in parallel at each stage of the encoder and decoder. These CNN layers are responsible for capturing short-range local dependencies, enabling the model to refine the feature maps before they are processed by the Transformer layers. This hybrid approach ensures that both local fine details and long-range contextual information are effectively captured, optimizing the model for accurate image reconstruction.

3.6 Model Training Environment

The model training was carried out on Kaggle's Distributed GPU-based environment, which provided access to high-performance computing resources for deep learning tasks. However, a significant constraint was the 12-hour runtime limitation imposed by Kaggle kernels. This limitation required an efficient training strategy that allowed for progress within the restricted time window.

To handle this constraint, a checkpoint-based approach was implemented. At regular intervals during training, the model's weights and state were saved to checkpoints, ensuring that the training could be resumed from the last checkpoint if interrupted. This approach was crucial for completing the training process, as it allowed the model to continue its training from where it left off, without losing valuable progress due to the runtime restrictions.

Additionally, model performance was monitored throughout the training process using a validation set. By leveraging these techniques, including checkpointing and patch-based training, the models were successfully trained within Kaggle's environment, ensuring the ability to handle large datasets despite runtime limitations.

Chapter 4

Results and Discussion

4.1 MoDL Architecture

The performance of the MoDL-QSM model was evaluated based on two key metrics: the Root Mean Square Error (RMSE) and the Structural Similarity Index (SSIM). To better illustrate the model's output, the inference results are shown in Figure 4.1, providing a visual representation of the model's ability to reconstruct the susceptibility maps.

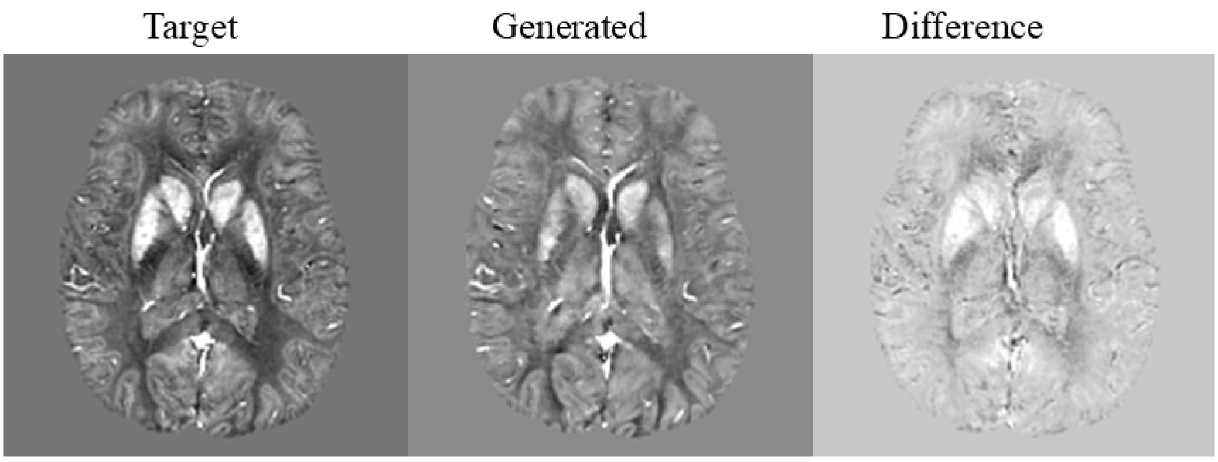

Figure 4.1: Inference result using MoDL QSM

As seen in Figure 4.1, the MoDL-QSM model generates reasonably accurate susceptibility maps. The difference with target is really minimal. Despite the complexity of the task, the model was able to recover meaningful patterns in the reconstructed maps.

Next, we compare the performance of our trained model with the results reported in the MoDL-QSM paper [16], based on RMSE and SSIM metrics in table 4.1. We also compared our results with few other models as mentioned in the paper. Despite the constraints of training time, GPU limitations, and a 12-hour runtime restriction on Kaggle, the MoDL-QSM model achieved

Model	**RMSE**	**SSIM**
MoDL-QSM (Training)	68.42	0.8306
MoDL-QSM (From Paper)	67.91	0.8446
AutoQSM	69.40	0.8549
QSMnet	55.07	0.8759
STAR-QSM	79.57	0.8180

Table 4.1: Comparison of RMSE and SSIM between the MoDL-QSM results from the training and the results from the MoDL-QSM paper. Also included are the RMSE and SSIM scores from other QSM methods.

results that were very close to those reported in the original paper. Specifically, the RMSE of our model was slightly higher at 68.42 compared to the 67.91 reported in the paper, and the SSIM was 0.8306, compared to 0.8446 in the paper. These results are promising, especially considering that we had to process the data ourselves and work within the limited training time on Kaggle. The RMSE and SSIM from other models, such as AutoQSM (69.40 and 0.8549) and QSMnet (55.07 and 0.8759), are also presented, demonstrating MoDL-QSM's competitive performance despite the training challenges.

Overall, the MoDL-QSM model demonstrated that it could be effectively trained even under constrained resources. The training took approximately 35 hours in total, utilizing a distributed data parallel system with GPUs, each featuring 16GB of VRAM. Due to memory limitations, the patch size was kept relatively small at only 48 x 48 x 48, which further constrained the available computational resources. Despite these limitations in training time, GPU capabilities, and dataset size, the model's performance remained close to the results reported in the original study. The slight differences in performance can be attributed to variations in data preprocessing and training conditions, but the MoDL-QSM model showed promising results within the given constraints.

4.2 SwinUNet2D Architecture

In this section, the results obtained from training the SwinUNet2D model on the given dataset are presented. The following figures and plots illustrate the model's performance, loss convergence over epochs, and the effectiveness of the chosen architecture. For the loss function, we utilized Mean Absolute Error (MAE or L1 loss) and combined it with 5% of the Structural Similarity Index (SSIM) loss to enhance image quality during training. Throughout the training process, we monitored the losses, compared performance metrics, and examined the model's behavior on both the training and validation datasets.

