Deep-learning based reconstruction of dynamic non-Cartesian fMRI:

Application to silent imaging

Master Thesis in Medical Technology

by

Marius Eldevik Rusaas

Department of Physics and Technology
University of Bergen

May 31, 2023

Scientific environment

This study has been carried out at the Department of Physics and Technology at the University of Bergen. The work has been performed in collaboration with Mohn Medical Imaging and Visualization Center (MMIV.no) at the Department of Radiology at Haukeland University Hospital.

Acknowledgements

I would like to express my gratitude to everyone who has helped and supported me during this last year and in my work with this thesis. First I would like to thank my supervisors, Frank Riemer and Renate Grüner, for introducing me to this exciting project. Thank you, Renate, for your support, feedback and investment in this project. Thank you, Frank, for always being available with invaluable feedback, providing me with everything I needed and your help with any problems that occurred along the way. I also appreciate that you gave me the opportunity to participate in the Lund ISMRM Nordic Chapter and the MMIV conference last year.

Thank you to everyone at MMIV for making me feel welcome and for all the knowledge that you so happily share. I would also like to thank Lars Ersland, Kenneth Hugdahl and Alex Craven from HUS/UIB for acquiring the Looping Star sequence and making this project possible, as well as Florian Wiesinger, Anne Menini, Ana Beatriz Solana and Brice Fernandez from GE Healthcare for granting access to the k-space and trajectory data. I also express my gratitude towards the Human Connectome Project for access to their data.

I would also like to thank Oliver Marcel Geier and Frank Riemer for letting me present my work at the 2023 MAGNETOMforum in Kristiansand and thank you to Eli Nina Eikefjord and PRESIMAL for supporting me on my trip.

Thanks to all of my friends and fellow students for all the love and support throughout my whole degree. A special thanks to Sven Alrik and Maria for being with me through all the ups and downs these five years. You always made studying a lot more fun. Last, but not least, I would like to thank my family for always being there for me, and a big thank you to Matilde Kirkeli Strand for all your love and support.

Marius Eldevik Rusaas
Bergen, June 2023

Abstract

Purpose: "Looping Star" is a silent magnetic resonance imaging (MRI) sequence that can be used for functional MRI (fMRI) examinations of patients using auditory stimuli or with issues with auditory perception such as auditory verbal hallucinations in patients with schizophrenia. However, the 3D radial k-space sampling of this sequence results in reduced image quality. In recent years, deep learning models for improved image reconstruction have emerged. The purpose of this master's thesis is to examine the usability of deep learning image reconstruction methods for the Looping Star sequence with the goal of increasing image quality.

Methods: In this project, a resting-state fMRI (rs-fMRI) dataset from the Human Connectome Project (HCP) was used as a basis for creating a dataset suitable for training deep neural networks. The coordinates of an acquired Looping Star k-space trajectory were used to downsample the fully sampled k-space of both an unprocessed and a processed HCP dataset. The downsampled k-space samples were reconstructed using a non-Cartesian reconstruction method to get pairs of downsampled Looping Star images and ground truth HCP images. The unprocessed dataset was used to train a 2D-UNet and a 3D-UNet deep learning model for image reconstruction. The two models were compared, and the best of these was further trained on the processed dataset. The performance of the final model was then evaluated using standard image quality metrics as well as through an rs-fMRI analysis.

Results: The 3D-UNet outperforms the 2D-UNet for this image reconstruction task and significantly reduces the error between the network input and the ground truth. The 3D-UNet achieves a 97% reduction of the mean square error between the Looping Star input and HCP ground truth on the processed dataset. The 3D-UNet also manages to preserve enough of the voxel variations that rs-fMRI analysis shows correlated activity in the posterior cingulate cortex (PCC) while reducing noise in the voxel time-series.

Conclusion: In this project, a dataset consisting of downsampled Looping Star images

was created, and artifacts resulting from the reconstruction of these images were successfully removed using a deep learning reconstruction approach. The network comparisons suggested that using 3D convolutions is more valuable than a deeper network of 2D convolutions for images with global artifacts. Furthermore, it has been demonstrated that the trained neural network retains some BOLD signal variations despite the pertruding image artifacts of the Looping Star images.

Contents

Abbreviations

Adam	Adaptive movement estimation
ATP	Adenosine Triphosphate
BOHB	Bayesian Optimization and Hyperband
BOLD	Blood Oxygenation Level-Dependent
CBF	Cerebral Blood Flow
CP	Conjugate Phase
CNN	Convolutional Neural Network
CSF	Cerebrosipnal Fluid
DFT	Discrete Fourier Transform
DL	Deep Learning
EPI	Echo Planar Imaging
FA	Flip Angle
FFT	Fast Fourier Transform
FID	Free Induction Decay
fMRI	Functional Magnetic Resonance Imaging
FOV	Field Of View
GRE	Gradient Echo
GN	Group Normalization
HCP	Human Connectome Project
HPO	Hyperparameter Optimization
HUS	Haukeland University Hospital
ICA	Independent Component Analysis
LR	Learning Rate
LS	Looping Star

MIRT	Michigan Image Reconstruction Toolbox
MLP	Multilayer Perceptron
MMIV	Mohn Medical Imaging and Visualization center
MRI	Magnetic Resonance Imaging
NMR	Nuclear Magnetic Resonance
NUFFT	Non-Uniform Fast Fourier Transform
PCC	Posterior Cingulate Cortex
ReLU	Rectified Linear Unit
Res	Residual
RF	Radiofrequency
RIESLING	Radial Interstices Enable Speedy Low-Volume imagING
ROI	Region of Interest
RS	Resting State
RUFIS	Rotating Ultra-Fast Imaging Sequence
SGD	Stochastic Gradient Descent
SNR	Signal-to-noise Ratio
Std	Standard Deviation
TE	Echo Time
TR	Repetition Time
WD	Weight Decay

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Magnetic Resonance Imaging (MRI), and especially functional MRI (fMRI), are subject to loud acoustic noise inside the scanners. The noise occurs due to rapid switching of the encoding gradients in the imaging system that induce high-frequency vibrations in the scanner hardware that fall within the acoustic spectrum [1]. The acoustic noise can exceed 100 dB in some imaging sequences, even reaching close to the human auditory pain level [2]. The loud acoustic noise can be unpleasant for many patients or research participants, and it will limit the possibility of imaging certain patient groups, such as small children, psychiatric patients or patients hypersensitive to sound [2]. The impact of the noise can be even higher for fMRI. The acoustic noise interferes with the auditory functional experiments by producing a direct cortical activation, making it hard to disentangle activity patterns from the stimuli and the acoustic scanner noise [3] [1]. It also interferes with non-auditory experiments by increasing the demands on attention systems, which can be even worse for patients with neurodegenerative diseases or psychiatric diseases who can have complications concerning attention to stimuli [3]. It has also been suggested that acoustic noise affects resting-state networks [2].

Different approaches to reducing acoustic noise have been proposed, such as active noise canceling [4] and using scanning sequences that produce less acoustic noise [5]. A recently developed imaging sequence for silent imaging named "Looping Star" (LS) has been demonstrated for T2*-imaging and T2* BOLD fMRI [6]. This sequence reduces the acoustic noise generated by the scanner by using smaller changes in gradient amplitudes between RF-pulses. The Looping Star sequence achieves a 31 dB reduction in sound level, which represents a 98% reduction in sound pressure compared with the echo-planar imaging (EPI) sequence most commonly used for fMRI. The way Loop-

ing Star uses the gradient coils results in a radial, non-Cartesian sampling that records fewer data points in k-space, resulting in lower image quality, reduced blood oxygenation level-dependent (BOLD) activation and larger variation between subjects [2]. With increased image quality, sequences like Looping Star can be a powerful tool in the research and diagnosis of neurodegenerative and psychiatric diseases, especially diseases whose symptoms include issues with auditory perception such as schizophrenia.

The emergence of AI and machine learning in recent years has also led to the incorporation of such algorithms in many areas of medicine, including image reconstruction. The large providers of MRI scanning equipment are now introducing new machine learning-based reconstruction methods for their scanners, such as Siemens' Deep Resolve [7], Phillips' SmartSpeed [8] and AIR Recon DL from GE Healthcare [9]. For newer, and less common sequences like Looping Star, there are no established AI tools available for improved reconstructions, and there has been done little research on the effect machine learning has on the BOLD signal in fMRI.

1.2 Objective

The purpose of this thesis is to explore the potential of leveraging deep learning reconstruction in the Looping Star sequence, which has also been proposed as future work by Wiesinger, Menini, and Solana [6] in the original Looping Star paper. The network implementation in this project is a 2D network inspired by the one used by Han et al. [10] for the reconstruction of radially undersampled MRI. This is also extended to a 3D network to match the shape of the Looping Star k-space. The main objectives of this thesis are:

(i) Prepare a trainable dataset by emulating the Looping Star acquisition on an fMRI dataset to create image pairs of inputs and ground truth labels.

(ii) Train different deep neural networks for image reconstruction.

(iii) Compare the performance of the networks trained in (ii).

(iv) Evaluate the network performance on fMRI and its ability to preserve BOLD signal variation.

Chapter 2

Background

2.1 MRI - Magnetic resonance imaging

Magnetic Resonance Imaging is one of the most advanced medical imaging techniques. MRI is based on the physics phenomenon known as nuclear magnetic resonance (NMR). In this section, the basic principles of MRI are described alongside principles of image reconstruction, functional MRI and the MRI sequence in focus in this project, Looping Star.

2.1.1 MRI physics

Nuclear spin

MRI is based on the property of nuclei that atoms with an odd number of protons and neutrons in the nuclei have a net nonzero nuclear spin [11]. This gives the nuclei a magnetic property. Hydrogen (^1H) is such a nucleus, with only one proton in the nucleus it has a net nonzero spin. MRI utilizes this and the fact that human body tissue contains a lot of water (up to 60 %), which consists of two hydrogen atoms. When no external electromagnetic forces affect the protons in water, all spins are randomly oriented at room temperature due to thermal motion. However, when an external magnetic field, $\mathbf{B_0}$ is applied, all proton spins are aligned with the direction of the magnetic field. The protons can either align in a parallel or anti-parallel fashion [11]. Of these two, the parallel orientation is a slightly lower energy state, and therefore a larger number of protons will align parallel than anti-parallel. This results in a net magnetization in the sample (tissue) that is non-zero. This net magnetization vector is called $\mathbf{M}$ and, considering a coordinate system with its z-axis along $\mathbf{B_0}$, this net magnetization is approximately entirely in the z-direction at rest, making $\mathbf{M} = \mathbf{M_z}$.

Even under the influence of the strong external magnetic field $\mathbf{B_0}$, the spin of the hydrogen proton experiences a small precession around the direction of $\mathbf{B_0}$, giving each spin a small magnetic component in the x-y direction ($\mathbf{M_{xy}}$). The angular frequency of this precession, ω_0, depends on the magnitude of the magnetic field $\mathbf{B_0}$ and can be explained by this equation [12]:

$$\omega_0 = \gamma \mathbf{B_0} \tag{2.1}$$

The gyromagnetic ratio, γ, is a constant that is unique for each nuclear isotope possessing a spin [12]. The angular frequency ω_0 is called the Larmour frequency and is an important concept in MRI. The precession of individual spins around the $\mathbf{B_0}$-field will not be coherent with other spins, causing the $\mathbf{M_{xy}}$ components of the spins to cancel each other out [12].

Bloch equation and excitation

To get a signal from the proton spins in MRI, it is necessary to change the direction of the net magnetization vector $\mathbf{M}$ away from its equilibrium orientation parallel to $\mathbf{B_0}$. In addition, $\mathbf{M}$ needs to oscillate in time in order to produce a signal in the form of an induced current in a coil [12]. The behavior of the magnetization vector as a result of magnetic interactions with an external magnetic field is described by the Bloch equation [12]:

$$\frac{d\mathbf{M}}{dt} = \gamma \mathbf{M} x \mathbf{B} \tag{2.2}$$

Where $\mathbf{B}$ is the total experienced magnetic field in all directions on $\mathbf{M}$. To get a signal, the individual spin precession around the $\mathbf{B_0}$-field is taken advantage of. This is done by applying a second magnetic field $\mathbf{B_1}$, that oscillates and is applied perpendicular to $\mathbf{B_0}$. If the $\mathbf{B_1}$ field is applied with an oscillating frequency equal to the Larmour frequency, the effective field on $\mathbf{M}$ is only $\mathbf{B_1}$ as $\mathbf{B_0}$ will be canceled out by the oscillations of the $\mathbf{B_1}$-field [12]. The resulting movement of the $\mathbf{M}$-vector can therefore be described by the altered Bloch equation:

$$\frac{d\mathbf{M}}{dt} = \gamma \mathbf{M} x \mathbf{B_1} \tag{2.3}$$

Due to its oscillation, the $\mathbf{B_1}$-field is referred to as a radio frequency field (RF-field), and when applied in a pulse, it is called a radio frequency (RF) - pulse. The effect of applying an oscillating $\mathbf{B_1}$-field with the same frequency as the Larmour frequency is a resonance effect. The precession of $\mathbf{M}$ around $\mathbf{B_0}$ in the z-direction will increase its angle to the z-axis, and $\mathbf{M}$ gets a component in the xy-plane that is oscillating with

the same angular frequency as the RF-pulse. The increased precession of the **M**-vector results in reduced magnitude of **M** in the $\mathbf{M_z}$ direction and increased magnitude in the $\mathbf{M_{xy}}$ direction [12]. This process of changing the direction of the **M**-vector to the xy-plane is referred to as excitation. The strength of the applied $\mathbf{B_1}$-field and duration of the applied RF-pulse decides the strength of the excitation. The strength of the $\mathbf{B_1}$-field decides the speed at which the **M**-vector is excited from the z to xy direction. This angular velocity is given by equation 2.4 [12]:

$$\omega_1 = -\gamma \mathbf{B_1} \tag{2.4}$$

If the pulse is applied for a time t_p, the angle of rotation is given by equation 2.5:

$$\alpha = \omega_1 t_p \tag{2.5}$$

This angle is called the "flip angle" of the RF-pulse [12].

Relaxation

As equation 2.5 describes, the magnetization vector **M** will be rotated an angle α away from the direction of $\mathbf{B_0}$ when the RF-pulse is turned on. When the RF-pulse is subsequently turned off, a process called relaxation starts, where the magnetization vector precesses towards the equilibrium state. The total magnetization vector continues its precession around the direction of the $\mathbf{B_0}$-field while the angle of the rotation (α) diminishes quickly. The macroscopic result of the relaxation is that the $\mathbf{M_{xy}}$ component of **M** gradually disappears and $\mathbf{M_z}$ gradually recovers [12].

During the relaxation process, the MRI signal can be observed and measured. This is due to the oscillating nature of the $\mathbf{M_{xy}}$. Since $\mathbf{M_{xy}}$ is rotating around the z-axis it is possible to detect this magnetic component through the induction of a current in a coil placed in the oscillating field [12]. The signal detected from $\mathbf{M_{xy}}$ is called the *Free Induction decay* (FID). $\mathbf{M_z}$ is not directly detectable as it does not rotate and hence will not induce a current [12]. In MRI, tissue contrast is achieved from two main types of biological contrast that occur due to the relaxation process. These are T1 and T2 relaxation.

T1 relaxation

As described, the macroscopic effect of the relaxation results in a gradual recovery of the longitudinal component of the magnetization vector, $\mathbf{M_z}$. The longitudinal relaxation process is referred to as *T1 relaxation* [12]. T1 relaxation describes the rate

of recovery of $\mathbf{M_z}$ through a time constant called the T1 relaxation time. The inverse of the relaxation time, $1/T1$ is referred to as the relaxation rate, R1 [12]. The relaxation of the longitudinal magnetization component following an RF-pulse is explained in equation 2.6:

$$M_z(t) = M_0\left[1 - exp\left(\frac{-t}{T_1}\right)\right] + M_z(0)exp\left(\frac{-t}{T_1}\right) \tag{2.6}$$

As equation 2.6 shows, the time constant T1 describes the time it takes for the longitudinal magnetization to regain approximately 63 % of its final value [13] (Fig 2.1).

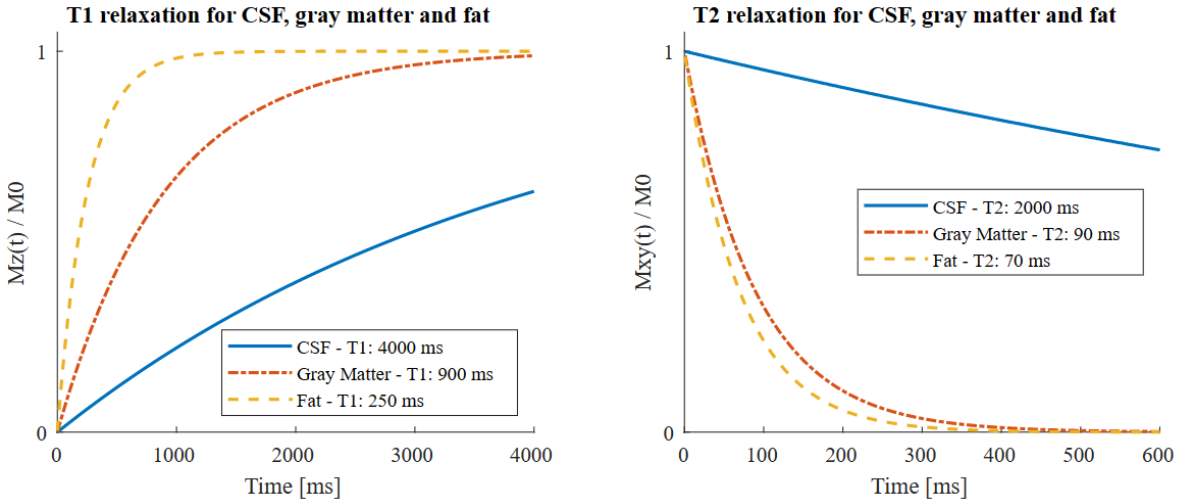

Figure 2.1: Equation 2.6 for T1 relaxation (left) and 2.7 for T2 relaxation (right) for CSF, gray matter and fat. CSF = Cerebrospinal Fluid

T2 relaxation

As well as recovery of the longitudinal component of the magnetization, the relaxation also leads to decay of the transversal magnetization $\mathbf{M_{xy}}$. This decay is termed *T2 relaxation* and, as with T1 relaxation, it is described by a time constant T2 and a corresponding relaxation rate $1/T2 = R2$ [12]. The relaxation on the transversal magnetization component following an RF-pulse is explained in equation 2.7:

$$M_{xy}(t) = M_{xy}(0)exp\left(\frac{-t}{T_2}\right) \tag{2.7}$$

As equation 2.7 shows, the time constant T1 describes the time it takes for the transversal magnetization to decay to approximately 37 % of its initial value [13]. One might expect that the T1 and T2 relaxations are directly related such that $\mathbf{M_z}$ is recovered as much as $\mathbf{M_{xy}}$ decay, but this is only the case in pure water [12] (Fig 2.1). In tissue and other medium, there exist fluctuating magnetic field variations that occur because

the proton spins are influenced by small magnetic fields from neighboring nuclei [13]. These fluctuations affect the spins, giving rise to variations in the Larmour frequency of individual protons [13]. This will make the spins in a voxel de-phase, leading to a more rapid decay of the transversal magnetization. This results in a T2 that is always shorter than T1 in vivo [12].

Another cause of loss of phase coherence is bulk inhomogeneity in the $\mathbf{B_0}$ magnetic field. This kind of inhomogeneity is always present, but unlike the random processes of T2 relaxation, the de-phasing that occurs as a result of this is caused by a constant and is potentially reversible [13]. The actual T2 relaxation is therefore commonly referred to as T2* to include the bulk inhomogeneity effect [12]. This is described as such:

$$\frac{1}{T2*} = \frac{1}{T2} + \gamma \Lambda B_0 \tag{2.8}$$

Here ΛB_0 should be interpreted as the bulk inhomogeneity within a single voxel. An important component of ΛB_0 is generated by inhomogeneities due to susceptibility differences between different tissue types and at interfaces between tissue and air [12].

2.1.2 Image formation

Several pieces of hardware components are used to separate and localize the MR signal in three dimensions. In this section, an overview of these and their uses are presented.

MRI equipment

The part of the MR system that generates and receives the signal consists of a set of magnets with different purposes. There are the main magnet coils, three gradient coils and an integral RF-transmitter coil [13]. The main magnet coils are used to generate the strong $\mathbf{B_0}$ magnetic field that primes the proton spins by directing them parallel or antiparallel to itself. The three gradient coils are oriented such that each represents one of the three orthogonal directions (x, y, and z) and they lie within the main magnet. Each of these coils can generate a magnetic field with a strength that changes with position along the orthogonal direction they represent [13]. These coils can then superimpose a magnetic field on top of $\mathbf{B_0}$ that results in a main magnetic field that varies along the direction of the applied gradient field [13]. The gradient coils are then used to spatially encode the MR signal. The RF coils are mounted inside the gradient coils and can be thought of as the "antenna" of the MRI system [13]. The RF-coils are used to generate the $\mathbf{B_1}$ field that transfers energy to the tissue, and later it is used to measure the induced signal from the relaxation process.

Slice selection

To spatially localize the signal it is necessary to excite only parts of the imaging object at a time. To achieve this, the RF-pulse needs to be selective such that it only affects protons in a defined plane in the object [12]. Applying a gradient field along one of the directions (x, y, and z) can alter the main magnetic field such that its strength changes with position [13]. Thus, considering equation 2.1, the Larmour frequency of protons in different locations becomes different. It is then possible to excite only one slice as only one slice will have a Larmour frequency corresponding with the RF-pulse.

Frequency and phase encoding

The remaining directional gradients after slice selection is used to separate the intra-slice signal after the RF-pulse. Typically, the z-gradient will be used for slice selection [12]. X and y gradients will then be used for processes called frequency encoding and phase encoding. Frequency encoding is the application of a gradient in the x direction following the RF-pulse. This applied gradient causes the spins in the slice to rotate at different frequencies depending on their position along the gradient [13]. The different frequencies permit differentiation of the signal along the plane of the gradient. The frequency encoding gradient is applied for the entire duration of the readout, which is important as the spins need to experience a different effective magnetic field strength to precess at different speeds during the readout.

Phase encoding utilizes the final gradient in the y-direction to make the signal differentiable in the y-direction. The phase encoding gradient is applied following an RF-pulse, but is then turned off before the signal is recorded. All the spins in the phase encoding plane are then precessing at the same frequency during readout. During the period the gradient was applied, the spins would have been precessing with different frequencies [13]. This means that when the gradient is turned off, spins along the phase encoding direction will be out of phase with each other and a phase shift happens along the gradient [12].

These two encoding steps are important to get the proper encoding of the MRI signal such that it can be properly sampled. MRI images are sampled in the frequency domain in a matrix called the *k-space*. To understand how this works one first needs to understand the Fourier transform, which is an important mathematical concept when it comes to image reconstruction in MRI.

Fourier transform

Fourier theory states that any function can be represented as a sum of sinusoid waves. This gave rise to the Fourier transform. The Fourier transform is a mathematical operation that transforms a function $f(t)$ in the temporal/spatial domain to a representation of that function in the frequency domain, $F(\omega)$. $F(\omega)$ contains the amplitude and phase of the frequency component ω in $f(t)$. In 1D, the Fourier transform is computed as such [14]:

$$F(\omega) = \int f(t) e^{-i2\pi\omega t} \, dt \tag{2.9}$$

Equation 2.9 shows that $F(\omega)$ is computed by taking the integral of the function $f(t)$ multiplied by the complex exponential representing the sinusoid with frequency ω. This is equivalent to finding the area under the curve of the product of function $f(t)$ and the sinusoid. The information in $F(\omega)$ is complex-valued, where the combination of the real and imaginary components gives an understanding of the amplitude and phase of the sinusoid at frequency ω. The Fourier transform is an inverse operation [14], where the original function $f(t)$ can be recovered from $F(\omega)$ using:

$$f(t) = \int F(\omega) e^{i2\pi\omega t} \, d\omega \tag{2.10}$$

Equation 2.9 shows how the Fourier transform works on a continuous function and the result will then become a continuous frequency spectrum. In practice, the recorded signals will be represented with discrete samples. This means that the ideal Fourier transform does not work on real-world data, instead, the *Discrete Fourier Transform* (DFT) is used. When performing the DFT on a discrete set of samples, the resulting frequency spectrum will also be discrete. Some important notes on the sampling technique of the signal can be made with respect to the DFT. The sampling resolution, i.e. how closely spaced the data samples are, decides the maximum frequency that can be measured as fewer data points will give a lower resolution in the signal. The duration, or the field of view (FOV) of the signal, decides the frequency resolution [14]. A shorter FOV will make it harder to tell different frequencies in the signal apart. Therefore more frequencies need to be binned together in the frequency domain. In a DFT, the total number of detectable frequencies is equal to the number of samples in the signal. The DFT is calculated using equation 2.11:

$$F_{\omega_n} = \sum_{k=0}^{N-1} f_k e^{-i2\pi\omega_n k/N} \tag{2.11}$$

The major issue with the DFT is that it requires a lot of calculations and will therefore be slow. In almost all applications today, a variant of the Discrete Fourier Transform algorithm called the *Fast Fourier Transform* (FFT) is used when the Fourier transform needs to be computed [15]. FFT is based on the fact that sinusoidal waves with a periodic increment of one integer will overlap at certain points. Utilizing this predictable manner of sinusoids, the number of calculations per sinusoid can be reduced, thus reducing computational load.

