SAPIENZA
UNIVERSITÀ DI ROMA

An Enhanced Architecture for Accelerating Magnetic Resonance Imaging Based on Res-U-Net

Faculty of Information Engineering, Informatics, and Statistics
Master Course on Artificial Intelligence and Robotics

Vincenzo Colella
ID number 1748193

Advisor
Prof. Christian Napoli

Co-Advisor
Prof. Stefano Giagu

Academic Year 2021/2022

An Enhanced Architecture for Accelerating Magnetic Resonance Imaging Based on Res-U-Net

Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: vincenzo.colella@hotmail.com

Abstract

This thesis proposes novel architectures to address the challenges associated with the reconstruction of undersampled MRI data using neural networks. The proposed approach involves designing and implementing a neural network, the Residual U-Net, along with two state-of-the-art preprocessing methods, Compressed Sensing and Parallel Imaging architectures, included to further improve the results obtained. Specifically, the GRAPPA (Generalized Autocalibrating Partially Parallel Acquisitions) and ESPIRiT (Eigenvector-based SPIRiT) algorithms will be applied to the MRI data to enhance the performance of the new architecture. The effectiveness of the proposed approach will be evaluated by training the new architecture on the NYU Health dataset in two distinct training instances, using two undersampling techniques, where the amount of data acquired is reduced by a factor of 4 (4x) and 8 (8x), respectively. The results obtained will be compared with those achieved by the teams that participated in the FastMRI challenge, using well-known metrics, such as SSIM, PSNR and NMSE. The purpose of this thesis is to investigate and analyze the potential of neural networks in enhancing the computational efficiency of MR imaging. The primary objective is to explore how neural networks can be leveraged to accelerate the entire scan process by a factor of four, thereby significantly reducing the total duration of the scan. The thesis contributes to the field of MRI reconstruction by providing insights into the effectiveness of the Res-U-Net and two specific preprocessing methods, GRAPPA and ESPIRiT, to improve the duration of Magnetic Resonance Imaging.

Contents

Chapter 1

Introduction

Magnetic Resonance Imaging (MRI) is a commonly used imaging technique that is non-invasive and does not use ionizing radiation, making it safe for imaging soft tissues. However, the lengthy scan time required for MRI can limit its application, particularly for patients who cannot remain still for extended periods or require urgent results. To address these challenges, researchers are developing faster MRI techniques that can produce high-quality images in shorter time frames, improving accessibility and convenience for patients. Parallel Imaging (PI) and Compressed Sensing (CS) have been proposed as effective approaches to acquire highly compressed MR data, resulting in reduced acquisition times. While Parallel Imaging exploits spatial information from multiple receiver coils to accelerate image acquisition by decreasing the number of phase-encoding steps required, Compressed Sensing MRI achieves high accelerations of 10 times or greater by acquiring random samples in k-space and incorporating image sparsity constraints during reconstruction using iterative algorithms. However, conventional reconstruction algorithms used to reconstruct MR images from undersampled data are not efficient, particularly in highly accelerated scenarios. Neural networks have been introduced as an alternative approach to address these issues. These models exploit both the algorithms presented to achieve an improved computational efficiency.

The context of the current study, as well as the research questions being addressed,

will be introduced briefly in the second chapter. Following that, the focus will shift to the selection of the neural network used for the project, as well as the underlying motivations for this particular choice. The fourth chapter will review the preprocessing techniques designed and implemented and the dataset used in the study. Lastly, in the final chapter, the training process and study results will be thoroughly examined and discussed.

Chapter 2

Background

2.1 Magnetic Resonances

A popular medical imaging method that is both highly efficient and widely used in recent years is magnetic resonance imaging (MRI). In the United States alone, medical professionals conduct about 30 million MRI scans each year; the technique is primarily used for detecting pathologies and disorders in a variety of fields, including oncology, neurology, orthopaedics, and more [10]. The main benefit of MRI over other diagnostic imaging tests is its capacity to generate extremely detailed images, allowing medical professionals to spot tiny fractures or tumours that might not be visible on X-rays or CT scans.

Tumours, joint disease, soft tissue damage, and internal organ damage are just a few of the conditions and injuries that an MRI scan can diagnose, as MRI scans can be used to look at internal organs like the liver, womb, or prostate gland as well as the brain, spinal cord, bones, joints, breasts, heart, and blood vessels. MRI is typically used for research, diagnosis, and treatment planning [8].

The generation of signals, detection of those signals, and reconstruction of images are three of the steps involved in the process of acquiring MRI images. The first step in the procedure involves positioning the patient within a large magnet and then transmitting radio waves through his body, which results in the protons absorbing

energy and spinning in an erratic pattern. When the radio waves are turned off, the protons will move back to where they were before and will release energy that a receiver coil will detect and absorb. A computer will then process this signal in order to produce an image of the area of the body that is being investigated.

The generation of a signal is accomplished by first aligning the protons in the patient's body with the magnetic field, which is done with the assistance of a powerful magnetic field. Tesla (T) units are used to express the magnitude of this magnetic field's strength. The image quality improves in direct proportion to the magnitude of the magnetic field.

The detection of signals requires the utilization of a receiver coil, which measures the amount of energy given off by the protons as they move back to their starting position. The receiver coil is positioned so that it is encircling the area of interest, and it monitors for shifts in the magnetic fields that are brought about by the returning protons.

Reconstructing an image requires converting the signals that are picked up by the receiver coil into an image that a radiologist or another qualified medical professional can decipher. This procedure entails several stages, some of which are referred to as filtering, the Inverse Fourier transformation, and image reconstruction.

K-space is a term used in magnetic resonance imaging (MRI) that refers to the mathematical representation of the spatial frequency domain of the MRI signal. When the image is directly obtained by the technician operating the MR scanner, the image will be in k-space representation, on which the human operator can apply filters for better processing of the data. Specifically, k-space is a 3D grid that represents the spatial frequencies of the MRI signal. The x, y, and z axes of k-space correspond to the phase-encoding, frequency-encoding, and slice-select directions of the MRI scan, respectively. Each point in k-space represents a specific frequency and phase shift of the MRI signal.

The data collected in k-space is processed using a mathematical algorithm called the Inverse Fourier transform to create an image. The Inverse Fourier transform

converts the frequency domain data into the spatial domain, allowing the image to be reconstructed.

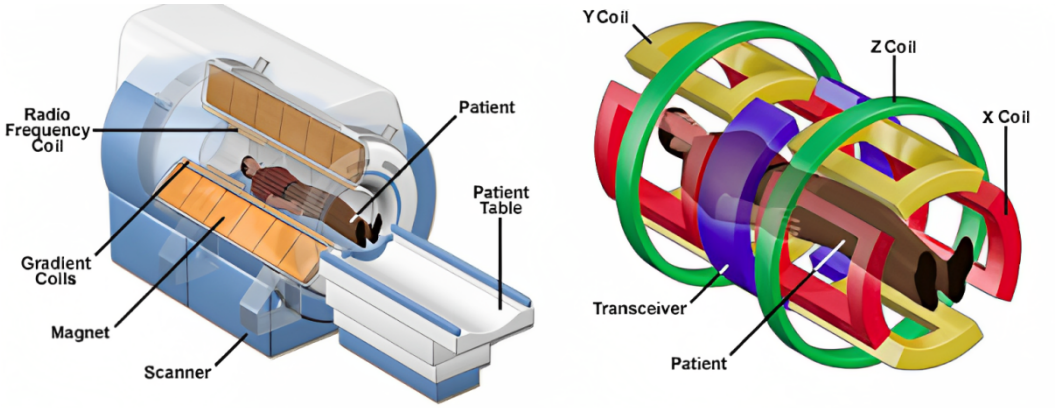

Figure 2.1. Schematic illustration of the MRI system, where the main components are indicated. *Source:* Coyne, 2012.

MRI scans do have some disadvantages, many of which are related to their extended duration. For some patient groups, like pregnant women, who must proceed with caution during an MRI, this can be a significant challenge. Even though MRI scans are thought to be safe during pregnancy, caution should still be exercised, especially during the first trimester, as contrast enhancements based on gadolinium, which are frequently used to enhance the clarity of MRI images, can pass through the placental tissue and be ingested by the fetus, having long-term effects. Studies have shown a marked increase in inflammatory, rheumatologic, and infiltrative skin conditions in pregnancies exposed to gadolinium-based MRI, with respect to the pregnancies which were not exposed to any MRI exam (123 versus 384,180 births, resulting in an adjusted risk difference of 45.3 per 1000 person years). Also, stillbirths and neonatal deaths were noticed as more frequent in the case of gadolinium MRI exposure [1].

The effects on patients who have cochlear implants are a further worry regarding the use of MRI scans. Despite the fact that 3T radiation has been used to demonstrate the safety of modern implants, some patients have reported complications like magnet rotation by 90 degrees, which led to a slight loss of magnetism [2]. Patients who have had cochlear implantation because of hearing loss and need routine MRI scans

for unrelated conditions may find this to be of particular concern.

Children and elderly people may have trouble staying still for long periods during an MRI scan, which can cause discomfort and distress. In particular, young children and babies may need a general anaesthetic to keep them still, while elderly individuals may experience particular challenges due to age-related conditions such as mobility limitations, cognitive impairment, and the presence of medical devices. These factors can impact the quality of MRI images and increase the risk of patient discomfort or injury.

As a result, scientists and medical experts have been looking into alternate methods to improve imaging quality while shortening scan times. Compressed sensing is one such method that makes use of sophisticated algorithms to reconstruct an image from a condensed set of measurements, allowing for faster scan times and reduced radiation exposure for patients. When scanning dynamic structures like the heart, where multiple images must be taken quickly after one another, this technique is especially helpful. Parallel imaging is another method that has gained popularity recently. It uses multiple coils to simultaneously acquire various signal components, which can create a higher-quality image more quickly by combining the data from these coils. This technique is particularly useful for imaging large regions of the body, such as the brain or spinal cord, where high spatial resolution is required. Parallel imaging is also useful for reducing scan times and minimizing patient discomfort. While speeding up the scan could address these issues, the quality of the final image might be affected. A poor-quality image may unnecessarily obstruct the diagnosis process or even result in erroneous or imprecise diagnoses. This can be particularly problematic when determining the diagnosis of complicated and serious medical conditions like cancer.

In conclusion, MRI scans like any diagnostic tool have certain limitations that must be considered. Due to these drawbacks, which include the prolonged scan time and the potential for complications in some patient populations, alternate techniques like compressed sensing and parallel imaging have been developed. By combining

these tools with cutting-edge technologies like machine learning, we may be able to shorten scan times without sacrificing image quality, which would help with complex medical condition diagnosis and treatment.

2.2 FastMRI Challenge

The fastMRI challenge is a public dataset and competition that aims to improve the efficiency of magnetic resonance imaging (MRI) scans by using machine learning techniques. It was launched in 2018 by Facebook AI Research and the New York University (NYU) School of Medicine.

The fastMRI dataset contains raw k-space data and corresponding images from over 8564 knee and brain MRI scans. The goal of the challenge is to develop algorithms that can accurately reconstruct high-quality images from undersampled k-space data, meaning achieve the goal to improve the speed of MRI scans by letting them acquire less data and then accurately reconstruct the resulting undersampled k-space image through algorithms [11].

The FastMRI challenge 2020 involved two tracks: the single-coil track and the multi-coil track. The former involved reconstructing high-quality images from undersampled k-space data using only one coil, while the latter utilized multiple coils to achieve the same goal. The multi-coil track was considered more challenging due to the increased complexity of the data and the need to account for variations in coil sensitivity. The challenge provides a training dataset, a test dataset, and a validation dataset, both of which contain raw k-space data and corresponding images.

The peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) between the reconstructed images and the ground truth images are the primary metrics used in the challenge to assess the algorithms, along with the well-known Normalized Mean Squared Error (NMSE). Specifically, the NMSE (Normalized Mean Squared Error) is calculated as the ratio of the mean squared error (MSE) between

the two images to the mean squared value of the original image and is a metric for comparing the differences between two images. The MSE formula is:

$$\frac{1}{D}\sum_{i=1}^{D}(x_i - y_i)^2 \tag{2.1}$$

where x and y are D dimensional vectors, and x_i denotes the value on the ith dimension of x. Once we divide the result by the mean squared value of the original image, we get the NMSE. This normalization makes the metric scale invariant and allows for a fair comparison of models trained on different datasets.

Another metric for comparing two images is the peak signal-to-noise ratio (PSNR), which is determined by dividing the maximum signal value by the two images' root mean square error (MSE). The PSNR formula is:

$$\text{PSNR} = 20 \cdot \log_{10}\left(\frac{\max(\text{I})}{\sqrt{\text{MSE}}}\right) \tag{2.2}$$

where MSE stands for mean square error, $\log_{10}$ is the base-10 logarithm, and max(I) is the highest possible pixel value.

The structural similarity between two images is measured by the SSIM (Structural Similarity Index) metric. Its formula is:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where σ_x and σ_y are the two variances of x and y, μ_x and μ_y are the two images pixel sample means, σ_{xy} is their covariance, and c_1 and c_2 are small constants to prevent division by zero. SSIM is a perception-based measure that considers image degradation as a perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. Unlike other techniques like MSE or PSNR that estimate absolute errors, SSIM takes into account structural information. This means that SSIM considers the strong inter-dependencies between pixels, especially when they are spatially close, which carry important information about the structure of objects in

the visual scene. In addition, SSIM considers the phenomenon of luminance masking, which refers to image distortions being less visible in bright regions, and contrast masking, which refers to distortions becoming less visible where there is significant activity or "texture" in the image [5].

For the reasons stated above MSE and NMSE, being errors, must be minimised, while SSIM and PSNR need to be maximized since the former represents image similarity while the latter shows relative image quality.

Machine learning algorithms have the potential to increase the effectiveness of MRI scans, as the fastMRI challenge has shown. The successful algorithms that emerged as winners against over 1500 participants (that often didn't submit their work but posted it online) have demonstrated the feasibility of reconstructing high-quality images from undersampled k-space data and consequently achieving an increase in the accessibility and cost-effectiveness of MRI scans. For scientists and programmers working to advance medical imaging, the fastMRI dataset and challenge continue to be invaluable resources.

2.3 Related Works

The state-of-the-art works in MRI reconstruction come from the 2020 FastMRI challenge [24], where Facebook evaluated 19 submissions from 8 different groups. For simplicity, we will only analyze the results for brain multicoil images, which are more challenging and interesting to study.

The best four teams, AIRS Medical, ATB, MRRecon (MRR), and Neurospin (Nspin), were chosen for the finals based on their performance. The projects had varying characteristics, with the number of model parameters ranging from 841,000 in the case of ResoNNance to 200 million in the case of AIRS. Teams used either GRAPPA or simple zero-filled initializations for coil estimation, with some employing ESPIRiT or centre-based estimation with U-Net refinement similar to that in the End-to-End Variational Network.

The challenge included two tracks: the 4x track focused on reconstructing brain MRI images from only 25% of the k-space data, while the 8x track aimed to reconstruct images from only 12.5% of the k-space data. These values are different from the actual image data, and tipically indicate that the 8x track will be the most difficult to reconstruct due to the high degree of undersampling in the data. In the 4x track, the top three teams, AIRS Medical, ATB, and Neurospin, achieved high SSIM values ranging from 0.959 to 0.964. AIRS Medical had the highest SSIM value of 0.967 for the T1 sequence, followed closely by ATB and Neurospin with SSIM values of 0.964 and 0.963, respectively. The T1POST and T2 sequences had similar results, with AIRS Medical achieving the highest SSIM values, followed by ATB and Neurospin.

In contrast, the FLAIR sequence had the lowest SSIM values for all three teams, with AIRS Medical achieving the highest value of 0.930, followed by ATB with 0.924 and Neurospin with 0.920. This indicates that FLAIR reconstruction is more challenging than the other sequences, and further research is needed to improve the reconstruction quality.

The top-performing algorithm, an 'AIRS-net', utilized a deep neural network architecture based on the U-Net model. It consisted of a cascade of 4 U-Nets, where the channels in each convolutional layer were split to work in both k-space and image space domains functionally. The model was initialized through multiple GRAPPA kernels while estimating coil sensitivities with ESPIRiT. An attention mechanism was employed to weigh the importance of different spatial locations in the image during the reconstruction process. This attention mechanism improved the model's ability to capture the fine details of the brain anatomy, leading to the high SSIM score. Moving on to the 8x track, the three leading teams, AIRS Medical, ATB, and Neurospin, achieved lower SSIM values compared to the 4x track. However, the SSIM values were still in the acceptable range, with values ranging from 0.942 to 0.952 for the T1, T1POST, T2, and FLAIR sequences averaged.

AIRS Medical achieved the highest SSIM values for the T1 and T1POST sequences, with SSIM values of 0.953 and 0.963, respectively. ATB and Neurospin achieved slightly lower SSIM values for these sequences. The T2 sequence had similar results, with AIRS Medical achieving the highest SSIM value. Once again, the FLAIR sequence had the lowest SSIM values for all three teams, with AIRS Medical achieving the highest value of 0.918, followed by ATB with 0.905 and Neurospin with 0.898. This highlights the difficulty of reconstructing the FLAIR sequence in both the 4x and 8x tracks.

Overall, the results of the 2020 fastMRI challenge demonstrate the effectiveness of deep learning methods in MRI reconstruction. The leading teams achieved high SSIM values for most sequences, with the FLAIR sequence being the most challenging. AIRS won all three tracks and had the largest model. However, large models were not always better, with ATB having a model with similar performance to Neurospin despite having 87% fewer parameters. Lastly, we note the AIRS model used a normalization routine to get the data into a consistent format for all coil configurations, as well as being the only team to use a GRAPPA reconstruction as the initialization.

Chapter 3

The Neural Network

3.1 Introduction to Neural Networks

Neural networks serve as the basis for modern artificial intelligence and machine learning. These computational models are influenced by the structure and function of biological neural systems, and they enable highly complex tasks by layering and interconnecting artificial neurons [6]. Neural networks have found use in numerous disciplines, including computer vision, natural language processing, and robotics, among others.

In the 1940s, Warren McCulloch and Walter Pitts proposed a simplified computational model of a biological neuron called the McCulloch-Pitts neuron (in Figure 3.1). In the decades that followed, Frank Rosenblatt introduced the perceptron, the first trainable neural network model capable of classifying linearly distinct patterns. The development of the backpropagation algorithm in the 1980s allowed for the efficient training of multi-layer perceptrons, signifying a significant milestone in the field of neural networks.

In recent times, the field of neural network research has experienced significant progress, particularly in the 21st century. This progress has been characterized by remarkable advancements in deep learning techniques, which have emerged as a dominant force in the domains of artificial intelligence (AI) and machine learning (ML)

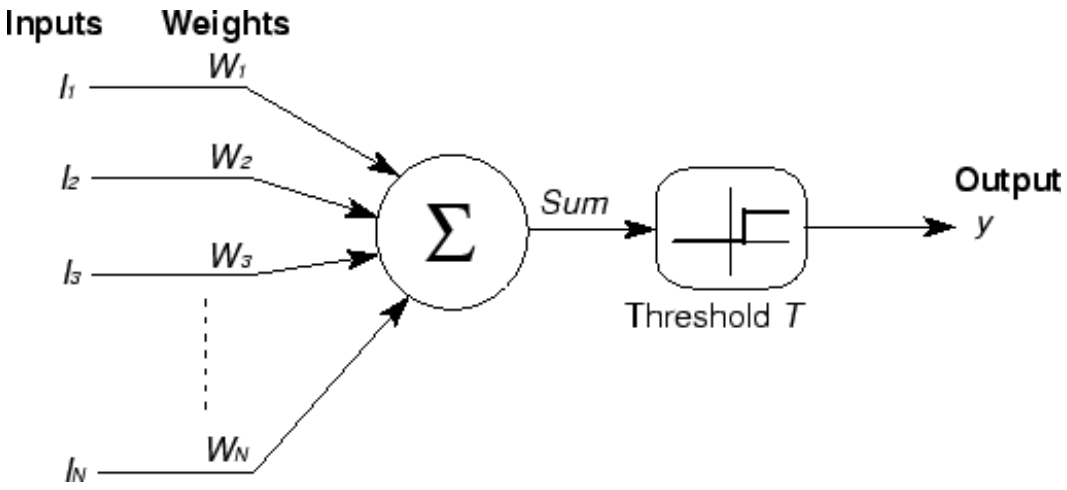

Figure 3.1. A McCulloch-Pitts neuron. *Source:* "A Concise History of Neural Networks"

applications. The advancement of contemporary hardware, particularly graphics processing units (GPUs), has enabled the training of neural networks that are progressively intricate and deep. As a result, it can be observed that these neural networks have attained remarkable levels of performance in diverse domains such as image recognition, language translation, and game playing.

A neural network is a computational model that consists of interconnected artificial neurons, which are also referred to as nodes or units. These nodes are organized into layers, with each layer performing a specific function in the network's overall computation. The neural network's architecture is designed to mimic the structure and function of the human brain, allowing it to learn and make predictions based on input data. In the domain of neural networks, it is widely recognized that there exist three fundamental categories of layers, namely the input layer, the hidden layer, and the output layer (Figure 3.2). These layers are integral components of the neural network architecture and are responsible for processing and transmitting information throughout the network. The input layer serves as the initial point of entry for data, while the hidden layer(s) process the information through a series of mathematical operations, ultimately leading to the output layer, which produces the final result or prediction [16].