The plot in Figure 4.2 illustrates the training and validation losses over 10 epochs. The blue line represents the total validation loss, while the orange line shows the total training loss. Notably,

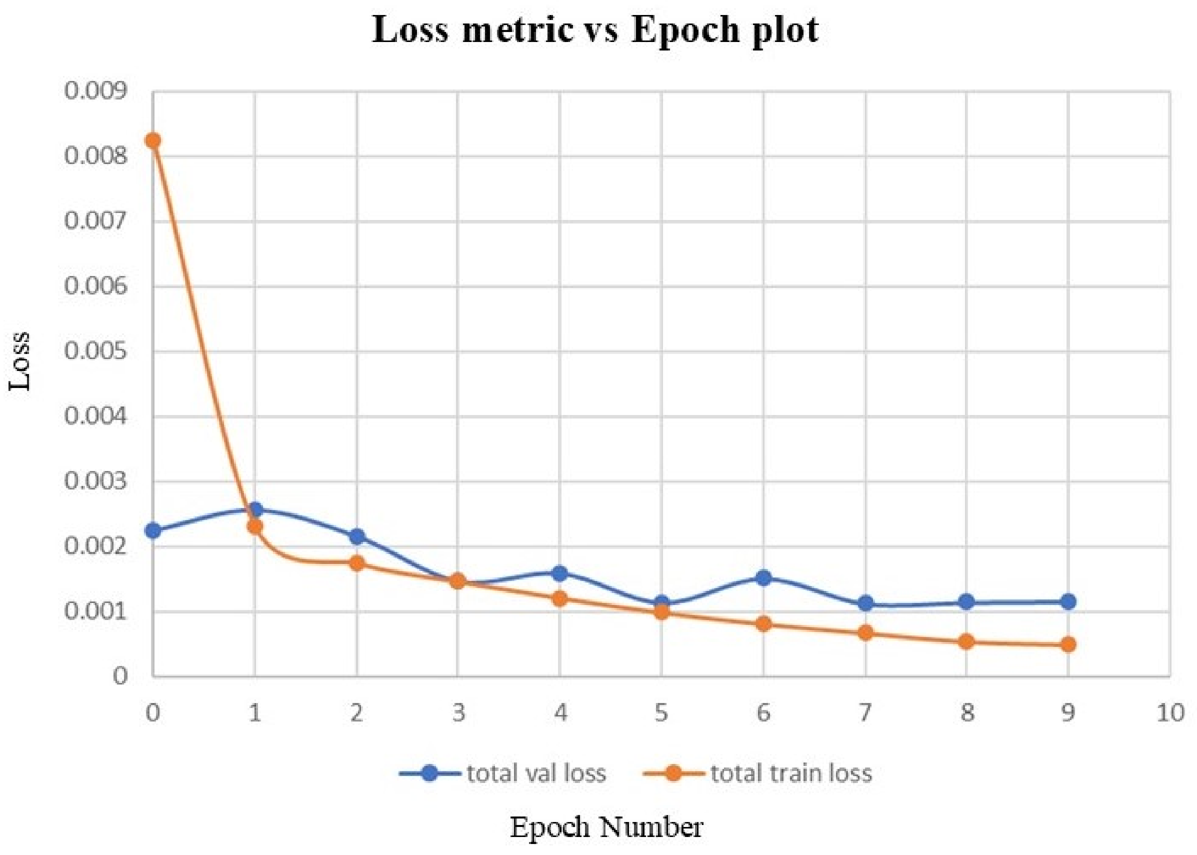

Figure 4.2: Plot for Total Loss with Epoch

the training loss dropped to abnormally low values from the very first epochs, which suggests that the model was quickly overfitting and memorizing the dataset. This rapid decrease in training loss, without corresponding improvements in the validation loss, is indicative of overfitting. Although the validation loss showed an initial decrease, it plateaued quickly, meaning the model struggled to generalize beyond the training data. This behavior suggests that the model had already learned the features of the training data too well and failed to generalize effectively to unseen data.

The output of the model was predominantly near-zero across the inferred slices, which suggests that the model struggled to extract relevant features from the data. This outcome can be attributed to several key factors, particularly the complexity of the model architecture in relation to the available data.

The Swin Transformer, a highly sophisticated model based on self-attention mechanisms, is designed to capture intricate spatial dependencies. However, this high-capacity model proved to be too complex for the relatively small and homogeneous dataset used in this study. While the dataset was augmented with multiple orientations, all the data came from the same subject, meaning that the model was exposed to very similar structures across orientations. Transformers excel at learning long-range dependencies in data, but the differences between orientations in this case were due to magnetic anisotropy, which represents fine details rather than large

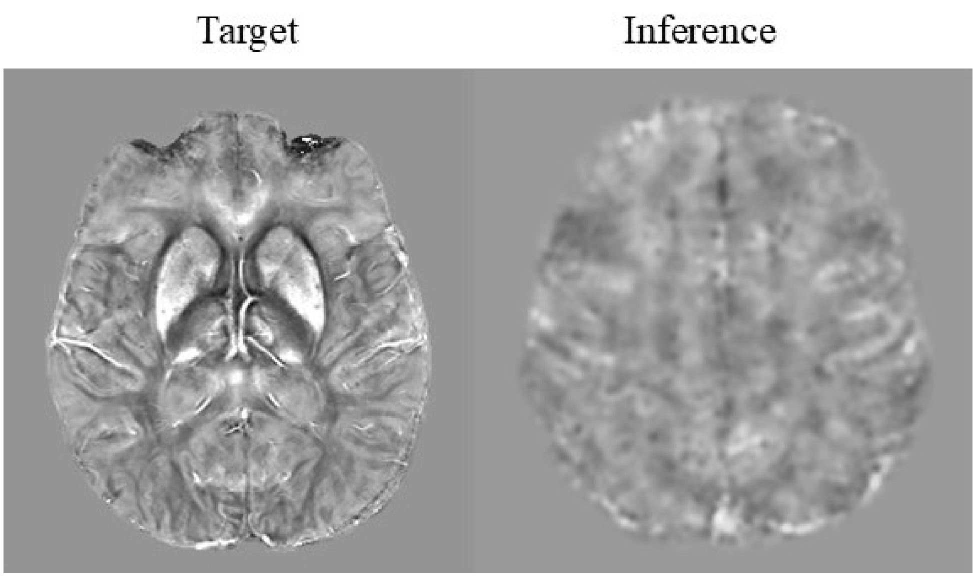

Figure 4.3: Inference Output of SwinUNet2D Architecture

structural differences. The model, therefore, struggled to distinguish these subtle variations because it was more focused on identifying larger-scale features, which led to the overfitting issue.

Furthermore, the model's use of aggressive downsampling during the inference process may have exacerbated the issue. The self-attention mechanism in transformers, while powerful, can result in the loss of fine-grained spatial information, particularly when applied to high-resolution imaging data with limited examples. During the downsampling process, essential low-level features and fine details may have been discarded, leading to outputs with little meaningful variation. This excessive downsampling, combined with insufficient training data, contributed to the model output being overwhelmingly close to zero.

In summary, the combination of an overly complex transformer model, a small dataset with limited variability (due to multiple orientations from the same subject), aggressive downsampling during inference, and the fine details caused by magnetic anisotropy, created a situation where the model was unable to effectively learn the intricate patterns in the data. Consequently, this led to the near-zero output observed in the inference stage. Despite these challenges, the insights gained from this training process can help guide future refinements in model architecture and data handling to improve performance.