K-space

K-space is an important concept in MRI as an MRI scanner collects its data in k-space. As mentioned in the frequency and phase encoding section, the k-space is a matrix in the frequency domain. Each pixel in this matrix contains information about the image contribution of different 2D sinusoidals, or stripe patterns [16]. In k-space, the position of data points in the matrix reveals the frequency and direction of that specific stripe pattern. If one imagines an arrow pointing from the center of k-space to a data point, the length of the arrow is proportional to the frequency of the stripe pattern (stripe density). The direction of the arrow is directly opposite the direction of the pattern. Horizontal patterns are located on the Ky-axis that goes through the middle of k-space, while vertical patterns are located on the Kx-axis [16] (Fig 2.2). The complex content of the data point gives away its amplitude and phase. What this means is that the matrix cells closest to the center of k-space contain information on low-frequency variations in the image. In other words, these cells contain information about the general shapes in the image. Cells further out in the k-space matrix will represent higher frequencies and will contain information about high-resolution details.

Figure 2.2 shows that the Fourier transform operation is used to move from the image domain to k-space, and the inverse Fourier transform is used when reconstructing the image from k-space. The FFT is used to get from frequency space (k-space) to image space as it is much faster. For a 2D k-space, a 2D FFT is needed for image reconstruction. For 3D k-space, 3D FFT is used.

In order to fill the data in the k-space matrix, spatial encoding by frequency and phase encoding is used. These encoding gradients are used to generate spatial representations of the stripe patterns that inhabit the data points in the k-space matrix. By applying a

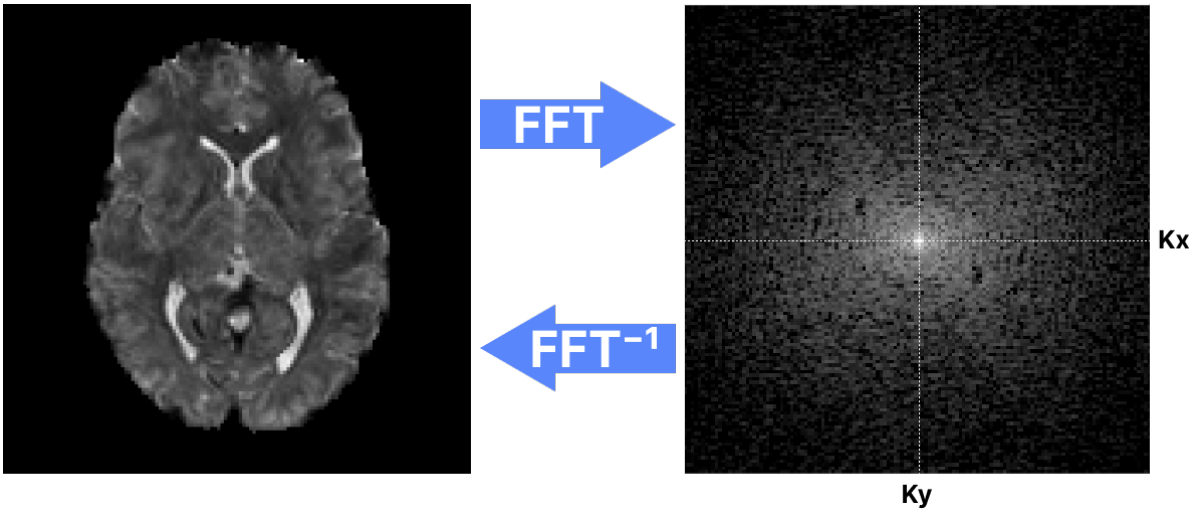

Figure 2.2: K-space example. Performing the FFT in the image of a brain (left) will result in the k-space matrix (right). An inverse FFT will reconstruct the image from k-space.

phase encoding gradient, the signal location is moved up and down in k-space. The density of the stripe pattern depends on the strength and duration of the gradient [16]. The frequency encoding gradient moves the signal location in the left and right direction in k-space. Combinations of the phase and frequency encoding gradients will generate oblique stripe patterns [16]. By applying combinations of phase and frequency encoding gradients, stripe patterns that correspond to any point in k-space can be created. The remaining information that needs to be recorded is the amplitude and phase of the stripe pattern. This is done by measuring the signal arising from the rotating magnetization [16]. Two receiver coils are used to record the signal and allow for recording both the amplitude and phase of the signal. The two coils are usually orthogonal to each other and will thus record two different signals. Phase and amplitude information can be retrieved by processing the signal from both coils.

2.1.3 Pulse sequences

Pulse sequences are combinations of RF-pulses and gradient encoding of the signal to sample the k-space in the desired way. The gradients can be used to either sample every pixel in the k-space matrix in a Cartesian way or use sequences that records non-Cartesian samples. Two time constants are important when looking at pulse sequences. The echo time (TE) is the time period between the RF pulse and the signal echo used for recording. Repetition time (TR) is the time between two consecutive RF-pulses.

Gradient echo sequence

One Cartesian pulse sequence that is widely used is the Gradient Echo sequence (GRE). In the GRE, a slice selection gradient is applied during the RF-pulse. The slice selection gradient will generate an echo from the original FID. Then, the phase encoding gradient, G_y, and a strong negative frequency encoding gradient, G_x, are applied before the echo arrives and the readout starts. Right before the echo, the G_x gradient is switched to positive and is kept constant during the readout (Fig 2.3a). This combination of G_y and G_x allows for the sampling of a full row in k-space for each RF-pulse (Fig 2.3b) [12]. The entire k-space is sampled by repeating the process with different strengths of the phase encoding gradient.

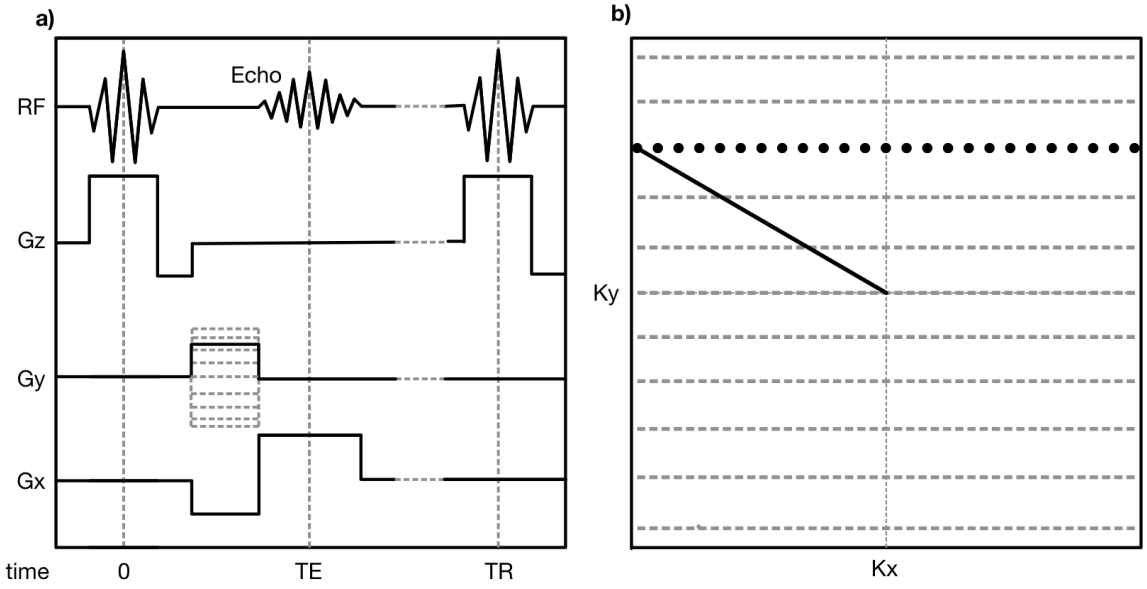

Figure 2.3: Gradient Echo imaging sequence. a) Sequence diagram explaining the use of x, y and z gradients in combination with RF-pulses. b) K-space trajectory from one RF-pulse using the black, whole line phase encoding gradient. RF = Radio frequency, TE = Echo time, TR = Repetition time. Figure inspired by Donald B. Plewes in [16]

Echo-planar imaging

Echo-planar Imaging (EPI) is a fast pulse sequence used for applications requiring very fast imaging, such as fMRI. EPI uses fast switching of high amplitude gradients to generate multiple echoes after one RF-pulse [13]. This gives EPI the opportunity to sample several rows in k-space from a single RF-pulse. Short pulses of the phase and frequency encoding gradients are used to maneuver up, down, left and right in k-space [12].

Non-Cartesian sampling sequences

The most common MRI sampling is a uniform collection of the data points in k-space, known as Cartesian sampling. Many alternatives to this Cartesian sampling have been proposed, such as spiral and radial scans. These non-Cartesian alternatives have advantages over Cartesian samplings, such as favorable robustness to motion [6], flow performance, and SNR efficiency [17].

Radial pulse sequence

One example of a non-Cartesian pulse sequence is the radial scan. In many of the non-Cartesian sampling methods, the notion of phase and frequency encoding gradients does not apply as the gradients are used differently [16]. In a radial scan, both gradients are applied simultaneously during readout with an incremental change in between RF-pulses (Fig 2.4a). With two gradients applied simultaneously from the start of the signal readout, the resulting k-space trajectory is a line starting at the center of k-space and extending out into the matrix (Fig 2.4b). With the proper choice of gradients, the k-space trajectory of the signal can be rotated in k-space from one RF-pulse to the next [16]. Thus the total acquired k-space resembles a wheel where the spokes of the wheel are the acquired k-space lines. This pulse sequence has the advantage that it samples more densely in the center of k-space than in the edges. The image can then be acquired using fewer RF-repetitions, without major effects on the image contrast [16].

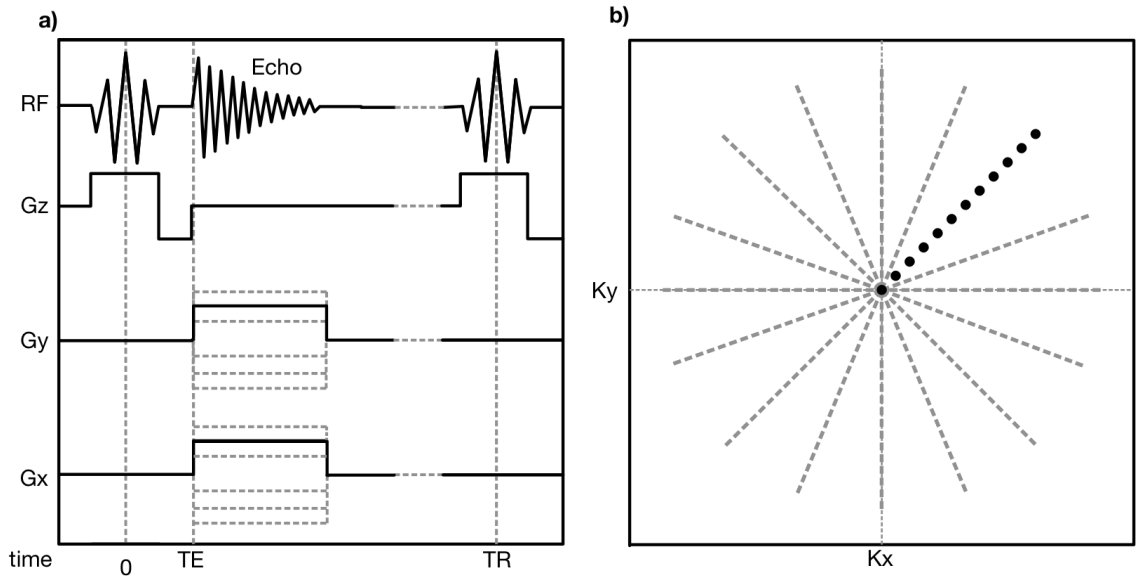

Figure 2.4: Radial imaging sequence. a) Sequence diagram explaining the use of x, y and z gradients in combination with RF-pulses. b) K-space trajectory from one RF-pulse using the fixed x and y gradients during the echo. RF = Radio frequency, TE = Echo time, TR = Repetition time. Figure inspired by Donald B. Plewes in [16]

2.2 Non-Cartesian reconstruction

The main disadvantage of the non-Cartesian imaging methods is the difficulty of reconstructing the resulting data [17]. One of the main advantages of Cartesian k-space is that the reconstruction can be performed with a simple 2D FFT. This is not possible for Non-Cartesian sampling as the data points are not uniform. Instead, approaches such as the conjugate phase reconstruction and gridding could be applied.

2.2.1 Conjugate phase reconstruction

One method for reconstructing non-Cartesian k-space samples is conjugate phase (CP) reconstruction. For a conjugate phase reconstruction at some data point, the acquired signal is multiplied by the conjugate of the phase at this point throughout the acquisition. Then the product is integrated over for the duration of the acquisition [17]. For a data point x_0, equation 2.12 describes the process.

$$m(x_0) = \int_0^T s(t) e^{-i2\pi \mathbf{k}(t)\cdot x} dx \tag{2.12}$$

In equation 2.12, s(t) is the signal from an acquisition [17]. As the process in equation 2.12 needs to be performed for all data points, the conjugate phase reconstruction is a slow process.

2.2.2 Density correction

One common issue with non-Cartesian sampling is that it is seldom uniform in k-space. A radial k-space acquisition will for example have a more densely sampled center of k-space, which is an issue as these samples will be overrepresented in the reconstruction. To account for this it is necessary to perform some weighting to account for sampling density [17].

2.2.3 K-Space resampling methods

A faster alternative to the conjugate phase reconstruction is to resample the k-space data onto a Cartesian grid and then use a DFT to reconstruct the image.

Grid-driven interpolation

The idea of grid-driven interpolation is to estimate the value at each grid point based on the immediately surrounding data. This approach is easy to implement if the location of the neighboring data points can be determined analytically. One disadvantage is that

it won't use all the input data, and lose some SNR efficiency as a result. But as most of the excess data is usually at low spatial frequencies where the SNR is already high, the loss in SNR is not very significant [17]. This approach does not require a density estimate.

Data-driven interpolation

Data-driven interpolation takes the data from each data point and adds their contribution to the surrounding grid points. Each input sample is conceptually considered to be convolved with a small kernel, which is wide enough to extend to the neighboring grid points, and each data point is in this way "resampled" at the adjacent grid points. In this approach, all data is used, as opposed to the grid-driven interpolation. That makes data-driven interpolation more SNR efficient, however, it requires a density estimate to correct for any non-uniformity in sample density [17].

2.3 fMRI - Functional MRI

Functional Magnetic Resonance Imaging (fMRI) is developed in order to measure local time-varying changes in oxygen consumption in the human brain at rest or in response to a cognitive task [18]. MRI images are sensitive to local changes in magnetic susceptibility which can occur when the oxygenation state of blood changes [19].

Previously, the decay of signal associated with local magnetic field inhomogeneities, called T2* relaxation, was considered a nuisance for MRI [19]. But when it was discovered that the presence of a paramagnetic substance in the bloodstream could provide useful contrast, sequences that could pick up these inhomogeneities were developed [19]. fMRI is a technique that utilizes this by measuring the changes in blood oxygenation that occur as a result of neural activity. By acquiring several brain volumes in a time-series, fMRI makes it possible to measure differences in blood oxygenation over time. A high temporal resolution is necessary to monitor the brain in real-time. Fast imaging techniques such as EPI are therefore often used for fMRI, which can achieve a temporal resolution of 1-3 seconds [19]. A technique known as statistical parametric mapping (SPM) is used to transform the time-series data into statistical maps of image difference [19]. For functional MRI data, several preprocessing steps must also be performed, such as removal of head movement effects, spatial smoothing, and statistical interference [19].

2.3.1 Blood oxygenation level-dependent contrast imaging

In 1935, Linus Pauling and Charles D. Coryell studied the magnetic properties of unbound and bound hemoglobin. They discovered that the hemoglobin molecule with oxygen bound to it (oxyhemoglobin) contains no unpaired electrons and the molecule is found to have zero magnetic moment [20]. In contrast to oxyhemoglobin, deoxyhemoglobin (oxygen-free hemoglobin) contains unpaired electrons. The magnetic susceptibility of deoxyhemoglobin, therefore, shows a profound paramagnetic contribution [20].

The discovery made by Pauling and Coryell [20] was further built upon by Ogawa et al. through several studies in 1990 [21], [22], [23]. They discovered that the difference in magnetic susceptibility between oxyhemoglobin and deoxyhemoglobin induced a local field change that is "felt" by both the water molecules in the blood and by those in the surrounding tissue [23]. This causes dephasing of the water signal in voxels in these regions [21]. This dephasing of the water signal was shown to give rise to an im-

age contrast in magnetic resonance imaging of the brain when the blood oxygenation level is lowered [22]. This contrast is known as the blood oxygenation level-dependent (BOLD) contrast [23]. Since the BOLD contrast depends on the state of blood oxygenation, physiological events that change the concentration of oxy/deoxyhemoglobin lend themselves to noninvasive detection through the accentuation of BOLD contrast.

Not much later, in 1992, Ogawa et al. investigated whether a local elevation in brain venous-blood oxygenation accompanies an increase in neural activity after visual stimulation [24]. They found that visual stimulation produces an easily detectable increase in the intensity of water proton MRI signals of 5-20 % in the primary visual cortex in the human brain [24]. This was consistent with the idea that neural activation increases regional cerebral blood flow and simultaneously increases venous-blood oxygenation and showed the possibility of using BOLD contrast to map human mental operations [24].

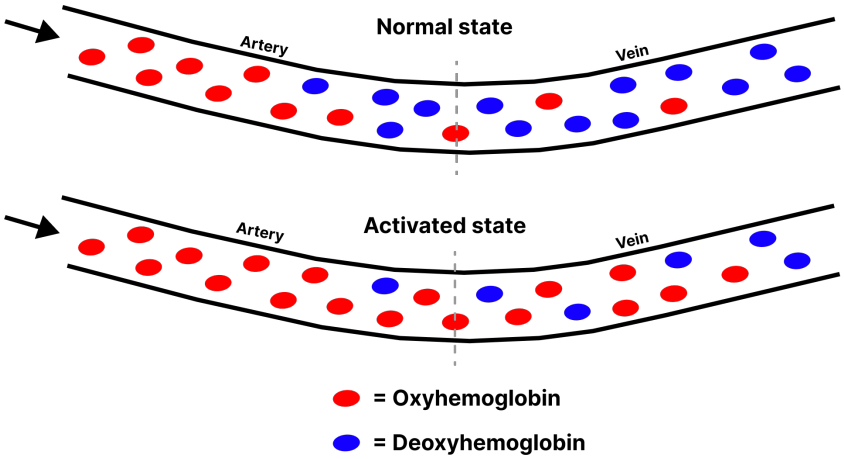

Figure 2.5: Blood flow during normal state (top) and after neuron activation in the area (bottom). Figure inspired by courses from SPM [25].

To explain why BOLD contrast imaging can be used to measure neural activation it is necessary to look at the brain metabolism. All the processes involving neural signaling in the brain require energy, which it gets in the form of adenosine triphosphate (ATP) [18]. When the neurons in a brain region are activated by a cognitive task, the increased neural activation is accompanied by increased ATP production via glucose consumption [26]. The production of ATP results in increased usage of local oxygen by the metabolic processes in the affected brain region [18]. As the local stores of oxygen are consumed and waste products from the neural activation build up, signals are sent to nearby blood vessels causing upstream arterial vessels to dilate. The dilation is accompanied by increased cerebral blood flow (CBF) to the area [26]. Increased CBF brings more oxygen to the region to restore the local oxygen level. However, more oxygen is

delivered than is needed to offset the increase in metabolic rate, resulting in an increase in oxygen levels [26] (Fig 2.5). As a result, neural activation in a brain region initially results in a build-up of deoxyhemoglobin and a decrease of oxyhemoglobin, followed by a response that reverses the situation with an increase in oxygenated hemoglobin and a decrease in deoxyhemoglobin above the levels of the resting condition [18]. This increase happens a second or two after the neural activation. This makes BOLD contrast imaging an indirect measure of neural activation. Since its discovery, BOLD contrast imaging is used in virtually all conventional fMRI experiments [18].

2.3.2 Sources of signal and noise in fMRI

In fMRI, the analysis is performed by comparing signal variations in each voxel in time in the imaged brain volumes. Several sources of signals play a role in the total T2* signal variations. In phantoms, noise has been found to be both "white", meaning it appears uniformly across the image, and appears as slow drifts in intensity that may be caused by temperature changes in imaging hardware [19]. In the human brain, low-frequency noise components that arise from head motion, slow global variations in blood oxygenation or cardiac and respiratory pulsations are also found [19].

When all external noise sources are accounted for (e.g. motion, hardware, etc), the remaining signal variation in the time-series data is caused by time-dependent changes in the BOLD signal. This signal can be decomposed into several frequency components. Each frequency component has a different source, and these sources can be either noise arising from physiological rhythms (heartbeat and respiration) or signal related to functionally activated brain areas [19]. The standard approach is to denoise the voxel time-series using band-pass filtering [27]. This removes low-frequency noise (e.g. intensity drifts, heartbeat and respiration) and high-frequency noise.

2.3.3 Task fMRI

Task activation fMRI seeks to induce different neural states in the brain as visual, auditory or other stimulus is manipulated during the scan [18], or mental tasks are performed. Activation maps are obtained by comparing the signals recorded during the different states. The stimuli is designed to induce two or more cognitive states in the participant while collecting MRI volumes continuously.

2.3.4 Resting state fMRI

A different class of fMRI to task-based functional imaging is resting-state fMRI (rs-fMRI). rs-fMRI is acquired in the absence of a stimulus or task, while the brain is "at rest" [28]. The data in rs-fMRI can be acquired with a dedicated resting-state scan where participants are instructed to simply rest [28] and refrain from any cognitive language or motor tasks as much as possible [29]. One of the advantages of rs-fMRI is that it does not require a specific task to be performed by the subject, making it suitable for use with patients who may have difficulty following instructions, such as those with neurological, neurosurgical or psychiatric conditions, as well as pediatric patients [28].

There are several ways of performing analysis of rs-fMRI data, but common to all is the need for large amounts of data. Several pre-processing steps are also necessary. This includes realignment and removal of artifacts from head motion and CSF signal [28], as well as slice timing, data normalization, smoothing and more.
Analysis of rs-fMRI data can reveal information on specific brain regions that show activation in a resting brain, but the data can also give information on the functional connectivity between different brain regions [28]. The analysis of rs-fMRI data can be separated into two subtypes: functional segregation and functional integration. Functional segregation focuses on specific brain regions and relies on the analysis of rs-fMRI activity while functional integration focuses on the relationships or connectivity between different brain areas and assesses the brain as an integrated network [28]. Proper interpretation of rs-fMRI data results and logical inferences of the results requires an understanding of anatomy, physiology and neuroscience [28]. Only the ROI-based functional connectivity method is used in this project, the rest are included for completion.

Functional segregation methods

Functional segregation methods divide the brain into regions according to their specific functions. ALFF (Amplitude of Low-Frequency Fluctuations) and ReHo (Regional Homogeneity) are functional segregation methods. ALFF measures the power of the BOLD signal within the low-frequency range, which only gives a measure of brain activity, and no connectivity information [28]. The ReHo analysis is a voxel-based measure of the similarity between the time-series of a given voxel and its nearest neighbors, measuring the synchrony of adjacent regions.

Functional integration methods

Functional integration focuses on the functional connectivity between regions of the brain and measures the synchrony of the BOLD time-series between different brain regions. The networks and pathways that are revealed by the integration methods may be a result of a direct anatomical connection or an indirect path. This makes it seldom useful to visualize the connections, but rather the brain regions that are presumably connected by these paths [28]. Common computational methods of functional integration include functional connectivity density (FCD) analysis, ROI-based functional connectivity analysis, independent component analysis (ICA) and graph analysis [28].

FCD is the most basic measurement of connectivity. It calculates the correlation of the BOLD time-series between each voxel and all other voxels in the brain, thus revealing how connected a voxel is, but not to which regions it is connected [28].

ROI-based functional connectivity analysis uses predefined regions-of-interest (ROI) to find other brain regions that are correlated to this ROI. Coupling of activation between different brain regions suggests that they are functionally connected and involved with the same underlying processes [28]. This method enables visualization of connections to the ROI and the strength of the connections. As ROI-based analysis requires a predefinition of the seeds (ROIs) the results will depend on the seed selection, making it vulnerable to bias. The selection is often based on prior results or can be derived from ALFF or ReHo calculations.

ICA uses multivariate decomposition to separate the BOLD signal into several independent functional networks that are temporally correlated [28]. ICA is data-driven, which means that it can be performed without any prior assumptions, except the number of independent components to identify [28]. While ROI-analysis only extract the regions connected to the ROI, ICA extracts all detectable networks [28]. Some drawbacks of ICA are that the underlying cause of the synchrony in the networks may be non-neural in origin (such as breathing or heartbeat), and ICA only presents the networks one by one with no communications between different brain networks [28].

2.4 Looping Star

Looping Star (LS) is an MRI pulse sequence based on the 3D radial Rotating Ultra-Fast Imaging Sequence (RUFIS) [6]. RUFIS uses a rapid succession of low flip angle RF pulses to acquire the FIDs [30]. RUFIS is a radial pulse sequence where small steps in the gradient fields between pulses rotate the acquiring trajectory in increments. In addition to the fast property of this sequence, the small steps in gradient fields also come with the benefits of reduction in acoustic noise. The loud acoustic noise generated in the MR scanner arises from interactions between the gradient coils and the magnet when switching of the gradient coils induce Lorentz forces and eddy currents that act on the coils and connecting elements [2]. This produces mechanical vibrations in the hardware. By using small gradient steps, as in RUFIS, the acoustic noise in the MR scanner can be mitigated and this results in quiet scanning. The Looping Star sequence extends RUFIS by adding a gradient-refocusing mechanism and full 3D spatial encoding [6]. The acquisition scheme, as well as prior results of the Looping Star sequence, will be explored in the following sections.