Each individual neuron within a given layer is intricately linked to every other

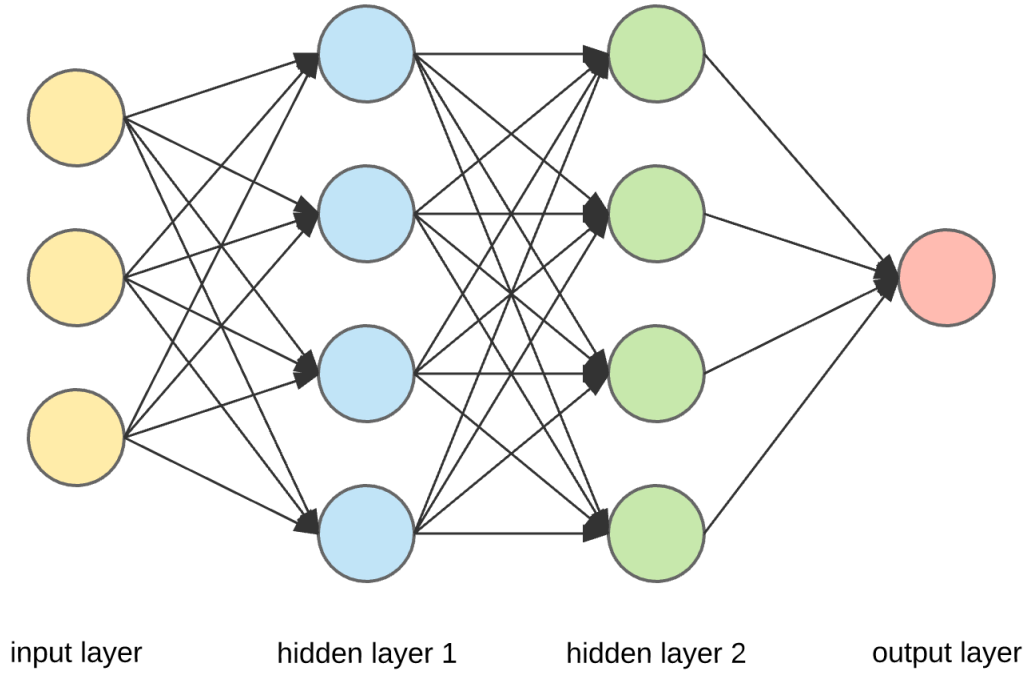

Figure 3.2. Schematic representation of the three main categories of layers

neuron in the subsequent layer through a complex network of weighted connections. The weights in a neural network are indicative of the degree of connectivity between neurons, and their respective magnitudes are acquired through the process of training. The processing of incoming data by neurons is accomplished through the application of an activation function, which serves to determine the output of the neuron based on the weighted sum of its inputs. Examples of such activation functions include the sigmoid and rectified linear unit (ReLU) [15]. The activation function serves the purpose of introducing non-linearity into the model, thereby enabling the neural network to learn complex and non-linear relationships that may exist within the data. This is achieved by applying a mathematical function to the output of each neuron in the network, which allows for the transformation of the input signal into a more expressive and informative representation. This step is crucial for the neural network to effectively capture the intricate and nuanced patterns that may exist within the data, which may not be possible with a linear model. Therefore, the activation function plays an essential part in the success of the neural network in

learning and making accurate predictions. Several types of neural networks have been developed to address different problem domains.

Feedforward Neural Networks (FNNs) are a type of artificial neural network where the flow of data is unidirectional, moving from the input layer through the hidden layers and ultimately to the output layer. This architecture is characterized by the absence of feedback connections, which means that the output of each layer is only connected to the subsequent layer. The input layer receives the raw data, which is then processed by the hidden layers using a series of mathematical transformations. Finally, the output layer produces the network's prediction or classification based on the input data. This unidirectional flow of information is a defining feature of FNNs and is what distinguishes them from other types of neural networks [18].

Recurrent Neural Networks (RNNs) are a type of neural network architecture that possess connections that form directed cycles, enabling them to retain and manipulate information from preceding time steps. This unique characteristic of RNNs allows them to effectively model sequential data, such as natural language, speech, and time series data. By leveraging the information from previous time steps, RNNs can capture the temporal dependencies and patterns that exist within the data, making them a powerful tool for a wide range of applications in machine learning and artificial intelligence. The architectural design under consideration is deemed highly appropriate for sequence-to-sequence tasks, which encompass a wide range of applications including but not limited to time series prediction and language modelling [29].

Convolutional Neural Networks (CNNs) are a class of deep neural networks that employ convolutional layers to extract and learn spatial hierarchies from input data. This makes them particularly well-suited for tasks that involve processing images or other grid-like data. By leveraging the power of convolutional layers, CNNs are able to effectively capture and represent the underlying patterns and structures present in the input data. As a result, they have become a popular and widely-used approach for a variety of computer vision tasks, including image classification, object

detection, and semantic segmentation, among others. Convolutional neural networks (CNNs) have been widely employed in various applications, including but not limited to image recognition, object detection, and segmentation [26].

Segmentation neural networks refer to a class of specialized computational architectures that are specifically designed to partition images into meaningful and distinct segments or regions. The primary objective of these networks is to accurately identify and classify the various components of an image, based on their unique characteristics and features. This process involves the use of advanced algorithms and techniques that enable the network to analyze and interpret the visual information contained within the image and to generate precise and reliable segmentations that can be used for a wide range of applications. The objective of these networks is to allocate a categorical label to every individual pixel present in the input image, thereby generating a segmented output. The process of image segmentation holds paramount importance in a multitude of applications, including but not limited to autonomous vehicles, medical imaging, and scene understanding. Its significance lies in the ability to partition an image into distinct and meaningful regions, thereby facilitating the extraction of relevant information. Overall, segmentation neural networks represent a powerful and versatile tool for image analysis and processing, with significant potential for further development and refinement in the future.

The Fully Convolutional Network (FCN) is considered one of the earliest architectures used for segmentation. Unlike traditional neural networks, FCN is entirely made up of fully connected layers and thus can process input images of any size. Its unique architecture enables it to generate dense pixel-wise predictions, making it a popular choice for image segmentation.

However, Convolutional Neural Networks (CNNs) are a type of neural network that can be repurposed for segmentation tasks by replacing fully connected layers with convolutional ones. This modification enables the network to process images of varying sizes and produce dense pixel-wise outputs.

While a typical CNN provides one prediction for each input image by gradually

decreasing the representation size along a contracting path, an FCN follows an expansive path and uses the data from the contracting path to provide one prediction per image pixel in segmentation tasks.

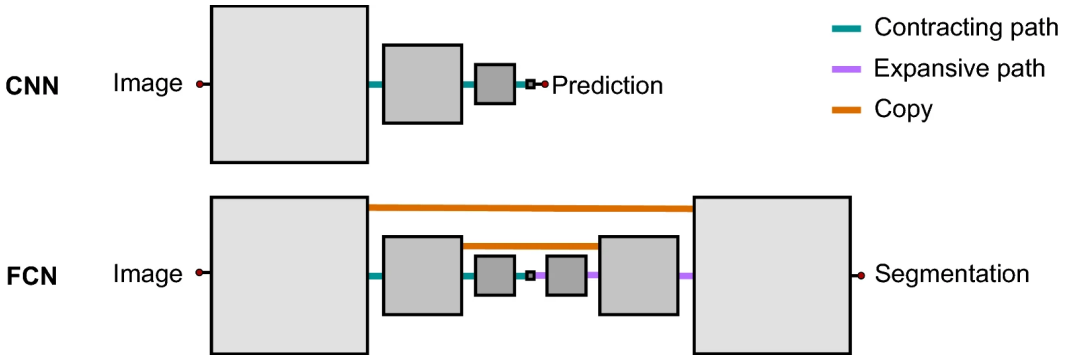

Figure 3.3. Schematic drawing of a convolutional neural network (CNN) and fully convolutional network (FCN)

These approaches have proven to be highly effective in image recognition tasks, as it allows for the efficient analysis of complex visual data.

Various other architectural models have been devised to enhance the performance of Fully Convolutional Networks (FCNs). Among these models are U-Net and DeepLab. These models have been developed with the aim of addressing the limitations of FCNs and improving their efficacy in various applications. The U-Net architecture has gained significant popularity in the field of biomedical image segmentation due to its utilization of a symmetric encoder-decoder structure, which is augmented with skip connections. This design choice has been shown to enhance the localization of objects within the image and facilitate the recovery of fine-grained details. In contrast, DeepLab is a convolutional neural network architecture that integrates atrous convolutions and spatial pyramid pooling techniques to effectively capture multi-scale contextual information. This approach has resulted in great performances on various segmentation benchmarks. [4].

VarNet is a neural network architecture for variational inference in probabilistic models. This approach leverages the power of neural networks to enable efficient and effective modelling of complex distributions. VarNet is a promising development in the field of machine learning, as it has the potential to significantly enhance the

accuracy and efficiency of probabilistic modelling tasks. Despite not being specifically designed for segmentation tasks, it presents itself as a noteworthy alternative to conventional models [27]. The VarNet model is comprised of a series of normalizing flow transformations that are subsequently followed by a "base" distribution, which is typically a Gaussian distribution. Normalizing flows represent a class of generative models that operate by transforming a basic probability distribution, such as a Gaussian distribution, into a more intricate distribution through the application of a sequence of invertible transformations. This approach enables the generation of complex distributions that are capable of capturing the underlying structure of the data. One of the primary benefits of employing normalizing flows is their ability to facilitate the effective computation of the log-likelihood of the model. This is a crucial requirement for probabilistic inference.

3.2 U-net

The U-Net model, developed at the University of Freiburg's Computer Science Department for biomedical image segmentation [28], serves as the baseline provided by Facebook AI for the FastMRI challenge. This well-known convolutional neural network is frequently used in biomedical segmentation tasks and is a safe and reliable option for the task. The U-Net architecture is similar to that of the Fully Convolutional Network [20], with an Encoder extracting features and a Decoder building the segmentation map. The Encoder, which is a contracting part, consists of two 3x3 convolutions followed by a max-pooling operation with a pooling size of 2x2 and stride of 2, which is repeated four times while doubling the number of filters in the convolutional layers after each down-sampling. In more detail, a convolutional layer creates a set of output feature maps by applying a number of filters to the input data. Each filter extracts a specific feature from the input data, by sliding over the input data during the convolution operation, and determining at each position the dot product between the filter and the input. This operation produces a feature

map that shows the locations of each feature in the input data. On the other hand, the max pooling operation decreases the dimensionality of images by lowering the number of pixels in the output from the previous convolutional layer. When using max pooling, the input image is divided into a number of non-overlapping rectangles, and the maximum value of each rectangle is then calculated. The input image's size is decreased through this operation while its most crucial components are kept. A pooling layer's main goal is to gather features from maps produced by the convolution over the image. Average pooling and maximum pooling are two popular pooling techniques that summarize the average presence of a feature and the most activated presence of a feature, respectively [3]. The Decoder has a similar sequence of up-sampling and two convolution operations, with the inputs in the first convolutions taking into account both the previous block's activation and the activations from the corresponding block in the Encoder. This sequence is also repeated four times, with the number of filters divided by two at each stage, followed by a 1x1 convolution operation to generate the final segmentation map without sacrificing spatial resolution (Figure 3.4).

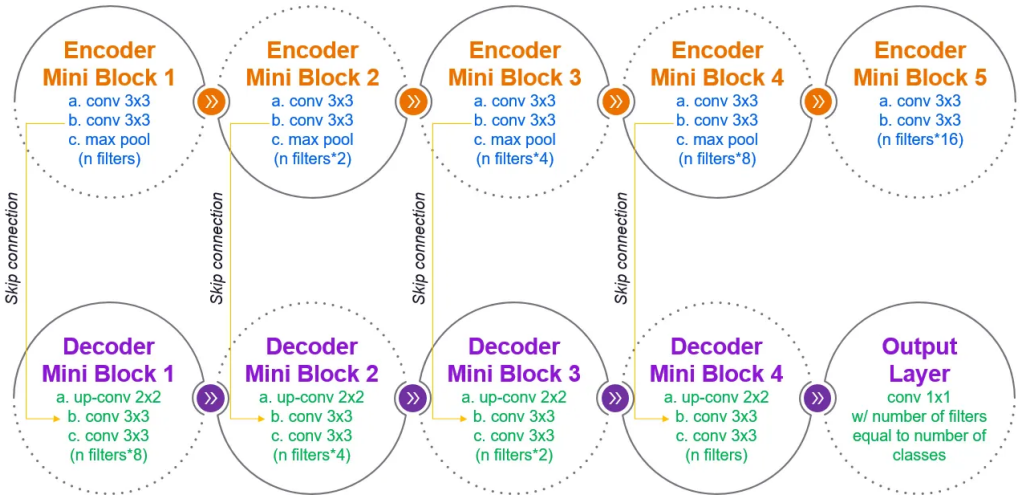

Figure 3.4. The flow of the architecture. *Source:* Vidushi Bhatia: "U-net Implementation from Scratch using TensorFlow"

The symmetric design of the U-Net architecture allows for a direct comparison of the Encoder and Decoder blocks, as well as the incorporation of skip connections

that connect the output of an Encoder block to the corresponding Decoder block (Figure 3.5). Skip connections, in particular, can serve to prevent the neural network from becoming trapped in a local minimum by providing an alternative pathway for the gradient to flow. [9]. The U-Net model has been modified and adapted for a variety of tasks such as image denoising, super-resolution, and image-to-image translation, demonstrating its effectiveness. Its popularity and success have resulted in numerous follow-up studies and variations, including changes to the Encoder and Decoder blocks, the addition of attention mechanisms, and the use of deeper and wider network architectures. Moreover, its versatility, with its adaptability to various tasks, provided me with the opportunity to incorporate new features to improve the model's performance as will be evidenced in the following section.

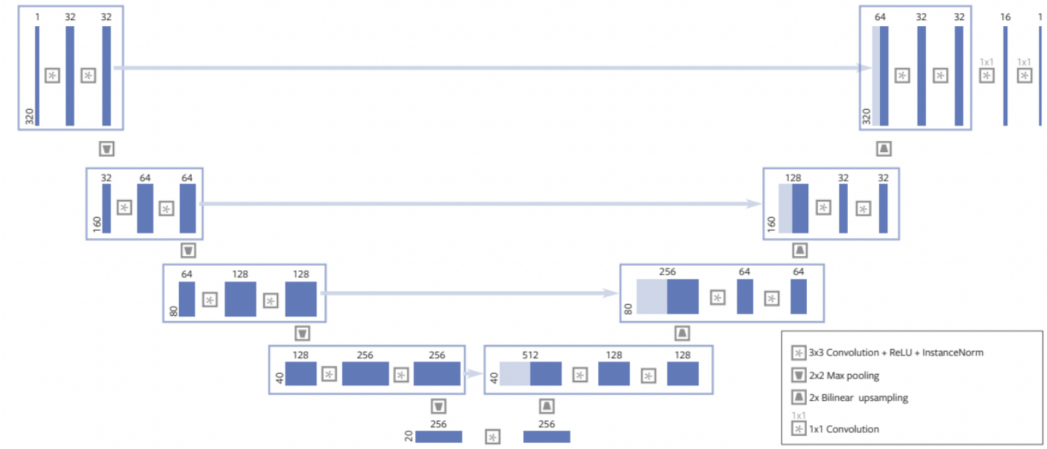

Figure 3.5. The U-Net structure

3.3 Res-U-net

The proposed neural network, which aims to increase the accuracy and quality of the results, is a modified version of the U-Net architecture by V. Lievin [21]. The primary components of the U-Net, like the encoder-decoder structure, are still present. To retain more data from the dataset and, as a result, produce higher-accuracy results, additional features are included in this architecture.

One of the key features of the proposed network is the use of 'Dilated Convolution' blocks. In contrast to conventional convolutions, these convolutions also include a dilation factor that controls the spacing between the kernel points. Dilated convolution operations expand the filter by introducing gaps between the filter values; the size of the gaps is determined by the dilation rate, which is a hyperparameter (which can be changed arbitrarily). By setting the dilation rate to 1, we would perform a regular convolution [19].

Because the filter is still the same size but has gaps between the values, the dilation rate effectively expands the receptive field of the filter without increasing the number of parameters. This can be useful in situations where a larger receptive field is needed, but increasing the size of the filter would lead to an increase in the number of parameters and computational complexity.

Dilated convolutions are able to capture a wider context than standard convolutions because the spacing between the kernel points is increased. They are used in the Res-U-Net architecture to accurately segment medical images by capturing both local and global features of the input image. They require two tensors produced by the Convolutional Encoder as input and are made up of the BatchNorm, ReLu, and Dropout functions. A tensor with the dimensions 64x256x3x3 is the output of the Dilated Convolution block and is sent to the Convolutional Decoder.

Dilated convolutions have been used successfully in various applications such as semantic segmentation where a larger context is needed to classify each pixel and audio processing where the network needs to learn patterns with longer time

dependencies.

The Residual Block is yet another crucial component of the proposed network, introduced as part of the ResNet architecture [17]. A stack of layers called a residual block is set up so that each layer's output is added to a layer further down the stack. They are, in other words, particular skip connection blocks that learn residual functions with reference to the layer inputs instead of learning unreferenced functions. More formally, we let the stacked nonlinear layers fit another mapping of

$$F(x) = H(x) - x \tag{3.1}$$

that denotes the desired underlying mapping as $H(x)$. It transforms the original mapping into

$$F(x) + x \tag{3.2}$$

The term "Residual Block" refers to how $F(x)$ behaves like a residual.

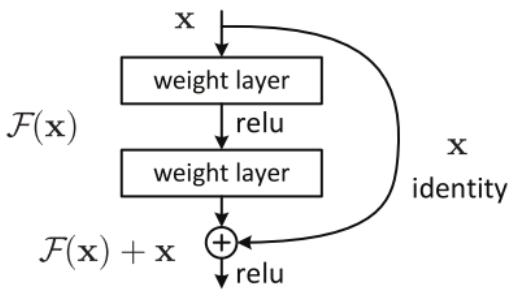

Figure 3.6. The structure of the residual block

The idea is that optimizing the residual mapping is simpler than optimizing the original, unreferenced mapping. In the most extreme case, fitting an identity mapping by a stack of nonlinear layers would be more difficult than pushing the residual to zero if an identity mapping were optimal. This method can help to alleviate the issue of vanishing gradients during training by allowing information to flow from the first to the last layers and then summing the processed data with the trained data. The designed architecture blends techniques to better handle the residual data and processed data, allowing for more optimized information transmission with the aim

of retaining more data from the dataset and thus producing more accurate results. The structure of my network, as computed by TensorBoard, is illustrated in Figure 3.7.

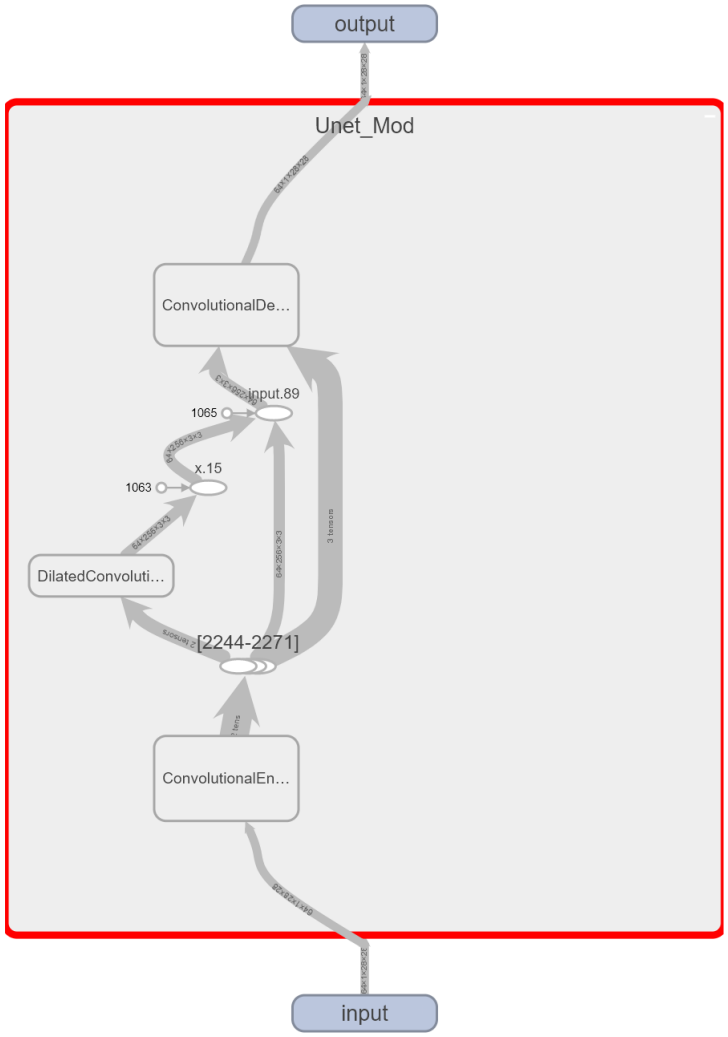

Figure 3.7. Structure of the Res-U-Net computed by TensorBoard

Chapter 4

Preprocessing of the Dataset

4.1 The Dataset

This study is based on the official FastMRI dataset provided by the radiology department at NYU. The said dataset includes MRI scans of the knee and brain that were performed using single-coil and multi-coil methods, respectively. However, our research is primarily focused on multi-coil brain images. The dataset consists of 6,970 fully sampled brain MRIs that have been de-identified in accordance with HIPAA regulations by NYU Langone Health, captured on 3 and 1.5 Tesla magnets. The images are divided into 3,001 at 1.5T and 3,969 at 3T. Axial T1, Axial T1 weighted, T2 weighted, FLAIR and T1 weighted with contrast agent (T1 POST) are the sequences that can be found in the raw dataset. Each one of them is used to highlight certain aspects of the picture: T2 weighted sequences are primarily used to identify pathological changes in neural tissue, whereas T1 weighted sequences are helpful for examining the normal anatomy of the brain. While T1 weighted is referred to as the T1 time measured in the absence of a contrast agent, post-contrast T1 is time measured after the application of gadolinium is used to calculate extracellular volume (ECV), a surrogate parameter for the extracellular matrix. FLAIR is a particular inversion recovery sequence that removes cerebrospinal fluid signal from the resulting images but has a long inversion time [13]. Grey matter is brighter than

white matter in brain tissue on FLAIR images, which are similar to T2 weighted images, but cerebrospinal fluid is dark rather than bright [14]. It's worth mentioning that some T1-weighted acquisitions include contrast materials.