Chapter 5

Evaluation

5.1 Assessment of Issues

5.1.1 Societal Issues

The use of deep learning in Quantitative Susceptibility Mapping (QSM) reconstruction has the potential to enhance diagnostic precision and efficiency. However, its adoption may widen healthcare disparities, as access to high-quality computational tools is often limited to well-funded institutions. This could create a gap between advanced medical centers and under-resourced facilities. Additionally, trust in automated reconstruction methods may be a barrier, as both patients and clinicians may be skeptical about relying on complex algorithms. Ensuring widespread access and proper education on these advancements is necessary for equitable healthcare improvements.

5.1.2 Health and Safety Issues

In medical imaging, even minor inaccuracies can lead to serious consequences. Since QSM reconstruction using deep learning lacks a direct ground truth, the results are approximations that may not always reflect true tissue properties. This raises concerns about misdiagnoses and incorrect treatment plans. Furthermore, full reliance on automated reconstructions without human verification could introduce risks in critical care settings. To ensure patient safety, robust validation procedures and human oversight must be maintained.

5.1.3 Legal and Cultural Issues

The legal framework surrounding computational medical tools is still evolving. Questions of liability arise when incorrect reconstructions lead to misinterpretations—whether responsibility falls on the developers, medical institutions, or individual practitioners remains unclear. Additionally, cultural resistance to automation in medicine is significant. Many healthcare professionals are accustomed to traditional imaging methods and manual verification, making them hesitant to embrace new computational approaches. Integrating these technologies into established workflows while maintaining regulatory clarity is essential for acceptance.

5.1.4 Ethical Issues

One of the biggest concerns with deep learning-based QSM reconstruction is the lack of interpretability. Understanding how conclusions are reached is crucial for clinicians, yet many reconstruction methods function as complex, opaque systems. This makes it difficult to justify treatment decisions based on the generated maps, especially when discrepancies arise between automated outputs and traditional diagnostic methods. Without clear explanations, clinicians may hesitate to fully trust or adopt these techniques in critical decision-making. Bias in training data is another ethical challenge—if models are trained on non-representative datasets, they may produce unreliable results for certain patient populations, leading to disparities in diagnostic accuracy. Addressing these concerns requires continuous evaluation, improved model transparency, and regulatory guidelines to ensure that deep learning-based medical tools are both reliable and equitable.

5.1.5 Evaluation of Environment and Sustainability

Developing and maintaining deep learning models requires significant computational power. The high energy consumption of training these models, along with ongoing updates and storage needs, contributes to environmental concerns. The extensive use of high-performance hardware and cloud-based systems increases the carbon footprint of medical computing. To promote sustainability, optimizing model efficiency and reducing unnecessary computational costs should be prioritized, ensuring that medical advancements do not come at a significant ecological expense.

5.2 Future Works

There are several avenues for future work to build upon the findings of this thesis and further advance the field of QSM reconstruction.

5.2.1 Transition to 3D Reconstruction

One of the most significant next steps is to transition the model from 2D to 3D reconstruction. While this work primarily focused on 2D slices, extending the model to process full 3D volumes will provide more accurate and holistic susceptibility maps. The ability to capture the entire volume of the brain, considering the spatial relationships between different regions, will allow for more precise diagnosis and treatment planning. A 3D model will also be better equipped to handle complex anatomical structures, providing a more realistic representation of tissue properties.

5.2.2 Incorporation of Generative Adversarial Networks (GANs)

Another key area of future work is the incorporation of **Generative Adversarial Networks (GANs)** to improve the quality of the reconstructed susceptibility maps [31]. GANs have shown promise in various imaging tasks, particularly in enhancing image realism and reducing artifacts. In the context of QSM reconstruction, GANs can be used as a post-processing step to refine the output of the deep learning model, improving its perceptual quality and reducing noise. By using a GAN-based approach, the model can learn through a dynamic loss analysis.

5.2.3 Expanding Data Augmentation Techniques

Moreover, the model can be improved by expanding the data augmentation techniques used during training. For example, in **QSMNet+** [44], the authors introduced effective data augmentation techniques to enhance the model's generalization across clinical scenarios. They applied susceptibility scaling and sign inversion to the susceptibility maps, which simulated variations in tissue susceptibility and phase map sign changes. Susceptibility scaling randomly multiplies the susceptibility values to capture a range of tissue types and pathological conditions, while sign inversion mimics different MRI phase data. These techniques help the model handle a broader spectrum of clinical cases, including abnormal tissue structures and hemorrhagic lesions.

Additionally, rotation, scaling, and noise injection were applied to simulate real-world variations like patient positioning, slice thickness, and scanner artifacts. These transformations ensure the model is robust to changes in orientation, size, and noise, which are common in clinical imaging. This augmentation strategy increases the dataset's diversity, helping the model generalize better across diverse clinical scenarios.

5.2.4 Integration into Clinical Workflows

Finally, integrating the developed model into clinical workflows will be a crucial next step. This would involve validating the model's performance in real-world settings and assessing its impact on clinical decision-making. Collaboration with hospitals and imaging centers to conduct prospective studies will help evaluate the model's effectiveness in assisting radiologists and clinicians in diagnosing diseases, planning treatments, and improving patient outcomes.

Chapter 6

Conclusion

This thesis explored the potential of deep learning in improving Quantitative Susceptibility Mapping (QSM) reconstruction, focusing on addressing the ill-posed nature of dipole inversion. By leveraging data-driven techniques, the study demonstrated the feasibility of deep learning-based QSM, particularly in balancing computational efficiency with reconstruction accuracy. The investigation of both 3D and 2D models provided valuable insights into the strengths and limitations of different architectural approaches. While the MoDL-QSM model successfully captured complex susceptibility patterns, the transformer-based model faced challenges due to data constraints and overfitting.

Despite these advancements, key challenges remain in ensuring model interpretability, clinical reliability, and robust generalization across diverse datasets. The integration of physics-informed learning and anisotropic susceptibility modeling presents promising directions for further improvement. Future work should focus on optimizing model architectures, incorporating advanced generative models, and refining data preprocessing techniques to enhance real-world applicability. Ultimately, the successful adoption of deep learning for QSM reconstruction will require a careful balance between technical innovation and clinical practicality, ensuring that these advancements contribute meaningfully to medical imaging and patient care.