2.4.1 Looping Star pulse sequence

The Looping Star sequence is organized into segments, loops and spokes for capturing a FID image and subsequent echo images (Fig 2.6). The sequence starts with short RF-pulses that are used to excite the FID as 3D radial spokes [2]. These radial spokes are organized in segments. One segment exists in one plane in k-space and consists of several spokes, forming a radial k-space trajectory [6]. Several RF-pulses are needed to excite all the spokes in the segment, as one spoke is excited per RF-pulse.

The trajectory of the signal for one segment is decided by keeping the magnitude of the readout gradient constant for the entire segment, but its direction increment by an in-plane rotation from one spoke to the next to get the radial sampling [6]. The rotation is performed using incremental changes in the G_x and G_y gradients (Fig 2.6A). The trajectory of each spoke in the segment is considered a pathway that will be refocused towards the center of k-space in a loop that exceeds the k-space limit (Fig 2.6B) [6]. The encoding gradients perform the refocusing of the spoke signal, making the sequence a self-refocusing sequence.

The excitation and consequent refocusing of the k-space trajectory for the spokes is called a loop. The first loop records the initial FID of the segment at zero echo time (TE = 0) and the following loops form equidistant echoes of the FID [6]. For the second

loop onward, during the gradient refocusing, the signal from the current spoke (dephasing outwards) gets contaminated by the signal from the next spoke (refocusing inwards) [6]. Separating the dephasing (echo-out) and rephasing (echo-in) signal contributions is favorable as the signal mixing increases image noise. Separation methods tend to either sacrifice high spatial frequency information or increase scan time [6]. To further reduce acoustic noise during the scanning, the gradient waveforms can be smoothed. This results in slightly bent radial spokes (An example of this can be seen in the real Looping Star k-space in figure 3.1).

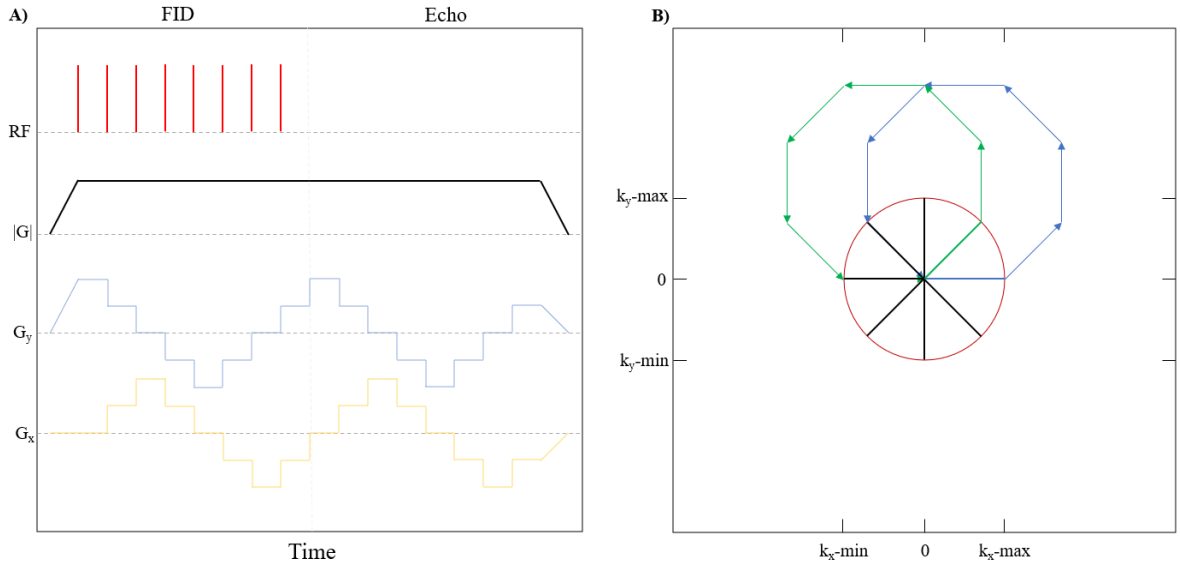

Figure 2.6: Diagram of the Looping Star sequence for one segment with 8 spokes per loop. A) shows the pulse diagram of the sequence for one segment with two loops: the FID and one echo. B) shows the k-space trajectory of the dephasing and refocusing for the first (blue) and second (green) spokes. FID = Free induction decay, RF = Radiofrequency. Figure inspired by Wiesinger, Menini, and Solana [6].

To ensure 3D spatial encoding, the planar k-space trajectory of the segments needs to be rotated. This can be done analytically in a specific manner or randomly [6]. As the k-space trajectory of the Looping Star sequence consists of many radial disks, the k-space is non-Cartesian. This means that non-Cartesian image reconstruction methods must be used, such as conjugate phase reconstruction or gridding to Cartesian k-space followed by Fourier transformation.

Looping Star can also be used to capture BOLD-weighted images for fMRI. To do this, one echo with echo time close to the T2* of gray matter should be obtained, a spatial resolution of at least 2-4 mm and a volume repetition time of no more than 2-4 seconds should be achieved [2].

2.4.2 The need for silent imaging: LS in practice

Some studies have looked at the effects of the Looping Star sequence on the acoustic noise levels in the MRI scanner [6] [2] [1]. Loud acoustic noise inside the scanner during acquisition remains a major concern as it causes discomfort and anxiety in patients [2]. Quiet imaging can improve scanning comfort and address anxious patient groups, such as pediatric, geriatric, and non-cooperative patients. In the context of fMRI, Looping Star might resolve acoustic noise concerns, such as in resting-state examinations, auditory processing, and sleep studies [6]. The Looping Star sequence has achieved reductions of 31 dB in A-weighted noise level and 20 dB reduction in peak noise level. This represents more than a 98% reduction in sound pressure compared with EPI [2].

The Looping Star sequence has been used to capture T2*-weighted images of the head both from the FID and later echoes. The FID covers the full head while the later echo primarily depicts the brain due to shorter T2* signal, live time or destructive signal interferences due to off-resonance effects in combination with 3D radial sampling [6]. In fMRI, the BOLD activation from the Looping Star sequence has been measured to be $\approx 5\%$ and matching the expected motor brain areas in a finger tapping motor experiment [6]. The Looping Star sequence has also been demonstrated as sensitive to the BOLD response in an auditory oddball paradigm [1], and a resting-state experiment [2] with some limitations such as reduced image quality, reduced tSNR and lower test-retest reliability compared to an EPI acquisition.

2.5 Machine learning

In classical computer science, programs are programmed according to a rigid set of rules specifying how the program should behave [31]. Machine learning is a relatively modern family of computer science algorithms that aim to solve some of the complex problems that are not easily solved using rigid rules. Machine learning algorithms learn from experience by adapting to observational data or other environmental interactions it encounters [31]. There are many types of machine learning models that behave in different ways, but most of them share some core components: The models need data they can learn from, an initial model of how to transform the data, an objective function (often called loss function) that quantifies how well the model is performing and they need an algorithm that adjust the model parameters based on its performance [31].

There are several different kinds of machine learning problems. Supervised learning describes tasks where the goal of the models is to predict labels from a set of input features. Supervised learning uses datasets that contain both input features and labels corresponding to the input features that act as a ground truth. For datasets that don't have clear labels for their input features, unsupervised learning algorithms can be used. Contrary to supervised learning, where the model has a clear objective, the task of unsupervised learning is to discover some kind of pattern in the dataset that can be interesting, e.g. finding data clusters or principal components. [31]

2.5.1 Generalization

The goal of machine learning is to discover underlying patterns in the data that will generalize and give good performance when solving problems on unseen data. This task is difficult as it can be difficult to tell if the model has discovered a general pattern, or just memorized the training data. When a model fits perfectly to its training data, but fails to generalize to unseen data, the model is overfitting [31].

Splitting the available data into training, validation and testing sets gives the opportunity to test the generalization of a model and detect overfitting. The model performance on the training set alone does not say much of the model's performance on unseen data, but by holding out parts of the data, the model can be tested on previously unseen data in the validation and test sets. The difference between the performance on unseen data and training data can give an indication of whether the model is overfitting or not [31]. The validation set is normally used for selecting the best model during model selection schemes or during training. The testing set is used for the final evaluation of the model

performance on unseen data.

There are several ways to reduce overfitting during training. One of these is early stopping, where the training is stopped if the model fails to improve the performance on the validation data. Another is to introduce data augmentations to the training data. This will increase the data diversity which is an effective way of increasing the model generalization [32].

2.6 Deep learning

Deep learning is a subset of machine learning models that focus on using many layers of neural networks inspired by the neuron connections in the brain. The goal of these layers is to learn patterns in the input structures by updating its model parameters during training. Deep learning models are termed deep because the models are represented by many subsequent layers of data transformation where each layer transforms the features extracted by the previous layer [31]. One of the key advantages of deep learning compared with other machine learning algorithms is that it replaces labor-intensive feature engineering that is often needed to preprocess the data correctly. In deep learning, feature extraction is done automatically, allowing many more feature choices than what is possible for a human [31].

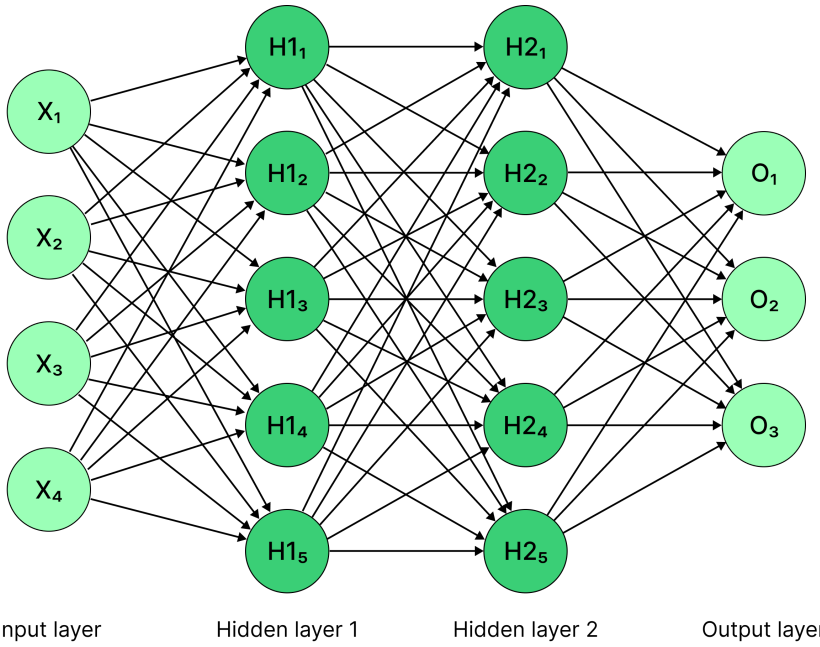

Figure 2.7: Multilayer perceptron with two hidden layers with 5 neurons each. This network has 4 inputs and 3 outputs. Each neuron in the hidden layers receives information from all neurons/inputs in the previous layer.

2.6.1 Multilayer perceptron

The multilayer perceptron (MLP) was the first neural network that uses deep layers and can be called a deep neural network [31]. A standard MLP consists of neurons organized in several layers where all neurons are fully connected to the neurons in neighboring layers (Fig 2.7). The neurons then receive input from the neurons in the previous layer through the connections. The connections between the neurons are weighted so the output from one layer will be multiplied by a weight before being inputted to the next neuron. Each neuron also contains a bias, which is a fixed value that is added to the inputs to the neuron. The connection weights and neuron biases are the trainable parameters of the network.

A plain MLP can be considered a linear model [31]. When the weighting and addition of the bias in the layers are the only effects the network has on the input, any increased value in the network input will result in an increased network output. To change the linearity of the multilayer perceptron, each neuron will also have an activation function. Activation functions decide whether the neuron should be activated or not based on the input value to the neuron. A very popular activation function is the *rectified linear unit* (ReLU) [31]:

$$ReLU(x) = max(0,x) \tag{2.13}$$

The ReLU activation function provides a very simple nonlinear transformation. ReLU will only keep the positive input it receives as all negative input will be set to zero. The derivative of the ReLU function is also simple as it is zero below an input value of zero or less and one with positive input.

2.6.2 Forward propagation

Forward propagation is the operation that takes all the inputs, passes them through the network, and results in the outputs. In this operation, all inputs are multiplied by the weights of the first layer, and the bias is added (Eq. 2.14). Then the output from this layer is calculated by passing all neuron values through the activation function of the layer (Eq. 2.15). This process is then repeated for all layers (Eq. 2.16 and 2.17):

$$\mathbf{Z}^1 = \mathbf{w}^1\mathbf{X} + \mathbf{b}^1 \tag{2.14}$$

$$\mathbf{A}^1 = \sigma(\mathbf{Z}^1) \tag{2.15}$$

$$\mathbf{Z}^l = \mathbf{w}^l\mathbf{A}^{(l-1)} + \mathbf{b}^l \tag{2.16}$$

$$\mathbf{A}^l = \sigma(\mathbf{Z}^l) \tag{2.17}$$

Here $\mathbf{X}$ is the input to the network, $\mathbf{w}^l$ are the weights in layer l, $\mathbf{b}^l$ are the biases in layer l, $\mathbf{Z}^l$ are the result from weights and biases in layer l, σ is the activation function and $\mathbf{A}^l$ is the output from the activation function and the input to the next layer. The forward propagation stores all intermediate variables [31], such as the hidden layer outputs. Storing the intermediate variables is important when the network is trained.

2.6.3 Network training

In order to achieve good results from a neural network, it is important to optimize the parameters of the network to the task at hand. To do this, the network needs to be trained. Training a neural network involves passing large amounts of data through the network using forward propagation before updating the network parameters based on some error between the predicted network output and the expected output based on the ground truth labels. The error between the network output and ground truth is calculated using a loss function. The mean square error (MSE) is an example of a loss function often used in deep learning regression tasks [33]:

$$L(\mathbf{y_s}, \hat{\mathbf{y}}_\mathbf{s}) = \frac{1}{m}\sum_{s=1}^{m}(\mathbf{y_s} - \hat{\mathbf{y}}_\mathbf{s})^2 \tag{2.18}$$

L is the loss function, m is the number of batches, $\mathbf{y}$ is the expected result, and $\hat{\mathbf{y}}$ is the predicted output. In deep learning, the loss function is used to calculate gradients for all the trainable parameters in the network. This is done in a process called backpropagation.

Backpropagation

Backpropagation refers to the process of traversing the network in reverse order to calculate the gradients of the network parameters [31]. The goal of backpropagation is to calculate the partial derivative $\partial L/\partial w$ of the loss function with respect to every weight w in the network (or bias b) [34]. This partial derivative reveals how quickly the loss function changes when a change in the parameter happens. These partial derivative terms then becomes the gradient of the loss function, as changing the parameters in the opposite direction of the partial derivatives would result in reduced loss. To calculate the partial derivatives, backpropagation uses the chain rule of derivatives to propagate the network backward to find the partial derivatives of the loss function with respect to each parameter. The chain rule is as follows [33]:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} \tag{2.19}$$

By propagating the error backwards through the network using the chain rule, all the partial derivatives for the weights, $\frac{\partial L}{\partial w}$, and the biases, $\frac{\partial L}{\partial b}$, can be calculated. The chain rule shows why it is important to store the intermediate values from the forward pass when training a neural network, as the intermediate values are necessary to calculate the partial derivatives throughout the network.

Gradient descent

After backpropagation through the network is performed and all the gradients are calculated, the network parameters need to be updated. To do this, the computed gradients are used in a process called gradient descent. In gradient descent, all the network weights and biases are updated by moving them in their negative gradient. To avoid changing the parameters too much, the gradients are multiplied by a small magnitude called the learning rate [34]:

$$\theta_t = \theta_{t-1} - \alpha \nabla L(\theta_{t-1}) \tag{2.20}$$

Here θ are all the trainable parameters in the network, t is the training iteration, α is the learning rate, and $\nabla L(\theta)$ is the gradient of the loss function L with respect to all the parameters. By subtracting the gradients, multiplied by the learning rate, from the parameter value, the parameters are moved in the direction that will reduce the value of the loss function [31]. Backpropagation and gradient descent are repeated for each batch passed through the network during training, updating the network parameters each time until they converge or the training budget runs out. The full training process is then subsequent operations of forward propagation, backpropagation and gradient descent to update the network weights and biases with the goal of minimizing the loss from the loss function (Fig 2.8a). This is a time-consuming process, and the training data often needs to be passed through the network several times to achieve a good result. In deep learning, one pass of the entire set of training data through the network is called an epoch [31].

Adam

Adaptive movement estimation (Adam) is an optimization algorithm that combines gradient descent and several other optimization techniques into one efficient learning algorithm. It combined stochastic gradient descent, momentum, adagrad and RMSprop into one learning algorithm [31]. Stochastic gradient descent reduces the computational

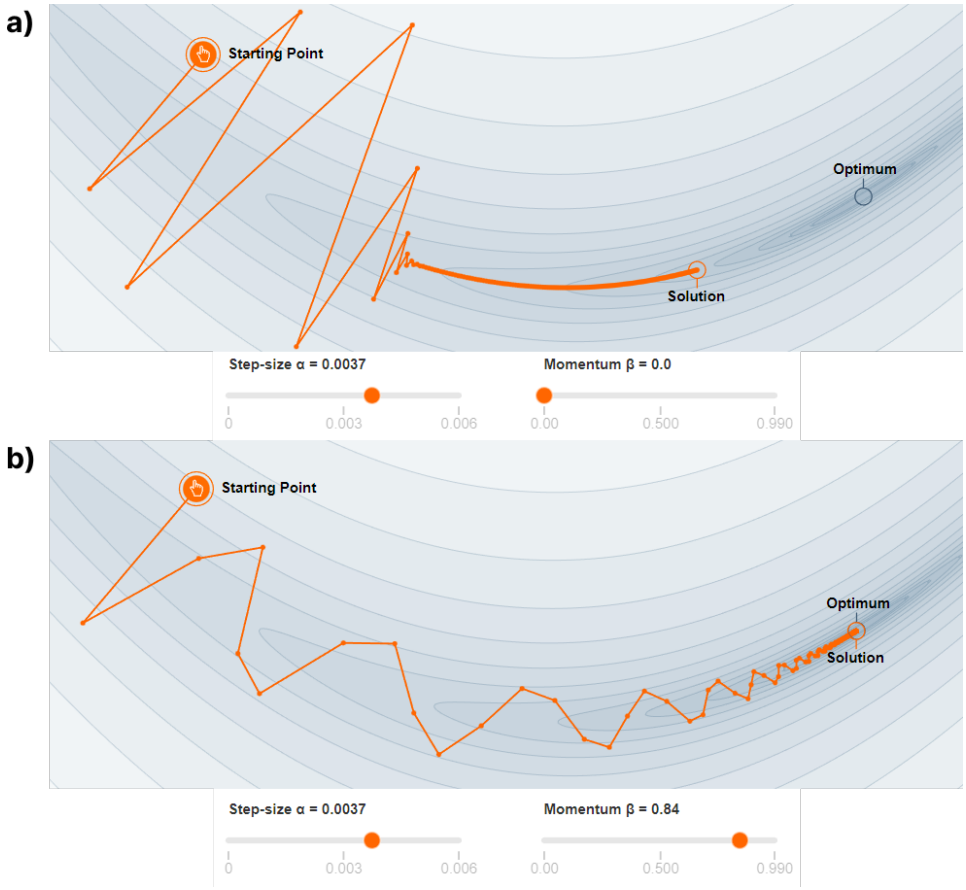

Figure 2.8: Visualization of gradient descent of a network with two parameters. a) gradient descent using a learning rate of 0.0037 without momentum. The solutions oscillate between the "ridges" of the solution space. The solution does not reach the optimum during training. b) Addition of a momentum of 0.84 reduce the oscillations and reach the optimum during training. Figures are generated from [35].

cost of gradient descent by computing the gradients of a random set of samples from the data instead of the full data [31]. Momentum replaces the computation of the instantaneous gradient for each iteration with one that has been averaged over multiple past gradients, giving the gradients a momentum [31] (Fig 2.8b). Momentum can lead to faster optimization as it can dampen oscillations in parameter space. Adagrad is a learning rate optimization method that uses the variance of past gradients to compute individual learning rates for every gradient coordinate [31]. RMSprop is an extension of the Adagrad that solves one issue with the Adagrad which is that the learning rate decreased at a predefined schedule. RMSprop helps Adam compute individual adaptive learning rates for different parameters [36]. By combining all these optimization techniques, Adam becomes a robust and efficient optimization algorithm that has become very popular in deep learning [31]

2.6.4 Hyperparameters

Hyperparameters in deep learning are the parameters that control the learning algorithm and the structure of the underlying model. They are called "hyperparameters" because they are set before the training process begins, and they are not learned by the model during training as the model parameters are [31]. Examples of hyperparameters in deep learning include the learning rate, batch size, number of hidden layers, number of neurons in each layer, activation functions, regularization parameters, and weight initialization schemes.

2.6.5 Hyperparameter optimization

The performance of every machine learning model depends on its hyperparameters. The hyperparameters control the training algorithm and the structure of the neural model, and no hyperparameter choice is best for every machine learning case. The choice of hyperparameters is often set in a trial-and-error manner or left to their default values [31]. Deciding on hyperparameters in a trial-and-error manner can be time-consuming, especially with large models and large datasets. Automation of the hyperparameter selection procedure is therefore favorable. Hyperparameter optimization (HPO) algorithms are designed to do this. HPO algorithms aim to reduce the error on a hold-out validation set by attempting different sets of hyperparameters. These algorithms are often also constrained by additional objectives such as training time or other budgets that limit the total running time [31]. Bayesian Optimization and Hyperband (BOHB) is an HPO that attempts to find good hyperparameters on a limited budget.

Bayesian optimization and hyperband

Bayesian Optimization and Hyperband (BOHB) is an HPO that combines two different hyperparameter optimization methods with different strengths to achieve better and faster hyperparameter optimization.

Bayesian optimization (BO) is an optimization algorithm that is used to select the next set of hyperparameters to try. Each machine learning algorithm can be modeled as an objective function of its hyperparameters, where the goal of the HPO is to find the combination of hyperparameters that optimize the objective function [37] (e.g. to maximize test set performance). In machine learning, the objective function is often a black-box function that can only be observed at tested data points. Bayesian optimization uses a probabilistic surrogate model of the objective function that maps from the hyperparam-

eter space to the performance of the objective function. At each step of the algorithm, BO uses the surrogate model and an acquisition function to find the next data point (set of hyperparameters) to test (Fig 2.9). The acquisition function is based on the probabilistic surrogate model and uses a trade-off of exploration of new areas in the hyperparameter space and utilization of areas close to good sets of hyperparameters to find the next set to test [37]. A common choice of acquisition function is the expected improvement over the currently best observed value [37].

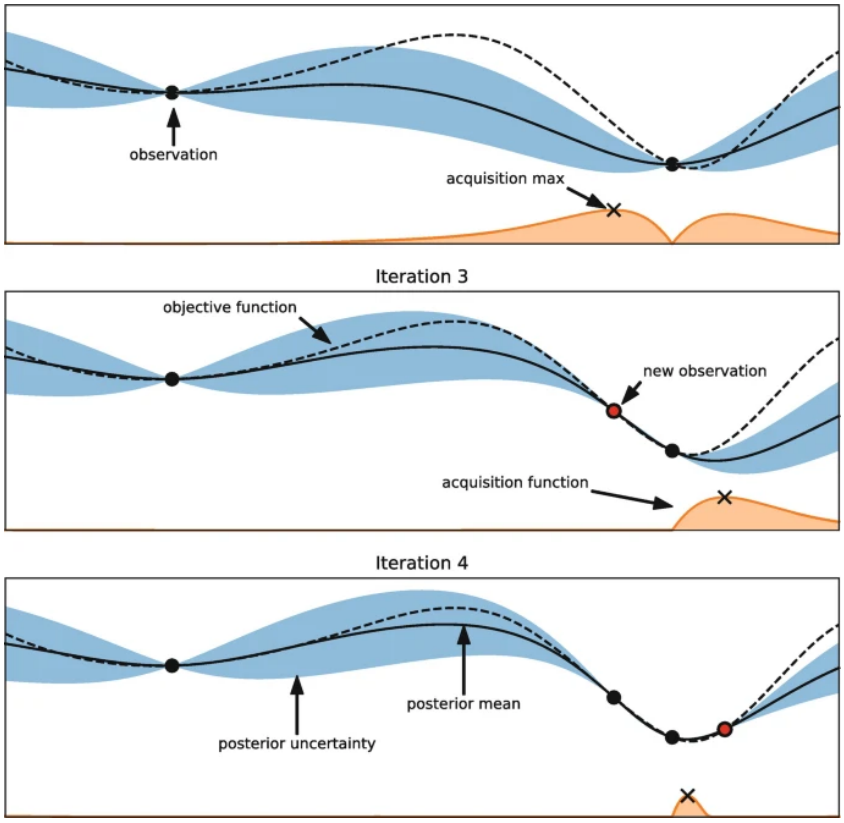

Figure 2.9: Illustration of Bayesian Optimization on a 1-d function. The dashed line represents the objective function to optimize. The surrogate model is represented by a prediction (whole black line) and an uncertainty (blue band). The acquisition function (yellow) is used to choose a new observation based on its maximum value. The acquisition function is a trade-off between the value and the uncertainty of the surrogate function. The acquisition max is a point where the value of the predicted value of the surrogate model is low and its uncertainty is high. The figure is retrieved from [38]

Hyperband is an optimization strategy that extends the successive halving (SH) algorithm [39]. Successive halving is an approach that attempts to optimize its budget (e.g. total training time) by dropping poorly performing models. In SH, the budget is uniformly allocated to a number of models with different hyperparameter configurations, then the performance of all models is evaluated, and the worst half is dropped. This process is then repeated until only one model remains [39]. One issue with the SH algorithm is that it requires manual input of how many different hyperparameter configurations to run. Hyperband extends SH by not using a predefined number of con-

figurations, but rather considering several different numbers of configurations [39]. In this way, the hyperband algorithm attempts to both allocate a small portion of the budget to many configurations leading to aggressive halving and giving a few models a large budget.