Each file contains around 30 slices of fully sampled images from the MRI samples and is stored in the Hierarchical Data Format (HDF). The dataset consists of five distinct components: training, validation, multi-coil test, multi-coil challenge, and single-coil challenge, which were randomly allocated by the NYU Health department. This systematic organization allows for the evaluation and comparison of various MRI reconstruction algorithms, contributing to the overarching goal of the FastMRI project.

Table 4.1. Number of scans for the different contrasts and scanner field strengths of the brain raw dataset.

Field Strength	1.5T	3T	Total
T1	375	407	782
T1 POST	849	641	1490
T2	1651	2515	4166
FLAIR	126	406	532

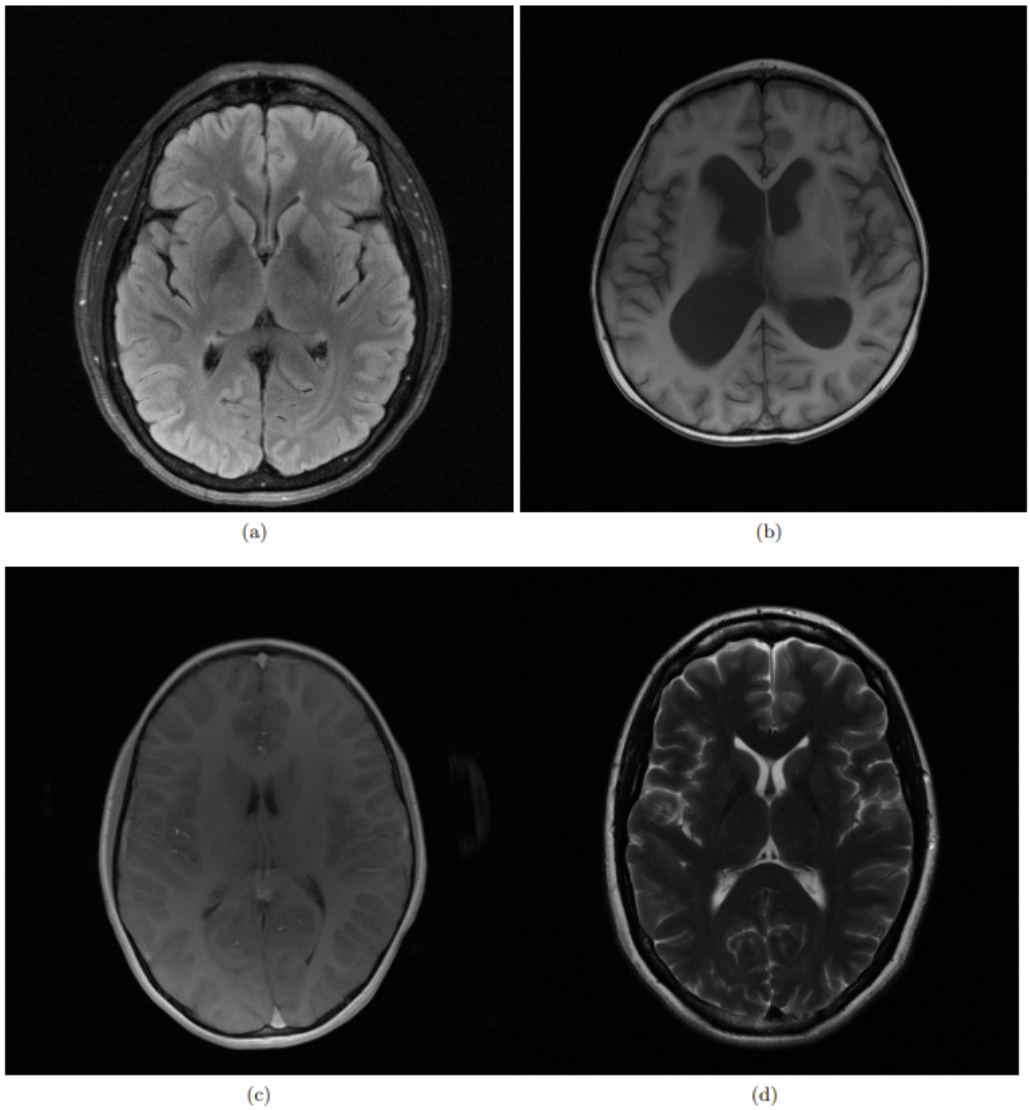

Figure 4.1. Axial brain MRI images with different contrasts: (a) FLAIR, (b) T1 weighted (c) T1 weighted with contrast agent (T1 POST), and (d) T2 weighted.

4.2 GRAPPA

Generalized Autocalibrating Partially Parallel Acquisitions (GRAPPA) [22] is an automatic coil-by-coil reconstruction technique that poses parallel imaging reconstruction as an interpolation problem. GRAPPA is based on the principle of partially acquiring data in k-space, the spatial frequency domain of the MRI signal and reconstructing the missing information using a linear combination of adjacent k-space data points. GRAPPA weights, which are determined during the calibration phase, are utilized to model the relationship between the acquired data and the missing data.

The calibration phase works by acquiring a fully sampled region in the centre of the k-space, known as the autocalibration signal (ACS). The ACS is applied to the undersampled data to reconstruct the missing k-space lines. In order to estimate the GRAPPA weights, a system of linear equations is developed by correlating the acquired ACS data with the missing target data. Solving this system of equations using least squares or other optimization techniques yields the weights. Then, GRAPPA weights are applied to the acquired undersampled data during the reconstruction phase to predict the missing k-space lines. The inverse Fourier transform is then applied to the reconstructed k-space to produce the final image.

Consider $S(\mathbf{k})$ to be the image in the k-space domain, computed using the Fourier Transform from the original image data. The k-space data acquired can be represented as:

$$D(\mathbf{k}) = S(\mathbf{k}) \otimes C(\mathbf{k}),$$

where $D(\mathbf{k})$ represents the acquired k-space data, $C(\mathbf{k})$ represents the coil sensitivity map in k-space, and $\otimes$ is the convolution operation.

GRAPPA can estimate the missing k-space data, $S_m(\mathbf{k})$, using a linear combination of neighboring acquired data points:

$$S_m(\mathbf{k}) = \sum_{i=1}^{N} w_i D(\mathbf{k} - \mathbf{k}_i),$$

where w_i represents the GRAPPA weights, N represents the number of neighboring points, and $\mathbf{k}_i$ represents the k-space coordinates of the neighboring points.

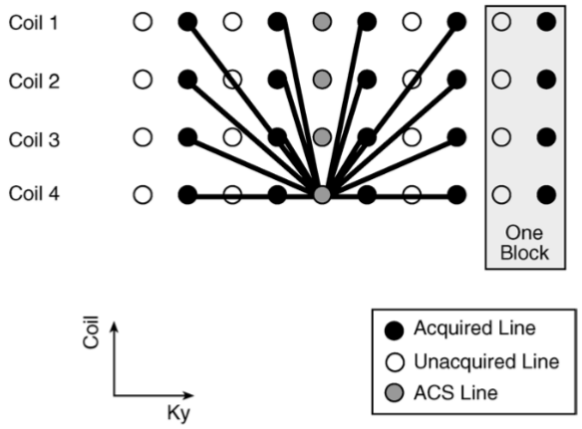

Figure 4.2. Schematic representation of Grappa algorithm. *Source:* Griswold, M.A., Jakob, P.M., Heidemann, R.M., Nittka, M., Jellus, V., Wang, J., Kiefer, B. and Haase, A. (2002), Generalized autocalibrating partially parallel acquisitions (GRAPPA).

Based on the paper by Griswold et al., the implementation takes as input the masked k-space tensor and the randomly computed mask that simulates undersampled k-space data. After applying the GRAPPA algorithm, as you can see from the code (in Code 4.1) the function returns the preprocessed masked k-space tensor of the MR image. I implemented this function as the initial algorithm to preprocess the data before feeding the images into the neural network. A *get_low_frequency_lines* function is defined within the code to compute the low-frequency line indices in the k-space. The *masked_kspace* tensor is a multi-coil masked input k-space tensor with the shape (num_coils, rows, cols, 2), where 2 represents the real and imaginary components of the complex data; the output tensor *preprocessed_masked_kspace* has the same dimensions as the input. The mask tensor is the mask applied to shape (1, 1, cols 1). The *get_low_frequency_lines* function calculates the indices of the low-frequency lines in k-space by finding the first and last non-masked lines in the middle of k-space. These low-frequency lines are subsequently used as the GRAPPA algorithm's calibration region. Then, the variable *num_low_freqs* stores the number of low-frequency lines in the k-space, and the variable *pad* computes the padding

necessary to center the low-frequency lines in the mask tensor. The variable *calib* stores the calibration region derived from the masked k-space tensor.

As inputs to the grappa function are the masked k-space tensor and the calibration region. The *kernel_size* parameter is set to (5, 5) and *coil_axis* is set to 0 to indicate that the coil axis is the first axis of the input tensors. The *preprocessed_masked_kspace* tensor is obtained by using the *to_tensor* function to convert the GRAPPA algorithm's output, which will be computed from the *pygrappa* library [23], to a tensor. The final return value of the *apply_grappa* function is the *preprocessed_masked_kspace* tensor.

The following images depict the process of selecting a slice of the brain from an MR file (a) as the ground truth (goal), and then masking this image using an 4x and 8x acceleration factor to simulate undersampled data (b). The slice is then reconstructed using the grappa algorithm, which takes around 6 seconds, resulting in the reconstructed image (c). As shown in Figure 4.3, in the case of the 4x acceleration factor, the SSIM metric varies between 0.74 for the undersampled image and the ground truth image, indicating significant data loss due to masking. However, our preprocessing algorithm improves the SSIM value to 0.77, resulting in a more accurate and visually pleasing image for further analysis. In the other case, with the image even more masked, the SSIM values drop to 0.60 with the acceleration factor set to 8x. Using the GRAPPA algorithm, the quality represented by SSIM increases to 0.68. Nevertheless, for specialists, an image that is still degraded may not be suitable for analysis. This motivated us to introduce another technique to further process the image before feeding it into the main neural network.

Code 4.1

```
def apply_grappa(masked_kspace, mask):
    def get_low_frequency_lines(mask):
        l = r = mask.shape[-2] // 2
        while mask[..., r, :]:
            r += 1

        while mask[..., l, :]:
            l -= 1

        return l + 1, r

    l, r = get_low_frequency_lines(mask)
    num_low_freqs = r - l
    pad = (mask.shape[-2] - num_low_freqs + 1) // 2
    calib = masked_kspace[:, :, pad:pad + num_low_freqs].clone()
    preprocessed_masked_kspace =
    ↪  grappa(tensor_to_complex_np(masked_kspace),
    ↪  tensor_to_complex_np(calib), kernel_size=(5, 5), coil_axis=0)
    return to_tensor(preprocessed_masked_kspace)
```

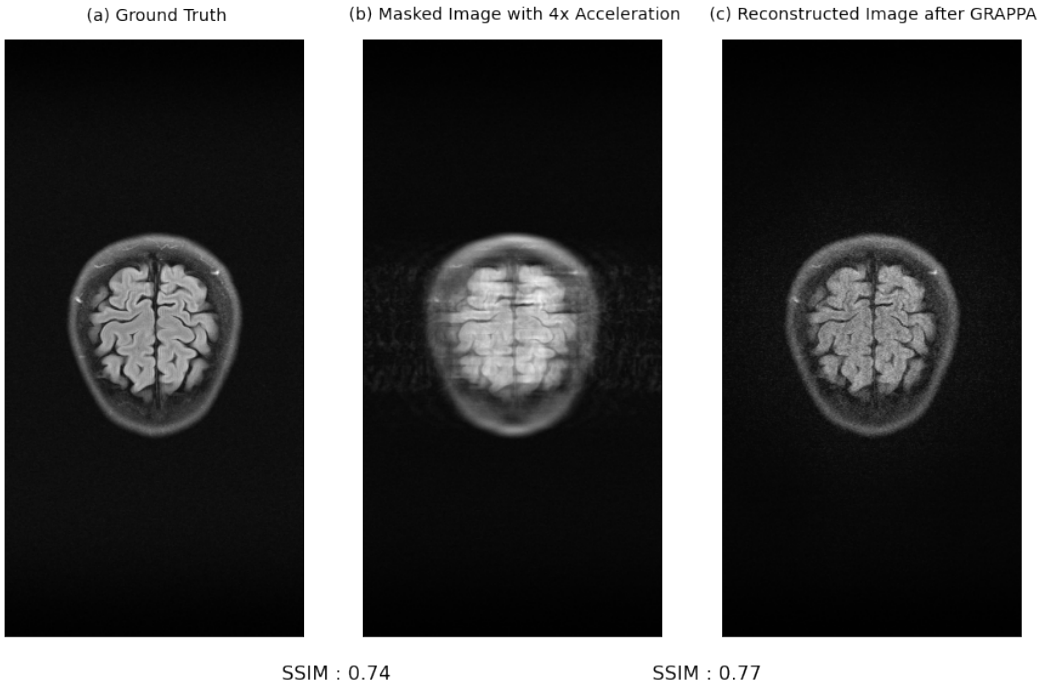

Figure 4.3. Simulation of GRAPPA on a 4x acceleration factor data

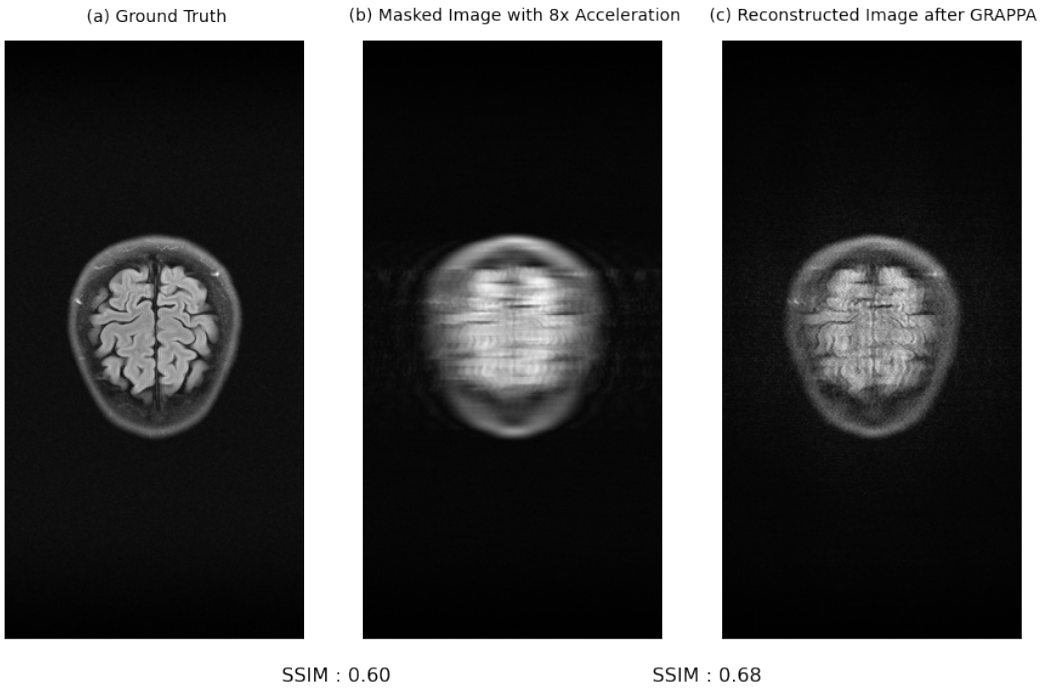

Figure 4.4. Simulation of GRAPPA on an 8x acceleration factor data

4.3 ESPIRiT

Prior to the introduction of the ESPIRiT, SENSE and GRAPPA were the most frequently used parallel MRI reconstruction techniques. SENSE is a coil-by-coil reconstruction method that uses fully sampled reference data to estimate the coil sensitivities, which are then used to unfold the aliased images. Separate coil sensitivity maps are necessary for SENSE, though they can be time-consuming to acquire and risk including errors if not estimated correctly. GRAPPA, on the other hand, is a data-driven approach that uses the variations in coil sensitivity to infer the missing k-space data from nearby acquired data points. GRAPPA relies on the regularity of the sampled k-space data rather than explicit knowledge of the coil sensitivities. When used alone, GRAPPA may experience noise amplification and limited accuracy in highly undersampled data.

By combining the advantages of SENSE and GRAPPA, ESPIRiT [30] is a novel strategy that fills the gap between them. The central concept of ESPIRiT is to simultaneously estimate the coil sensitivity maps and the missing k-space data using an eigenvalue decomposition method. This is accomplished by rewriting the parallel MRI problem as an eigenvector problem, where the coil sensitivities are represented by the eigenvectors with the largest eigenvalues and the missing k-space data is encoded by the eigenvectors with the smallest eigenvalues.

The foundation of ESPIRiT is the idea of subspace-based methods, which have been extensively applied to image and signal processing. The calibration matrix for the ESPIRiT method is built from the acquired k-space data, which captures the inherent correlations between various coil channels and the spatial organization of the coil sensitivities.

$$C = [S_1, S_2, ..., S_n]^H [y_1, y_2, ..., y_n] \tag{4.1}$$

where n is the number of coils, H is the Hermitian transpose, and y_i denotes the k-space data acquired by the i-th coil.

The underlying structure of the coil sensitivities and the missing k-space data are then revealed by decomposing the calibration matrix into its eigenvalues and eigenvectors, using the following equation:

$$C = U\Lambda V^H \tag{4.2}$$

where U and V are unitary matrices, and Λ is a diagonal matrix containing the eigenvalues of C.

The coil sensitivity maps are created using the eigenvectors linked to the largest eigenvalues, and they are then used to unfold the aliased images similarly to SENSE.

$$S_{est} = [u_1, u_2, ..., u_k] \tag{4.3}$$

Utilizing the innate structure of the acquired data, the eigenvectors linked to the smallest eigenvalues are used to estimate the missing k-space data in a GRAPPA-like manner.:

$$y_{miss} = [v_{k+1}, v_{k+2}, ..., v_n]^H [y_1, y_2, ..., y_n] S_{est}^H \tag{4.4}$$

where S_{est}^H is the conjugate transpose of the estimated sensitivity maps.

The unfolded images and the estimated missing k-space data are then combined with an inverse Fourier Transform to create the reconstructed images.

$$x = F^{-1}(S_{est} y_{unfolded} + y_{miss}) \tag{4.5}$$

where F^{-1} denotes the inverse Fourier Transform, and $y_{unfolded}$ is the unfolded k-space data obtained using the estimated sensitivity maps.

The map of all G_q's eigenvalues and eigenvectors is shown in Figure 4.5. For data from an eight-channel head-coil, G_q has been calculated as the Fourier transform of the reconstruction operator W using a 24 x 24 k-space calibration region and a 6 x 6 kernel size. Eigenvalues are arranged on the left in order of decreasing magnitude.

Where there is signal in the image, Eigenvalues equal to one are visible. Right: The eigenvector maps' magnitude and phase for each eigenvalue at all spatial locations. The eigenvectors that correspond to eigenvalues of 1 seem to be sensitivity maps, as expected. The sensitivities' magnitude and phase closely match the bottom row's individual coil images' magnitude and phase. The eigenvectors are defined only up to multiplication with an arbitrary complex number. Due to this, each location's eigenvector norm is normalized to one, and the 8th channel is used as a reference with zero phase.