References

[1] Zhang, H., Nguyen, T.D., Zhang, J., Marcille, M., Spincemaille, P., Wang, Y., Gauthier, S.A. and Sweeney, E.M. “QSMRim-Net: Imbalance-Aware Learning for Identification of Chronic Active Multiple Sclerosis Lesions on Quantitative Susceptibility Maps.” *NeuroImage: Clinical*, volume 35:p. 102979, 2022. Originally posted on bioRxiv: 2022.01.31.478482

[2] Si, W., Guo, H., Zhang, Y., Zhang, J., Wang, S. and Feng, Q. “Quantitative susceptibility mapping using multi-channel convolutional neural networks with dipole-adaptive multi-frequency inputs.” *Frontiers in Neuroscience*, volume 17:p. 1165446, 2023

[3] Liu, C., Li, W., Tong, K., Yeom, K. and Kuzminski, S. “Susceptibility-weighted imaging and quantitative susceptibility mapping in the brain.” *J. Magn. Reson. Imaging*, volume 42, no. 1:pp. 23–41, Jul 2015. Epub 2014 Oct 1

[4] Acosta-Cabronero, J., Williams, G., Cardenas-Blanco, A., Arnold, R., Lupson, V. and Nestor, P. “In vivo quantitative susceptibility mapping (qsm) in alzheimer’s disease.” *PLoS One*, volume 8:p. e81093, 2013

[5] Acosta-Cabronero, J., Betts, M.J., Cardenas-Blanco, A., Yang, S. and Nestor, P.J. “The whole-brain pattern of magnetic susceptibility perturbations in parkinson’s disease.” *Brain*, volume 140, no. 1:pp. 118–131, 2017

[6] Elkady, A., Cobzas, D., Sun, H., Blevins, G. and Wilman, A. “Discriminative analysis of regional evolution of iron and myelin/calcium in deep gray matter of multiple sclerosis and healthy subjects.” *J. Magn. Reson. Imaging*, volume 48:pp. 652–668, 2018

[7] Wang, Y. and Liu, T. “Quantitative susceptibility mapping (qsm): Decoding mri data for a tissue magnetic biomarker.” *Magn. Reson. Med.*, volume 73, no. 1:pp. 82–101, Jan 2015. Epub 2014 Jul 17

[8] Li, W., Wang, N., Yu, F., Han, H., Cao, W., Romero, R., Tantiwongkosi, B., Duong, T. and Liu, C. “A method for estimating and removing streaking artifacts in quantitative susceptibility mapping.” *Neuroimage*, volume 108:pp. 111–122, Mar 2015. Epub 2014 Dec 20

[9] Liu, J., Liu, T., de Rochefort, L., Ledoux, J., Khalidov, I., Chen, W., Tsiouris, A., Wisnieff, C., Spincemaille, P., Prince, M. and Wang, Y. "Morphology enabled dipole inversion for quantitative susceptibility mapping using structural consistency between the magnitude image and the susceptibility map." *Neuroimage*, volume 59, no. 3:pp. 2560–2568, Feb 1 2012. Epub 2011 Sep 8

[10] Wei, H., Dibb, R., Zhou, Y., Sun, Y., Xu, J., Wang, N. and Liu, C. "Streaking artifact reduction for quantitative susceptibility mapping of sources with large dynamic range." *NMR Biomed.*, volume 28, no. 10:pp. 1294–1303, Oct 2015. Epub 2015 Aug 27

[11] Shmueli, K., de Zwart, J.A., van Gelderen, P., Li, T.Q., Dodd, S.J. and Duyn, J.H. "Magnetic susceptibility mapping of brain tissue in vivo using mri phase data." *Magnetic Resonance in Medicine*, volume 62, no. 6:pp. 1510–1522, 2009

[12] Chen, Y., Jakary, A., Avadiappan, S., Hess, C.P. and Lupo, J.M. "Qsmgan: Improved quantitative susceptibility mapping using 3d generative adversarial networks with increased receptive field." *NeuroImage*, volume 207:p. 116389, 2020

[13] Yoon, J., Gong, E., Chatnuntawech, I., Bilgic, B., Lee, J., Jung, W., Ko, J., Jung, H., Setsompop, K., Zaharchuk, G., Kim, E.Y., Pauly, J. and Lee, J. "Quantitative susceptibility mapping using deep neural network: Qsmnet." *NeuroImage*, volume 179:pp. 199–206, 2018

[14] Li, M., Chen, C., Xiong, Z., Liu, Y., Rong, P., Shan, S., Liu, F., Sun, H. and Gao, Y. "Ir2qsm: Quantitative susceptibility mapping via deep neural networks with iterative reverse concatenations and recurrent modules." *arXiv preprint arXiv:2406.12300*, 2024

[15] Gao, Y., Zhu, X., Moffat, B.A., Glarin, R., Wilman, A.H., Pike, G.B., Crozier, S., Liu, F. and Sun, H. "xqsm: Quantitative susceptibility mapping with octave convolutional and noise-regularized neural networks." *NMR in Biomedicine*, volume 34, no. 5:p. e4461, 2021

[16] Feng, R., Zhao, J., Wang, H., Yang, B., Feng, J., Shi, Y., Zhang, M., Liu, C., Zhang, Y., Zhuang, J. and Wei, H. "MoDL-QSM: Model-based deep learning for quantitative susceptibility mapping." *NeuroImage*, volume 240:p. 118376, 2021

[17] Lai, K.W., Aggarwal, M., van Zijl, P., Li, X. and Sulam, J. "Learned proximal networks for quantitative susceptibility mapping." *arXiv preprint arXiv:2008.05024*, 2020

[18] Polak, D., Bredies, K., Langkammer, C., Schweser, F., Ropele, S., Bredies, K., Langkammer, C., Schweser, F. and Ropele, S. "Variational network for quantitative susceptibility mapping." *arXiv preprint arXiv:2006.14151*, 2020

[19] Oh, G., Bae, H., Ahn, H.S., Park, S.H. and Ye, J.C. "Cycleqsm: Unsupervised qsm deep learning using physics-informed cyclegan." *arXiv preprint arXiv:2012.03842*, 2020

[20] Zhang, M., Zhang, Y. and Wei, H. "A subject-specific unsupervised deep learning method for quantitative susceptibility mapping using implicit neural representation." *NeuroImage*, volume 256:p. 119248, 2022

[21] Xiong, Z., Jiang, W., Gao, Y., Liu, F. and Sun, H. "Qsmdiff: Unsupervised 3d diffusion models for quantitative susceptibility mapping." *arXiv preprint arXiv:2403.14070*, 2024

[22] Li, Y., Li, W., Qin, W., Liang, X., Xu, J., Xiong, J., Xia, J. and Xie, Y. "Comparison of supervised and unsupervised deep learning methods for medical image synthesis between computed tomography and magnetic resonance images." *BioMed Research International*, volume 2020:pp. 1–10, 2020

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, and Polosukhin, I. "Attention is all you need." *Advances in Neural Information Processing Systems*, volume 30, 2017

[24] Zhang, C., Liu, L., Dai, J., Liu, X., He, W., Chan, Y., Xie, Y., Chi, F. and Liang, X. "Xtransct: Ultra-fast volumetric ct reconstruction using two orthogonal x-ray projections for image-guided radiation therapy via a transformer network." *Physics in Medicine and Biology*, volume 69, no. 8, 2024