2.6.6 Convolutional neural networks

Convolutional neural networks (CNNs) are a special kind of neural network that is developed for processing data that has a grid-like structure [33]. An MLP ignores the structure of grid-based data and treats all input pixels on the same footing. A more preferable approach would be to use the prior knowledge that nearby pixels are typically related to each other [31]. CNNs have been designed especially for this purpose. As the name implies, CNNs are centered around the mathematical operation called convolution [33]. Convolutions are used to create the intermediate output values between convolutional layers, called feature maps. CNNs often also use pooling layers to reduce the number of features in the network.

Convolutions

CNNs use convolutional kernels to create feature maps from the input data [33]. Instead of weights and biases connecting every input feature to every neuron, as in the MLP, the convolutional kernels are used to stride over the input and create the feature map (Fig 2.10). The feature map is filled by performing an elementwise multiplication between the convolutional kernel and a portion of the input that has the same size as the kernel (e.g. 3x3). The resulting matrix values are summed together to one pixel value in the feature map. The first feature map value is the result of the multiplication by the kernel in the top left corner portion of the input, and by striding the kernel over the input, all feature map pixels can be covered. One kernel is used to create one map, and by applying several kernels, several different feature maps can be created from the input.

In a CNN, it is the kernel values that are optimized during training. Since every kernel is used for the entire input image, the CNN layers will have a lot fewer trainable parameters than a fully connected layer. This is because the features in the feature map are calculated as a result of kernel weights and biases that are shared across the entire input.

The size of the output feature map from a convolution will depend on the size of the input features and some kernel parameters. The size of the kernel and the number

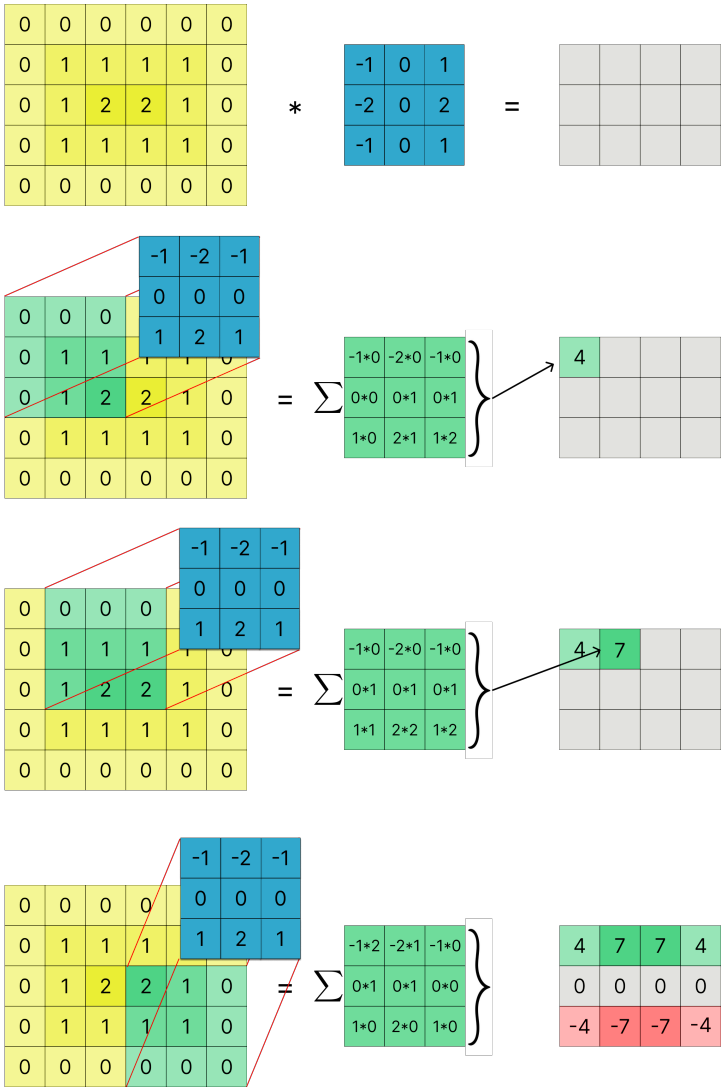

Figure 2.10: Example of convolution of an image using a convolutional kernel of size 3x3. This example shows a horizontal edge detection kernel.

of pixels it is moved between calculations (stride) will change the feature map size. Padding the input features can increase the size of the feature map [31].

Pooling

A typical layer in a CNN consists of the convolutional kernels described above, an activation function applied to the feature space to avoid linearity, and finally, a process called pooling [33]. In many deep learning applications, it is desirable that the output features are sensitive to the entire input to the network. An MLP does this by fully connecting all neurons in neighboring layers to each other. However, for a CNN with only convolutional kernels in its layers, this will be done gradually by aggregating the information with the kernels [31]. Pooling will accelerate this process by reducing the spatial resolution of the input features. Like the convolutional layers, the pooling

operator consists of a fixed window that slides over its input, and for each iteration, it will reduce the portion of the input it is currently operating on to one value. This can either be the average of all the feature values it is currently operating on (average pooling) or the maximum of these values (maximum pooling). Figure 2.11 shows an example of the pooling procedure of a 2x2 pooling kernel with a stride of 2.

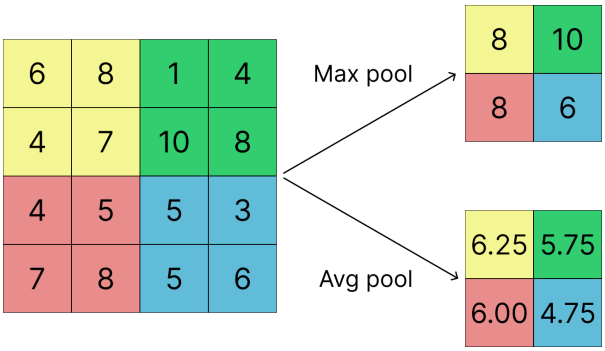

Figure 2.11: Example of the maximum and average pooling procedures with a pooling kernel of 2x2 and a stride of 2. This will halve the number of features in each dimension

UNet

The UNet is a CNN architecture introduced by Ronneberger, Fischer, and Brox [40] in 2015 (Fig 2.12). The UNet is a fully convolutional neural network that uses an encoder-decoder structure. The encoder functions as an analysis path where the network discovers essential features in the input, while the decoder is a synthesis path where the output from the encoder is interpreted to produce the final output [41]. The encoding and decoding are performed in layers. One layer consists of one encoder block and one decoder block. In the encoder blocks, the number of feature maps is increased while the size of each feature map is decreased. The decoding blocks reverse this process by increasing the size of each feature map while reducing the number of feature maps. Each block in the encoder consists of a set of two convolution procedures that increase the number of feature maps, each followed by an activation function, and, at the end of the block, a max pooling operation that reduces the size of the feature maps. The blocks in the decoder consist of an upsampling that increases the feature maps' size and two convolutions that reduce the number of feature maps, each followed by an activation function [40] [41]. As the encoder applies pooling to reduce the redundancy in the features, much high-resolution information is lost and cannot be recovered by the decoder [42]. To mitigate this, the UNet employs skip connections between the encoder and decoder block in the same layer. In a skip connection, the feature maps from the last set of convolutions in the encoder block are concatenated to the output from the upsampling in the previous decoder block. The name, UNet, stems from the U-shape of

the network that results from the pooling and upsampling between the layers (Fig 2.12).

The original UNet [40] and its 3D-version [41] were designed for image segmentation tasks, using a final activation of either a softmax or sigmoid activation function. But its encoder-decoder structure makes it applicable to several image-to-image tasks, such as regression, denoising and image reconstruction [42].

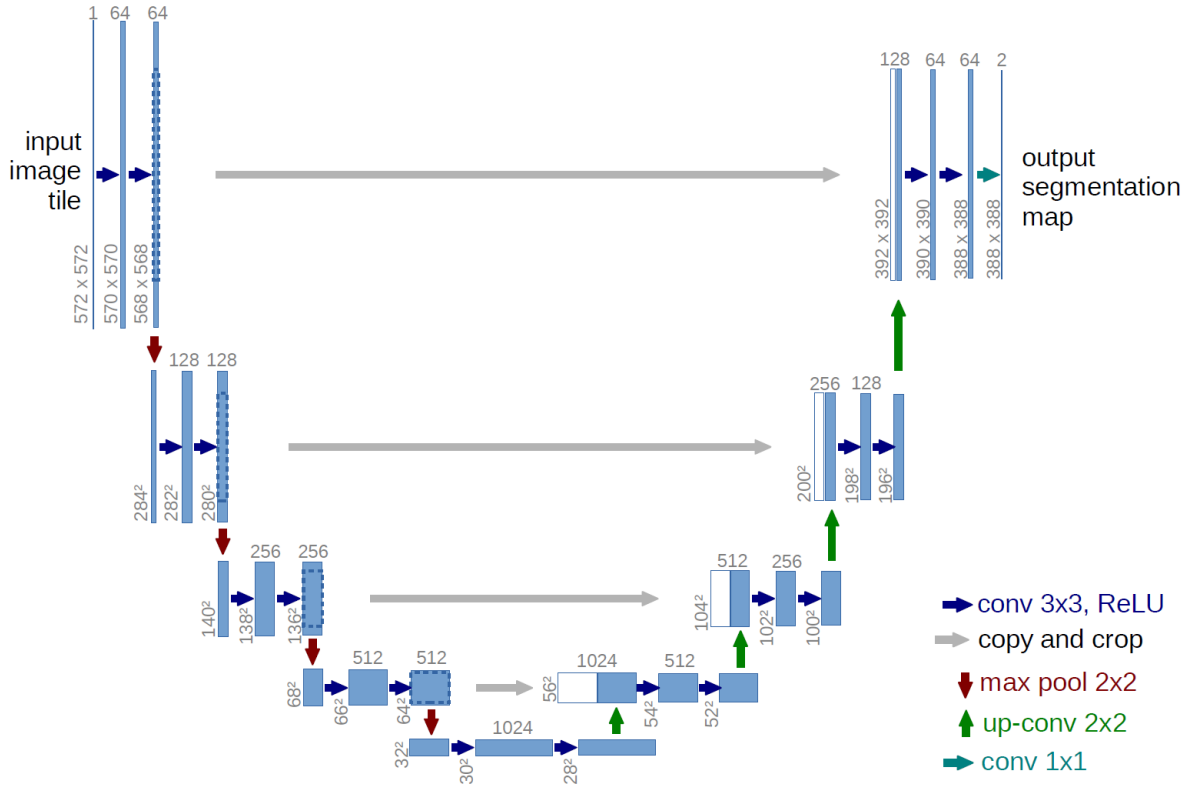

Figure 2.12: The original UNet as proposed by O. Ronneberger et al. in 2015. Reprint from [40]

Chapter 3

Methods

This chapter presents the methodology of this project. One thing to keep in mind is that during the project, a conscious choice was made to switch between an unprocessed and a processed dataset. Many steps in the methodology were the same for these two datasets, and the methodology is thus presented as one.

3.1 Data acquisition

3.1.1 Image data collection

The data used in this project was resting-state fMRI data, provided by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University. The data was downloaded from https://db.humanconnectome.org. The HCP data contained images from 181 participants imaged on a 7T MR system. The HCP rs-fMRI images are imaged using EPI for high temporal resolution. EPI is quite sensitive to specific image artifacts [43]. In the first parts of this project, an unprocessed dataset is used. As many of the artifacts present in the unprocessed HCP images occur due to the EPI sequence, the same artifacts would not appear in a Looping Star scan.

To avoid the EPI artifacts, a dataset consisting of processed rs-fMRI images was also used as a more accurate representation of Looping Star images. This dataset was used in the later parts of this project. This dataset has been processed according to the HCP functional processing pipeline that includes processes described in Glasser et al. [44]. The functional preprocessing pipeline is separated into the *fMRIVolume* and the *fMRISurface* pipelines. fMRIVolume removes spatial distortions, realigns volumes, nor-

malizes the 4D image and masks the data with the final brain mask [44]. fMRIVolume ensures that no overt volume smoothing is performed. fMRISurface is used to bring the time-series from the volume into the CIFTI grayordiante standard space [44]. Data from this preprocessing step was not used. The preprocessing change the image parameters of the unprocessed data (Tab 3.1).

Table 3.1: Image data information of the unprocessed and processed 7T rs-fMRI datasets from the HCP. FOV = Field of View

Image reconstruction parameters	Unprocessed image data	Processed image data
Initial image matrix size	130 x 130 x 85	113 x 136 x 113
Voxel size [mm^3]	1.6 x 1.6 x 1.6	1.6 x 1.6 x 1.6
FOV [mm^3]	208 x 208 x 136	180.8 x 217.6 x 180.8
Number of slices	85	113
Number of volumes	900	900
Temporal resolution [s]	1	1

3.1.2 Looping Star trajectory

The k-space trajectory that was used was acquired using a Looping Star fMRI protocol that records the FID and one echo. Due to the nature of the Looping star acquisition, the echo k-space signal consists of both the echo-out signal from the echo and the echo-in signal from the previous FID signal (Fig 3.1). In this acquisition, these two signals have been separated to dampen signal noise. Separation of the signal results in more data samples per spoke, as each spoke contains both the echo-in and echo-out signal. Table 3.2 contains information on the Looping Star protocol. As the Looping Star trajectory consists of rotated disks of radial samplings, k-space will be more densely sampled at the center. Therefore, in addition to the k-space trajectory, density correction factors (DCF) corresponding to the k-space samples were provided to avoid over-representation of the densely sampled center. These were used during the reconstruction of the Looping Star images, which are described in the next subsection.

Table 3.2: Information on the imaging parameters and k-space sampling details from the Looping Star protocol. FOV = Field of View, FA = Flip Angle

Imaging parameters	
FOV [cm]	19.2
Slice thickness [mm]	3
FA	3°
Looping Star sampling details (Fig 3.1)	
Number of segments	21
Number of spokes per loop	24
Number of samples per spoke	128

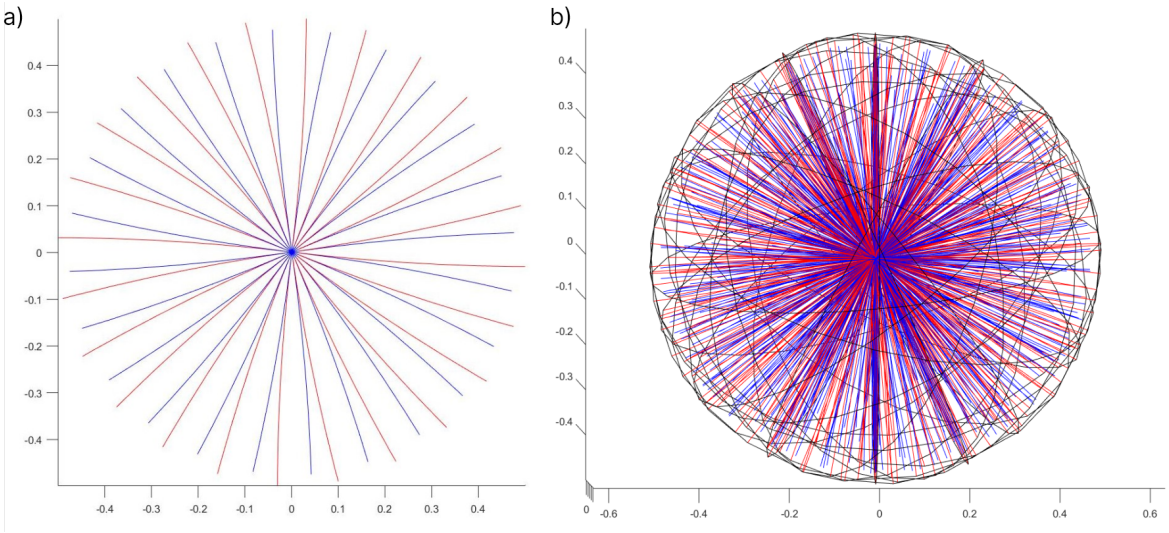

Figure 3.1: Looping Star k-space. a) shows the echo-in (red) and echo-out (blue) trajectories in a single radial segment. b) shows the full Looping star echo k-space trajectory consisting of 21 segments. Black lines show the outline of each segment of a radial trajectory.

3.1.3 Looping Star acquisition emulation

To create the Looping Star images, the rs-fMRI data from the Human Connectome Project was resampled as if it was acquired with the Looping Star k-space trajectory. As the non-Cartesian Looping Star k-space contains fewer k-space samples than the Cartesian sampling of the original HCP data, the resampling will hereby be referred to as downsampling. The downsampling was performed using the Michigan Image Reconstruction Toolbox (MIRT) by Jeffrey A Fessler [45] in MATLAB, (2022b), Natick, Massachusetts: The MathWorks Inc. [46]. MIRT enables the use of a permutable mathematical operator with an inverse that is useful for problems that require reconstruction of a function from nonuniform Fourier samples, such as non-Cartesian MRI. The operator takes k-space coordinates as input, which enables the operator to emulate MRI signal acquisition based on the k-space coordinates [45]. This is done in the forward operation of the operator. In the adjoint operation, the operator performs nonuniform FFT by conjugate phase reconstruction for the reconstruction of a set of k-space samples that correspond to the coordinates that built the object.

By using the acquired Looping Star k-space coordinates to build the operator, the forward operation of the mathematical operator emulated the Looping Star acquisition on any volume, while the adjoint operation reconstructed the emulated k-space samples by conjugate phase reconstruction. The mathematical operator was used by creating k-space samples corresponding to the Looping Star k-space coordinates using the forward operation. Then, density correction factors were multiplied with the corresponding k-

space samples before the image was reconstructed using the adjoint operation of the mathematical operator.

Due to the spherical shape of the Looping Star trajectory, the best reconstruction by the mathematical operator from MIRT was achieved when the samples are sampled from a square volume and the volume was reconstructed with a square matrix shape. Therefore, the matrix size of the input HCP data was increased by adding zero-padded slices before downsampling, and the reconstruction was performed with a square logical mask as a template. One issue with the reconstruction using the mathematical operator was that it produced image artifacts when reconstructing the image from the downsampled k-space. At small image matrix sizes, this artifact affected the image by shading areas of the brain in the image. Increasing the image size before reconstruction pushed this artifact outside of the brain volume, minimizing its disturbance on the brain.

The downsampling process was the same for both the unprocessed and processed datasets. The only variation was the size of the image matrix and reconstruction masks that were necessary to push the artifacts outside the brain.

3.2 Dataset preparation

Increasing the matrix size by concatenation of zero-padded slices did not increase the size of the brain in the images. The entire increased matrix size was therefore not needed for training the convolutional neural networks. After the emulated Looping Star images were acquired, the matrix size of the images was reduced before training. The final image matrix size was decided by the number of slices necessary to cover the entire brain. For the unprocessed dataset, the final matrix size was 128x128x90 while the final matrix size was 128x128x113 for the processed dataset. The downsampling process results in pairs of images with one HCP rs-fMRI time-series and one Looping Star time-series per participant.

3.2.1 Dataset splitting

The data was separated into training, validation, and testing sets for training. 15 random participants were selected as the testing set. From the remaining 166 participants, 10 were selected as a validation set used for model selection and evaluation during training. For the training and validation set, 50 consecutive volumes were downsampled. Several volumes per participant were downsampled to let the models train on volumes with rs-fMRI BOLD variations. For the testing set, all 900 time-series volumes were

downsampled for all the participants. This was necessary as the full sequence was needed to perform fMRI analyses on the participants. Dataset information is displayed in table 3.3. All datasets consisted of LS reconstructions and corresponding original HCP image data. Data augmentations were added to training and validation sets to increase the dataset size and diversity. The augmentations consisted of flipped and rotated image volumes from all the participant's data. The dataset splitting and augmentation were identical for the unprocessed and processed data, and the testing participant IDs were also the same for both datasets.

Table 3.3: Number of participants and number of volumes per participants for the training, validation and testing datasets

	Number of participants	Volumes per participants
Training set	156	50
Validation set	10	50
Testing set	15	900

3.2.2 Data loading

Two separate data loaders were employed when loading the data for training. Two loaders were used because of the substantial amount of training data and the large file sizes of the images. The first loader was responsible for loading participant data into the computer memory, while the second loader facilitated the transfer of data from the computer memory to the models.

Throughout the training process, the first data loader fetched the data in batches of four participants. All the volumes for these four participants were loaded into the computer memory and fed to the second loader. The second loader performed image augmentations and created training batches by randomly selecting data from the batch stored in the computer memory. The network was then trained on all volumes, including augmentations, for the four participants in one batch before proceeding to load a new batch into the computer memory. Utilizing two data loaders was done to circumvent the time-consuming task of repeatedly loading the large image files into computer memory during training. By loading batches consisting of four participants, the loaders ensured that the network was not trained solely on data from one participant at a time, thereby promoting a more balanced training.

3.3 Network training

Two deep-learning models were explored to attempt to improve the quality of Looping Star images for both the processed and unprocessed datasets. PyTorch version 1.13.0 was used for network modeling and training [47]. PyTorch is an open-source optimized tensor library for deep learning using GPU and CPU. Ray Tune was used for hyperparameter tuning and model selection for the processed dataset [48]. Tune is a library in Ray for hyperparameter tuning that employs a wide range of search algorithms and hyperparameter optimization tools. Ray version 1.6.0 was used in this project. An NVIDIA GeForce RTX 3080 Ti graphics card with CUDA version 11.7 was used for GPU training. PyTorch was used within Python version 3.9.13 [49]. In this project, a 2D-UNet and a 3D-UNet were trained for unprocessed data. A 3D-Unet was also trained for processed HCP data. Mean square error (MSE) between model output and ground truth images was used as the loss function for training and was also used in validation during training.

3.3.1 Network training on unprocessed data

Both a 2D-UNet and a 3D-UNet were trained on the unprocessed dataset. The use of a 2D-UNet for deep learning image reconstruction of a radial sampling in MRI was inspired by Han et al. [10]. They managed to improve the image quality and reduce the image artifacts that were a result of the non-Cartesian sampling. The Looping Star sequence use rotated radial samplings to get its 3D-sampling, which is different from the sampling used by Han et al. They used one radial sampling per image slice. The noise and artifacts resulting from such a k-space sampling are expected to be uniform across the slices. This is not the case for Looping Star data. The application of a 3D-UNet, typically used for image segmentation, for this project was done to attempt to improve the network performance by allowing the network to see global artifacts across the entire volume through 3D convolutions.

Training of the 2D-UNet and 3D-UNet used similar optimization schemes. The Adam optimizer was used for updating the network weights during training. Adam performs faster weight optimization compared to e.g. SGD. The only difference in the optimization schemes was the learning rates used to train the models. An initial learning rate of $1x10^{-3}$ was used for the 2D-UNet and a learning rate of $5x10^{-5}$ was used for the 3D-UNet. Due to the large amount of data, which included several volumes per participant, the networks were trained for 20 epochs. The final model for both networks was saved as the instance where it performed best on the validation set, even if that point

was reached before the 20 epochs had passed.

3.3.2 2D-UNet specifics

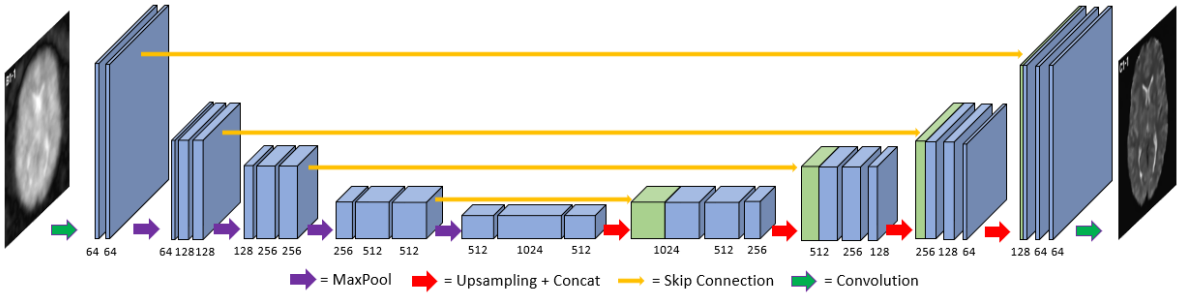

Figure 3.2: 2D-UNet architecture. Blue blocks are feature maps. The height and width show the size of the maps, while the length indicates the number of channels. Green blocks are feature maps from the encoder in the same layer. Stacks of three feature maps make up one encoder/decoder block, where 2D convolution, 2D batch normalization and ReLU activation are performed between the feature maps in each block. The numbers below the cubes indicate the number of channels for each block. Figure inspired by Han et al. [10].

For the 2D-UNet, the implementation by Milesial was applied [50]. This implementation was chosen due to its similarity with the 2D-UNet used by Han et al. [10]. The 2D-UNet took 1 input channel and returned 1 output channel (1 slice). The architecture consisted of an initial double convolution, an encoder-decoder structure with 5 layers and a final convolution with kernel size 1 (Fig 3.2). Each encoder block consisted of two sets of convolutions in the following order: 2D convolution, 2D batch normalization, followed by an activation function. The convolutions in the encoders increased the number of channels. The two sets of convolution were followed by a max pooling operation.

The decoder blocks started with a bilinear upsampling of the output features from the previous layer. Bilinear upsampling scaled the feature map size by a factor of two and filled in the blank features using linear interpolation. Skip connections concatenated the upsampled features with the output from the last convolution in the encoder block in the same layer. At the end of the decoder, a set of convolutions similar to the ones in the encoder was used to create the output from the decoder block. The convolutions in the decoders reduced the number of channels.