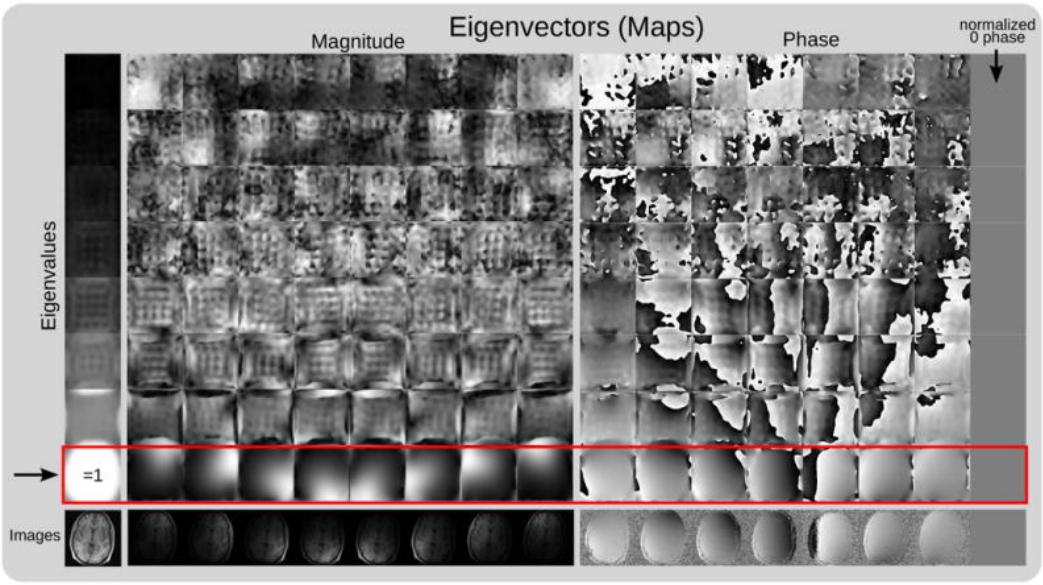

Figure 4.5. Explicit sensitivity maps from autocalibration data using eigenvalue decomposition. *Source :* Original Paper

ESPIRiT has a number of benefits over conventional parallel MRI techniques. The first benefit is that it offers a uniform framework for parallel MRI reconstruction that combines the advantages of both GRAPPA and SENSE. Second, it simplifies the reconstruction process and lowers computational complexity by not requiring separate estimation of the coil sensitivity maps and the missing k-space data. Thirdly, because the eigenvalue decomposition method is less sensitive to noise and errors in the acquired data, it is inherently robust to noise and artefacts.

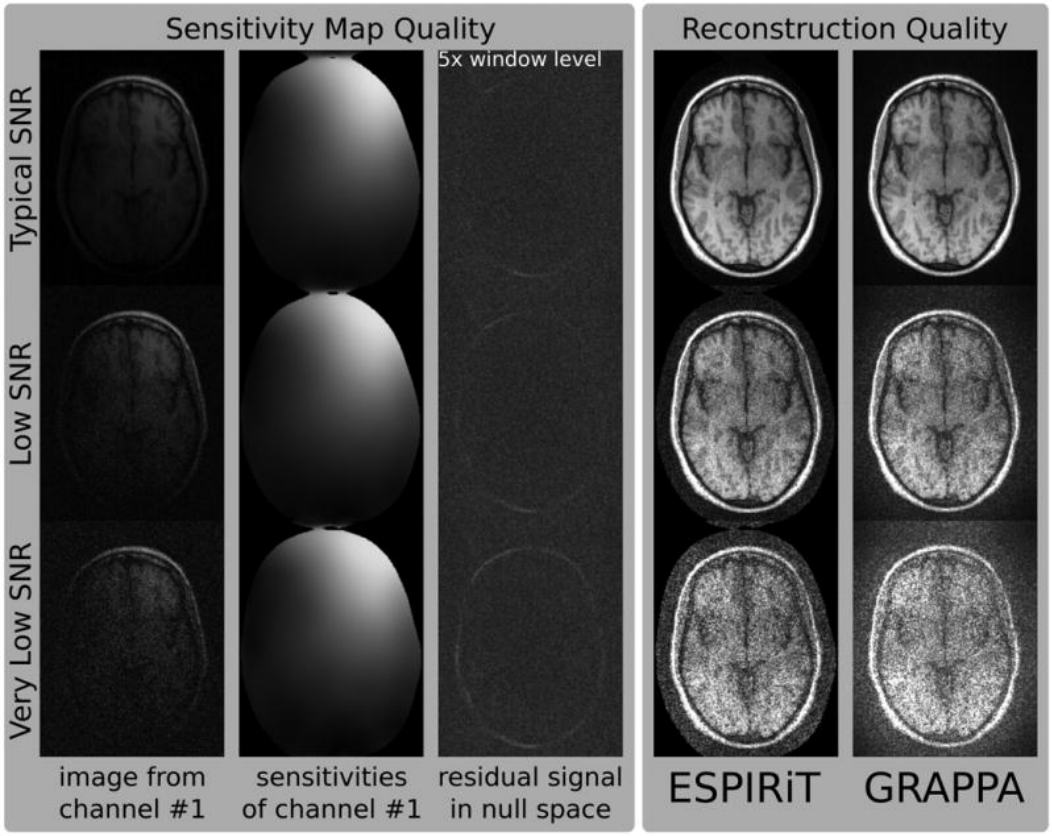

Figure 4.6. Reconstrution after addition of noise *Source :* Original Paper

By introducing noise to fully-sampled data (noise levels: 1, 10, and 20), it has been possible to examine the impact of noise on the calibration of the sensitivity maps. In the first column of the Figure 4.6 the images that are fully sampled and correspond to the first channel at various noise levels. In the second column is shown the first channel's estimated sensitivity map as determined by ESPIRiT calibration. The projection of the fully sampled original data (scaled by a factor of 5) onto the nullspace determined by the sensitivities is shown in the third column. The results indicate that ESPIRiT, working alone without the intervention of neural networks or other algorithms, produces more accurate results than GRAPPA.

Another illustration of the variations between undersampled MRI data reconstruction using various algorithms is shown in Figure 4.7. The residual signal is reduced to its minimum by using the NLINV algorithm and ESPIRiT directly on the slice, but it is not completely reduced by SENSE.

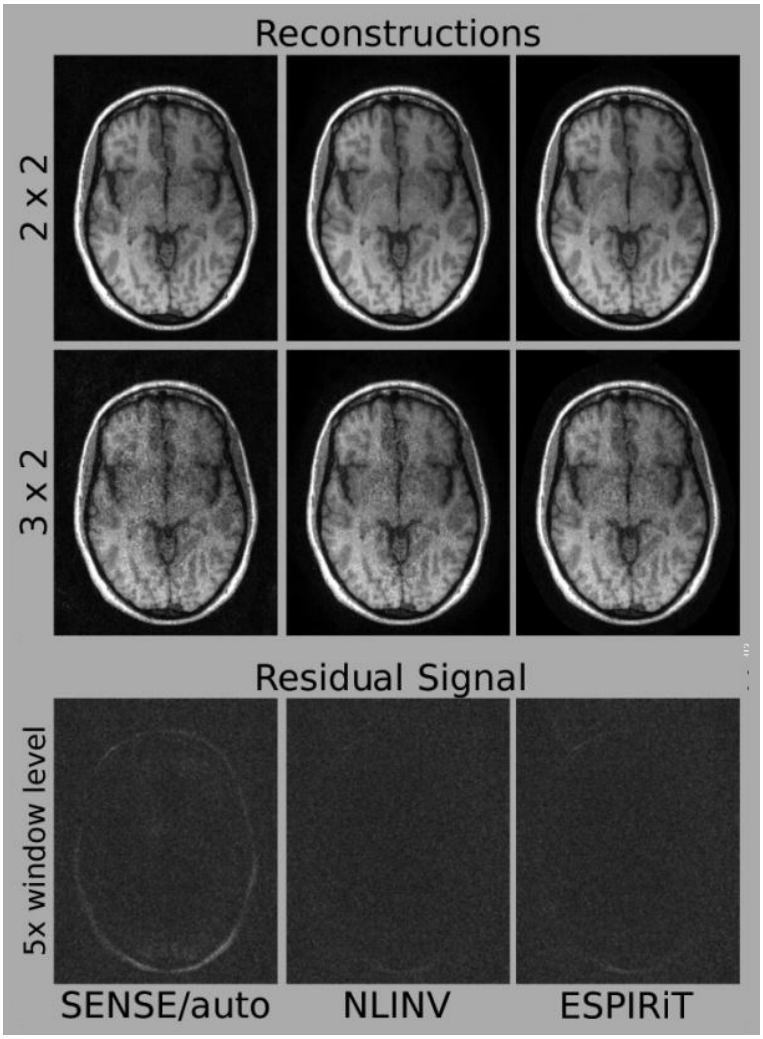

Figure 4.7. Images of human brain undersampled and reconstructed using SENSE, NLINV, GRAPPA and ESPIRiT *Source : Original ESPIRiT Paper*

NLINV is another reconstruction algorithm used to produce high-quality images from MRI data that is highly undersampled. It is a non-linear inverse reconstruction method that involves iteratively solving a non-linear problem to estimate the underlying image from the acquired k-space data, using the idea of regularized non-linear inversion. Regarding the specific implementation of ESPIRiT (Code 4.2), the *standardize_multi_coil_image* function takes in the k-space data and reconstructed image as input and returns the combined image. It first performs a few transformations of the image axes and then creates and runs the SigPy implementation [25] of the ESPIRiT calibration function with the k-space data as the only input. This process may take a few seconds and will return ESPIRiT coil sensitivity maps of the same size as the k-space data. The function then combines the coil sensitivity maps with the image data using element-wise multiplication and summation along the coil dimension. The idea behind coil combination is to weight each coil image by its corresponding sensitivity map so that the resulting combined image has a higher signal-to-noise ratio (SNR) and reduced artefacts.

Code 4.2

```
def standardize_multi_coil_image(kspace, image):

# Permute k-space data so that the coil dimension is the last
kspace_perm = np.moveaxis(kspace, 0, 2)

# Add a batch dimension to k-space data
kspace_perm = np.expand_dims(kspace_perm, axis=0)

# Create an ESPIRiT calibration object using the k-space data
espirit = mri.app.EspiritCalib(kspace)

# Compute the ESPIRiT operator using the calibration object
S = espirit.run()

# Combine the coil images using the ESPIRiT operator
M_comb = np.sum(np.multiply(np.conj(S), image),axis=0)

# Add a batch dimension to the combined image and return it
M_comb = np.expand_dims(M_comb, axis=0)

return M_comb
```

4.4 Implementation of the Preprocessing Algorithms

In Code 4.3 is shown a fundamental class that applies both GRAPPA and ESPIRiT transformations to the data before feeding it to the neural network. The arguments of this class are the input k-space of shape (*num_coils*, *rows*, *cols*), the mask to apply on the data, the target image (ground truth), a few attributes related to the h5 image, the file name of the image, and the serial number of the slice. It returns a tuple containing the zero-filled processed image, the reconstruction target, the mean and standard deviations used for normalization, the filename, and the slice number [31]. Specifically, the k-space data is first converted to a PyTorch tensor using the *to_tensor* function. Then, if a mask function was provided, a mask is applied to the k-space data using the *apply_mask* function. Next, GRAPPA is applied to try to reconstruct the masked image. Since GRAPPA works in the k-space but ESPIRiT works with both the k-space data and the zero-filled solution (real image data), the Inverse Fourier Transform is used to get the zero-filled solution, which is then passed as input to the *standardize_multi_coil_image* function. This function returns the calibrated and reconstructed image follwing ESPIRiT algorithm. Note that the function takes complex Numpy arrays as input, so both tensors are first transformed into Numpy arrays. After this, a few minor transformations are applied to the data, such as cropping the data to the correct size based on the target image and taking the absolute value of the data. Since the challenge is multicoil, Root-Sum-of-Squares is applied to get the final image with all the coils in one image. Finally, the tensor is normalized, and the final tuple is returned.

Code 4.3

```
class UnetDataTransform_Grappa:
    def __call__(
        self, kspace, mask, target, attrs, fname, slice_num):

        # transform the input k-space into a Tensor
        kspace_torch = to_tensor(kspace)

        # apply mask
        masked_kspace, mask = apply_mask(kspace_torch,
        self.mask_func, seed=seed)

        # apply Grappa
        masked_kspace = apply_grappa(masked_kspace, mask)

        # Apply inverse Fourier transform to get the actual image
        image = fastmri.ifft2c(masked_kspace)

        # apply ESPIRiT
        image = standardize_multi_coil_image(
        tensor_to_complex_np(masked_kspace),
        tensor_to_complex_np(image))

        # Transform the resulting image into a Tensor
        image = to_tensor(image)

        [...]
        # Returns the tuple with processed data
        return UnetSample(
            image=image,
            target=target_torch,
            mean=mean,
            std=std,
            fname=fname,
            slice_num=slice_num,
            max_value=max_value)
```

Chapter 5

Training and Results

The training was conducted entirely on a single Tesla V100-SXM2 with 32GB of memory, which was provided to us by the 'Istituto Nazionale di Fisica Nucleare' (Italian National Institute for Nuclear Physics), and took around a week to complete for each training instance (4x and 8x). The difference between these two training cases is that for the 4x track, I set the acceleration factor to 4 during data masking, while for the 8x track, I set it to 8. Consequently, the entire dataset was masked with the respective acceleration factor prior to the application of preprocessing algorithms and the start of neural network training.

The model was trained using SSIM as the loss function for both tracks since it is the main metric that I aimed to maximize in this project. To achieve this, I set the loss function to (1-SSIM). A mechanism was implemented to save the checkpoints of the model with the highest SSIM values over the validation set, which was used as a measure of the model's performance on the validation data.

5.1 4x Track

The acceleration factor R is defined as the ratio of the amount of data required for a fully sampled image to the amount collected in an accelerated acquisition [7]. This implies that when using a 4x acceleration factor, only 75% of the potential data

will be sampled. Figures 5.1 and 5.2 provide a clear illustration of the impact of the undersampling technique. To create these figures, I selected samples from the dataset and plotted them side-by-side with the corresponding undersampled images, which were obtained using a 4x acceleration factor.

The 4x track was easier to train, and the neural network, equipped with the preprocessing methods, had no difficulty in converging to a minimum loss result. A learning rate of 10^{-3} was used for the first 40 epochs of the training process. After 40 epochs, the learning rate was decreased to 10^{-4} to help the model converge to a better minimum. This gradual decrease in the learning rate is a common technique known as learning rate decay, and it is often used to improve the performance of neural networks during training.

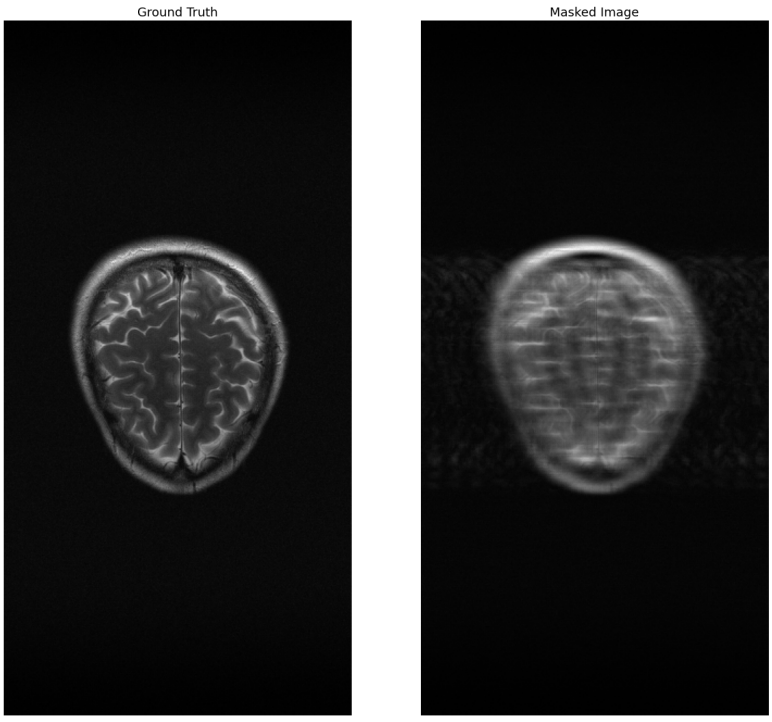

Figure 5.1. Randomly taken MR images from the dataset, on which is applied an 4x mask

Figure 5.3 displays the progression of the loss function during the initial training phase. The three lowest loss function values are highlighted, with an all-time low of 0.03225. It is noteworthy that these values were obtained directly from the training

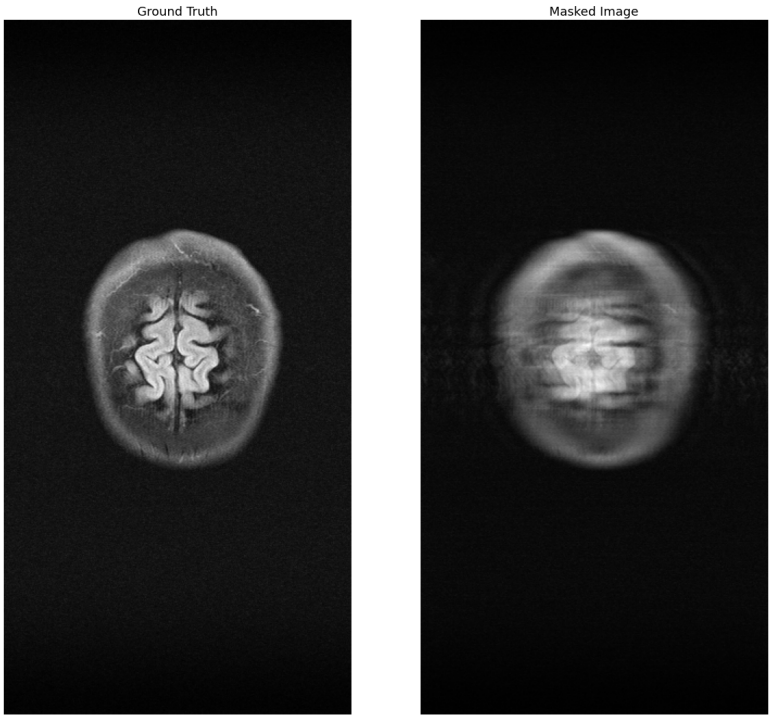

Figure 5.2. Randomly taken MR images from the dataset, on which is applied an 4x mask

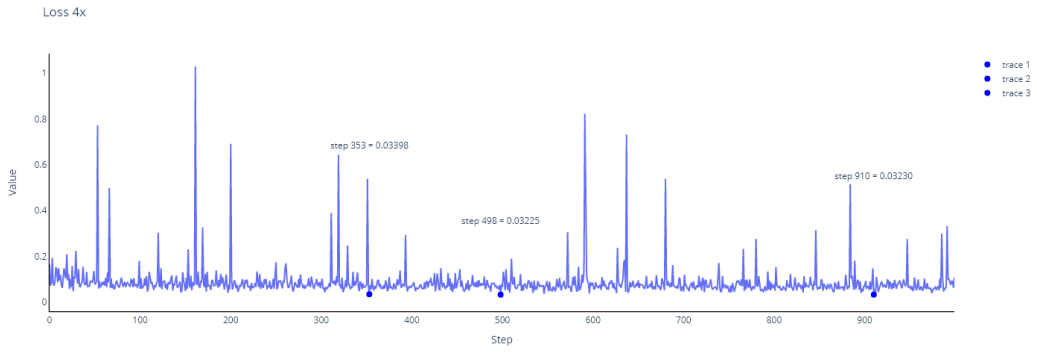

Figure 5.3. Loss function progression over 50 epochs in the 4x track

data, and therefore, their attainment is relatively straightforward. Despite this, the corresponding SSIM values for these optimal loss function values are remarkable, reaching an impressive value of 0.96775.

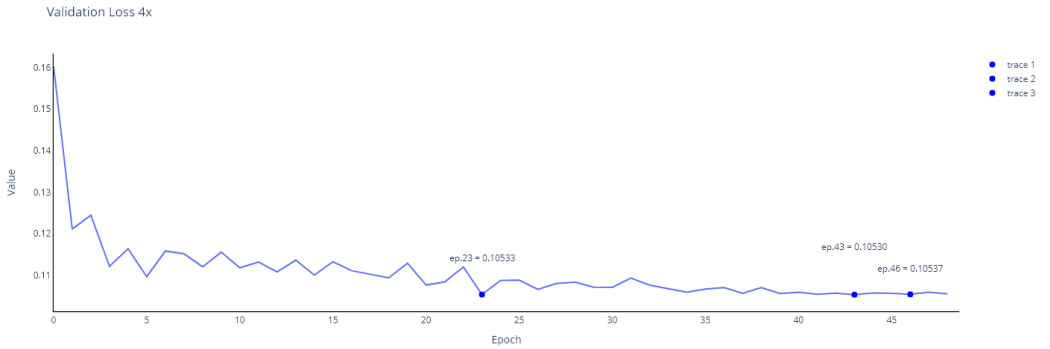

Figure 5.4. Validation Loss function progression over 50 epochs in the 4x track

The validation loss graph (Figure 5.4) provides a more comprehensive illustration of the learning evolution throughout the 50 epochs. The model's performance improves rapidly in the first few epochs, attributable to the high learning rate, but then gradually stagnates and cannot achieve lower values for a prolonged period. Around the end of the training, the optimal validation loss value of 0.10537 is the best obtained.

The metrics shown in Figures 5.5, 5.6 and 5.7 are coherent as the best results are got in the final epochs of the training, starting from the 40th to the 50th epoch. In particular, the best result for each metric is got at the 43rd epoch for the PSNR, 37.5703, and at the 46th epoch for both NMSE and SSIM, respectively 0.01446 and 0.9284.

On the following pages, there are figures showing MR images generated during the architecture training, with their corresponding SSIM values displayed to track the loss metric. The actual images are on the left, while the error compared to the ground truth is on the right. In a later picture (5.12), the corresponding ground truth images are presented, which are the actual images the model aims to generate, and from which the SSIM values are computed.