[25] Huang, B., Liu, X., Fang, L., Liu, Q. and Li, B. "Diffusion transformer model with compact prior for low-dose pet reconstruction." *arXiv preprint arXiv:2407.00944*, 2024

[26] Chen, J., Lu, Y., Yu, Q., Luo, X., Adeli, E., Wang, Y., Lu, L., Yuille, A.L. and Zhou, Y. "Transunet: Transformers make strong encoders for medical image segmentation." *arXiv preprint arXiv:2102.04306*, 2021

[27] Hatamizadeh, A., Nath, V., Tang, Y., Yang, D. and Roth, H.R. "Swin unetr: Swin transformers for semantic segmentation of brain tumors in mri images." *arXiv preprint arXiv:2201.01266*, 2022

[28] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929*, 2020

[29] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. and Guo, B. "Swin transformer: Hierarchical vision transformer using shifted windows." *arXiv preprint arXiv:2103.14030*, 2021

[30] Cai, Y., Long, Y., Han, Z. et al. “Swin unet3d: a three-dimensional medical image segmentation network combining vision transformer and convolution.” *BMC Medical Informatics and Decision Making*, volume 23:p. 33, 2023

[31] Isola, P., Zhu, J.Y., Zhou, T. and Efros, A.A. “Image-to-image translation with conditional adversarial networks.” *arXiv preprint arXiv:1611.07004*, 2017

[32] Liu, C. “Susceptibility tensor imaging (sti) of the brain.” *Magnetic Resonance in Medicine*, volume 66, no. 3:pp. 746–754, 2011

[33] Liu, C., Wei, H., Gong, N.J., Cronin, M., Dibb, R. and Decker, K. “Quantitative susceptibility mapping: Contrast mechanisms and clinical applications.” *Tomography*, volume 1, no. 1:pp. 3–17, 2015

[34] Wei, H., Zhang, Y., Gibbs, E., Chen, N.K. et al. “Joint 2d and 3d phase processing for quantitative susceptibility mapping: Application to 2d echo-planar imaging.” *NMR in Biomedicine*, volume 30, no. 4, 2016

[35] Mason, E. “How to read an mri.”

[36] Stehling, M.K., Turner, R. and Mansfield, P. “Echo-planar imaging: magnetic resonance imaging in a fraction of a second.” *Science*, volume 254, no. 5028:pp. 43–50, 1991

[37] Shi, Y., Feng, R., Li, Z. et al. “Towards in vivo ground truth susceptibility for single-orientation deep learning qsm: A multi-orientation gradient-echo mri dataset.” *NeuroImage*, volume 261:p. 119522, 2022

[38] Smith, S.M. “Fast robust automated brain extraction.” *Human Brain Mapping*, volume 17, no. 3:pp. 143–155, 2002

[39] Schofield, M.A. and Zhu, Y. “Fast phase unwrapping algorithm for interferometric applications.” *Optics Letters*, volume 28, no. 14:pp. 1194–1196, 2003

[40] Schweser, F., Deistung, A. and Reichenbach, J.R. “Toward online reconstruction of quantitative susceptibility maps: superfast dipole inversion.” *Magnetic Resonance in Medicine*, volume 69, no. 6:pp. 1581–1593, 2011

[41] Sun, H. and Wilman, A.H. “Harmonic artifact reduction for susceptibility mapping (harperella): A novel method for improving quantitative susceptibility mapping near the edges of the brain.” *Magnetic Resonance in Medicine*, volume 71, no. 3:pp. 1151–1157, 2014

[42] Li, W., Wu, B. and Liu, C. “5223 sti suite : a software package for quantitative susceptibility imaging.”, 2013

[43] Li, W., Wu, B., Avram, A.V. and Liu, C. "Magnetic susceptibility anisotropy of human brain in vivo and its molecular underpinnings." *NeuroImage*, volume 59, no. 3:pp. 2088–2097, 2012

[44] Jung, W. and et al.,. "Qsmnet+: A deep learning framework for quantitative susceptibility mapping from mri phase images." *IEEE Transactions on Medical Imaging*, volume 39, no. 9:pp. 2957–2968, 2020

Appendix A

Model Architecture Codes

A.1 Code for MoDL Architecture

```python
from keras.models import Model
from keras.layers import Input, BatchNormalization, Dropout, Lambda,
   Add, LeakyReLU, Multiply, Activation, Flatten, Layer
from keras.layers import Conv3D
from keras.regularizers import l2
from keras import backend as K
import tensorflow as tf
import numpy as np
from scipy.io import loadmat
import os

weight_decay=0.0005
os.environ["CUDA_VISIBLE_DEVICES"]="0"

def initial_conv(x,filters,kernel_size,strides=(1,1,1),padding='same'
   ):
    """
    This function creates a convolution layer followed by ReLU
    """
    x=Conv3D(filters,kernel_size,strides=strides,padding=padding,
             W_regularizer=l2(weight_decay),
             use_bias=False,
             kernel_initializer='he_uniform')(x)

    x=Activation('relu')(x)
    return x
```

```python


def Res_block(x, k=1, dropout=0.0):
    """
    This function creates a ResBlock
    """

    init=x

    x = Conv3D(16*k,(3,3,3),padding='same',
                W_regularizer=l2(weight_decay),
                use_bias=False,
                kernel_initializer='he_uniform')(x)

    x = BatchNormalization(axis=4)(x)
    x=Activation('relu')(x)

    if dropout > 0.0: x=Dropout(dropout)(x)

    x = Conv3D(16*k,(3,3,3),padding='same',
                W_regularizer=l2(weight_decay),
                use_bias=False,
                kernel_initializer='he_uniform')(x)
    x = BatchNormalization(axis=4)(x)
    m = Add()([init,x])
    m=Activation('relu')(m)

    return m

def pad_tensor(x,ind1,ind2,ind3,matrix_size):
    """
    This function will pad the output patches to match the size of
   dipole kernel
    """
    paddings = tf.constant([[0,0],[int((matrix_size[0]-ind1)/2), int
   ((matrix_size[0]-ind1)/2)], [int((matrix_size[1]-ind2)/2), int((
   matrix_size[1]-ind2)/2)],[int((matrix_size[2]-ind3)/2),int((
   matrix_size[2]-ind3)/2)],[0,0]])
    px=tf.pad(x,paddings,'CONSTANT')
    return px