The 2D-UNet was altered to fit a regression task by dropping the final softmax activation before output. Convolutional kernel size, pooling kernel size, activation function and padding was kept constant for all encoder/decoder blocks (Tab 3.4). For training of the 2D-UNet, the second data loader created training batches of 90 random slices

Table 3.4: Hyperparameters of 2D-UNet network.

2D-UNet Hyperparameters	
Convolutional kernel size	3x3
Number of trainable parameters	31 036 481
Pooling kernel size	2x2
Padding	1
Activation Function	ReLU
Upsampling Algorithm	Bilinear
Number of channels per layer	64, 128, 256, 512, 1024

from the participant data in the computer memory. The size of 90 was set due to GPU memory limitations.

3.3.3 3D-UNet specifics

For the 3D-UNet, the implementation by Adrian Wolny was used (pytorch-3dunet version 1.3.9) [51]. This implementation facilitated regression tasks by giving the opportunity to skip the final sigmoid/softmax activation layers commonly used for segmentation tasks. The 3D-Unet architecture that was used took 1 input channel and returned 1 output channel (1 volume). The encoder-decoder structure of the architecture consisted of 4 layers (Fig 3.3). The encoder blocks in each layer consisted of two sets of convolutions using the following order for each convolution: group normalization, 3D convolution, followed by an activation function. The two sets of convolutions were followed by max pooling between the encoding layers. Each convolution in the encoder increased the number of channels.

The decoder blocks started with upsampling of the output feature maps from the previous layer. Skip connections concatenated the upsampled feature maps and the output maps from the encoder block on the same layer. The decoder blocs also used two sets of convolutions of the total concatenated features. The two sets of convolutions in the decoder blocks followed the same order as in the encoder blocks. Each convolution in the decoder reduced the number of channels. Convolutional kernel size, pooling kernel size and padding was kept constant for all layers (Tab 3.5).

The increase and decrease of the feature maps between the encoders and decoders were performed in steps in the double convolution blocks (Fig 3.3). The size of the feature maps was increased by half of the output size in the first of the two convolutions and further increased to the correct output size in the last convolution. The number of layers, feature maps and kernel sizes were chosen to be kept as the implementation

Table 3.5: Hyperparameters of 3D-UNet network.

3D-UNet Hyperparameters	
Convolutional kernel size	3x3x3
Number of trainable parameters	16 318 691
Pooling kernel size	2x2x2
Padding	1
Activation Function	ReLU
Upsampling Algorithm	Nearest Neighbor
Number of channels per layer	64, 128, 256, 512

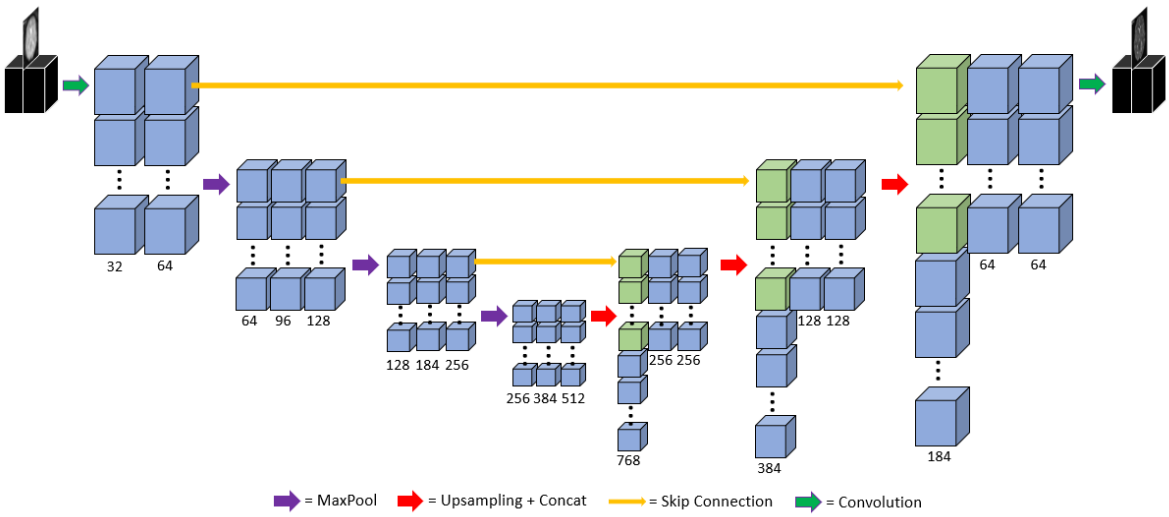

Figure 3.3: 3D-UNet architecture. Blue cubes are one feature map and stacked feature maps represent feature channels. Green blocks are feature maps from the encoder in the same layer. Groups of feature channels make up one encoder/decoder block. Group normalization, 3D convolution and ReLU activation are performed between each block's feature channels. The numbers beneath the stacks of blocks indicate the number of channels.

default and not increased due to limitations of GPU memory. For training of the 3D-UNet, the second data loader created training batches consisting of 1 volume from the participant data in the computer memory. This size was set due to GPU memory limitations.

3.3.4 Model selection for processed data

For training of the model on processed data, the 3D-UNet was initialized with the best weights from the training on unprocessed data. This was done to avoid starting from scratch with randomly initiated weights in the model training. A model selection scheme was performed to select a good set of model hyperparameters for this training, as the starting point differed from the training on unprocessed data.

6 different models with different model/optimization hyperparameters were tested on

a subset of the training data consisting of 5 volumes per participant in the training set. Group normalization size of the network, learning rate and weight decay of the optimizer were the tested hyperparameters. As the amount of data and model architecture were large, a Bayesian Optimization and Hyperband (BOHB) hyperparameter search algorithm was used [37]. BOHB chooses hyperparameters from an input space of possible choices (Tab 3.6). All models were trained with the Adam optimizer and with the 3D-UNet hyperparameters in table 3.5.

Table 3.6: Hyperparameters ranges available for BOHB search algorithm

Hyperparameter	Hyperparameter space	Variable type
Group normalization size	2, 4, 8	Discrete choice
Learning rate	$1x10^{-6} \rightarrow 1x10^{-4}$	Loguniform choice
Weight decay	$0.0 \rightarrow 1x10^{-3}$	Loguniform choice

From the hyperparameter ranges (Tab 3.6), the BOHB chose 6 sets of hyperparameters (Tab 3.7). The search algorithm included early stopping of poorly performing models. This meant that models that performed poorly on the validation set were dropped during the model selection process. The model selection process ran with a maximum number of epochs of 22 for one model. Based on this total number of epochs, the search algorithm dropped two models at appropriate times during the selection process. After dropping two models, the BOHB algorithm restarted training for the remaining models. The algorithm chose epoch number two and six as appropriate times to drop models. The two worst-performing models at these intermediate epochs were then dropped. After the 6th epoch, the final two remaining models were then trained for the last 14 epochs and the model that performed best on the validation set after this was chosen for training on the complete dataset.

Table 3.7: Hyperparameters of models attempted in model selection. GN = Group Normalization

Model	GN size	Learning rate	Weight decay
1	2	2.1714e-05	0.0004888
2	4	1.9387e-06	0.0005063
3	8	3.5720e-06	0.0008863
4	8	3.3545e-05	0.0004613
5	4	6.8463e-06	0.0002887
6	8	2.0852e-06	0.0003865

3.3.5 Network training on processed data

The model that performed best in the model selection scheme was trained using the same number of epochs and volumes per participant as for the unprocessed data. This

model was then trained for 20 epochs on the complete training set of 50 volumes per participant for 156 participants. The model was saved after each epoch if the validation loss had improved from the previous epoch. The final model was then saved when it reached its best performance on the validation set.

3.4 Model evaluation on test set

Several image quality metrics were used to evaluate the performance of the different models and to examine which model performed best on unseen data. The metrics were used to compare the 2D-UNet and 3D-UNet for the unprocessed data. The metrics that were used were: Mean square error (MSE), structural similarity (SSIM), peak signal-to-noise ratio (PSNR) and signal-to-noise ratio (SNR).

3.4.1 Mean square error

MSE was calculated between ground truth and the different model outputs, as well as the Looping Star reconstruction. MSE was calculated over all volumes per participant in the test set. The models were then evaluated on their ability to reduce the MSE of the Looping Star reconstruction. The MSE calculation was performed using the *immse()*-function in MatLab using the following equation [46]:

$$MSE(x,y) = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2 \tag{3.1}$$

Here x and y are the corresponding data points in the ground truth and reconstructed image.

3.4.2 Structural similarity

SSIM was calculated between the first ground truth volume and the first volume from the different model outputs, as well as from the Looping Star reconstruction. SSIM gives a value between 0 and 1 based on how structurally similar the two inputs are. SSIM was calculated in MatLab using the *ssim()*-function. Equation 3.2 shows the SSIM equation [52].

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.2}$$

Here μ_x is an average of x, σ_x^2 is a variance of x and σ_{xy} is a covariance of x and y. c_1 and c_2 are small constraints given by: $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$, where L is the dynamic range of the pixel values. k_1 and k_2 are constants: $k_1 = 0.01$ and $k_2 = 0.03$ [52].

3.4.3 Peak signal-to-noise ratio

PSNR was calculated between ground truth and the different model outputs, as well as the Looping Star reconstruction. PSNR was calculated over all volumes for the

participants in the test set. PSNR was calculated using the *psnr()*-function in MatLab as such:

$$PSNR(x,y) = 10log_{10}\left(\frac{PEAK^2}{MSE(x,y)}\right) \tag{3.3}$$

Here peak is the maximum value possible for the data type of x and y, and MSE is the mean square error between x and y (Eq. 3.1).

3.4.4 Signal-to-noise ratio

SNR was calculated for the ground truth HCP data, Looping Star reconstruction and the model outputs. The SNR is the only one of these four image quality metrics that do not compare the different reconstructions to the ground truth HCP data. The SNR was calculated for one volume per participant by finding the standard deviation of noise in four regions of the image with only air (no signal from the brain) and finding the mean of the signal in an area of the brain. the SNR was then calculated according to equation 3.4.

$$SNR = \frac{S}{SD(N)} \tag{3.4}$$

Here S is the mean signal while SD(N) denotes the standard deviation of the noise. SNR was calculated for the first volume in all participants.

3.5 fMRI analysis

As Looping Star can be a powerful tool in fMRI due to the low acoustic noise, any evaluation of LS reconstruction methods needs to include an evaluation of the BOLD signal preservation and the images' usability in an fMRI analysis. In this project, the BOLD signal preservation was evaluated by comparing the time-series of individual voxels in the original HCP data and the LS and 3D-UNet reconstruction. As the networks were trained on resting state fMRI data, a seed-based resting-state analysis of the posterior cingulate cortex (PCC) was performed for all image reconstructions and compared. The PCC was chosen as it is a part of the default mode network (DMN). the DMN is a well-known brain network known for showing strongly correlated activity at rest [53].

3.5.1 Voxel variation correlation

Correlations between the time-series of individual voxels were calculated to look at how the network performed on time-series data. The correlation between the original HCP time-series and Looping Star reconstruction time-series as well as the correlation between original HCP and 3D-UNet reconstruction time-series were calculated for all the voxels in the brain. The network's ability to preserve and enhance the voxel variation was evaluated by comparing the correlation between the time-series of the individual voxels. The correlation was calculated using the Pearson correlation coefficient, which measures the linear dependence of two variables:

$$\rho(A.B) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{A_i-\mu_A}{\sigma_A}\right)\left(\frac{B_i-\mu_B}{\sigma_B}\right) \tag{3.5}$$

Here A and B are the different variables, μ is the variable mean and σ is the standard deviation of the variable. Before calculating the correlation coefficients, each time-series was denoised using a simple bandpass filter that removed the contributions of low frequencies below 0.008 Hz and high frequencies above 0.09 Hz. This was performed in MatLab by removing the contributions of the frequency bins below 0.008 Hz and above 0.09 Hz.

The voxel variations were displayed for eight voxels in two different example regions of the brain, one in the PCC and one in the frontal lobe. The region in the frontal lobe was chosen as this was an area where the deep learning reconstruction performed worse in the resting state fMRI analysis and thus it could demonstrate weaknesses of the 3D-UNet reconstruction.

The number of improved voxel correlations was calculated for all the participants in the testing set to get a more complete view of network performance across all the participants. Correlation maps for four example participants were also displayed to look for patterns or areas where the 3D-UNet performed well. The correlation maps were calculated by subtracting the correlation coefficient between Looping Star reconstruction and original HCP data from the correlation coefficient between 3D-UNet reconstruction and original HCP data for every voxel in the brain.

3.5.2 Resting state fMRI analyses

The resting state fMRI preprocessing and analyses were performed using the functional connectivity toolbox CONN [54] (RRID:SCR_009550) release 22.a [55] and SPM [25] (RRID:SCR_007037) release 12.7771. The following descriptions have been retrieved from the process log in CONN.

Preprocessing

Functional and anatomical data were preprocessed using a flexible preprocessing pipeline [27]. Functional and anatomical data were normalized into standard MNI space, segmented into grey matter, white matter, and CSF tissue classes, and resampled to 1.6 mm isotropic voxels. This was done using SPM unified segmentation and normalization algorithm [56][57] with the default tissue probability map template. Last, functional data were smoothed using spatial convolution with a Gaussian kernel of 8 mm full-width half maximum (FWHM).

Denoising

In addition, functional data were denoised using a standard denoising pipeline [27], followed by bandpass frequency filtering of the BOLD time-series between 0.008 Hz and 0.09 Hz.

First-level analysis

Seed-based connectivity maps (SBC) were estimated characterizing the spatial pattern of functional connectivity with a seed area. Seed regions included 164 HPC-ICA networks [55] and Harvard-Oxford atlas ROIs [58], but only the connectivity map of the PCC is included in this thesis. Functional connectivity strength was represented by Fisher-transformed bivariate correlation coefficients from a weighted general linear

model (weighted-GLM [27]), estimated separately for each seed area and target voxel, modeling the association between their BOLD signal time-series.

Group-level analyses

Group-level analyses were performed using a General Linear Model (GLM [27]) for five participants in the testing set. For each individual voxel, a separate GLM was estimated, with first-level connectivity measures at this voxel as dependent variables, and groups or other subject-level identifiers as independent variables. Voxel-level hypotheses were evaluated using multivariate parametric statistics with random-effects across participants and sample covariance estimation across multiple measurements. Inferences were performed at the level of individual clusters (groups of contiguous voxels).

Chapter 4

Results

4.1 K-space downsampling

The dataset of 181 participants from the Human Connectome Project was downsampled according to the dataset preparation mentioned in section 3.1. The downsampling process (described in 3.1.3) results in pairs of images with one HCP rs-fMRI time-series and one Looping Star time series per participant. Figure 4.1 shows the downsampling result from one volume from the testing dataset of both the unprocessed and processed images.

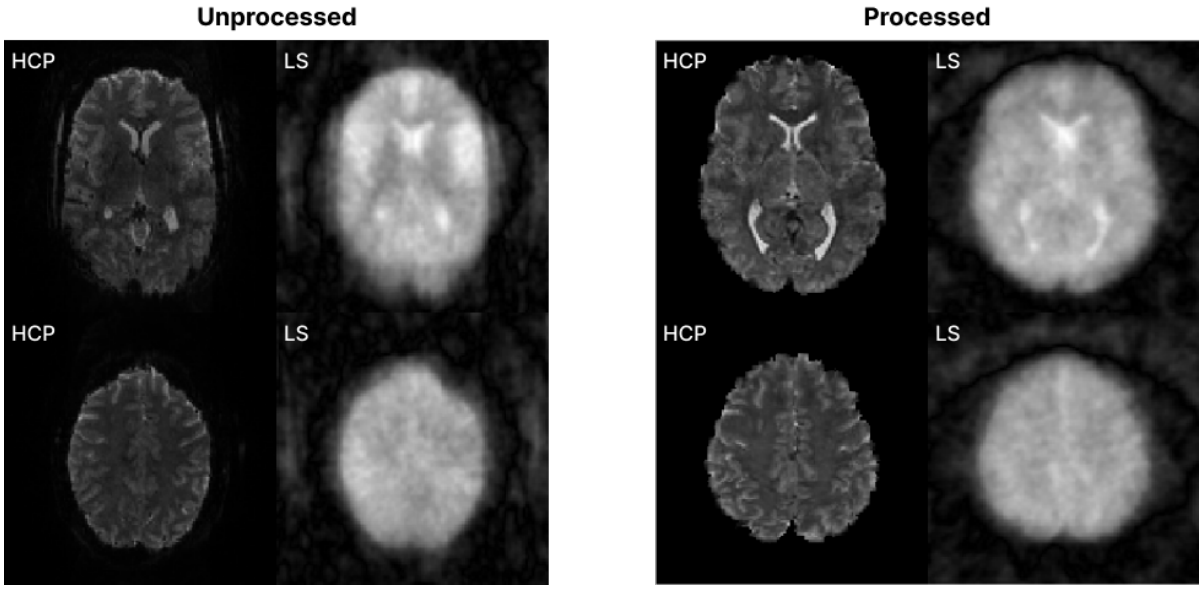

Figure 4.1: Example of one downsampled volume for one test set participant from both the unprocessed (left) and processed (right) datasets. HCP) Original HCP rs-fMRI data. LS) Image reconstructed from downsampled k-space according to Looping Star k-space trajectory. The two rows show different image slices from one volume.

Different reconstruction parameters were attempted to improve the image quality of the downsampled dataset. The parameter with the most effect on image quality was

the matrix size of the image and of the reconstruction mask. Reconstructing the image using the original matrix size results in image artifacts that disturbed the imaged object. Squaring and increasing the image matrix with zero-padded slices pushed this artifact further out of the FOV. Testing revealed that increasing the image matrix size to 150x150x150 pushed the artifact adequately outside the brain in the unprocessed HCP data (Fig 4.2). For processed data, the volume matrix size was increased to 170x170x170 to push the artifact out (Fig 4.3).

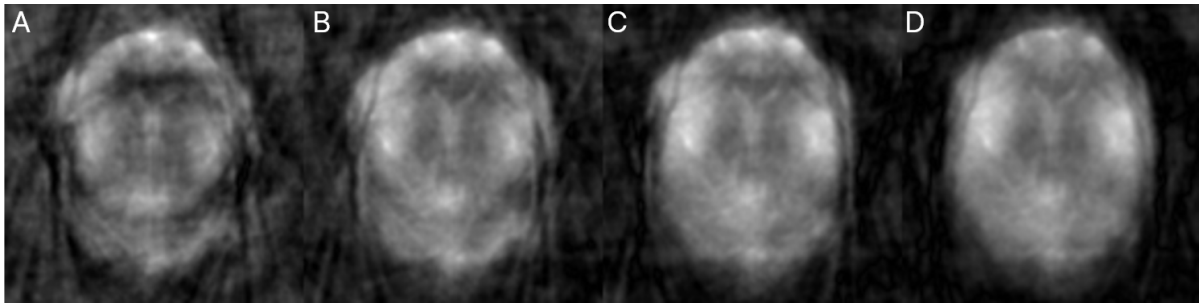

Figure 4.2: Reconstruction of emulated Looping Star images using different matrix sizes (Unprocessed HCP dataset). A) Original matrix size: 130x130x85. B) Squared reconstruction matrix to largest original dimension: 130x130x130. C) Increased square reconstruction matrix: 140x140x140. D) Reconstruction matrix further increased to push artifacts out: 150x150x150. All images are cropped for visualization.

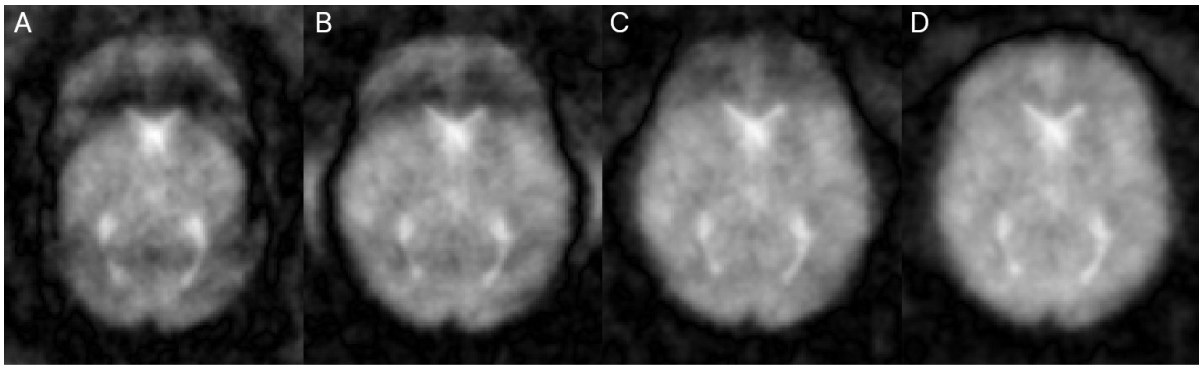

Figure 4.3: Reconstruction of emulated Looping Star images using different matrix sizes (Processed HCP dataset). A) Original matrix size: 113x136x113. B) Squared reconstruction matrix to largest original dimension: 136x136x136. C) Increased square reconstruction matrix: 150x150x150. D) Reconstruction matrix further increased to push artifact out: 170x170x170. All images are cropped for visualization.

Figure 4.1 shows that the Looping Star reconstructions have a larger general image intensity compared to the HCP data, as the brain in the images looks brighter. However, this results in less visible contrast between tissues, and the images appear blurred. Reconstruction artifacts and noise also plague the images. It appears that areas in the Looping Star images with a high signal intensity have a tendency to bleed into neighboring voxels, giving less contrast between voxels.

4.2 Network training on unprocessed data

Figure 4.4: Training and validation loss during training of 2D-UNet and 3D-UNet. Validation loss is blue while training loss is yellow. The 2D-UNet losses are represented by dashed lines and triangle markers. The 3D-UNet losses are represented by solid lines and star markers. MSE = Mean square error.

2D-UNet

Training of the 2D-UNet on unprocessed data lasted for 38 hours. The training and validation loss during training shows that the performance of the 2D-UNet converged early for the training data, as the loss plateaus after only two epochs. The performance on validation data still improved until the 15th epoch (Fig 4.4). Moving forward, the instance of the model that performed best on validation data during training was used. The best performance was achieved after the 15th epoch.

3D-UNet

Training of the 3D-UNet on unprocessed data lasted for 90 hours. The performance metrics of the 3D-UNet show that the model performed best on validation data after the 7th epoch (Fig 4.4). The model kept improving the training loss after this, but as the validation loss did not improve, the model was saved at epoch number 7. Figure 4.4 shows that the distance between the validation and training loss is narrower for the 3D-UNet compared to the 2D-UNet. This suggests that the 3D-UNet model is less overfitted to the training data compared to the 2D-UNet.

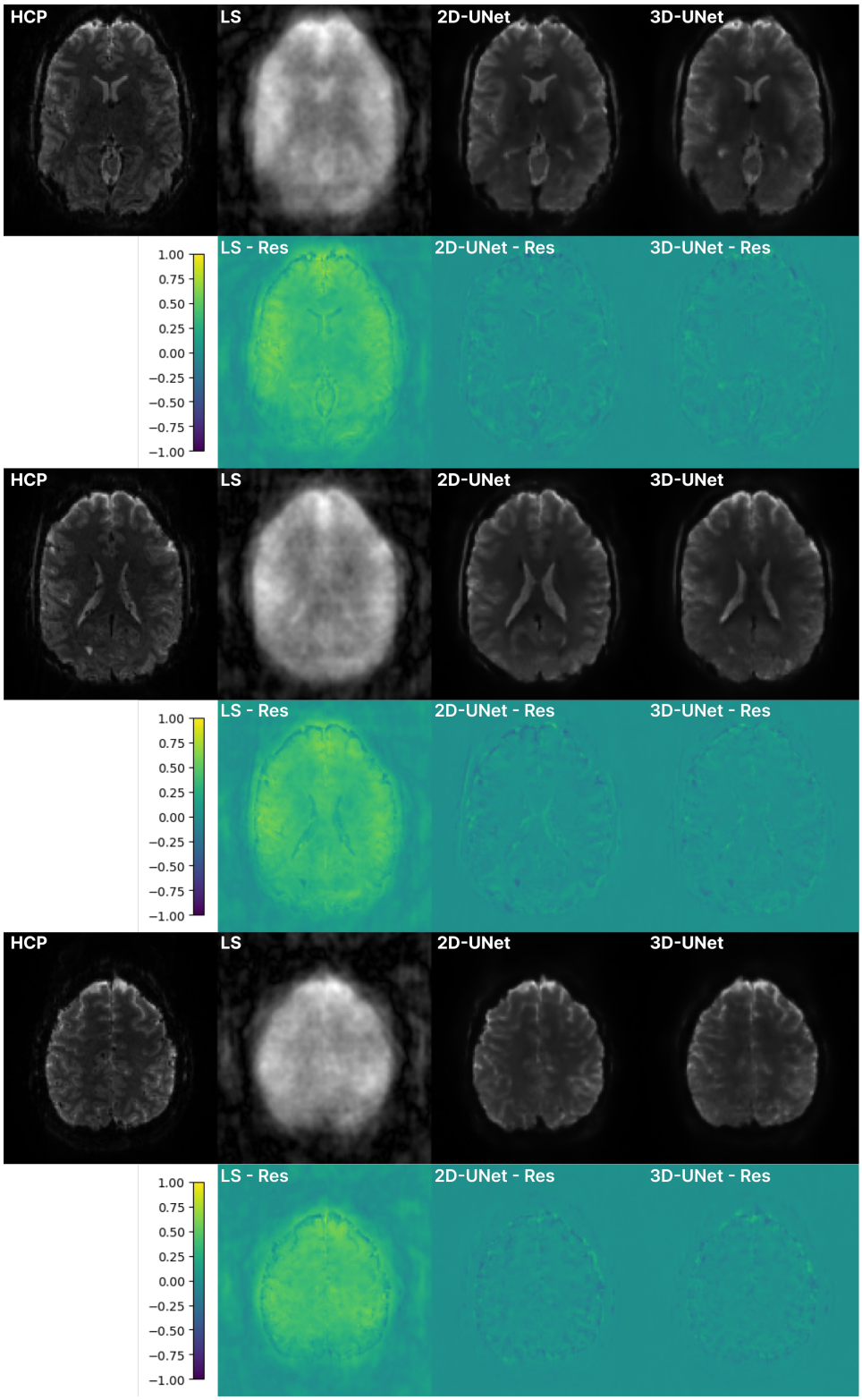

Figure 4.5: Comparison of unprocessed HCP data to LS data and UNet reconstructions. The figure shows three different slices from the first volume of one participant from the test set. HCP) Slices from ground truth HCP volume, LS) Slices from Looping Star volume, 2D-UNet) Slices from 2D-UNet reconstructed volume, 3D-UNet) Slices from 3D-UNet reconstructed volume. LS - Res) The residual between the above Looping Star volume and the ground truth HCP, 2D-UNet - Res) The residual between the above 2D-UNet reconstructed volume and the ground truth HCP, 3D-UNet - Res) The residual between the above 3D-UNet reconstructed volume and the ground truth HCP.