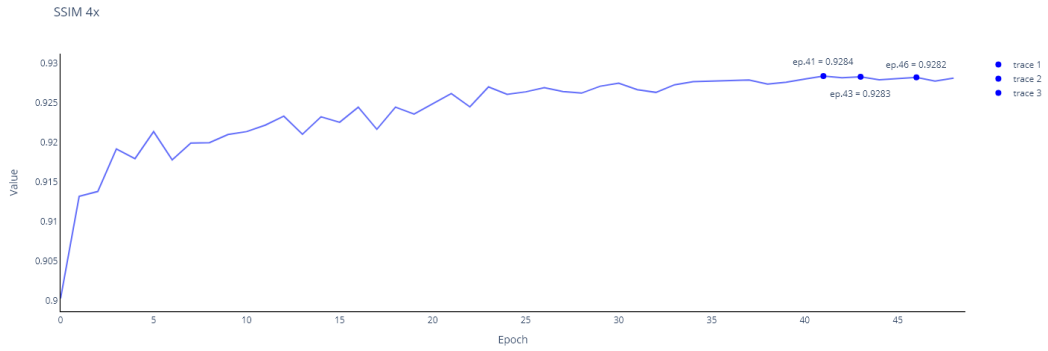

Figure 5.5. SSIM progression over 50 epochs in the 4x track

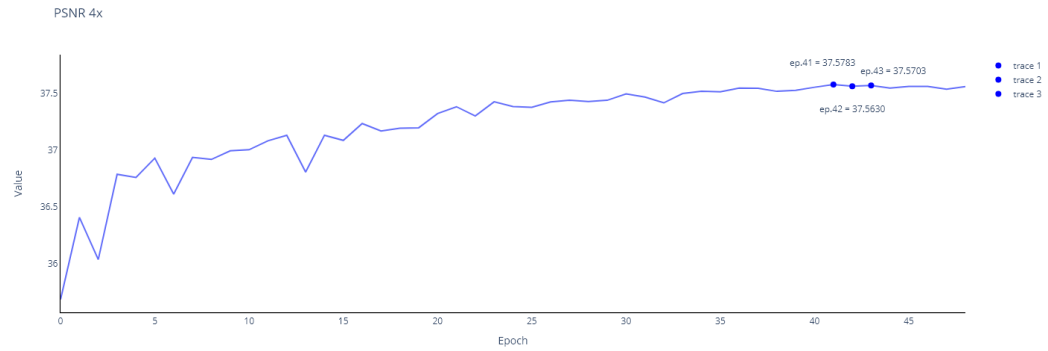

Figure 5.6. PSNR progression over 50 epochs in the 4x track

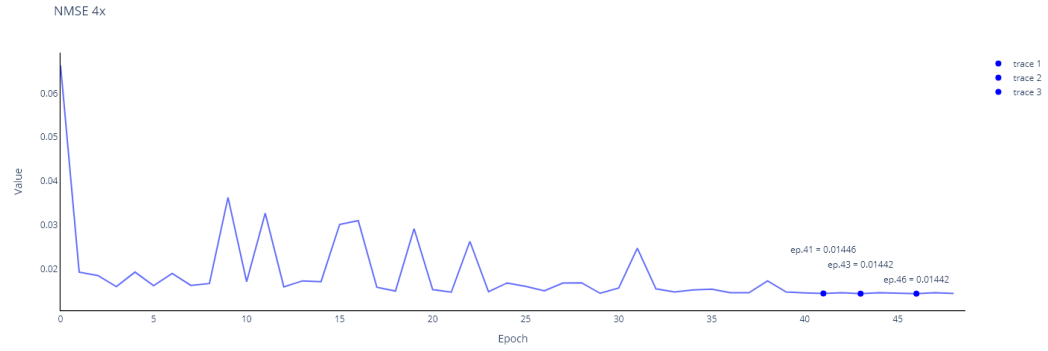

Figure 5.7. NMSE progression over 50 epochs in the 4x track

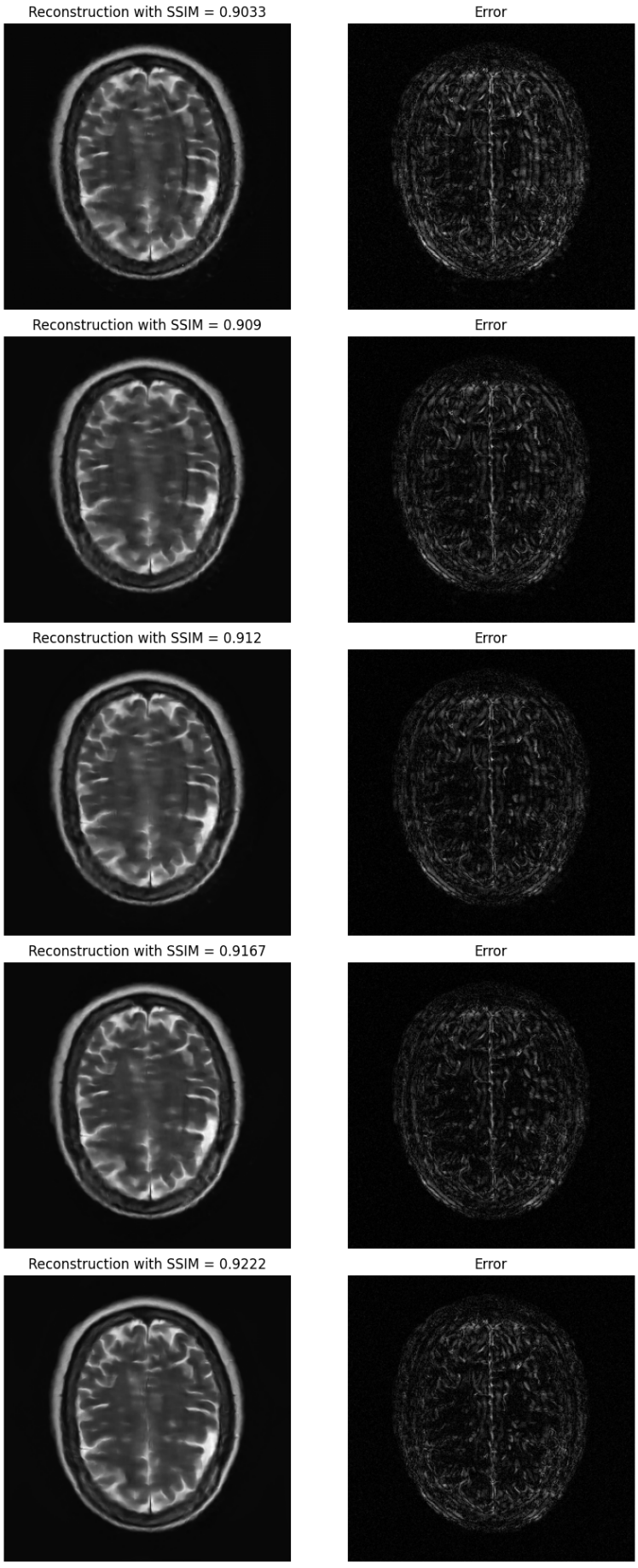

Figure 5.8. Image reconstruction improved over the epochs. File number 2676, 4x track

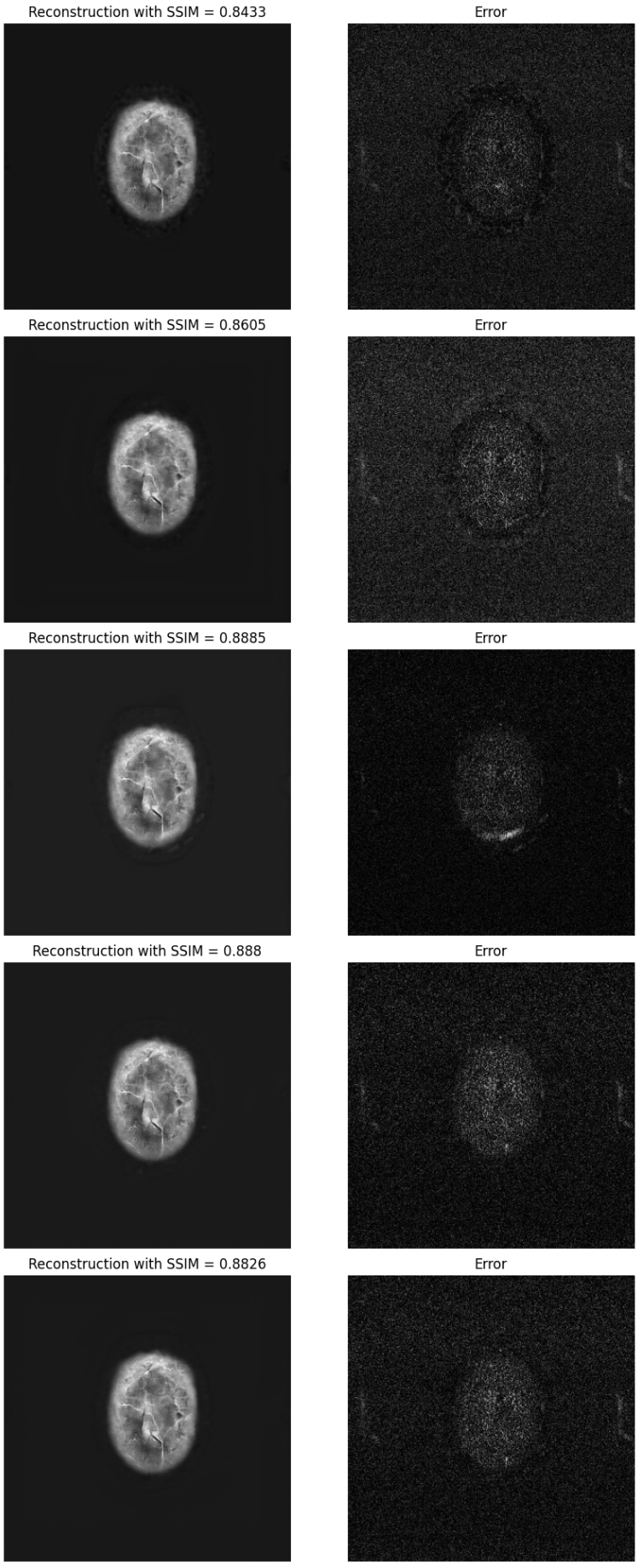

Figure 5.9. Image reconstruction improved over the epochs. File number 5331, 4x track

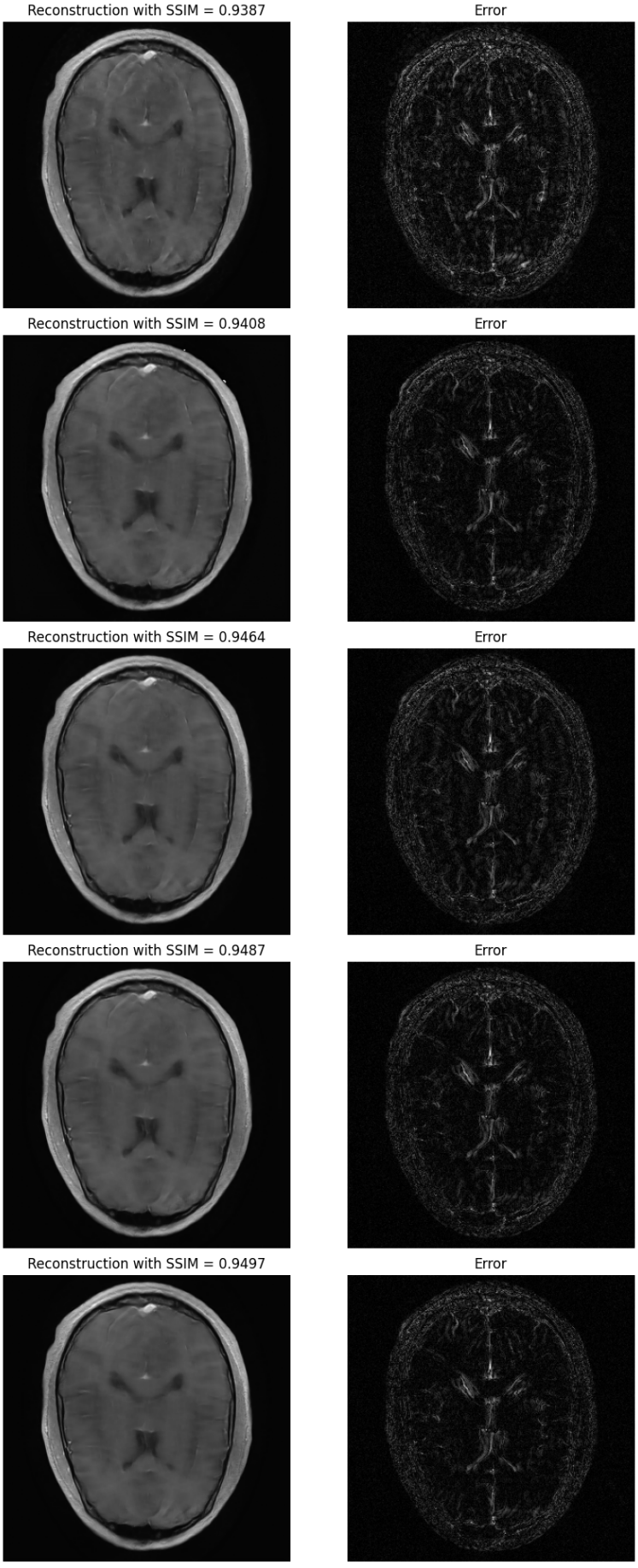

Figure 5.10. Image reconstruction improved over the epochs. File number 9605, 4x track

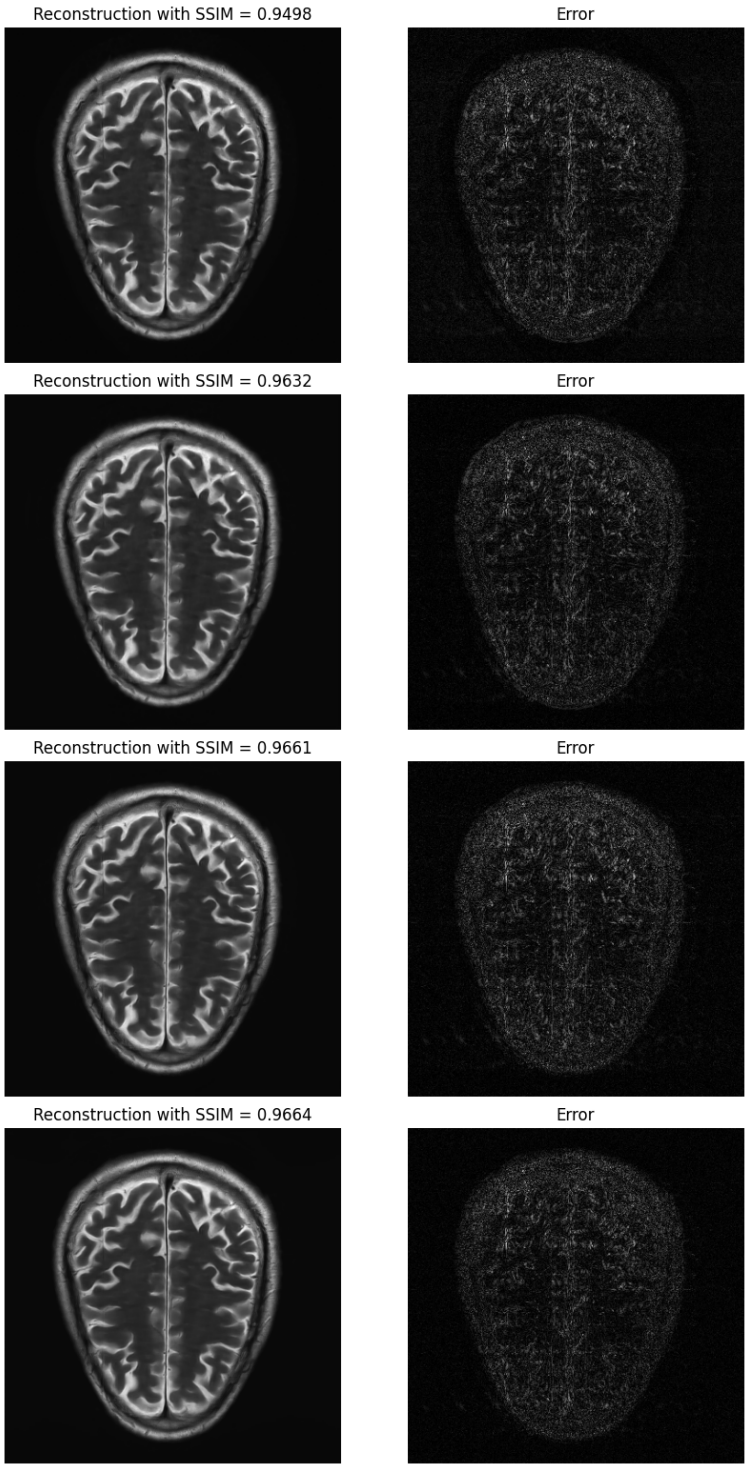

Figure 5.11. Image reconstruction improved over the epochs. File number 15800, 4x track

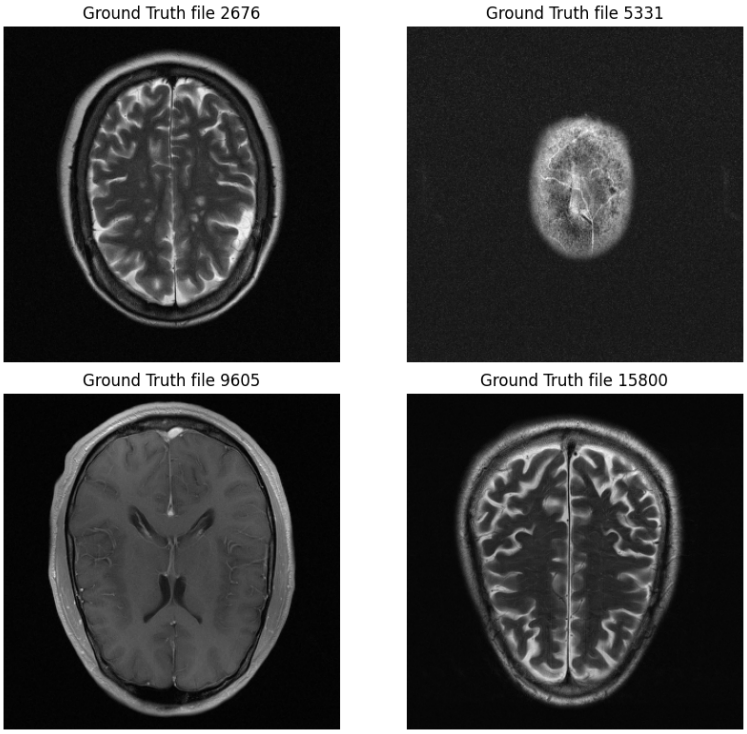

Figure 5.12. Ground Truth for the images

5.2 8x Track

In the 8x track, the results were consistently worse than in the previous track, due to the data being deeply undersampled, with only around 60% of the full data being kept. To address this issue, the training parameters were modified to better suit the more challenging training process.

One important modification was the implementation of an algorithm called '*auto_find_lr*', developed by the Lightning Team [12]. This algorithm tests different learning rates and selects the most appropriate one after a few steps of training; in our case, the learning rate that was preferred was 10^{-4}. This approach was chosen after careful consideration of the previous training in the 4x track, as high-weight updates could be problematic with such undersampled data.

In addition to the '*auto_find_lr*' algorithm, weight decay and dropout probability were introduced in this track to further improve the model's performance. Weight decay is a regularization technique that aims to prevent overfitting by adding a penalty term to the loss function that encourages the model to have smaller weight values. This is because larger weights can lead to overfitting as they may cause the model to become too specialized to the training data and not generalize well to new data. Overfitting is a common phenomenon in machine learning that we try to avoid which occurs when a model is excessively trained on a particular dataset, leading to the model learning the idiosyncratic features or noise in the data instead of the underlying patterns. Consequently, such a model may perform exceptionally well on the training data but may not generalize well to new, unseen data. Dropout is another technique used to prevent overfitting by randomly dropping out some of the neurons during training. This means that some of the neurons in the model will be ignored during each training iteration, which forces the model to learn more robust and generalizable features. Dropout can be thought of as a way to prevent the model from relying too heavily on any one feature or combination of features, which can lead to overfitting. These modifications were aimed at improving the

model's ability to handle the challenges posed by the deeply undersampled data and ultimately produce higher-quality results. I set both of these values to a standard value, specifically dropout probability was set to 0.5 and weight decay to 10^{-5}.

Moreover, I opted for 70 epochs in this run, for many reasons. First, increasing the acceleration factor from 4x to 8x required more training time to capture the finer details, and also the data has more complex features and patterns that the architecture needs to decrypt and that require additional training time to learn. Finally, using a larger number of epochs helped the model to gradually refine its understanding of the data over time with a lower learning rate than before. In pictures 5.13, 5.14 and 5.15, is shown a random slice of MR data that has been processed with an 8x accelerated mask.