```

```

def AH_op(x,ind1,ind2,ind3,matrix_size):
    """
    This function is the A^H operator as described in paper
    """
    phase=x[0]
    D=x[1]

    #
    phase=tf.dtypes.cast(phase,tf.complex64)
    D=tf.dtypes.cast(D,tf.complex64)


    phase=tf.transpose(phase,perm=[0,4,1,2,3])
    D=tf.transpose(D,perm=[0,4,1,2,3])

    #scaling factor
    SF=np.sqrt(matrix_size[0]*matrix_size[1]*matrix_size[2])
    SF=tf.dtypes.cast(tf.convert_to_tensor(SF),tf.complex64)

    #A_H_A
    ksp_phase=tf.signal.fft3d(phase)/SF
    ty=tf.signal.ifft3d(tf.multiply(D,ksp_phase))*SF
    ty=tf.transpose(ty,perm=[0,2,3,4,1])
    phase=tf.transpose(phase,perm=[0,2,3,4,1])
    ty=tf.concat([ty,phase],axis=-1)
    #cut to the original size
    y=ty[:,int((matrix_size[0]-ind1)/2):int((matrix_size[0]-ind1)/2)+
   ind1,int((matrix_size[1]-ind2)/2):int((matrix_size[1]-ind2)/2)+
   ind2,int((matrix_size[2]-ind3)/2):int((matrix_size[2]-ind3)/2)+
   ind3,:]
    y=tf.dtypes.cast(y,tf.float32)
    return y

def A_op(x,ind1,ind2,ind3,matrix_size):
    """
    This function is the A operator as described in paper
    """
    sus=x[0]
    D=x[1]

    #dipole kernel
    sus=tf.dtypes.cast(sus,tf.complex64)
    D=tf.dtypes.cast(D,tf.complex64)

    sus0=sus[:,:,:,:,0]
    sus1=sus[:,:,:,:,1]

    #scaling factor
    SF=np.sqrt(matrix_size[0]*matrix_size[1]*matrix_size[2])
    SF=tf.dtypes.cast(tf.convert_to_tensor(SF),tf.complex64)
    #A
    ksp_sus0=tf.signal.fft3d(sus0)/SF
    ksp_sus1=tf.signal.fft3d(sus1)/SF

    ty=tf.signal.ifft3d(tf.multiply(D[:,:,:,:,0],ksp_sus0)+ksp_sus1)*
   SF
    #cut to the original size
    y=ty[:,int((matrix_size[0]-ind1)/2):int((matrix_size[0]-ind1)/2)+
   ind1,int((matrix_size[1]-ind2)/2):int((matrix_size[1]-ind2)/2)+
   ind2,int((matrix_size[2]-ind3)/2):int((matrix_size[2]-ind3)/2)+
   ind3]
    y=tf.expand_dims(y,axis=-1)
    y=tf.dtypes.cast(y,tf.float32)
    return y


def term2(inputs,ind1,ind2,ind3,matrix_size):
    """
    This function performs the term: -t_k A^H A x^k-1 as described in
    paper
    """
    sus=inputs[0]
    D=inputs[1]
    alpha=inputs[2]
    pad_sus=Lambda(pad_tensor,output_shape=(matrix_size[0],
   matrix_size[1],matrix_size[2],1),arguments={'ind1':ind1,'ind2':
   ind2,'ind3':ind3,'matrix_size':matrix_size})(sus)
    A_sus=Lambda(A_op,output_shape=(ind1,ind2,ind3,1),arguments={'
   ind1':ind1,'ind2':ind2,'ind3':ind3,'matrix_size':matrix_size})([
   pad_sus,D])
    pad_sus=Lambda(pad_tensor,output_shape=(matrix_size[0],
    matrix_size[1],matrix_size[2],1),arguments={'ind1':ind1,'ind2':
    ind2,'ind3':ind3,'matrix_size':matrix_size})(A_sus)
    AH_sus=Lambda(AH_op,output_shape=(ind1,ind2,ind3,2),arguments={'
    ind1':ind1,'ind2':ind2,'ind3':ind3,'matrix_size':matrix_size})([
    pad_sus,D])

    weight2=Lambda(lambda y:-y[1]*y[0])
    x=weight2([AH_sus,alpha])
    return x

def term1(y,ind1,ind2,ind3,matrix_size):
    """
    This function performs the term: t_k A^H y as described in paper
    """
    pad_phase=Lambda(pad_tensor,output_shape=(matrix_size[0],
    matrix_size[1],matrix_size[2],1),arguments={'ind1':ind1,'ind2':
    ind2,'ind3':ind3,'matrix_size':matrix_size})(y[0])
    out_y=Lambda(AH_op,output_shape=(ind1,ind2,ind3,2),arguments={'
    ind1':ind1,'ind2':ind2,'ind3':ind3,'matrix_size':matrix_size})([
    pad_phase,y[1]])
    return out_y

def init(x):
    """
    This function creates a zero keras tensor of the same size as x
    """
    return tf.zeros_like(x)

def My_init(shape, dtype='float32',name=None):
    """
    This function creates the learnable step size in gradient descent
    """

    value =4.

    return K.variable(value, name=name)



class MyLayer(Layer):
    """
    This function creates the custom layer in Keras
    """

    def __init__(self,**kwargs):

        super(MyLayer,self).__init__(**kwargs)

    def build(self,input_layer):
        self.step=self.add_weight(name='step',
                                  shape=(1,1),
                                  initializer=My_init,
                                  trainable=True)
        super(MyLayer,self).build(input_layer)

    def call(self, input_layer):
        return self.step*tf.ones(tf.shape(input_layer))





def define_generator(num_iter,is_train,Normdata,matrix_size
   =[210,224,160],input_shape=[48,48,48],output_size=[48,48,48]):
    """
    This function is called to create the generator model
    """

    CosTrnMean=Normdata['CosTrnMean']
    CosTrnStd=Normdata['CosTrnStd']
    CosTrnMean=CosTrnMean[np.newaxis,np.newaxis,:,:]
    CosTrnMean=np.tile(CosTrnMean,(input_shape[0],input_shape[1],
   input_shape[2],1))
    CosTrnStd=CosTrnStd[np.newaxis,np.newaxis,:,:]
    CosTrnStd=np.tile(CosTrnStd,(input_shape[0],input_shape[1],
   input_shape[2],1))

    #######G block###########
    init_input=Input(shape=input_shape+[2])
    conv1=initial_conv(init_input,32,(3,3,3))
    print(conv1.shape)
    wide_res1=Res_block(conv1,k=2,dropout=0.5)
```

```
 wide_res2=Res_block(wide_res1,k=2,dropout=0.5)
 wide_res3=Res_block(wide_res2,k=2,dropout=0.5)
 wide_res4=Res_block(wide_res3,k=2,dropout=0.5)
 wide_res5=Res_block(wide_res4,k=2,dropout=0.5)
 wide_res6=Res_block(wide_res5,k=2,dropout=0.5)
 wide_res7=Res_block(wide_res6,k=2,dropout=0.5)
 wide_res8=Res_block(wide_res7,k=2,dropout=0.5)

 conv2=initial_conv(wide_res8,32,(1,1,1))
 print(conv2.shape)
 conv3=initial_conv(conv2, 32, (1,1,1))
 output=Conv3D(filters=2,kernel_size=(1,1,1),strides=(1,1,1),
padding='same')(conv3)

 print(output.shape)
 basic_model=Model(init_input,output)

 ########P(physical) block#########
 y_init=Input(shape=input_shape+[1])     #input phase
 mask=Input(shape=input_shape+[1])       #input voxel_size
 msk=Lambda(lambda x:tf.tile(x,(1,1,1,1,2)))(mask)
 D_input=Input(shape=matrix_size+[1])        #input dipole kernel
 size=tf.keras.backend.int_shape(y_init)

 y_input=Lambda(term1,output_shape=(size[1],size[2],size[3],2),
arguments={'ind1':size[1],'ind2':size[2],'ind3':size[3],'
matrix_size':matrix_size})([y_init,D_input])
 Alpha= MyLayer()(y_input)
 y_input=Multiply()([y_input,Alpha])

 #iterative
 for i in range(num_iter):
     if i==0:
         layer_input=y_input
     else:
         term_output=Lambda(term2,output_shape=(size[1],size[2],
size[3],2),arguments={'ind1':size[1],'ind2':size[2],'ind3':size
[3],'matrix_size':matrix_size})([x_output,D_input,Alpha]) #lamda_9
  #lamda_20 #lamda_31
         layer_output=Add()([x_output,term_output])     #add_9
add_11 add_13
```

```
                layer_input=Add()([layer_output,y_input])    #add_10
    add_12  add_14

            layer_input=Lambda(lambda x: (x-CosTrnMean)/CosTrnStd)(
    layer_input)
            layer_input=Multiply()([layer_input,msk])
            fx_output=basic_model(layer_input)   #model_1
            fx_output=Multiply()([fx_output,msk])
            x_output=Lambda(lambda x: x*CosTrnStd+CosTrnMean)(fx_output)


        if is_train:
            model=Model([y_init,mask,D_input],fx_output)
            return model
        else:
            x_output=Multiply()([x_output,msk])
            model=Model([y_init,mask,D_input],x_output)
            return model
```