4.2.1 Reconstruction results on the test set

Visual inspection of the model reconstructions on the testing set show that both the 2D and 3D-UNet reduce the image intensity from the Looping Star images, and by subtracting the different reconstructions from the ground truth it is clear that both models also reduce the error (Fig 4.5).

The performance on MSE, SSIM, PSNR and SNR for each reconstruction and all participants in the testing set are presented in table 4.1. MSE, SSIM and PSNR were calculated using the ground truth HCP data as a reference. This table shows that both models manage to improve the quality of the Looping Star images on all metrics for all participants.

Table 4.1: Performance metrics on unprocessed data for 15 testing participants reconstructed with LS, 2D-UNet and 3D-UNet. ↓ indicates that a lower value is better for this metric while ↑ indicates that a higher value is better. MSE = Mean square error, SSIM = Structural similarity, PSNR = Peak signal-to-noise ratio, SNR = Signal-to-noise ratio, LS = Looping Star, HCP = Human connectome project, Std = Standard deviation.

	MSE [$\times 10^{-3}$] ↓			SSIM [$\times 10^{-2}$] ↑			PSNR [dB] ↑			SNR ↑			
Participant	LS	2D-UNet	3D-UNet	LS	2D-UNet	3D-UNet	LS	2D-UNet	3D-UNet	HCP	LS	2D-UNet	3D-UNet
1	3.090	0.076	0.062	24.3	85.7	87.9	15.1	31.2	32.1	147	12	266	207
2	2.443	0.070	0.079	24.2	85.8	87.9	16.1	31.5	31.0	253	11	192	170
3	4.164	0.074	0.053	22.1	86.4	88.2	13.8	31.3	32.8	256	16	320	264
4	3.377	0.096	0.086	25.5	84.2	86.2	14.7	30.2	30.6	271	15	338	263
5	3.431	0.088	0.066	25.3	85.1	87.0	14.7	30.6	31.8	255	13	286	234
6	3.702	0.085	0.063	23.3	84.9	87.6	14.3	30.7	32.0	220	15	262	228
7	3.012	0.068	0.057	24.1	86.6	88.5	15.2	31.7	32.4	303	14	230	252
8	3.639	0.116	0.096	24.3	82.1	84.3	14.4	29.4	30.2	152	10	204	178
9	2.917	0.083	0.073	27.5	84.7	86.3	15.4	30.8	31.3	316	14	309	273
10	3.579	0.086	0.075	23.3	83.7	85.7	14.5	30.6	31.3	199	9	176	170
11	4.181	0.094	0.075	22.3	83.7	86.3	13.8	30.3	31.3	230	11	244	182
12	4.090	0.087	0.080	22.3	85.9	86.5	13.9	30.6	30.9	303	14	313	246
13	3.343	0.098	0.087	25.9	83.7	85.4	14.8	30.1	30.6	208	16	259	237
14	3.779	0.078	0.066	22.1	86.6	88.6	14.2	31.1	31.8	233	14	286	251
15	3.727	0.093	0.075	23.8	84.3	86.5	14.3	30.3	31.3	266	11	206	199
Mean	**3.498**	**0.086**	**0.073**	**24.0**	**84.9**	**86.9**	**14.6**	**30.7**	**31.4**	**241**	**13**	**259**	**224**
Std	**0.490**	**0.012**	**0.012**	**1.6**	**1.3**	**1.2**	**0.64**	**0.60**	**0.72**	**51**	**2**	**50**	**36**

Table 4.2: Reconstruction performance on the unprocessed test set. Bold numbers indicate the best performance for each metric. MSE = Mean square error, SSIM = Structural similarity, PSNR = Peak signal-to-noise ratio, SNR = Signal-to-noise ratio, HCP = Human connectome project

Reconstruction	MSE [$\times 10^{-3}$] ↓	SSIM [$\times 10^{-2}$] ↑	PSNR [dB] ↑	SNR ↑
Ground truth HCP	-	-	-	241 ± 51
Looping Star	3.498 ± 0.490	24 ± 2	14.6 ± 0.6	13 ± 2
2D-UNet	0.086 ± 0.012	85 ± 1	30.7 ± 0.6	$\mathbf{259 \pm 50}$
3D-UNet	$\mathbf{0.073 \pm 0.012}$	$\mathbf{87 \pm 1}$	$\mathbf{31.4 \pm 0.7}$	224 ± 36

Table 4.2 is a summary view of table 4.1 with mean and standard deviation values. This table shows that the 3D-UNet performs better on MSE, SSIM and PSNR, which are all of the performance metrics that compare the reconstruction to the ground truth.

This, along with the visual inspection of figure 4.5 and 4.6 suggest that the 3D-UNet is better at reconstructing the Looping Star images. The 2D-UNet only performs better on the SNR, which is the image quality metric that did not use a ground truth image as reference.

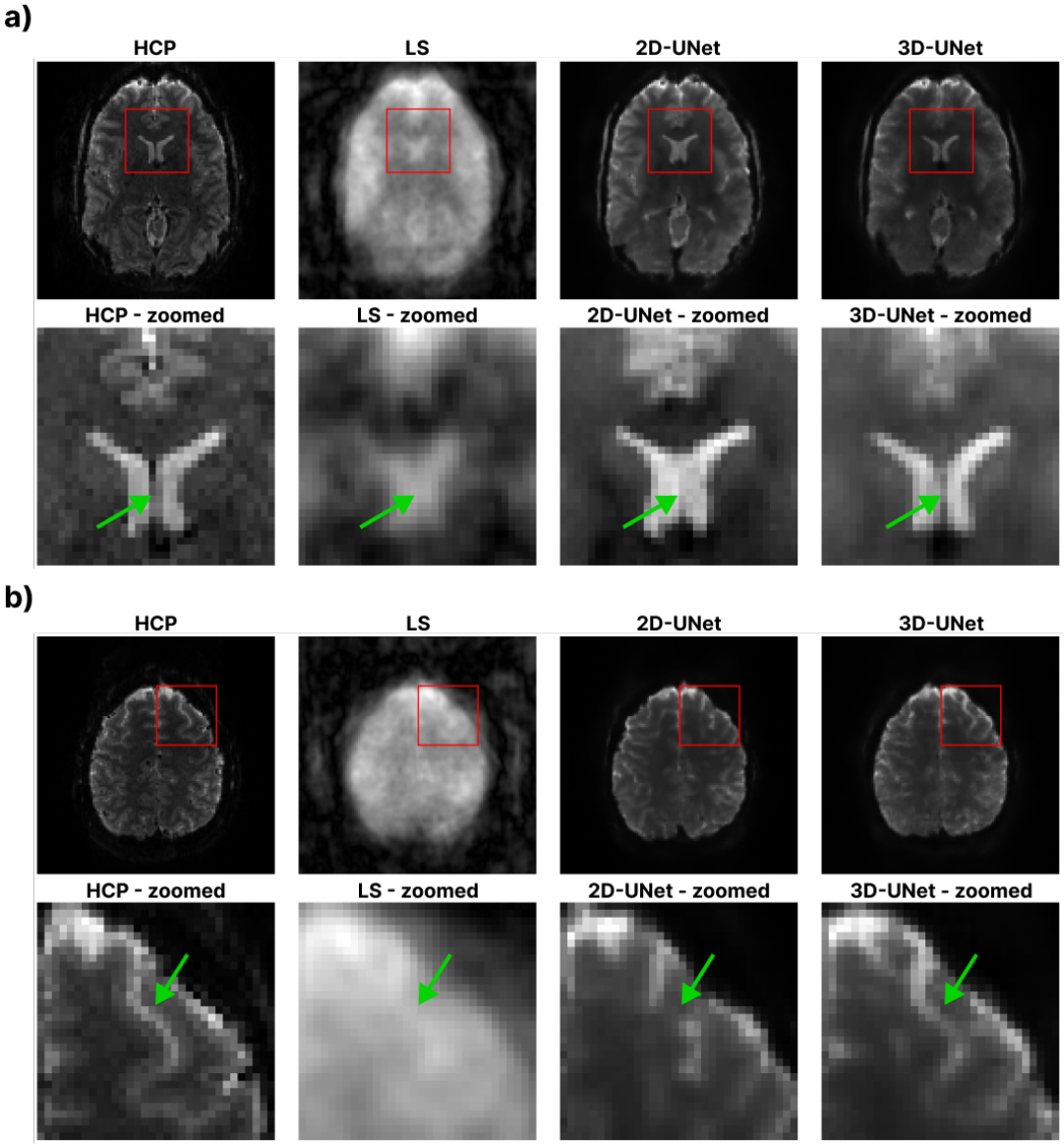

*Figure 4.6: Closer look at two examples (a and b) of image details from one example participant in the test set. The images are from the same participant as figure 4.5. Red squares display the location of the zoomed-in images 2nd and 4th rows. Green arrows highlight specific anatomical structures for a closer look. HCP) Slices from ground truth HCP volume. LS) Slices from Looping Star volume, 2D-UNet) Slices from 2D-UNet reconstructed volume, 3D-UNet) Slices from 3D-UNet reconstructed volume. * - zoomed) Zoomed in sections of the image above.*

The residual plots in figure 4.5 show that the residual from both models is largest at sharp edges, such as between air and tissue. The residual plots also show that the 2D-UNet has a more prominent error in large structures of the brain, such as the ventricles, compared to the 3D-UNet. This is also shown in figure 4.6a, where the green arrows

show that the 3D-UNet has managed to separate the two ventricles while the 2D-UNet has not. Figure 4.6b also shows that the 3D-UNet is better at recreating the folds in the brain. The green arrows point to a section where the 2D-UNet has not managed to connect the folds while the 3D-UNet has.

4.3 Network training on processed data

This section details the training and evaluation procedure for the processed dataset. The SNR result from table 4.2 as well as the theoretical risk of the neural network to adapt to the EPI artefacts and noise patterns was the motivation for the switch to training on processed data. For this, the best network on the unprocessed data was used as a pretrained network that is further optimized. The 3D-UNet was used since it performed best on the unprocessed training set.

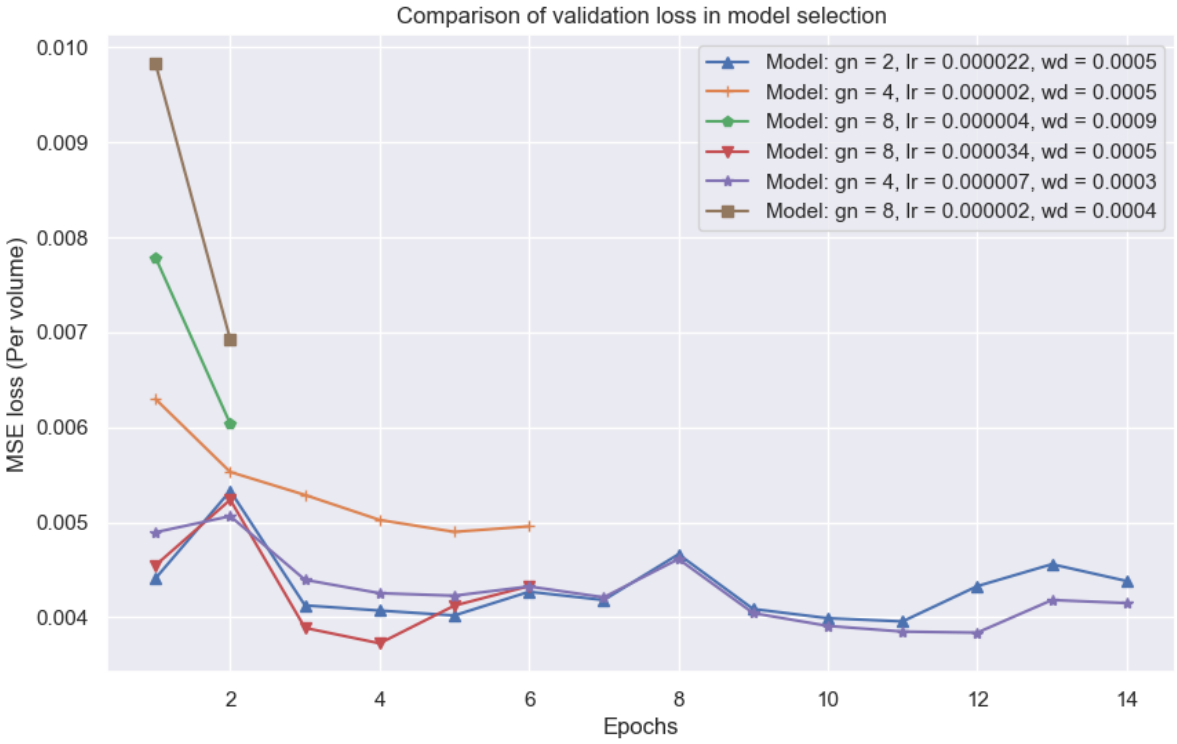

Figure 4.7: Validation loss over 14 epochs for 6 models tested in model selection. Poorly performing models on the validation set are terminated after two and six training epochs. MSE = Mean square error, gn = Group normalization, lr = Learning rate, wd = Weight decay

4.3.1 Model selection

The total model selection scheme lasted for 55 hours. Models 3 and 6 (from Tab 3.7) were dropped after two epochs and models 2 and 4 were dropped after six epochs. The models were dropped due to low performance on the validation set at the intermediate epochs (Fig 4.7). Models 1 and 5 were then evaluated after the last model selection epoch, and model 5 achieved the best validation loss after the model selection scheme. Based on this result, the best set of hyperparameters was found to be the hyperparameters used by model 5 (Tab 4.3). These hyperparameters were then used to train the final model on processed data.

Table 4.3: Hyperparameters of the best-performing model in model selection scheme

Group normalization size	4
Learning rate	6.8463e-06
Weight decay	0.0002887

4.3.2 Training 3D-UNet with best hyperparameters

Training of the optimal model for 20 epochs lasted for 99 hours. The model performed best on the validation set after epoch number 18 (Fig 4.8). This instance of the model was then saved for use in image reconstruction. Figure 4.8 show that the validation loss reaches its lowest point when the training loss has converged and stopped improving. The graphs show that the training loss improves a lot during the first epochs while the validation loss does not. This suggests some overfitting to the training data.

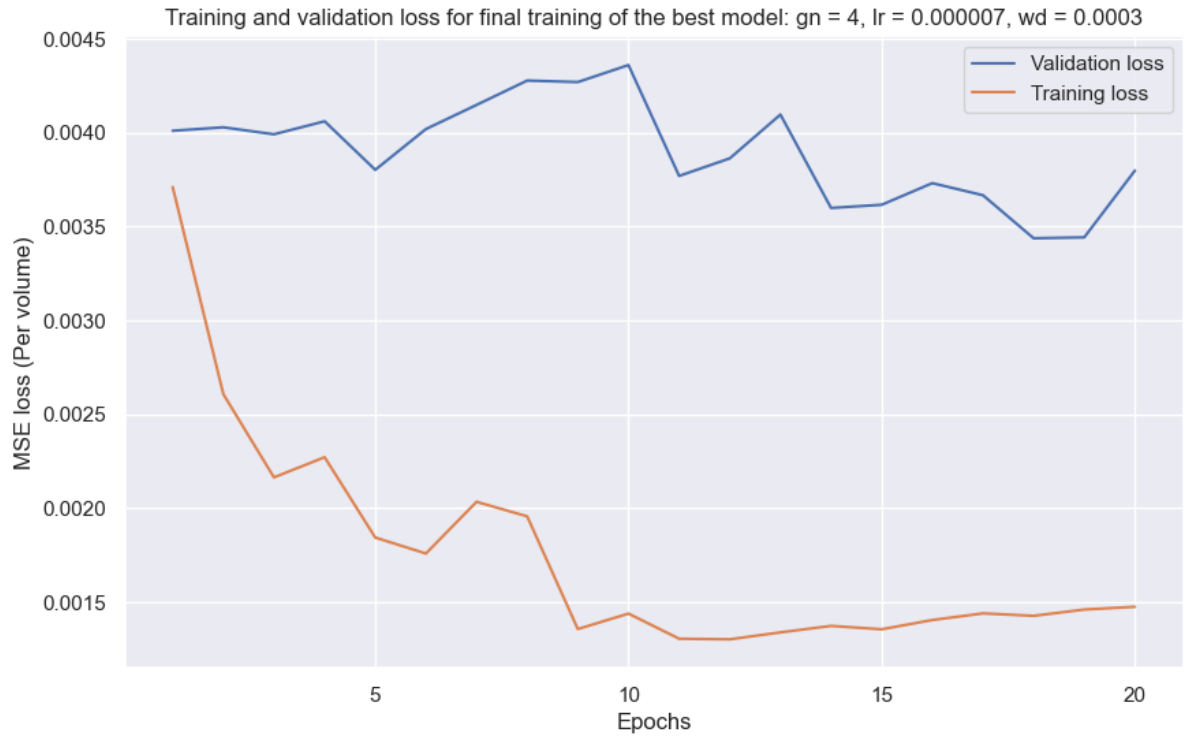

Figure 4.8: Training and validation loss of the model with the best-performing hyperparameters from the model selection. MSE = Mean square error, gn = Group normalization, lr = Learning rate, wd = Weight decay

4.3.3 Evaluation on test set

As for the unprocessed data, the model performance was evaluated using MSE, SSIM, PSNR and SNR for all participants (Tab 4.4).

Table 4.4: Performance metrics on the processed data for 15 testing participants for both LS reconstruction and 3D-UNet reconstruction. ↓ indicates that a lower value is better for this metric while ↑ indicates that a higher value is better. MSE = Mean square error, SSIM = Structural similarity, PSNR = Peak signal-to-noise ratio, SNR = Signal-to-noise ratio, LS = Looping Star, HCP = Human connectome project, Std = Standard deviation.

	MSE [$\times 10^{-3}$] ↓		SSIM [$\times 10^{-2}$] ↑		PSNR [dB] ↑		SNR ↑	
Participant	LS	3D-UNet	LS	3D-UNet	LS	3D-UNet	LS	3D-UNet
1	3.717	0.097	11.6	90.6	14.1	30.1	25	569
2	5.857	0.162	9.0	89.5	12.1	27.9	24	566
3	3.072	0.087	12.5	90.4	15.0	30.6	23	525
4	4.289	0.096	11.7	89.6	13.5	30.2	22	412
5	3.322	0.090	12.0	90.3	14.6	30.5	22	525
6	3.133	0.084	12.0	90.3	14.8	30.7	22	599
7	4.570	0.078	11.4	89.5	13.3	31.1	23	531
8	4.474	0.090	10.4	90.3	13.3	30.5	21	547
9	3.076	0.086	12.9	89.7	15.0	30.7	25	499
10	3.889	0.102	11.2	90.1	13.9	29.9	21	533
11	7.373	0.579	5.6	81.7	11.2	22.4	21	568
12	4.103	0.078	11.0	91.6	13.7	31.1	23	585
13	3.673	0.101	12.1	89.6	14.3	29.9	25	538
14	3.573	0.078	12.0	91.0	14.3	31.1	23	532
15	3.727	0.097	11.5	89.8	14.1	30.1	24	619
Mean	**4.123**	**0.127**	**11.1**	**89.6**	**13.8**	**29.8**	**23**	**543**
Std	**1.153**	**0.127**	**1.8**	**2.3**	**1.05**	**2.19**	**1**	**48**

Table 4.5: Reconstruction performance on processed test set for the LS reconstruction and the 3D-UNet reconstruction. MSE = Mean square error, SSIM = Structural similarity, PSNR = Peak signal-to-noise ratio, SNR = Signal-to-noise ratio, HCP = Human connectome project.

Reconstruction	MSE [$\times 10^{-3}$] ↓	SSIM [$\times 10^{-2}$] ↑	PSNR [dB] ↑	SNR ↑
Looping Star	4.12 ± 1.15	11 ± 2	13.8 ± 1.1	23 ± 1
3D - UNet	0.13 ± 0.13	90 ± 2	29.8 ± 2.1	543 ± 48

The summary view of the performance metrics (Tab 4.5) shows that the 3D-UNet reconstruction succeeds in reducing the reconstruction errors from the Looping Star reconstruction and performs better in every metric. The network reduces the MSE between the Looping Star images and the ground truth with **97%**.

By comparing the SNR results on unprocessed (Tab 4.2) and processed data (Tab 4.5), it is clear that the network trained on processed data reduced the image noise to a larger extent. The noise reduction is also visible in figure 4.9 where the residual plots show that the residuals outside the brain are close to zero. As for the unprocessed data, the majority of the error in the 3D-UNet reconstruction appears in the edges of large structures, such as between the brain and air and the edges of the ventricles.

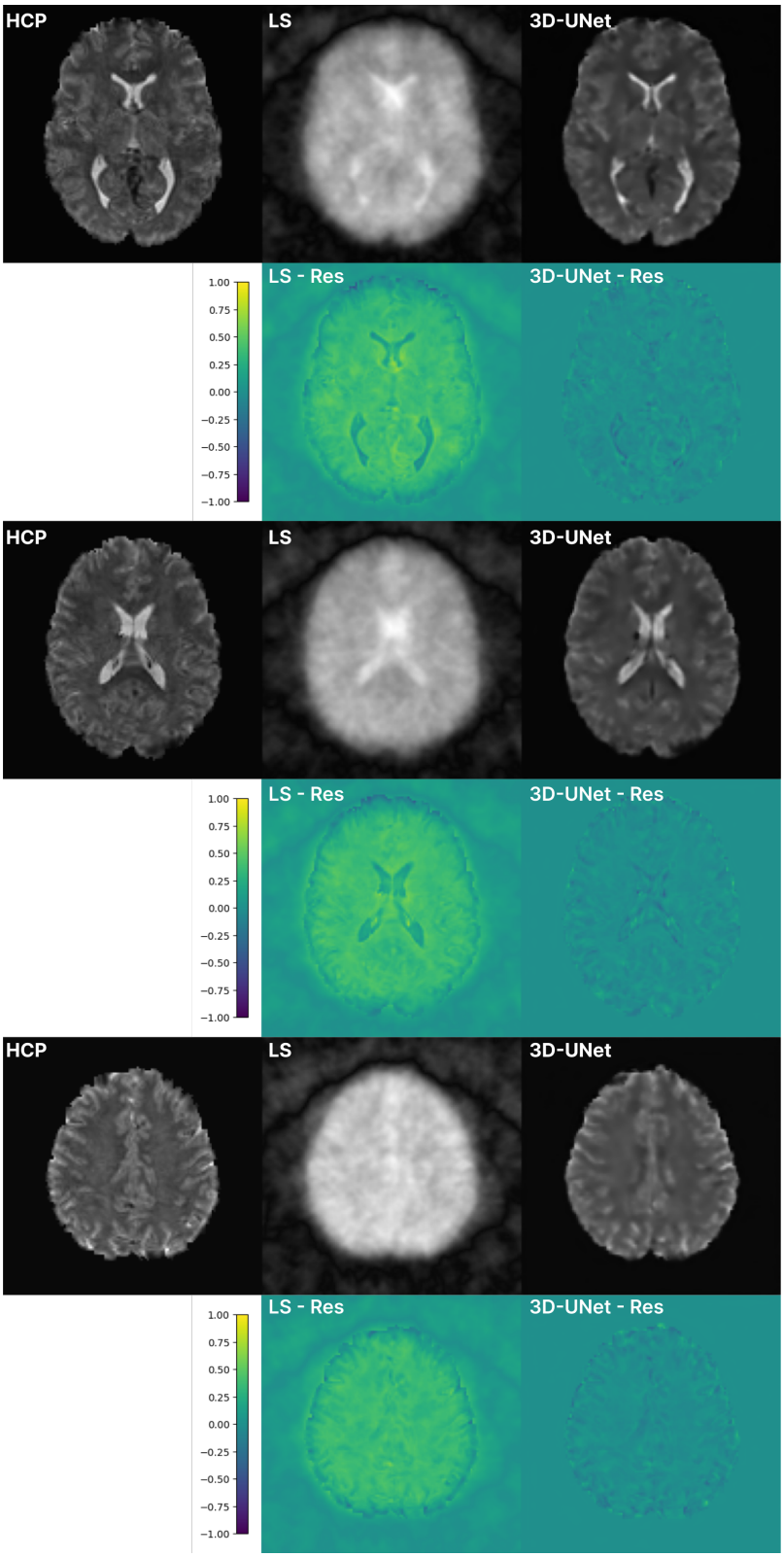

Figure 4.9: Comparison of processed HCP data to LS data and 3D-UNet reconstruction. The figure shows three different slices from the first volume of one participant from the test set. HCP) Slices from ground truth HCP volume, LS) Slices from Looping Star volume, 3D-UNet) Slices from 3D-UNet reconstructed volume. LS - Res) The residual between the above Looping Star volume and the ground truth HCP. 3D-UNet - Res) The residual between the above 3D-UNet reconstructed volume and the ground truth HCP.