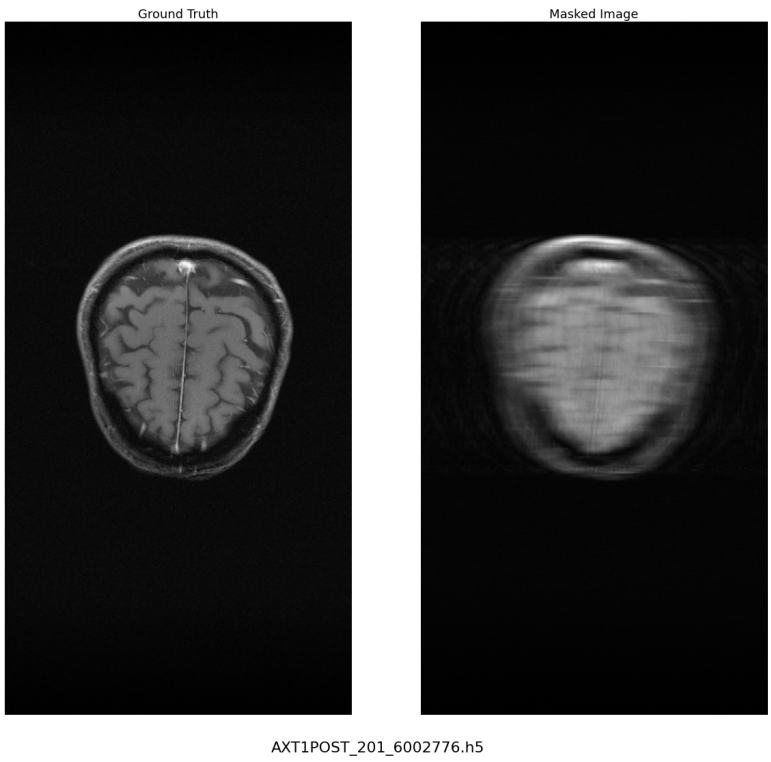

Figure 5.13. Randomly taken MR images from the dataset, on which is applied an 8x mask

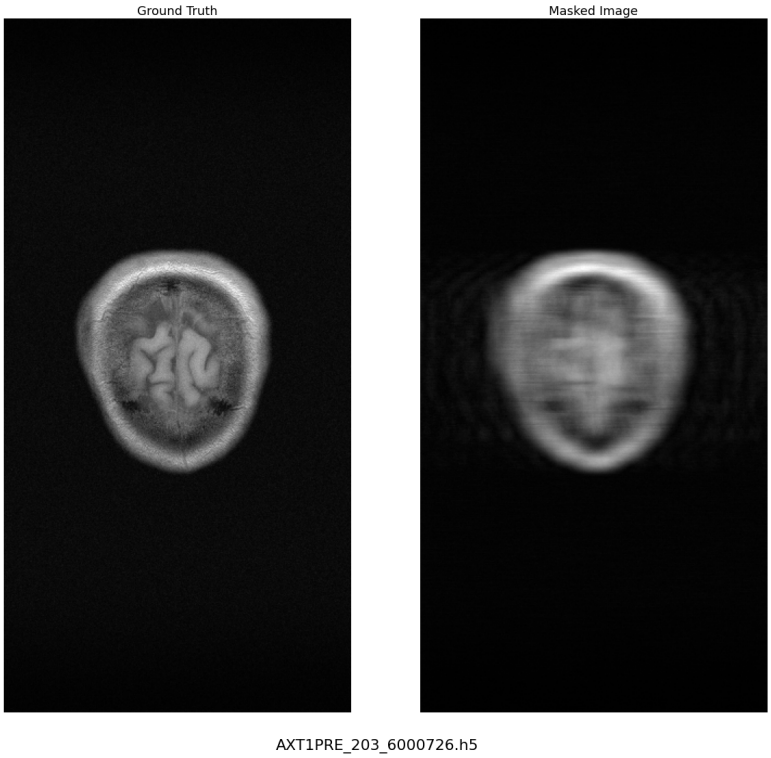

Figure 5.14. Randomly taken MR images from the dataset, on which is applied an 8x mask

Figure 5.16 displays the progression of the loss function during the 70 epochs of the training. The values shown are computed on the training dataset, and for this motivation many impressive values are reached. The best result is 0.04693, computed around half of the training process.

The validation loss graph (Figure 5.17) provides a crucial overview of the model's learning progress during the 70 epochs. With a lower learning rate, the model's performance steadily improves in the right direction throughout the training process. By the 40th epoch, a stable value is reached, indicating that the model has reached its optimal performance. Then, it slightly improves its SSIM value over the final epochs, reaching a minimum of 0.12511 at the 65th epoch.

The metrics in Figure 5.18, 5.19 and 5.20 confirm what we observed in the previous graphs, with valid results achieved across all metrics by the end of the training process. The maximum SSIM achieved was 0.8984, which is lower than

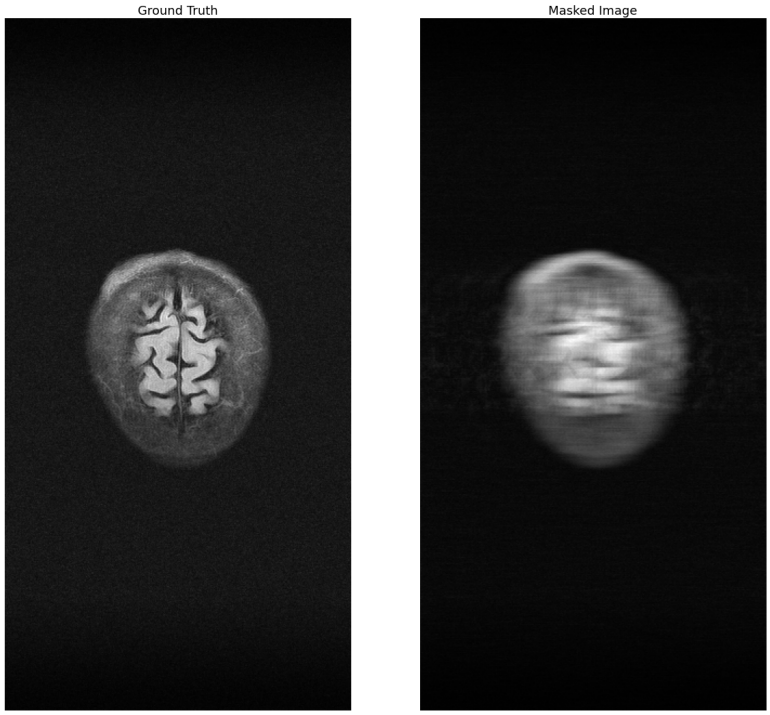

Figure 5.15. Randomly taken MR images from the dataset, on which is applied an 8x mask

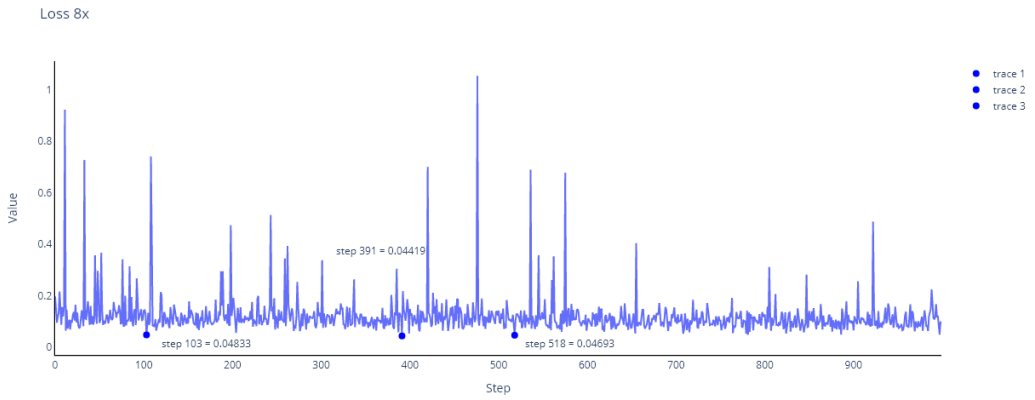

Figure 5.16. Loss function progression over 70 epochs in the 8x track

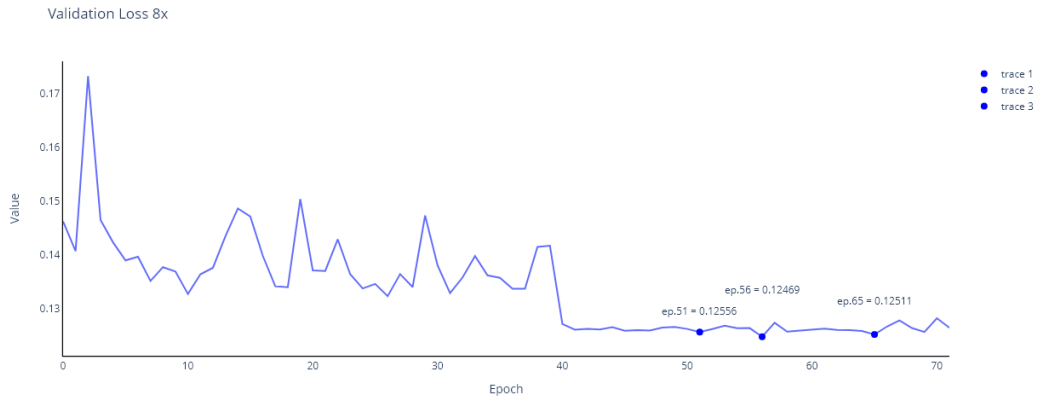

Figure 5.17. Validation Loss progression over 70 epochs in the 8x track

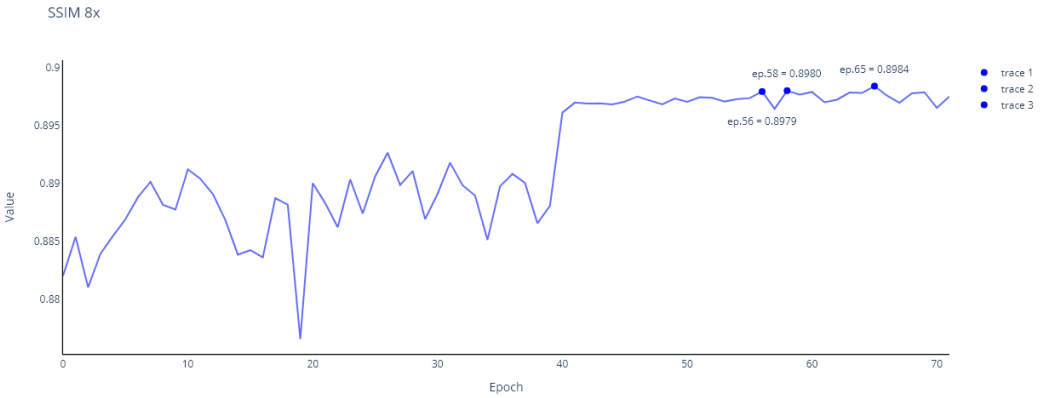

Figure 5.18. SSIM progression over 70 epochs in the 8x track

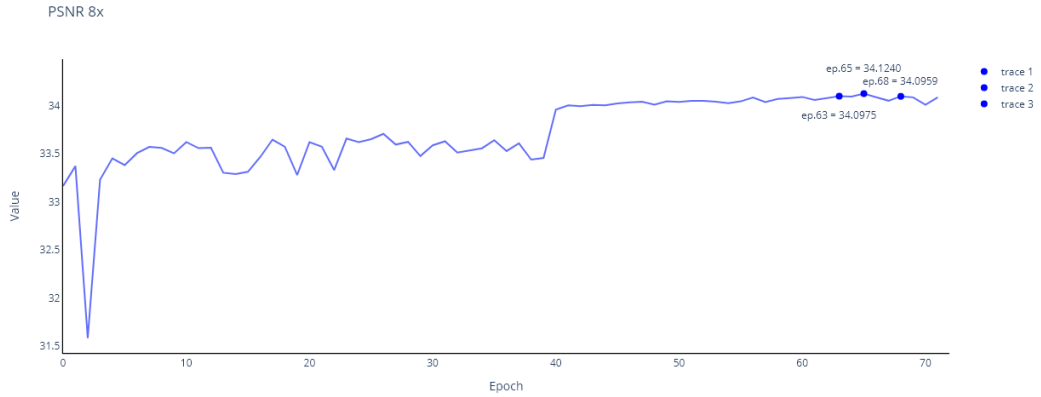

Figure 5.19. PSNR progression over 70 epochs in the 8x track

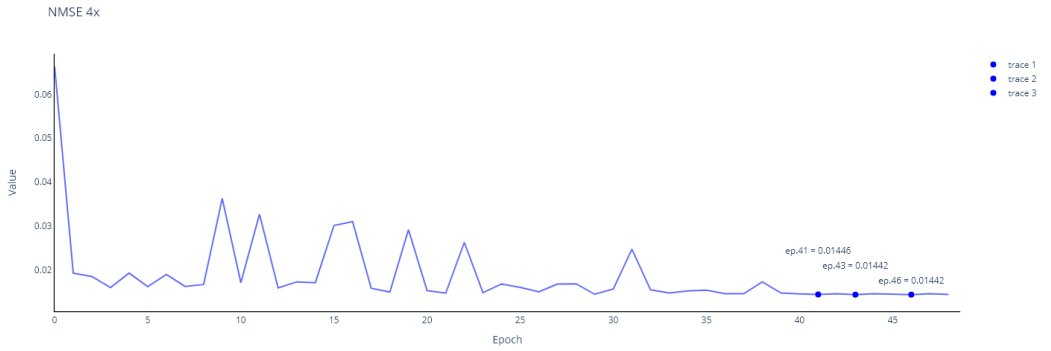

Figure 5.20. NMSE progression over 70 epochs in the 8x track

the previous track, but still a remarkable result considering how challenging it is to reconstruct images from only 60% of the available data.

In the following figures, I present some shots of the same slices of MR data taken every few epochs during the 70-epoch training process, for a total of four different images and five shots for every image. It is noticeable that the SSIM tends to increase over time, as expected. However, there are occasional dips in SSIM, and the results are consistently lower than those of the previous 4x track.