A.2 Code for SwinUnet2D Architecture

```
from keras.models import Model
from keras.layers import Input, BatchNormalization, Dropout, Lambda,
    Add, LeakyReLU, Multiply, Activation, Flatten, Layer
from keras.layers import Conv3D
from keras.regularizers import l2
from keras import backend as K
import tensorflow as tf
import numpy as np
from scipy.io import loadmat
import os

weight_decay=0.0005
os.environ["CUDA_VISIBLE_DEVICES"]="0"

def initial_conv(x,filters,kernel_size,strides=(1,1,1),padding='same'
    ):
    """
    This function creates a convolution layer followed by ReLU
    """
    x=Conv3D(filters,kernel_size,strides=strides,padding=padding,
             W_regularizer=l2(weight_decay),
             use_bias=False,
             kernel_initializer='he_uniform')(x)

    x=Activation('relu')(x)
    return x


def Res_block(x, k=1, dropout=0.0):
    """
    This function creates a ResBlock
    """

    init=x

    x = Conv3D(16*k,(3,3,3),padding='same',
               W_regularizer=l2(weight_decay),
               use_bias=False,
               kernel_initializer='he_uniform')(x)

    x = BatchNormalization(axis=4)(x)
    x=Activation('relu')(x)

    if dropout > 0.0: x=Dropout(dropout)(x)

    x = Conv3D(16*k,(3,3,3),padding='same',
               W_regularizer=l2(weight_decay),
               use_bias=False,
               kernel_initializer='he_uniform')(x)
    x = BatchNormalization(axis=4)(x)
    m = Add()([init,x])
    m=Activation('relu')(m)

    return m

def pad_tensor(x,ind1,ind2,ind3,matrix_size):
    """
    This function will pad the output patches to match the size of
   dipole kernel
    """
    paddings = tf.constant([[0,0],[int((matrix_size[0]-ind1)/2), int
   ((matrix_size[0]-ind1)/2)], [int((matrix_size[1]-ind2)/2), int((
   matrix_size[1]-ind2)/2)],[int((matrix_size[2]-ind3)/2),int((
   matrix_size[2]-ind3)/2)],[0,0]])
    px=tf.pad(x,paddings,'CONSTANT')
    return px


def AH_op(x,ind1,ind2,ind3,matrix_size):
    """
    This function is the A^H operator as described in paper
    """
    phase=x[0]
    D=x[1]

    #
    phase=tf.dtypes.cast(phase,tf.complex64)
    D=tf.dtypes.cast(D,tf.complex64)


    phase=tf.transpose(phase,perm=[0,4,1,2,3])
    D=tf.transpose(D,perm=[0,4,1,2,3])

    #scaling factor
    SF=np.sqrt(matrix_size[0]*matrix_size[1]*matrix_size[2])
    SF=tf.dtypes.cast(tf.convert_to_tensor(SF),tf.complex64)

    #A_H_A
    ksp_phase=tf.signal.fft3d(phase)/SF
    ty=tf.signal.ifft3d(tf.multiply(D,ksp_phase))*SF
    ty=tf.transpose(ty,perm=[0,2,3,4,1])
    phase=tf.transpose(phase,perm=[0,2,3,4,1])
    ty=tf.concat([ty,phase],axis=-1)
    #cut to the original size
    y=ty[:,int((matrix_size[0]-ind1)/2):int((matrix_size[0]-ind1)/2)+
   ind1,int((matrix_size[1]-ind2)/2):int((matrix_size[1]-ind2)/2)+
   ind2,int((matrix_size[2]-ind3)/2):int((matrix_size[2]-ind3)/2)+
   ind3,:]
    y=tf.dtypes.cast(y,tf.float32)
    return y