The residual plots for the Looping Star reconstruction in figure 4.9 show that the large structure edges (e.g. the ventricles) are among the places where most of the information is lost between ground truth and Looping Star reconstruction. This may suggest that much of the information in these areas were blurred during the downsampling and Looping Star reconstruction process, making it difficult for the network to accurately reconstruct these areas.

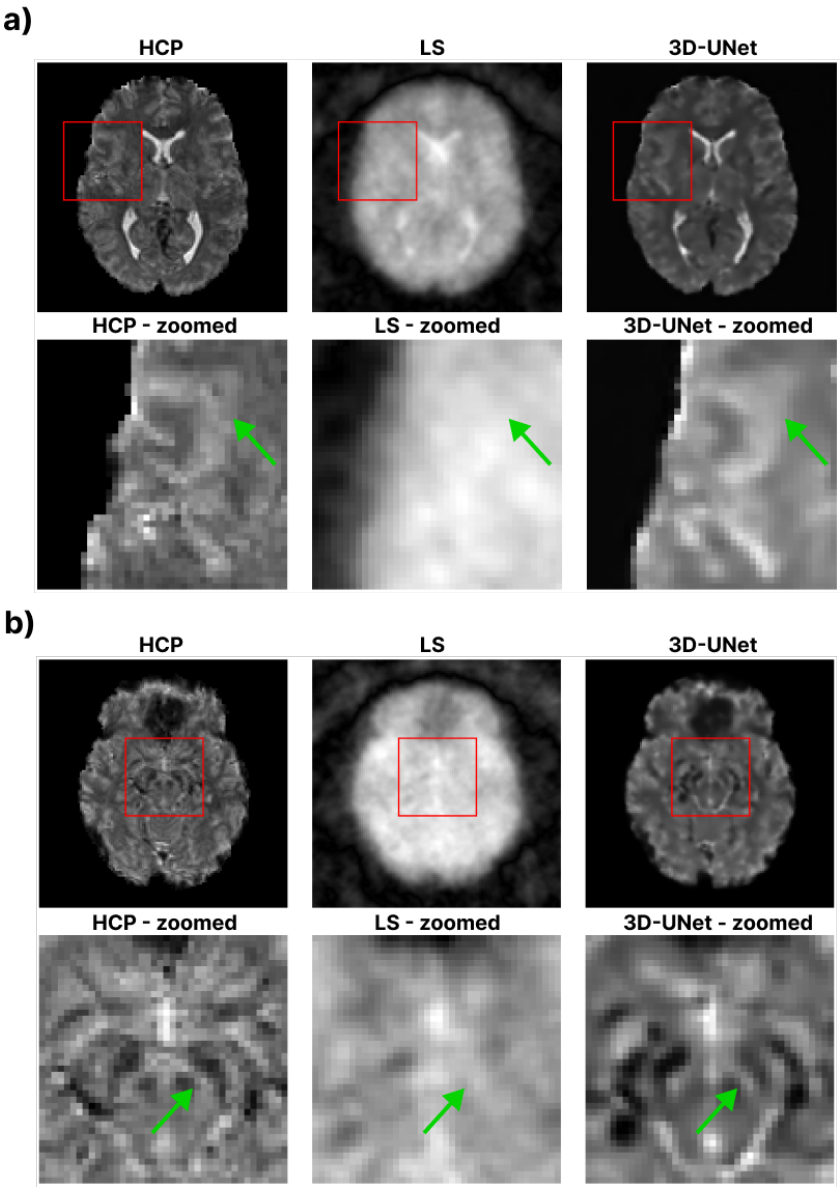

Figure 4.10: Closer look at some image details from the image comparison in figure 4.9 that are improved by the network. The images are from the same participant as figure 4.9. Red squares in the 1st and 3rd rows display the location of the zoomed-in images (2nd and 4th rows). Green arrows point to specific anatomical structures that are improved by the network. HCP) Slices from ground truth HCP volume, LS) Slices from Looping Star volume, 3D-UNet) Slices from 3D-UNet reconstructed volume.

Figure 4.10 and 4.11 shows highlighted results from one participant in the testing set. Figure 4.10 shows examples where the network accurately reconstructs detailed struc-

tures. Figure 4.11 shows examples where the network is not able to reconstruct the details, and the low-contrast details from the ground truth data are not recovered. All the examples show that the network output is more smoothed than the ground truth.

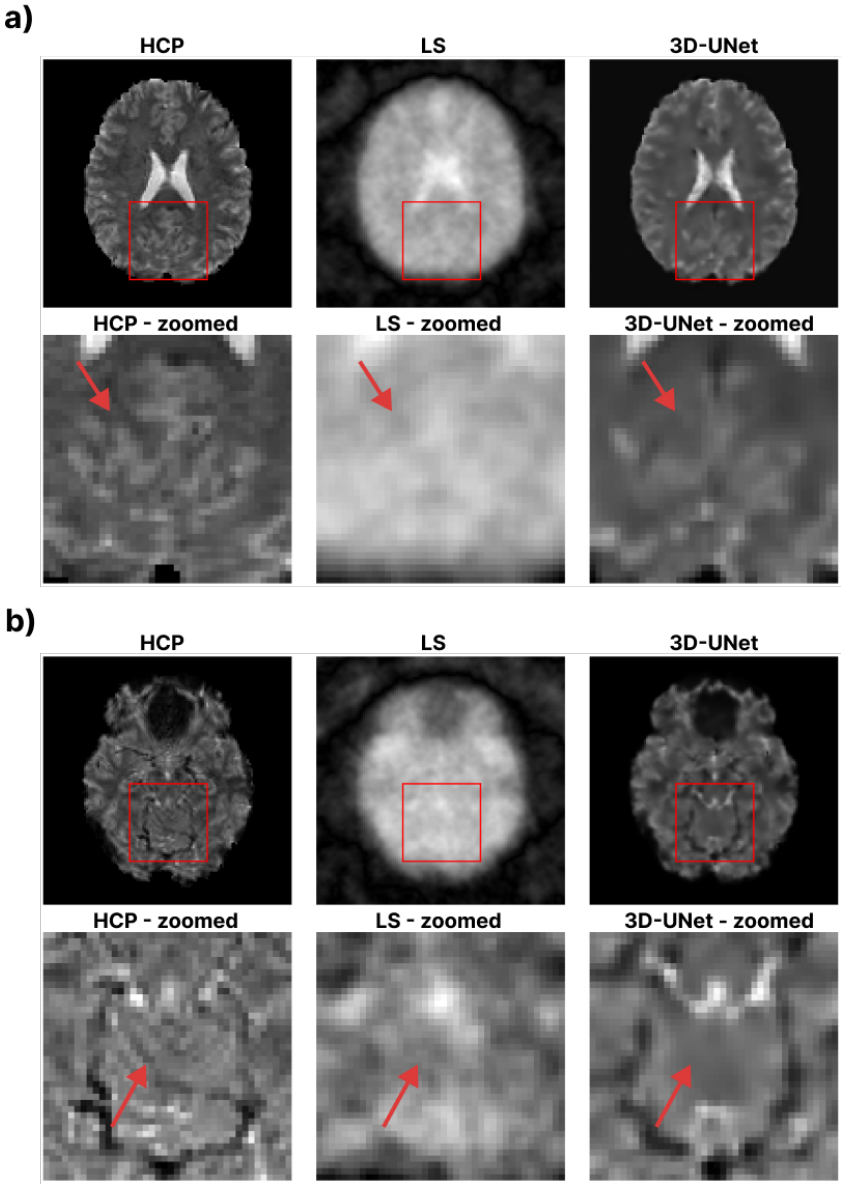

Figure 4.11: Closer look at some image details from the image comparison in figure 4.9 that are not completely recovered by the network. The images are from the same participant as figure 4.9. Red squares in the 1st and 3rd rows display the location of the zoomed-in images (2nd and 4th rows). Red arrows point to specific detail in the images that are not recovered by the network. HCP) Slices from the ground truth HCP volume, LS) Slices from the Looping Star volume, 3D-UNet) Slices from the 3D-UNet reconstructed volume.

4.4 fMRI analysis

4.4.1 BOLD variance comparison

This section contains the result of the voxel time-series correlations for the three different image reconstructions: fully sampled HCP data, downsampled Looping Star reconstruction and reconstruction by the 3D-UNet.

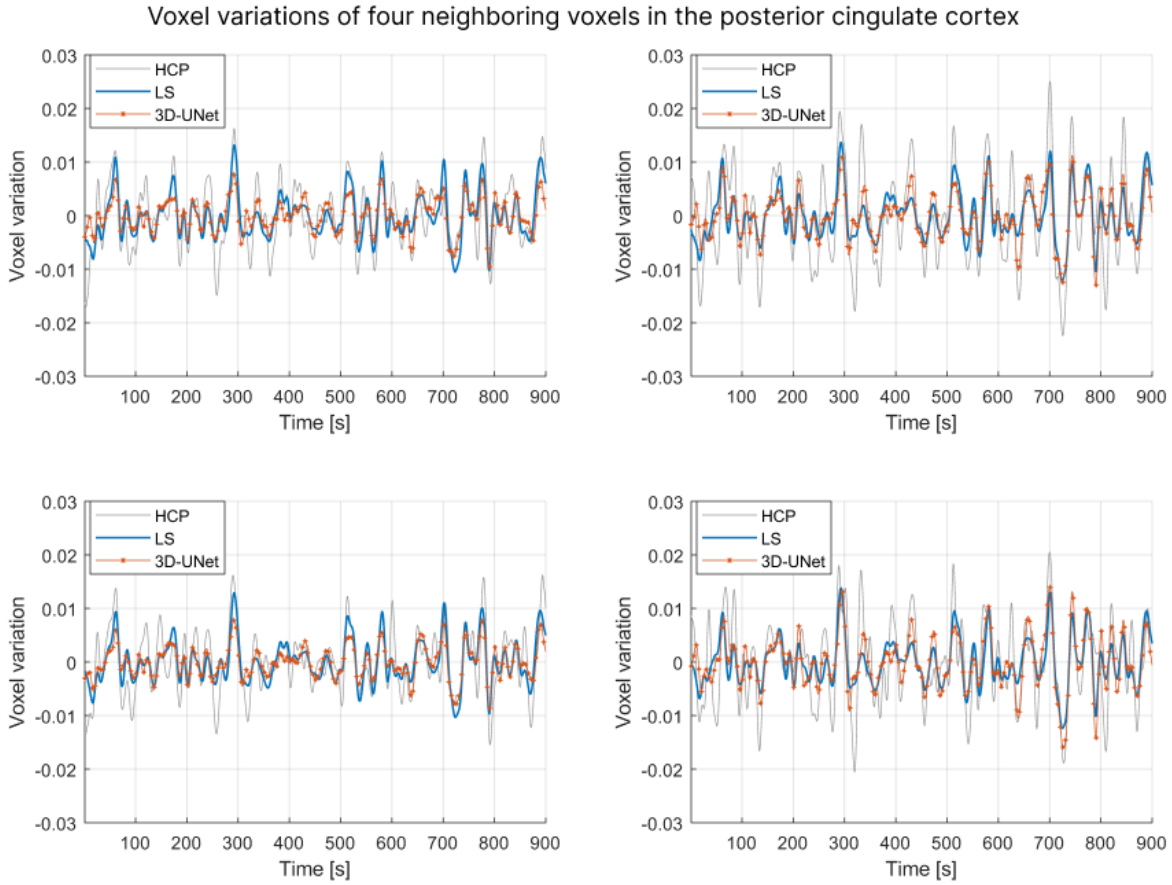

Figure 4.12: Voxel variations in four neighboring voxels located in the PCC. The visualization uses a moving average over 10 data points. HCP) Human connectome project data, LS) Looping Star data, 3D-UNet) 3D-UNet reconstructed data.

Figure 4.12 shows the time-series variation in four neighboring voxels in a region in the PCC that shows activation in all the different image reconstructions during the resting-state fMRI analysis (section 4.4.2). These graphs show that all three reconstructions have similar variation over time for all four voxels, and the graphs seem to correlate. This is also expressed in table 4.6a. Table 4.6a shows that the Looping Star and UNet reconstructions have some degree of correlation to the ground truth HCP variation, but the LS reconstruction is more strongly correlated in these four voxels. Although being less correlated in total, the UNet reconstruction does improve on some activation peaks. The UNet reconstruction manage to retain a lot of the information from the LS

reconstruction in these voxels, as is shown by the strong correlation between the UNet and LS reconstructions.

Table 4.6: Pearson correlation coefficient between voxel variation graphs for 8 voxels from two different regions in one subject from the testing set. Bold numbers indicate which reconstruction achieved the strongest correlation to the original HCP data for each example voxel. PCC = Posterior cingulate cortex, LS = Looping Star, HCP = Human connectome project

a) Figure 4.12	Pearson correlation coefficient		
PCC voxels	LS & HCP	UNet & HCP	UNet & LS
1	**0.71**	0.70	0.84
2	**0.68**	0.67	0.86
3	**0.75**	0.68	0.91
4	**0.60**	0.56	0.88
b) Figure 4.13	Pearson correlation coefficient		
Frontal lobe voxels	LS & HCP	UNet & HCP	UNet & LS
1	**0.13**	-0.02	0.14
2	**0.15**	0.11	-0.05
3	**0.47**	0.11	0.40
4	**0.38**	0.20	0.08

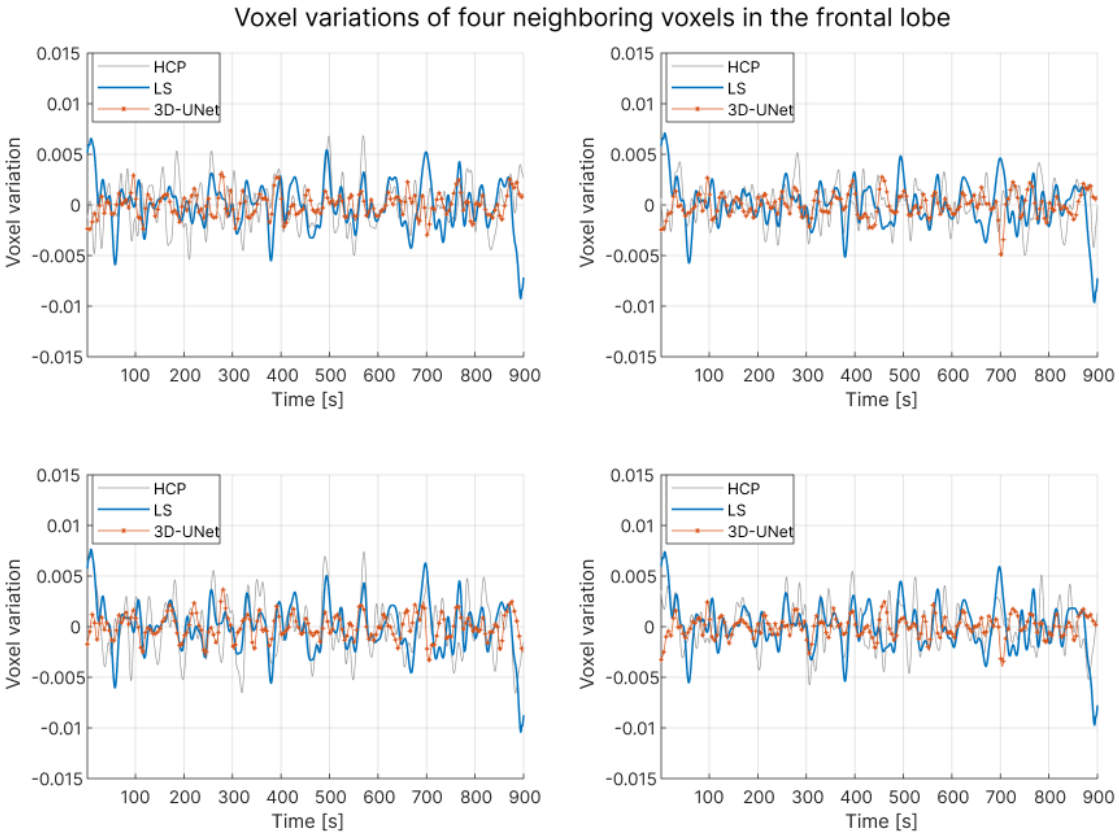

Figure 4.13: Voxel variations in four neighboring voxels located in the frontal lobe (right hemisphere). The visualization uses a moving average over 10 data points. HCP) Human connectome project data, LS) Looping Star data, 3D-UNet) 3D-UNet reconstructed data.

Figure 4.13 shows the time-series variation in four neighboring voxels in a region in the frontal lobe. Both figure 4.13 and table 4.6b demonstrate a low correlation between the UNet reconstruction and the two other time-series, and the UNet reconstruction makes no apparent improvements. Even correlations between UNet and LS are low, and the variation in the UNet reconstruction seems to be somewhat random. Low correlation between LS reconstruction and ground truth HCP may suggest that a lot of the BOLD variation is lost in the downsampling and any remaining variation is seen as noise by the neural network.

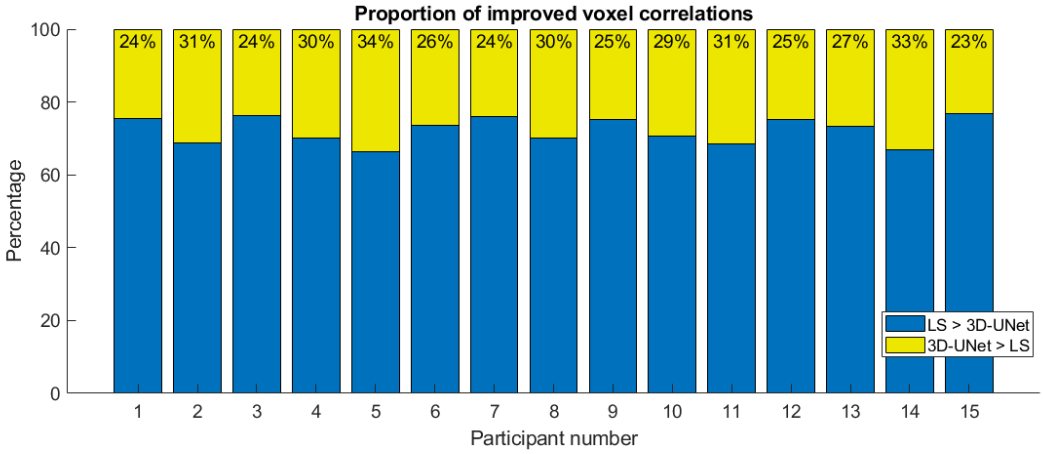

Figure 4.14: Percentage of voxel correlations improved by the 3D-UNet in the test set participants. LS = Looping Star

Figure 4.14 shows the percentage of voxel time-series correlations that were improved by the 3D-UNet. The correlations have only been calculated for the voxels that contain signal from the brain in the original HCP data. This figure shows that the proportion of voxel time-series correlations that are improved by the 3D-UNet for the test set is consistent between participants and stays between 20% and 40%. However, these values show that the 3D-UNet reconstruction fails to improve on the majority of the correlations, as most time-series get reduced correlation after the deep learning reconstruction.

Figure 4.15 shows maps of the difference in correlation coefficient for the voxel time-series in four participants from the test set. As figure 4.14, figure 4.15 also demonstrates that the Looping Star reconstruction had a stronger correlation in the majority of the voxels (blue areas in the maps). However, there are areas in the correlation maps that display a stronger correlation for the 3D-UNet reconstruction. One visible pattern is that the 3D-UNet reconstruction achieves a stronger correlation in the ventricles, as they are visible as yellow or green in the correlation maps (Fig 4.15). Other areas in the maps also show that the 3D-UNet achieved equal, and sometimes even stronger correlation to the voxel time-series of the HCP data. But apart from the improved cor-

relation in the ventricles, there are no apparent patterns in the figure on the strengths or weaknesses of the 3D-UNet reconstruction.

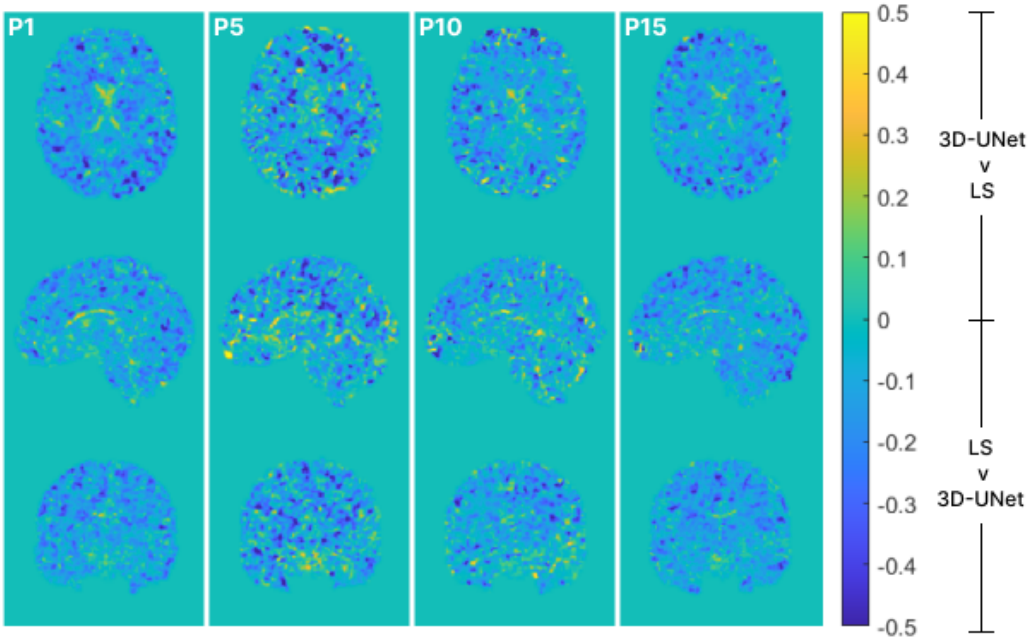

Figure 4.15: Maps showing the difference in correlation between 3D-UNet reconstruction and LS reconstruction for four test set participants. P denotes the participant number from the testing set. The maps are created by subtracting the correlation coefficient between LS time-series and original HCP time-series from the correlation coefficient between 3D-UNet reconstructed time-series and HCP time-series for every voxel. Which of the models that perform best on each individual voxel is displayed using the color range from the color bar. The bar furthest to the right display which color range indicate better performance for the 3D-UNet (3D-UNet > LS) or the LS (LS > 3D-UNet). LS = Looping Star.

4.4.2 Resting state fMRI - Single participant analysis

This section contains the result of the seed-to-voxel analysis for one example participant in the testing set. The displayed results (Fig 4.16, 4.17 and 4.18) show the brain voxels that are connected with the posterior cingulate cortex (PCC) of the default mode network (DMN) through correlated BOLD fluctuations. This is also referred to as activation.

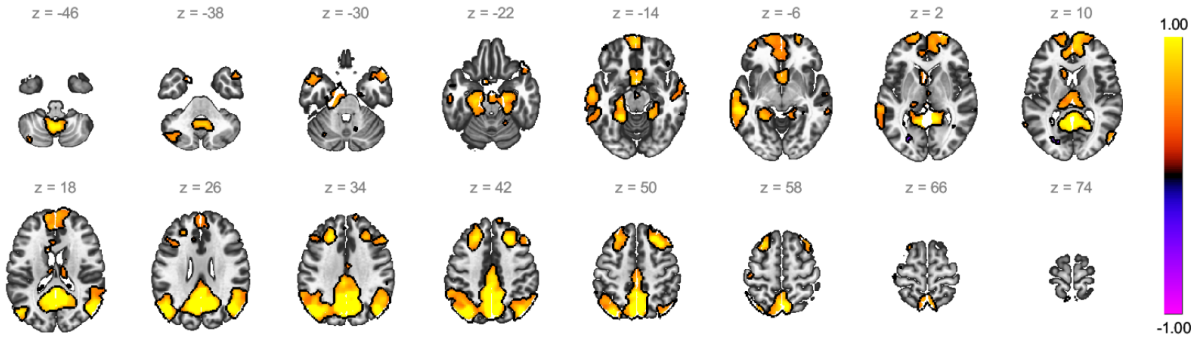

Figure 4.16: Seed to voxel analysis of the posterior cingulate cortex (PCC) for one example participant in the testing set for the original HCP data. The threshold for visualization is set to 0.4. z is the slice number from the center slice. The bright colors (yellow and orange) show areas with a positive correlation to the PCC. Dark colors (blue and purple) show areas with a negative correlation.

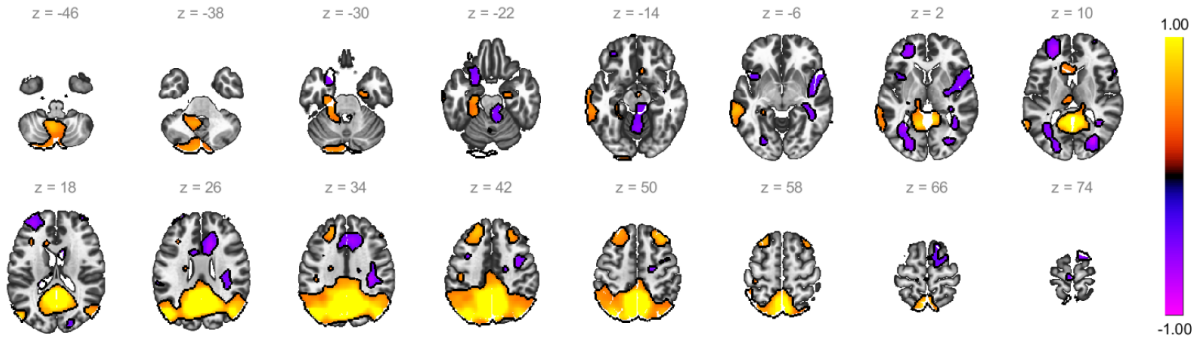

Figure 4.17: Seed to voxel analysis of the posterior cingulate cortex (PCC) for one example participant in the testing set for the Looping Star reconstruction. The threshold for visualization is set to 0.4. z is the slice number from the center slice. The bright colors (yellow and orange) show areas with a positive correlation to the PCC. Dark colors (blue and purple) show areas with a negative correlation.