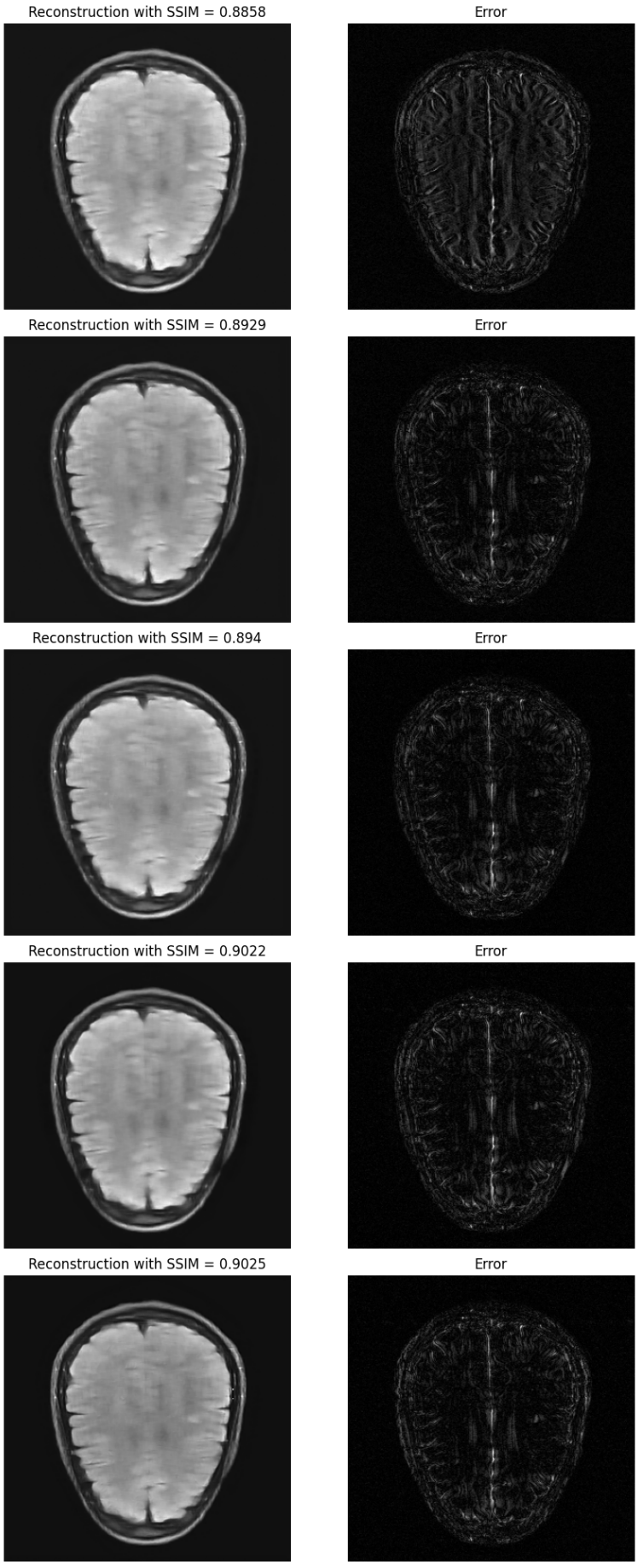

Figure 5.21. Image reconstruction improved over the epochs. File number 1493

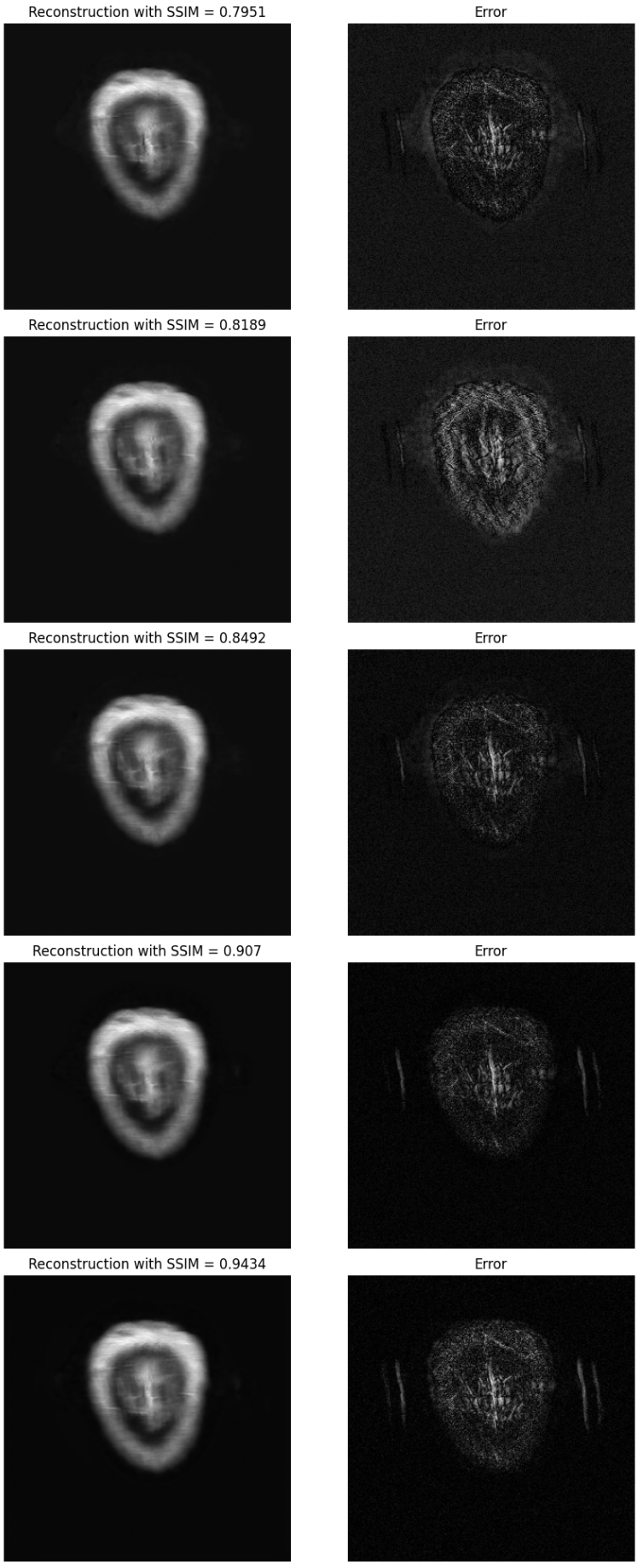

Figure 5.22. Image reconstruction improved over the epochs. File number 8193

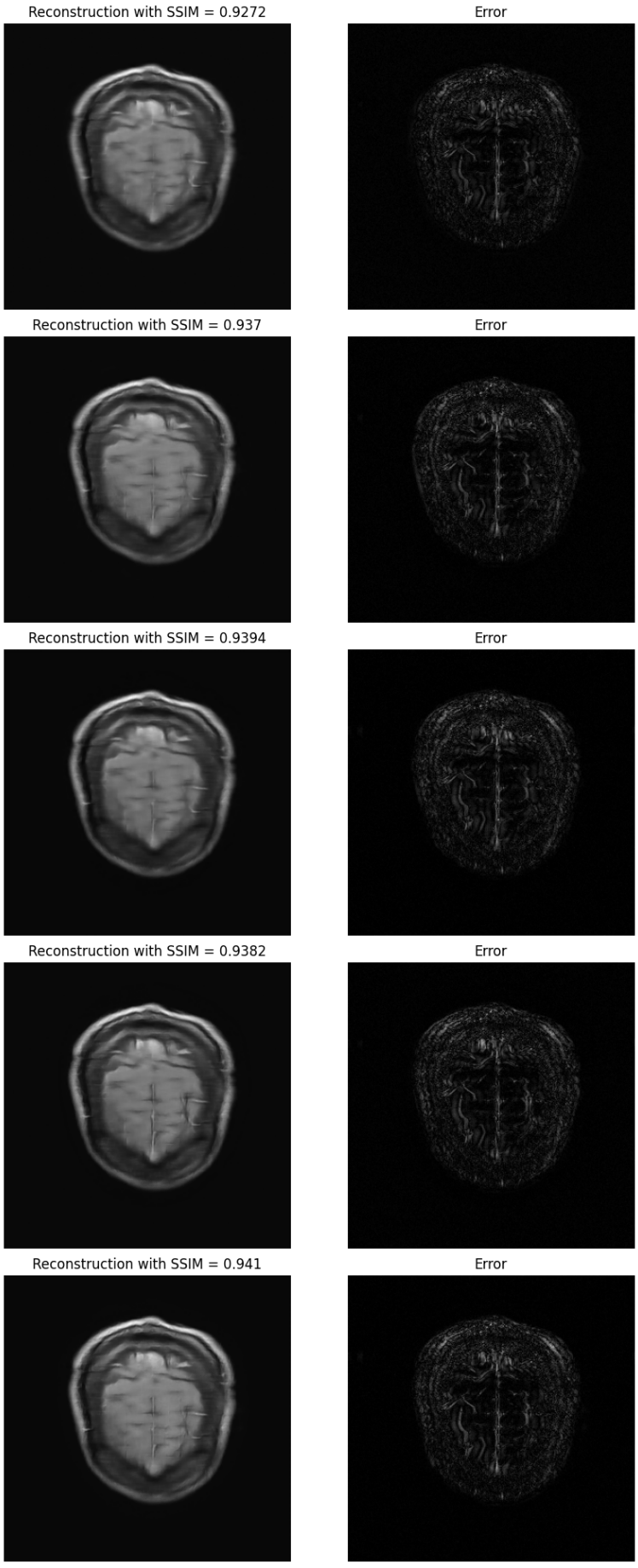

Figure 5.23. Image reconstruction improved over the epochs. File number 9977

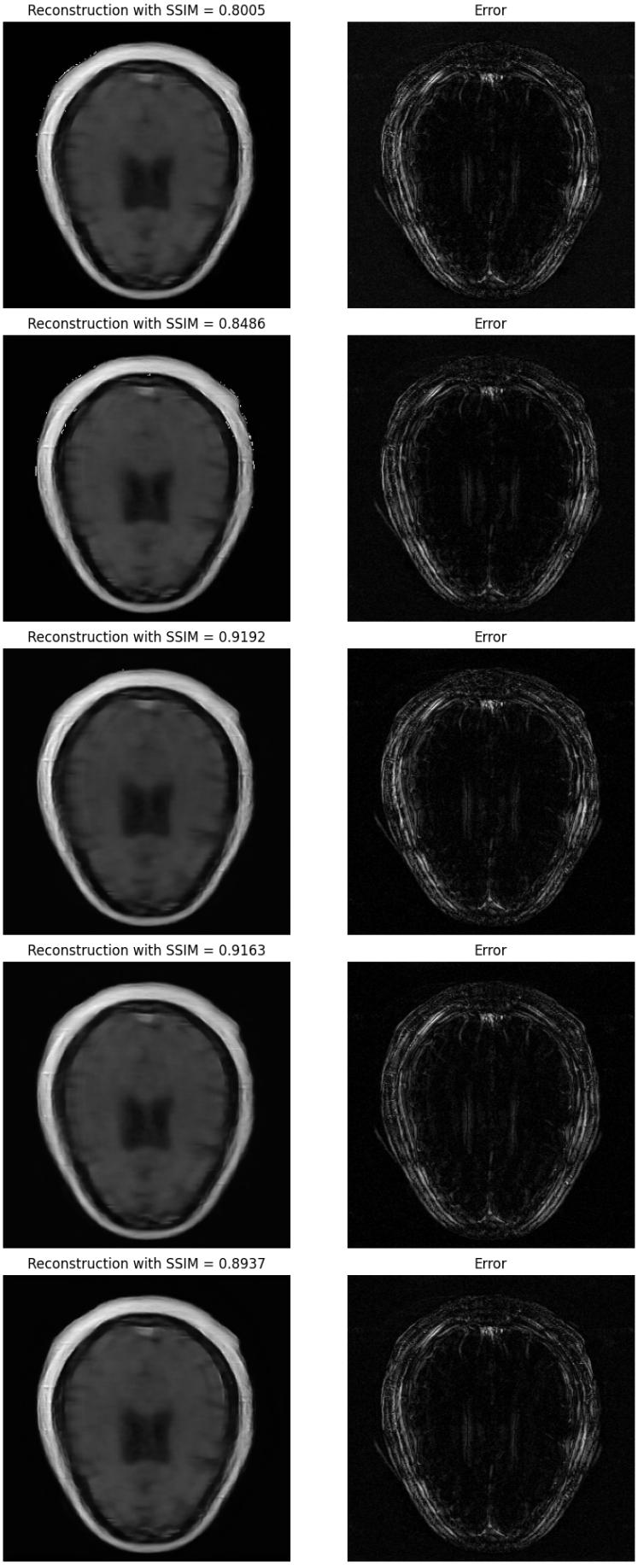

Figure 5.24. Image reconstruction improved over the epochs. File number 18076

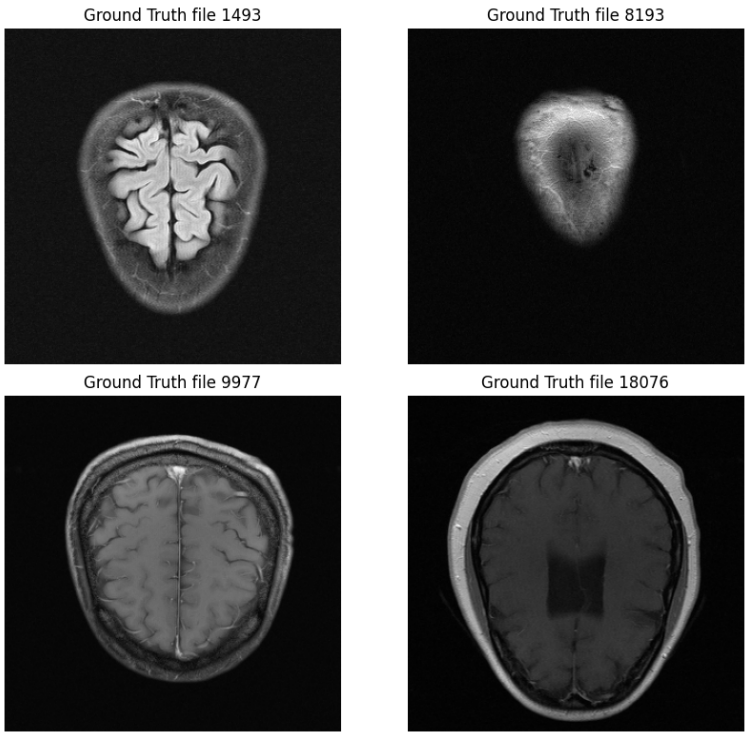

Figure 5.25. Image reconstruction improved over the epochs. File number 18076

5.3 Discussion

In this section, I will discuss the results of my work on accelerated MRI reconstruction. I addressed this problem by developing a neural network-based approach and incorporating two powerful preprocessing methods. To evaluate the effectiveness of my approach, I conducted experiments at two different acceleration factors, 4x and 8x. For the 4x acceleration factor, my neural network was able to converge to a minimum loss result effectively. I employed a learning rate of 10^{-3} for the first 40 epochs, followed by a decrease to 10^{-4} to facilitate convergence to a better minimum. I observed that the lowest loss function value was 0.03225, and the corresponding SSIM reached an impressive value of 0.96775 on the training split, while the average validation set score reached 0.9284.

After examining the best reconstruction of image file 15800 in the 4x track, we zoomed in on approximately the same region with different levels of magnification (Figure 5.26). While we observed that a few details were missing, overall the image quality was great. Similarly, in file 9605, as shown in Figure 5.27, the image quality is generally appreciable; however, there are some regions that appear blurred.

When examining Figure 5.28, it becomes apparent that there is an imprecision in the reconstruction. Specifically, the brain image is not demarcated correctly, making it easy to spot few errors.

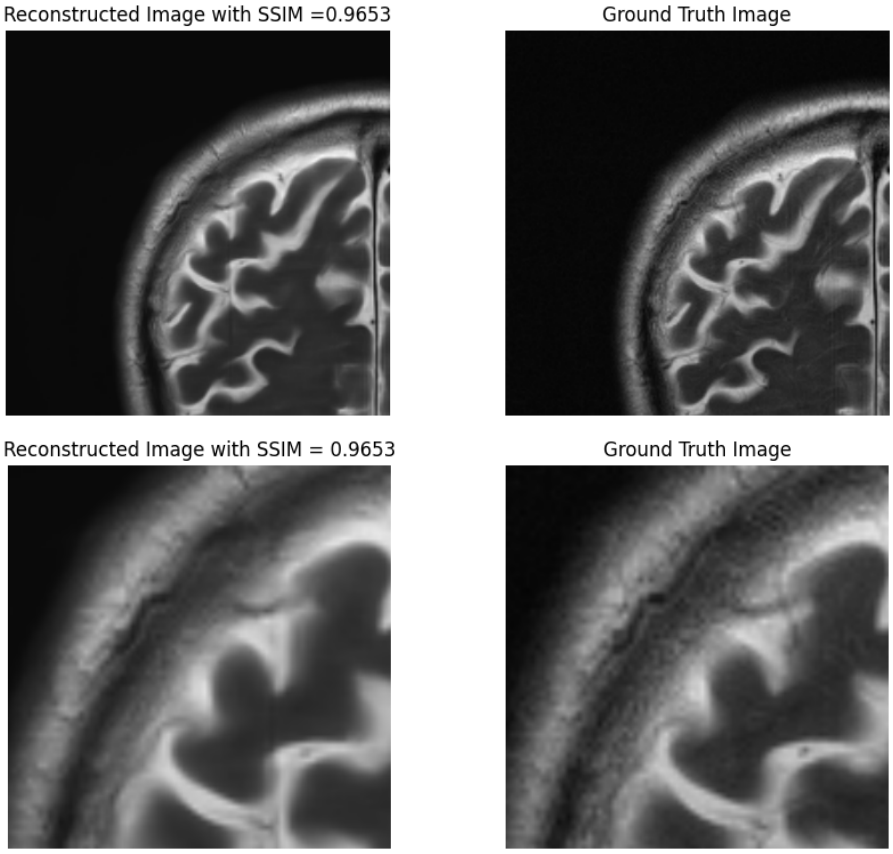

Figure 5.26. Images of file 15800 in the 4x track, at 2x and 3x magnification levels

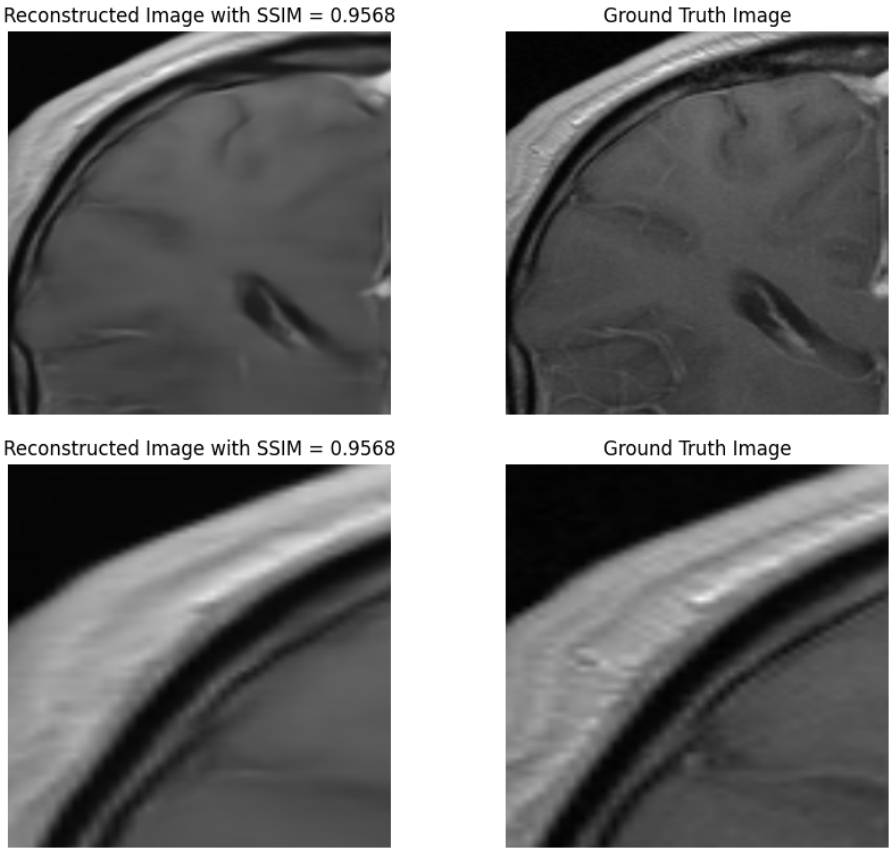

Figure 5.27. Images of file 9605 in the 4x track, at 2x and 3x magnification levels

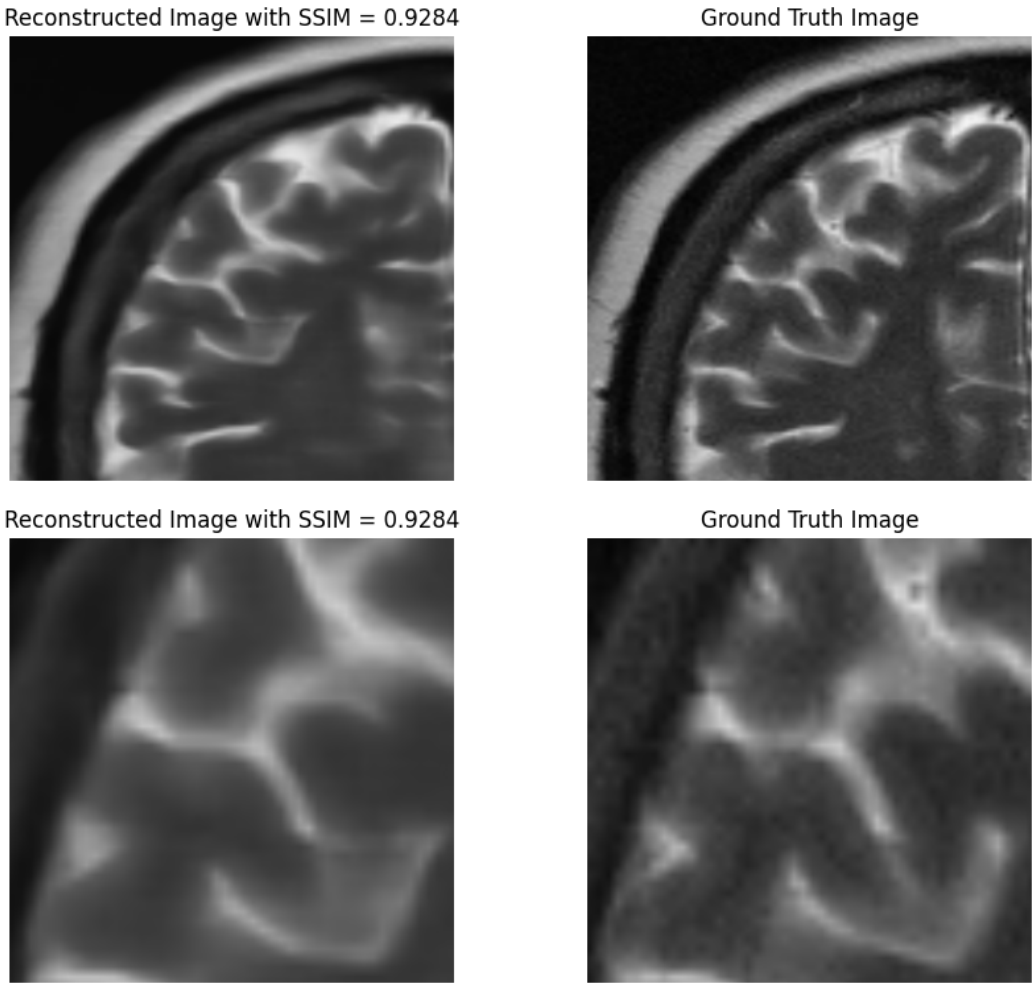

Figure 5.28. Images of file 2676 in the 4x track, at 2x and 3x magnification levels

The validation loss graph provided a comprehensive illustration of the learning evolution throughout the 50 epochs. The model's performance improved rapidly in the first few epochs, thanks to the high learning rate. Toward the end of the training, the optimal validation loss value of 0.10537 was the best obtained. The best results for each metric were achieved during the final epochs of the training, including a PSNR of 37.5703 at the 43rd epoch and NMSE and SSIM values of 0.01446 and 0.9284, respectively, at the 46th epoch. These results demonstrate the effectiveness of our model for the 4x acceleration factor.

For the 8x acceleration factor track, we faced a more challenging scenario due to the deeper undersampling of the data, with only around 60% of the full data being retained. To address this issue, we modified the training parameters and implemented the *auto-find-lr* algorithm, weight decay, and dropout probability. The *auto-find-lr* algorithm helped us select a more suitable learning rate of 10^{-4}, while weight decay and dropout probability were set to 0.5 and 10^{-5}, respectively, to prevent overfitting.

We increased the number of training epochs to 70 for the 8x track, which allowed the model to capture finer details and complex patterns in the data. The best loss function value achieved during training was 0.04693, and the validation loss graph showed a steady improvement in performance throughout the training process. The model reached a stable value by the 40th epoch and achieved a minimum validation loss of 0.12511 at the 65th epoch.

The metrics further confirmed the effectiveness of our approach for the 8x acceleration factor, with a maximum SSIM of 0.8984 achieved by the end of the training process. Although this value is lower than that obtained for the 4x track, it is still a remarkable result given the challenge of reconstructing images from only a small part of the available data.

By examining two specific files and zooming in 2x and 3x on a slice (Figures 5.29 to 5.31), we can compare the images to the ground truth and analyze them. We can observe that multiple regions of the image are too blurred to be useful. In this

8x track, the data was deeply undersampled, leading to a reconstruction that has limited practical value for specialists. While the result is exceptional considering how undersampled the data was, these figures would not be useful to radiologists as they lack details and exhibit blurry brain structures.