```

```python
def A_op(x,ind1,ind2,ind3,matrix_size):
    """
    This function is the A operator as described in paper
    """
    sus=x[0]
    D=x[1]

    #dipole kernel
    sus=tf.dtypes.cast(sus,tf.complex64)
    D=tf.dtypes.cast(D,tf.complex64)

    sus0=sus[:,:,:,:,0]
    sus1=sus[:,:,:,:,1]

    #scaling factor
    SF=np.sqrt(matrix_size[0]*matrix_size[1]*matrix_size[2])
    SF=tf.dtypes.cast(tf.convert_to_tensor(SF),tf.complex64)
    #A
    ksp_sus0=tf.signal.fft3d(sus0)/SF
    ksp_sus1=tf.signal.fft3d(sus1)/SF

    ty=tf.signal.ifft3d(tf.multiply(D[:,:,:,:,0],ksp_sus0)+ksp_sus1)*
   SF
    #cut to the original size
    y=ty[:,int((matrix_size[0]-ind1)/2):int((matrix_size[0]-ind1)/2)+
   ind1,int((matrix_size[1]-ind2)/2):int((matrix_size[1]-ind2)/2)+
   ind2,int((matrix_size[2]-ind3)/2):int((matrix_size[2]-ind3)/2)+
   ind3]
    y=tf.expand_dims(y,axis=-1)
    y=tf.dtypes.cast(y,tf.float32)
    return y


def term2(inputs,ind1,ind2,ind3,matrix_size):
    """
    This function performs the term: -t_k A^H A x^k-1 as described in
    paper
    """
    sus=inputs[0]
    D=inputs[1]
    alpha=inputs[2]
    pad_sus=Lambda(pad_tensor,output_shape=(matrix_size[0],
   matrix_size[1],matrix_size[2],1),arguments={'ind1':ind1,'ind2':
   ind2,'ind3':ind3,'matrix_size':matrix_size})(sus)
    A_sus=Lambda(A_op,output_shape=(ind1,ind2,ind3,1),arguments={'
   ind1':ind1,'ind2':ind2,'ind3':ind3,'matrix_size':matrix_size})([
   pad_sus,D])
    pad_sus=Lambda(pad_tensor,output_shape=(matrix_size[0],
   matrix_size[1],matrix_size[2],1),arguments={'ind1':ind1,'ind2':
   ind2,'ind3':ind3,'matrix_size':matrix_size})(A_sus)
    AH_sus=Lambda(AH_op,output_shape=(ind1,ind2,ind3,2),arguments={'
   ind1':ind1,'ind2':ind2,'ind3':ind3,'matrix_size':matrix_size})([
   pad_sus,D])

    weight2=Lambda(lambda y:-y[1]*y[0])
    x=weight2([AH_sus,alpha])
    return x

def term1(y,ind1,ind2,ind3,matrix_size):
    """
    This function performs the term: t_k A^H y as described in paper
    """
    pad_phase=Lambda(pad_tensor,output_shape=(matrix_size[0],
   matrix_size[1],matrix_size[2],1),arguments={'ind1':ind1,'ind2':
   ind2,'ind3':ind3,'matrix_size':matrix_size})(y[0])
    out_y=Lambda(AH_op,output_shape=(ind1,ind2,ind3,2),arguments={'
   ind1':ind1,'ind2':ind2,'ind3':ind3,'matrix_size':matrix_size})([
   pad_phase,y[1]])
    return out_y

def init(x):
    """
    This function creates a zero keras tensor of the same size as x
    """
    return tf.zeros_like(x)

def My_init(shape, dtype='float32',name=None):
    """
    This function creates the learnable step size in gradient descent
    """

    value =4.

    return K.variable(value, name=name)



class MyLayer(Layer):
    """
    This function creates the custom layer in Keras
    """

    def __init__(self,**kwargs):

        super(MyLayer,self).__init__(**kwargs)

    def build(self,input_layer):
        self.step=self.add_weight(name='step',
                                        shape=(1,1),
                                        initializer=My_init,
                                        trainable=True)
        super(MyLayer,self).build(input_layer)

    def call(self, input_layer):
        return self.step*tf.ones(tf.shape(input_layer))





def define_generator(num_iter,is_train,Normdata,matrix_size
    =[210,224,160],input_shape=[48,48,48],output_size=[48,48,48]):
    """
    This function is called to create the generator model
    """

    CosTrnMean=Normdata['CosTrnMean']
    CosTrnStd=Normdata['CosTrnStd']
    CosTrnMean=CosTrnMean[np.newaxis,np.newaxis,:,:]
    CosTrnMean=np.tile(CosTrnMean,(input_shape[0],input_shape[1],
    input_shape[2],1))
    CosTrnStd=CosTrnStd[np.newaxis,np.newaxis,:,:]
    CosTrnStd=np.tile(CosTrnStd,(input_shape[0],input_shape[1],
   input_shape[2],1))

    #######G block###########
    init_input=Input(shape=input_shape+[2])
    conv1=initial_conv(init_input,32,(3,3,3))
    print(conv1.shape)
    wide_res1=Res_block(conv1,k=2,dropout=0.5)
    wide_res2=Res_block(wide_res1,k=2,dropout=0.5)
    wide_res3=Res_block(wide_res2,k=2,dropout=0.5)
    wide_res4=Res_block(wide_res3,k=2,dropout=0.5)
    wide_res5=Res_block(wide_res4,k=2,dropout=0.5)
    wide_res6=Res_block(wide_res5,k=2,dropout=0.5)
    wide_res7=Res_block(wide_res6,k=2,dropout=0.5)
    wide_res8=Res_block(wide_res7,k=2,dropout=0.5)

    conv2=initial_conv(wide_res8,32,(1,1,1))
    print(conv2.shape)
    conv3=initial_conv(conv2, 32, (1,1,1))
    output=Conv3D(filters=2,kernel_size=(1,1,1),strides=(1,1,1),
   padding='same')(conv3)

    print(output.shape)
    basic_model=Model(init_input,output)

    ########P(physical) block#########
    y_init=Input(shape=input_shape+[1])     #input phase
    mask=Input(shape=input_shape+[1])       #input voxel_size
    msk=Lambda(lambda x:tf.tile(x,(1,1,1,1,2)))(mask)
    D_input=Input(shape=matrix_size+[1])        #input dipole kernel
    size=tf.keras.backend.int_shape(y_init)

    y_input=Lambda(term1,output_shape=(size[1],size[2],size[3],2),
   arguments={'ind1':size[1],'ind2':size[2],'ind3':size[3],'
   matrix_size':matrix_size})([y_init,D_input])
    Alpha= MyLayer()(y_input)
    y_input=Multiply()([y_input,Alpha])

    #iterative
    for i in range(num_iter):
        if i==0:
            layer_input=y_input
```

```
        else:
            term_output=Lambda(term2,output_shape=(size[1],size[2],size[3],2),arguments={'ind1':size[1],'ind2':size[2],'ind3':size[3],'matrix_size':matrix_size})([x_output,D_input,Alpha]) #lamda_9 #lamda_20 #lamda_31
            layer_output=Add()([x_output,term_output])     #add_9 add_11 add_13
            layer_input=Add()([layer_output,y_input])     #add_10 add_12  add_14

        layer_input=Lambda(lambda x: (x-CosTrnMean)/CosTrnStd)(layer_input)
        layer_input=Multiply()([layer_input,msk])
        fx_output=basic_model(layer_input)    #model_1
        fx_output=Multiply()([fx_output,msk])
        x_output=Lambda(lambda x: x*CosTrnStd+CosTrnMean)(fx_output)


    if is_train:
        model=Model([y_init,mask,D_input],fx_output)
        return model
    else:
        x_output=Multiply()([x_output,msk])
        model=Model([y_init,mask,D_input],x_output)
        return model
```

B.Sc.
Engg.
EEE
BUET

Search Towards an Advanced Deep Learning Algorithm for MRI Quantitative Susceptibility Mapping (QSM) with Magnetic Anisotropy

Tasmin Khan

January
2024