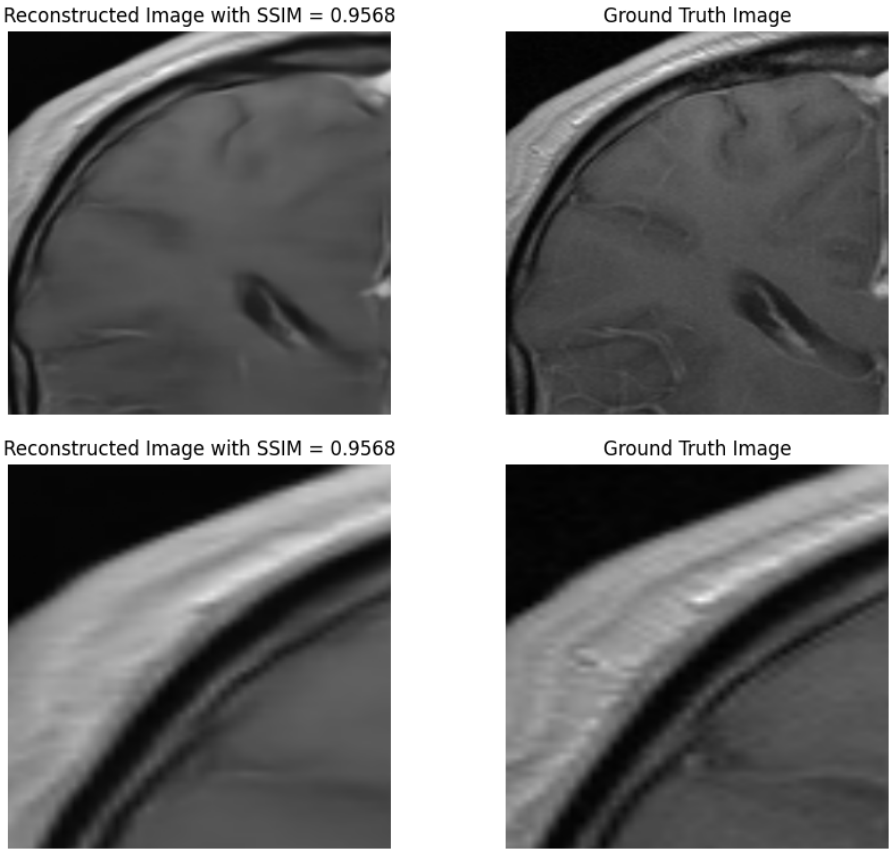

Figure 5.29. Images of file 9605 in the 8x track, at 2x and 3x magnification levels

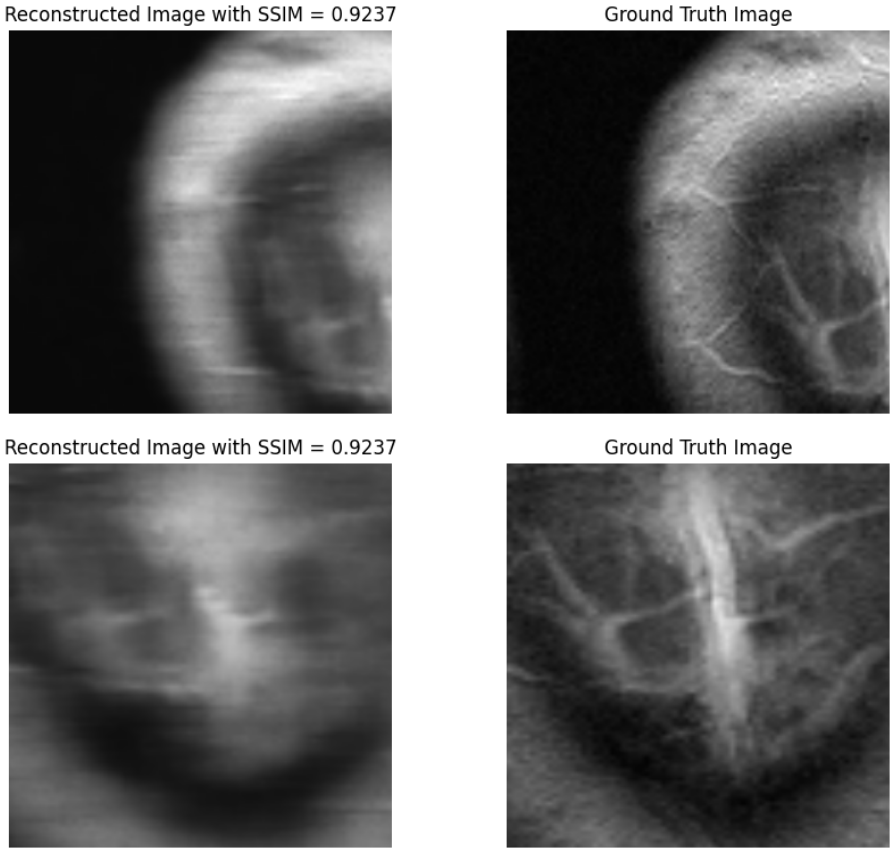

Figure 5.30. Images of file 8193 in the 8x track, at 2x and 3x magnification levels

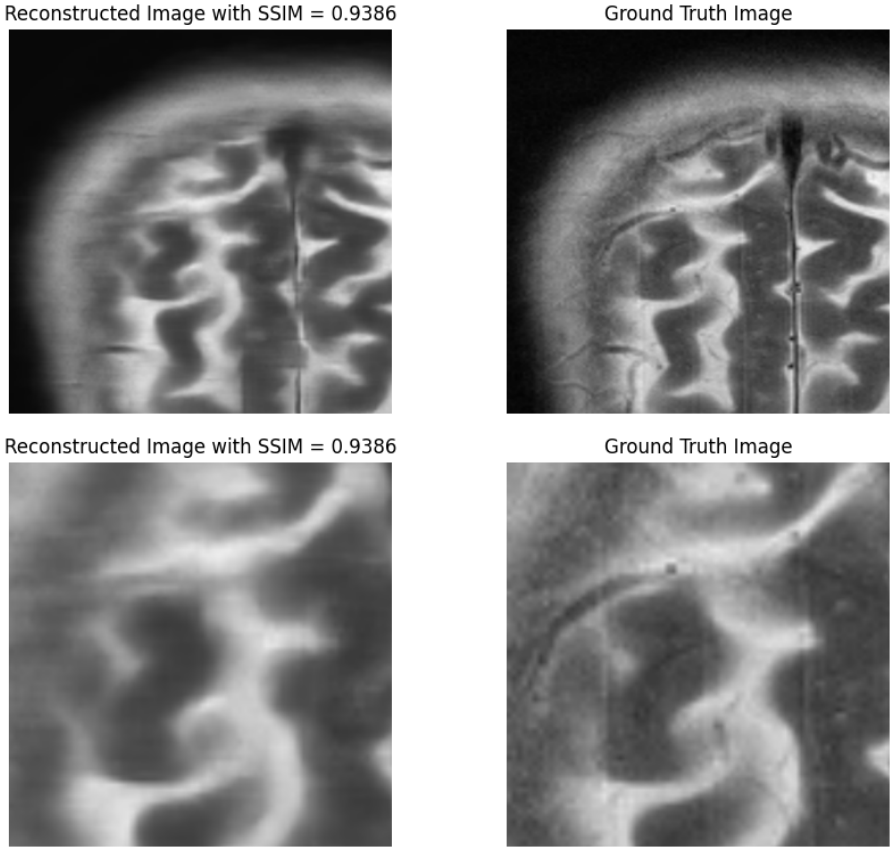

Figure 5.31. Images of file 8170 in the 8x track, at 2x and 3x magnification levels

In the next table, a summary of the teams' results along with our architecture, for both 4x and 8x tracks.

Table 5.1. Summary of SSIM results for each team in the FastMRI 2020

Team	4x Track	8x Track	Average
AIRS Medical	0.964	0.952	0.958
ATB	0.960	0.944	0.952
Neurospin	0.959	0.942	0.9505
Ours	**0.9284**	**0.8984**	**0.9134**

This table shows a summary of SSIM (Structural Similarity Index) results for each team in the FastMRI 2020 competition. The table indicates that the proposed architecture, represented by the "Ours" team, obtained slightly lower SSIM values compared to the top three teams (AIRS Medical, ATB, and Neurospin) in both the 4x and 8x tracks, with the "Average" column displaying the average SSIM score across both tracks. The "Ours" team had the lowest average SSIM score compared to the other teams in the FastMRI 2020 competition. To get a better idea of how the performance of my architecture compared to the other teams, I calculated the percentual difference between the SSIM score of the "ours" team and the SSIM scores of each of the other teams.

The percentual difference between our research and the team with the highest SSIM score, AIRS Medical, was 3.07%. The percentual difference between my research and the team with the second-highest SSIM score, ATB, was 2.52%. Similarly, the percentual difference between my research and the team with the third-highest SSIM score, Neurospin, was 2.37%. These percentual differences suggest that my architecture was less effective at reconstructing high-quality MRI images from undersampled k-space data compared to the algorithms of the other teams in the competition. However, it's important to note that percentual differences are just one way of comparing results, and experts have other ways to judge the results of reconstruction. My architecture has consistent and comparable results to these top teams, since these results are reasonably similar to the top 3 teams of the challenge.

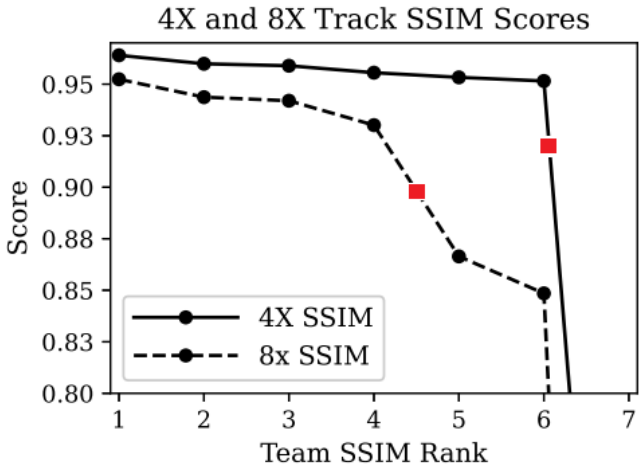

Figure 5.32. Comparison of SSIM scores

Figure 5.32 depicts a graph that displays the scores of the other teams in the competition side-by-side, with the addition of my team's result to the graph (in red). Based on the given results, the proposed architecture would rank just after the 6th team in the 4x track and just after the 4th team in the 8x track, indicating great performance in general. I also compared all the other metrics obtained with those obtained by the other teams and they are all coherent with the SSIM. Our architecture is just after the 6th team in 4x and after the 4th team in the other metrics such as NMSE and PSNR.

According to several interviewed radiologists, the images generated at a 4x acceleration factor are comparable to traditional scans and are suitable and usable for diagnostic purposes in a hospital setting. However, these experts did note that there were some artefacts present in T1POST images that may impact the accuracy of certain diagnoses. In the case of the 8x track, the radiologists were more critical in their feedback. Many of them stated that none of the submitted images were deemed acceptable. The radiologists were influenced by the presence of hallucinations, as seen in Figure 5.33, which are considered unacceptable. These hallucinations were particularly problematic when they mimicked normal structures that were either non-existent or actually abnormal. Despite having high SSIM scores, these images

were not optimized in terms of the presence of hallucinations. This suggests that the optimization metric used may not be appropriate for detecting these features. In reconstructions made by my architecture, I did not observe any artifacts of this kind.

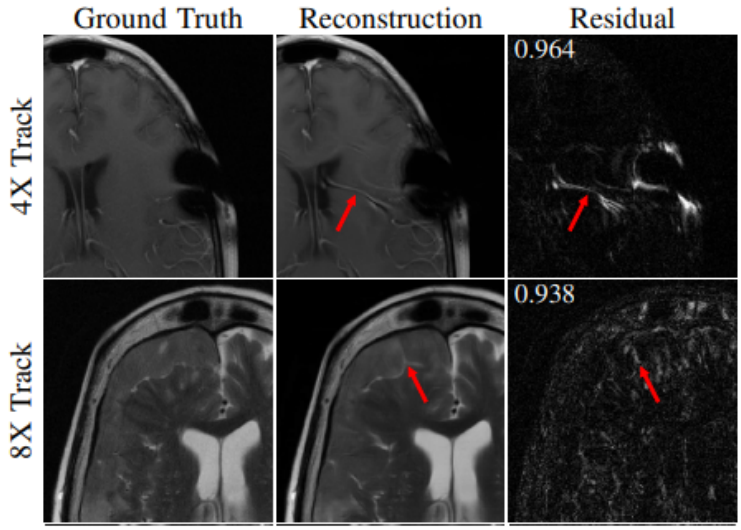

Figure 5.33. Hallucinations highlighted in reconstructions from other teams

Moreover, the results from the specific sequences are similar to ours, as all teams including ours had the lowest results in the FLAIR sequence, indicating that it was the most challenging to reconstruct.

Feedback was received from both participating and non-participating teams, with the second stating that they were unable to participate due to the high computational and storage requirements. In the future, researchers suggested that it would be beneficial to lower the barriers to entry, especially for academic groups that may have innovative methods but limited computing or storage resources.

Another feedback received was regarding the use of SSIM as the primary metric for evaluating the images, which tends to promote smoothing, rather than relying on radiologist interpretation. Both of these issues are of fundamental importance and require further investigation.

In conclusion, our work demonstrates the potential of deep learning techniques for accelerated MRI reconstruction. Our neural network-based approach was able to

effectively reconstruct high-quality images from undersampled k-space data, but the results at high acceleration factors (8x) were less operable by specialists due to their blurry and not detailed quality. After comparing my images and scores to those of other teams, it became apparent that the issues observed were common to their algorithms as well. This observation suggests that the problem may be inherent to the task itself, rather than specific to my approach. The results obtained for both the 4x and 8x acceleration factors showcase the effectiveness of our method and its ability to handle the challenges posed by undersampled data.

Chapter 6

Conclusions

In conclusion, this thesis has presented novel neural network architectures designed to address the challenges associated with reconstructing highly accelerated MRI data. We have developed a Residual-U-Net capable of handling complex data in conjunction with state-of-the-art preprocessing methods, namely ESPIRiT (eigenvalue approach to autocalibrating parallel MRI) and GRAPPA (Generalized autocalibrating partially parallel acquisitions). The successful implementation of these architectures demonstrates the potential of neural networks to enhance the computational efficiency of MR image reconstruction.

To assess the effectiveness of our approach, we trained the proposed architecture on the NYU MRI dataset using two distinct training instances with two undersampling techniques, reducing the amount of data acquired by factors of 4 (4x) and 8 (8x), respectively. The resulting reconstructions were compared to those achieved by teams participating in the FastMRI challenge, utilizing well-established metrics such as SSIM, PSNR, and NMSE.

Our findings indicate that our proposed methodology matches the performance of state-of-the-art methods employed by other teams in the FastMRI challenge. This contribution adds to the growing body of knowledge surrounding deep learning applications in medical imaging, with potential implications for the development of advanced techniques in accelerated MRI reconstruction.

When radiologists provided feedback on the reconstructed images, they reported that for the 4x track, there were often no significant differences between the reconstructions and the ground truth, and they could work with these images without any issues. This suggests that our results would likely be well-received by medical professionals in the context of the 4x track.

However, the results from the 8x acceleration factor instance study suggest that further optimization and enhancements are needed to produce higher-quality reconstructions for images obtained from deeply undersampled data. The lower accuracy results, blurred reconstructions, and hallucination features observed in the images indicate that these aspects require additional investigation and development, as they have not been thoroughly addressed by our work or that of other teams.

In light of these considerations, the algorithms studied have succeeded in significantly increasing the effectiveness of MRI scans by cutting scan times by up to four times while maintaining high image quality. These algorithms may increase the availability and affordability of MRI scans, which could significantly affect healthcare.

Future research should explore alternative training parameters or techniques to further improve model performance. Potential avenues include different regularization techniques, altering the loss function, extending training epochs, investigating new architectures, and ensuring the absence of image artefacts or false features (i.e., hallucinations). In summary, the findings of this study contribute to the field of MR image reconstruction and hold promise for advancements in medical imaging techniques.

Bibliography

[1] American College of Obstetrician and Gynecologists. *Guidelines for Diagnostic Imaging During Pregnancy and Lactation*, 2017. https://doi.org/10.1097/AOG.0000000000002355.

[2] Naif Bawazeer, Hella Vuong, Sophie Riehm, Francis Veillon, and Anne Charpiotb. *Magnetic resonance imaging after cochlear implants*, March 2019. https://doi.org/10.1016/j.joto.2018.11.001.

[3] Jason Brownlee. *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*, 2019. https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*, 2017.

[5] Wikipedia contributors. *Structural similarity*. https://en.wikipedia.org/w/index.php?title=Structural_similarity&oldid=1142355078.

[6] Li Deng and Dong Yu. *Deep Learning: Methods and Applications*, 2014. http://dx.doi.org/10.1561/2000000039.

[7] Deshmane, Gulani, Griswold, and Seiberlich. *Parallel MR imaging*, 2019. doi: 10.1002/jmri.23639. PMID: 22696125; PMCID: PMC4459721.

[8] Health Direct. *MRI scan*, 2021. https://www.healthdirect.gov.au/magnetic-resonance-imaging-mri.

[9] Bhumika Duttasep. *What are Skip Connections in Neural Networks?*, 2021. https://www.analyticssteps.com/blogs/what-are-skip-connections-neural-networks.

[10] Envision Radiology. *Facts about MRIs*, 2020. https://www.envrad.com/facts-about-mris/.

[11] Facebook AI and NYU Langone Health. *FastMRI*, 2020. https://fastmri.org/.

[12] Falcon and The PyTorch Lightning team. *PyTorch Lightning (Version 1.4) [Computer software]*, 2019. https://doi.org/10.5281/zenodo.3828935.

[13] Feger, Murphy, and Bell. *T1 mapping - myocardium*, 2015. https://doi.org/10.53347/rID-78604.

[14] Gaillard, Baba, and Bell. *MRI sequences, an overview*, 2015. https://doi.org/10.53347/rID-37346.

[15] Glorot, Bordes, and Bengio. *Deep Sparse Rectifier Neural Networks*, 2011. in Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, 2011, pp. 315-323.

[16] Simon Haykin. *Neural Networks and Learning Machines*, 2009. 3rd Edition.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*, 2015. https://arxiv.org/abs/1512.03385v1.

[18] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. *Multilayer feedforward networks are universal approximators*, 1989. https://doi.org/10.1016/0893-6080(89)90020-8.

[19] Sandeep Jain. *Dilated Convolutions*, 2023. https://www.geeksforgeeks.org/dilated-convolution/.

[20] Long Jonathan, Shelhamer Evan, and Darrell Trevor. *Fully convolutional networks for semantic segmentation*, 2015. https://arxiv.org/pdf/1411.4038.pdf.

[21] Valentin Lievin. *U-Net Model for Image Segmentation Problems using PyTorch 0.4*, 2018. https://github.com/vlievin/Unet.

[22] Tianrui Luo, Douglas Noll, Jeffrey Fessler, and Jon-Fredrik Nielsen. *A GRAPPA algorithm for arbitrary 2D/3D non-Cartesian sampling trajectories with rapid calibration*, 2019. https://pubmed.ncbi.nlm.nih.gov/31050011/.

[23] Nicholas McKibben. *Python implementations of GRAPPA-like algorithms.* https://github.com/mckib2/pygrappa.

[24] Matthew Muckley, Bruno Riemenschneider, Alireza Radmanesh, Sunwoo Kim, Geunu Jeong, Jingyu Ko, Yohan Jun, Hyungseob Shin, Dosik Hwang, Mahmoud Mostapha, Simon Arberet, Dominik Nickel, Zaccharie Ramzi, Philippe Ciuciu, Jean-Luc Starck, Jonas Teuwen, Dimitrios Karkalousos, Chaoping Zhang, Anuroop Sriram, Zhengnan Huang, Nafissa Yakubova, Yvonne Lui, and Florian Knoll. *Results of the 2020 fastMRI Challenge for Machine Learning MR Image Reconstruction*, 2020. https://arxiv.org/pdf/2012.06318.pdf.

[25] Frank Ong and Michael Lustig. *SigPy: A Python Package for High-Performance Iterative Reconstruction*, 2019. https://sigpy.readthedocs.io/en/latest/index.html.

[26] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*, 2015.

[27] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*, 2015.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*, 2015. https://arxiv.org/pdf/1505.04597.pdf.

[29] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*, 2019. https://arxiv.org/abs/1912.05911.

[30] Uecker, Lai, Murphy, Virtue, Elad, Pauly, Vasanawala, and Lustig. *ESPIRiT–an eigenvalue approach to autocalibrating parallel MRI: where SENSE meets GRAPPA.*, 2014. Magn Reson Med. 2014 Mar;71(3):990-1001. doi: 10.1002/mrm.24751. PMID: 23649942; PMCID: PMC4142121.

[31] Jure Zbontar, Florian Knoll, Anuroop Sriram, Tullie Murrell, Zhengnan Huang, Matthew Muckley, Aaron Defazio, Ruben Stern, Patricia Johnson, Mary Bruno, Marc Parente, Krzysztof Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdzal, Adriana Romero, Michael Rabbat, Pascal Vincent, Nafissa Yakubova, James Pinkerton, Duo Wang, Erich Owens, Lawrence Zitnick, Michael Recht, Daniel Sodickson, and Yvonne Lui. *fastMRI: An Open Dataset and Benchmarks for Accelerated MRI*, 2018. 1811.08839.

Desidero ringraziare tutti coloro che hanno contribuito al completamento del mio percorso accademico, sia coloro che mi hanno accompagnato da sempre, sia quelli che sono arrivati più recentemente nella mia vita.

Un ringraziamento speciale ai miei amici di sempre, Gabri e Ali, che mi sopportano e mi supportano dal lontano 2012. Siete stati la costante nella mia vita e non potrei chiedere amici migliori.

Voglio ringraziare anche tutta la Paula Gvng, composta da Angelica, Giorgia, Roberta, Bea, Martina, nonché le estensioni del cerchio, Chiara, Arianna e Socci.

Un grazie di cuore all'unico collega di cui ho mai avuto bisogno, Adriano, che ha commentato la mia tesi esclamando: "Ma è un nuovo cocktail?".

Un ringraziamento speciale anche ai nuovi arrivati, finti atleti ma veri amici: Michele, Riccardo, Flavia, e tutti gli altri. Siete stati una presenza preziosa durante tutto il mio percorso. Grazie per avermi fatto ridere e per avermi assecondato nelle mie idiozie ad atletica condividendo con me tante esperienze indimenticabili.

Desidero esprimere la mia profonda gratitudine verso Russo+Assogna, che sono stati per me come una famiglia romana durante tutto il mio percorso. Ringrazio di cuore Giulia, Marghe, Luca Piccolo, Luca Grande, Paola e Gianluca per avermi accolto a braccia aperte e per avermi fornito un ambiente così caloroso.

Desidero ringraziare in modo particolare la ragazza che è sempre stata al mio fianco, e che ha svolto un ruolo fondamentale nel raggiungimento di questo traguardo. Gippa, questa laurea magistrale è il nostro successo, perché tu hai sempre saputo incoraggiarmi, ascoltarmi e sostenermi, anche nei momenti difficili. Non riesco a ringraziarti abbastanza per esserti sempre presa cura di me, sia nei momenti felici che in quelli difficili. Grazie per non esserti mai stancata di me, anche dopo settimane insieme 24/24. Se sono qui oggi, lo devo in parte a te, e per questo ti sarò sempre grato.

Infine, voglio ringraziare la mia famiglia, che è sempre stata presente e non si è mai allontanata: Mamma, Papà, Laura, Simone, Davide e Beba. Grazie al vostro esempio, ho trovato la forza di perseverare negli studi, nonostante avessi sempre pensato di non essere all'altezza. Ora, guardando indietro, sono felice di dire che ce l'ho fatta, e sono grato per l'ispirazione che mi avete dato.