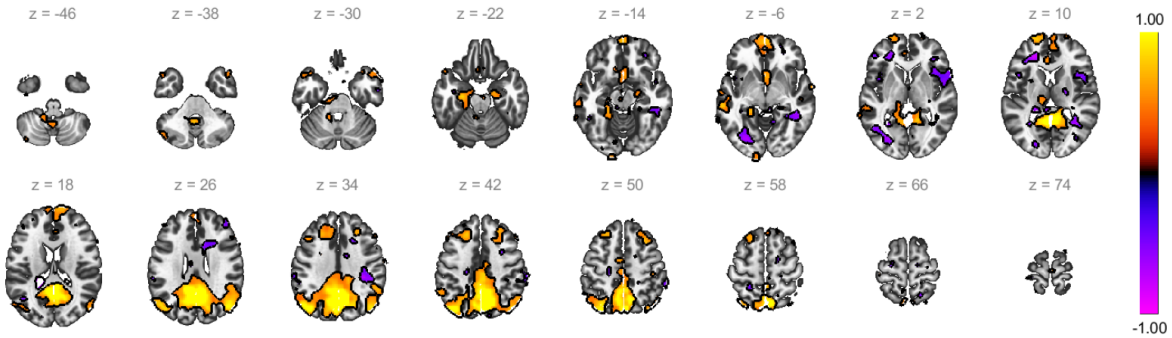

Figure 4.18: Seed to voxel analysis of the posterior cingulate cortex (PCC) for one example participant in the testing set after 3D-UNet reconstruction. The threshold for visualization is set to 0.4. z is the slice number from the center slice. The bright colors (yellow and orange) show areas with a positive correlation to the PCC. Dark colors (blue and purple) show areas with a negative correlation.

Figure 4.16, 4.17 and 4.18 show that for all three datasets (original HCP data, LS reconstruction and 3D-UNet reconstruction respectively), the largest group of correlated voxels were located in an area close to the PCC. The 3D-UNet reconstruction had fewer

correlated voxels compared to the original HCP data and the Looping Star reconstruction.

Figure 4.17 shows that the Looping Star reconstruction of the example participant had a larger area of correlated activity close to the PCC than the ground truth data. The 3D-UNet reconstruction reduced some of this activation from the Looping Star data in areas where it was not present in the ground truth HCP data. This resulted in an activation pattern from the 3D-UNet reconstruction that more closely resembled the pattern in the ground truth HCP data in many slices. However, The 3D-UNet has also suppressed some activation patterns from the Looping Star data that were accurate representations of the ground truth data. This is visible in slice z = 50 in figure 4.16, 4.17 and 4.18 where the activated areas in both hemispheres in the frontal lobe are smaller in the 3D-UNet reconstruction.

The 3D-UNet also managed to reduce some of the negative correlations introduced in the Looping Star reconstructions. These negative correlations can be seen in figure 4.17 as blue and purple spots of activation. The 3D-UNet image in figure 4.18 show fewer of these spots, and some are reduced to smaller sizes. Furthermore, the 3D-UNet has managed to retrieve some of the correlations that were lost in the Looping Star reconstruction. One example of this can be seen in the frontal lobe in slice z = 6.

4.4.3 Resting state fMRI - Group-level analysis

This section shows the result of the group-level seed-to-voxel analysis for five participants in the testing set. The displayed results (Fig 4.19, 4.20 and 4.21) show the brain regions that have BOLD fluctuations that correlate with those of the posterior cingulate cortex (PCC) of the default mode network (DMN) on a group level.

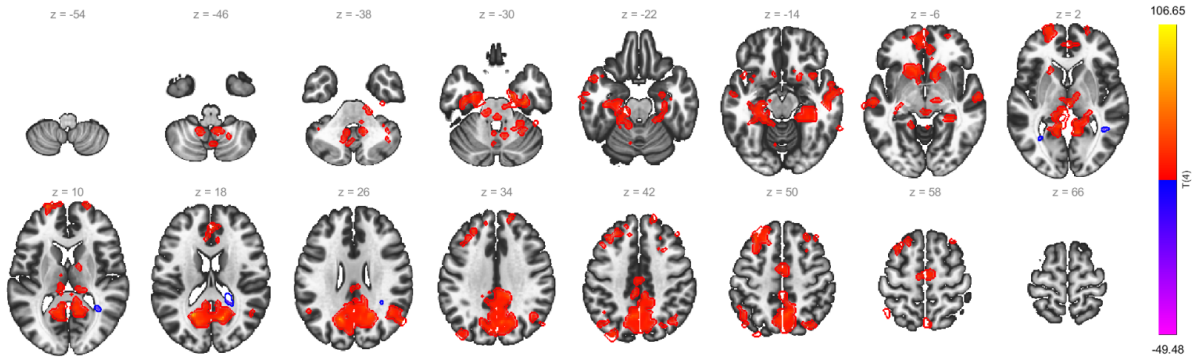

Figure 4.19: Group-level Seed to voxel analysis of the posterior cingulate cortex (PCC) for 5 participants in the testing set for the original HCP data. z is the slice number from the center slice. The bright colors (yellow and red) show areas with a positive correlation to the PCC. Dark colors (blue and purple) show areas with a negative correlation.

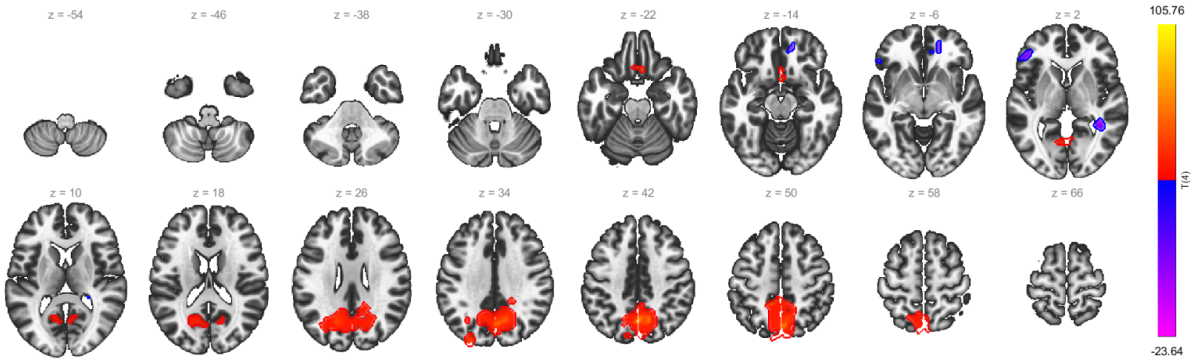

Figure 4.20: Group-level Seed to voxel analysis of the posterior cingulate cortex (PCC) for 5 participants in the testing set for the Looping Star reconstruction. z is the slice number from the center slice. The bright colors (yellow and red) show areas with a positive correlation to the PCC. Dark colors (blue and purple) show areas with a negative correlation.

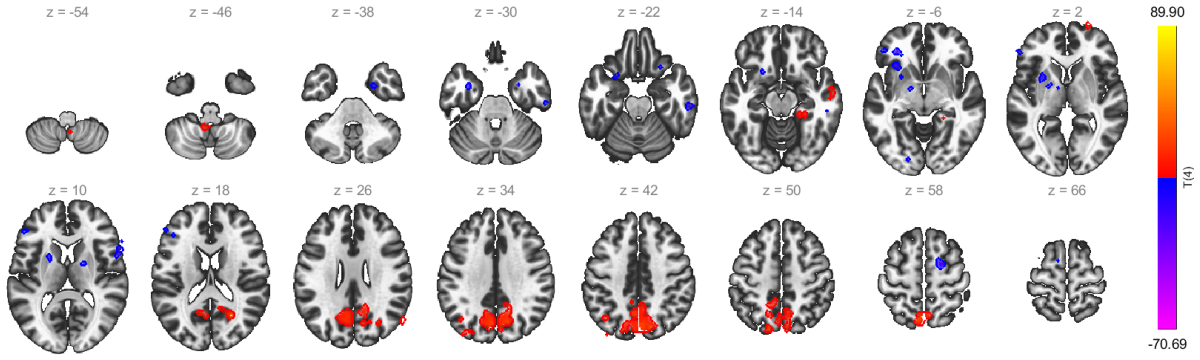

Figure 4.21: Group-level Seed to voxel analysis of the posterior cingulate cortex (PCC) for 5 participants in the testing set after 3D-UNet reconstruction. z is the slice number from the center slice. The bright colors (yellow and red) show areas with a positive correlation to the PCC. Dark colors (blue and purple) show areas with a negative correlation.

As with the single participant analysis, the 3D-UNet reconstruction show reduced activity correlation compared to the original HCP data and the LS reconstruction in the

group-level analysis (Fig 4.19, 4.20 and 4.21). In this analysis, the increased area of activation of the Looping Star (shown in figure 4.17) has been suppressed, implying that it was the result of noise in the Looping Star data. Apart from the area closest to the PCC, many areas of correlation in the HCP data (Fig 4.19) did not appear in the analysis of Looping Star data (Fig 4.20).

The group-level analysis of the 3D-UNet reconstructions revealed that these correlated regions have not been recovered by the deep learning reconstruction (Fig 4.21). The analysis of the 3D-UNet reconstruction from figure 4.21 also shows reduced activation in the area close to the PCC compared to the Looping Star reconstruction. The 3D-UNet did produce some small spots of correlated activity that corresponded with the analysis on HCP data, which the analysis on Looping Star data did not show. These were two small areas visible in slice z = -14 and one area in the frontal lobe in slice z = 2 (Fig 4.19, 4.20 and 4.21). However, the 3D-UNet has also produced spots of activation that were not present in either of the two other datasets.

Chapter 5

Discussion

The main motivation behind this master project was to explore the potential of leveraging deep learning reconstruction methods for the Looping Star sequence which have an application to silent imaging. As the Looping Star sequence facilitates fMRI acquisition, one of the objectives was also to examine what a DL reconstruction does to the small signal variations in fMRI. The work included data preparation by emulation of the Looping Star signal acquisition by the use of k-space resampling according to a Looping Star k-space trajectory. The resampling was done for 7T rs-fMRI data from the Human Connectome Project. The resampled data was then used to train deep neural networks with the goal of removing noise and artifacts from the Looping Star reconstruction. The neural networks were variations of the UNet [40] with both 2D and 3D convolutions and were trained using PyTorch [47]. The 2D and 3D networks were tested on an unprocessed dataset from HCP and the best of these were then trained on a processed dataset. The processed results were evaluated both with standard image quality metrics and through fMRI analysis.

In summary, this project developed and evaluated a neural network for Looping Star fMRI image reconstruction. The 3D-UNet outperformed the 2D-UNet in effectively reducing global artifacts and noise in the Looping Star images. Furthermore, the deep neural network demonstrated the potential to retain some BOLD signal variations, indicating promise for deep learning reconstruction in fMRI analysis. In this chapter, the results of the work in this project are discussed. The section will also discuss the limitations and potential errors of the proposed methods and provide some comments on future work.

5.1 Evaluation of the methodology

5.1.1 Data and dataset preparation

7T resting-state fMRI data from the human connectome project was used for this project. One of the motivations behind the project was to increase the usability of the Looping Star sequence in fMRI, e.g. in resting state examinations of schizophrenic patients without the presence of acoustic noise. This motivation was the reason why deep learning reconstruction was attempted for fMRI data.

7T data provides a lot of information for the networks to train on, and this was the major reason for choosing the 7T data for this project. By training on 7T data, the network learns from the best data available from the HCP. This would allow the networks to learn from the image detail present in the data. The HCP also has data from 3T experiments. Including 3T data in the network training pipeline could be favorable to training on one dataset only as the two datasets would increase data diversity, leading to more generalizable networks. Many open datasets exist, and many dataset combinations could be attempted during training to improve network performance. Training networks on many combinations of large datasets require a lot of storage space and computing power and is outside the scope of this project.

Unprocessed vs Processed data

The unprocessed data from the Human Connectome Project consist of images taken using the EPI sequence without any processing. The EPI sequence introduces certain artifacts like Nyquist ghosts, chemical shifts and image distortions [59]. Many of these artifacts are created as a result of the heavy use of gradients through e.g. eddy currents, and would not be present in sequences like Looping Star. These artifacts are removed in processing pipelines like the one used by the HCP [44]. The main motivation for training on processed data was to avoid learning EPI artifacts that could later be added to images acquired with different sequences. The switch from unprocessed to processed data was made during the work in this project, but the results on unprocessed data have still been included as they provide interesting results that have affected the choices of methodology on processed data. The major finding that affected the work on processed data was the discovery of the 3D-UNet as the better model for this reconstruction case.

One potential issue with the processed dataset is that the image-processing pipeline has removed some of the heterogeneity of the processed dataset. The spatial distortions in

the unprocessed dataset provide more data diversity. One result of the processing is therefore a more homogeneous dataset. Despite this, it was decided that the processed dataset is more representative of a Looping Star acquisition, which was the most important criterion for this project. The inclusion of data augmentation also helps mitigate the homogeneity of the processed dataset.

Image downsampling and Looping Star reconstruction

As mentioned above, the most important aspect of the choice of datasets was how representative the data was of real Looping Star data. This was also the case when it came to the downsampling of the HCP data and reconstruction of the downsampled k-space samples. Reconstructing the downsampled k-space using small matrix sizes resulted in image artifacts that affected the brain by shading large areas (Fig 4.2 and 4.3). These reconstruction artifacts are not typical of a Looping Star reconstruction and are probably a result of the downsampling and reconstruction process. Therefore it was decided that the best course of action was to attempt to decrease the impact of the artifact as much as possible. Different reconstruction parameters in MIRT [45] was attempted, but ultimately the best effect was achieved by increasing the image matrix by zero padding to push the artifacts outside of the brain. The zero padding was not visible by the networks as the images were cropped to smaller sizes before training.

Due to the low number of k-space samples acquired by the Looping Star sequence, the image quality of the reconstructed images is quite poor. The amount of image detail that is visible in the images is lower than in most other deep learning reconstruction studies, as these focus on slice-by-slice subsampling. This limitation on the image quality will also put limitations on the performance of the neural networks. For this project, it was decided that minimal image processing after the Looping Star reconstruction would be performed to make the pipeline from k-space data to deep learning reconstruction as short as possible. Future work with this methodology should explore different reconstruction techniques for 3D non-Cartesian k-space with the goal of improving the image quality of input data. There are few established tools for reconstructing 3D non-Cartesian MRI, and only MIRT [45] was used in this project. The RIESLING toolbox [60] is also available for 3D non-Cartesian MRI and includes NUFFT by gridding which is a good alternative to the conjugate phase reconstruction done in MIRT. Gridding to a Cartesian k-space could also enable other deep learning reconstruction techniques such as deep learning k-space interpolation [61].

5.1.2 Network architectures

The main objective of this project was the implementation of a deep neural network for the Looping Star sequence and a following examination of its usability. Studies have shown that deep neural networks, specifically convolutional neural networks built on the UNet have performed well in the reconstruction of accelerated MRI with both radial and Cartesian subsampling [10][62][63], as well as for the removal of streaking artifacts in CT [64]. The encoder-decoder structure of the UNet makes it good for extracting large features in the input images. As the reconstruction artifacts from undersampled k-space affect the entire image, it becomes more effective to use an encoder-decoder structure instead of back-to-back convolutions without pooling and upsampling.

One common approach when training neural networks, and specifically CNNs for undersampled data is to use 2D networks that consider each slice separately. This has been shown to work well for the case where each slice is subsampled separately. E.g. through subsampling each slice radially such that the full k-space becomes a cylinder of radially undersampled disks, one per slice. As figure 3.1 shows, this is not the case for the Looping Star sequence. In the Looping Star k-space, all the radial disks are rotated in k-space so that the edge of k-space form a sphere rather than a cylinder. This will result in reconstruction artifacts that, unlike the artifacts for a stack of radial disks, will affect the entire volume. For cylinder-shaped k-space, the k-space trajectory for each slice is uniform, making the reconstruction artifacts similar for each slice. This was the motivation behind attempting a 3D-UNet as well as a 2D-UNet for this task. The 3D-UNet was a modified version of a network used for image segmentation. Not much research has been done on the use of 3D models for image reconstruction. Some studies have been performed with the aim of scan acceleration from Cartesian subsampling using 3D models [65][66]. Few studies look at 3D models for non-Cartesian subsampling [61] and, to the best of available knowledge, no studies have been done with the use of 3D models for whole-volume non-Cartesian sampling.

As mentioned in section 3.3.2, the 2D-UNet architecture by Milesial [50] was chosen for its similarity to the one used by Han et al. [10]. As there are few articles detailing a similar usage of a 3D-UNet for reconstruction tasks, the 3D network architecture by Wolny et al. [51] was kept in its default state. This meant that the trained networks had certain differences such as operation order, depth and number of feature maps per encoder/decoder block. More extensive hyperparameter optimization can be run to determine which configurations of the 3D-UNet perform best for an image reconstruction case.

Other image reconstruction tasks often use a residual path, that involves adding the network input to the output [10][61][67]. This results in a network that only learns the residuals between input and ground truth. This was not done for this project. The reason for this is that the result of the non-Cartesian reconstruction of the Looping Star data by conjugate phase reconstruction is very different from the fully sampled HCP images. As a result, the residuals are large compared to those from the cited studies.

5.1.3 Network training

Unprocessed data

For training of the 2D-UNet, network architecture and hyperparameters were inspired by Han et al. [10]. The initial idea for the project was to use the same hyperparameters for the 3D-UNet for comparability. After a test run, it was clear that the 3D-UNet did not converge over the 20 epochs with a learning rate of 1×10^{-3}. The learning rate was therefore decreased for the 3D-UNet on a second run which yielded better convergence on the validation set and better reconstruction results. The new learning rate of 5×10^{-5} was chosen as it was in the mid-range of the learning rate schedule used by Han et al. The loss during training (Fig 4.4) showed that both models converged on validation loss during training, but more extensive hyperparameter optimization could be performed, as is done for the processed data. The Adam optimizer was used instead of SGD as it performs individual learning rate optimization for the parameters.

The initial idea for data loading during training was to load volumes and slices randomly from all the training set participants. This turned out to be impractical for training the 2D-UNet as the image data for each participant would need to be loaded each time a single image slice was randomly chosen by the data loader. This meant that every participant's data would be loaded 450 times per epoch as each subject had 50 volumes with 90 slices in each volume. As the loading time for one participant's data was not insignificant this would drastically increase training time. Therefore it was decided that the better approach was to load every participant's data once per epoch and train on batches of several random participants. This was also done when training the 3D-UNet for similarity. The data augmentation was performed after the participant data was loaded into computer memory. This meant that the training was performed on a subset of participants whose data was randomly loaded to the network, including augmented data. This was performed to avoid issues when splitting the dataset into training and validation datasets, as adding augmentations before loading would not split

the data by participant only, but by participant and participant augmentations. The two datasets would then most likely contain data from the same participants, only with augmentations.

Similar deep learning studies for image tasks usually train for more than 20 epochs, and often the number of epochs is in the hundreds [10][61]. The number of training epochs was chosen with the large amount of training data in mind. With 50 volumes per participant, every participant's data was seen by the network 1000 times over the 20 epochs. The graphs in Figure 4.4 give a reduced view of the training process. The majority of the loss reduction happens in the first epoch, and saving the loss per batch could give a better view of the training process. Both the training graphs for the 2D-UNet and the 3D-UNet show some degree of overfitting as the training loss improves where the validation loss does not. By saving the instances of the models that perform best on validation data, the degree of overfitting is as low as possible.

The number of volumes to downsample for every participant in the training set was a topic of discussion before training. 50 volumes were chosen as a good compromise between voxel variation and the number of epochs. Training on all 900 volumes per participant would provide more variation, but it would make for impractical training. With the batch loading process mentioned above, training on all 900 volumes would reduce the batch size and the network would then train on only one or two participants at a time. Passing 900 volumes from one participant to the network at a time would likely result in overfitting. The number of epochs would also need to be reduced to avoid training for weeks. The goal of training on more downsampled volumes per participant was to let the network see similar volumes with different voxel variations that would be the result of a varying BOLD signal. 50 consecutive volumes were chosen to try to limit the volume variations to mainly voxel variations, and not variations as a result of head motion. Choosing 50 evenly spaced volumes would give 18 seconds between each volume, increasing the probability of motion between the volumes.

Processed data

The data loading process was kept unchanged when moving from training on unprocessed to processed data. As the processed network would be the final network, a more extensive hyperparameter optimization was run to avoid sacrificing model performance due to poor hyperparameters. The hyperparameter ranges for learning rate and weight decay was chosen based on other deep learning reconstruction studies [10][52][61]. The choices of group normalization rate were set between the default network value of

8 and lower. The BOHB search algorithm was chosen as it was suggested by the Ray Tune library for problems with large models and a small number of hyperparameters [48]. BOHB would give good configurations after only a few trials and included early stopping that would reduce the training time [48]. The searching process only used five volumes per participant in an effort to reduce the extensive training time that would be necessary to train all models on 50 volumes per participant. The best model was chosen as the one that improved its validation score fastest.

5.1.4 fMRI analysis

For the fMRI analysis, it is worth discussing to what degree the downsampled Looping Star data can accurately simulate a Looping Star acquisition. The temporal resolution of the data from the HCP is one second per volume, while the temporal resolution that has been achieved for Looping Star is just under three seconds [6].

Several approaches were considered to make the fMRI analysis comparison as accurate as possible. Two of the proposed approaches attempt to artificially reduce the temporal resolution of the Looping Star data by reducing the number of volumes in the timeseries. One approach would be to only consider every third volume in order to simulate a temporal resolution of approximately three seconds. The second approach would combine three consecutive volumes by taking the mean value for each voxel across the three volumes. Both of these approaches would reduce the number of volumes and artificially reduce the temporal resolution. However, as the volume repetition time is limited by the amount of time it takes to acquire the signal for one volume, the second approach could be considered more realistic as it would combine the signal from the three consecutive time points.

Other studies have looked at the differences between Looping Star and EPI in fMRI acquisition [1][2] and therefore this was not of importance in this project. The main purpose of the fMRI analysis in this project was to compare the Looping Star reconstruction to the deep learning reconstruction. As the deep learning reconstruction is applied after the reconstruction of the Looping Star images, the most important requirement in the comparison of the two reconstructions is that they should have an equal temporal resolution. Therefore it was ultimately decided that the time series of the two reconstructions would not be changed to fit better into the actual temporal resolution of the Looping Star sequence. However, it should be noted that this can result in a Looping Star fMRI analysis that performs better in comparison to the ground truth data.

5.2 Result evaluation

The main objective of this project was the implementation of a deep neural network for the Looping Star sequence and a following examination of its usability. This is first explored in Section 4.2. In this section, a 2D-UNet that reconstructs each slice in the Looping Star volumes separately is compared to a 3D-UNet that reconstructs the entire volume through 3D convolutions. In section 4.3, this is expanded upon by training a 3D-UNet for processed data which is evaluated in an fMRI setting in section 4.4.

5.2.1 Network training on unprocessed data

Figure 4.4 displays the training progress for both the 2D-UNet and the 3D-UNet models trained on unprocessed data. Notably, the 3D-UNet had lower validation loss than the 2D-UNet throughout the entire training process. The 2D-UNet had a lower training loss for all 20 epochs. An interesting observation is the substantial difference between the training and validation loss of the 2D-UNet, suggesting a higher level of overfitting to the training data when compared to the 3D-UNet, which displayed a smaller difference. This may suggest that the 2D-UNet has learned the expected appearance of a brain instead of learning noise or artifact patterns.

Additionally, it is worth noting the large difference in training times between the two models. The training time for the 3D-UNet was more than double the training time of the 2D-UNet. Both networks were trained on the same dataset for an equal number of epochs, indicating that the increased training time is likely attributed to the higher computational load during backpropagation. Surprisingly, the 3D-UNet achieved its lowest validation loss after only 7 epochs, despite training for an additional 13 epochs thereafter. Different hyperparameter configurations could yield further improvement during the last epochs, leading to even better model performance. This consideration holds true for both models, and future work could involve an even more extensive model selection scheme to enhance the overall outcomes.

5.2.2 2D-UNet and 3D-UNet evaluation on unprocessed data

The findings presented in section 4.2 demonstrate that both the 2D-UNet and the 3D-UNet managed to reduce the error and increase the similarity with the ground truth (Tab 4.2). Notably, in this image reconstruction task, the 3D-UNet outperformed the 2D-UNet across all metrics that compared the network output to the ground truth. Surprisingly, the 3D-UNet achieved superior performance despite having nearly half the

number of trainable parameters compared to the 2D-UNet (Tab 3.4 and Tab 3.5). These results suggest that leveraging information from neighboring slices and features is more effective than extracting more features per slice when employing deep learning for Looping Star image reconstruction. Furthermore, the utilization of 3D convolutions in the network allowed for better learning and identification of global artifacts and noise patterns that varied among image slices. One conclusion that can be drawn from this is that the shape of the k-space trajectory and the artifacts it produces should be considered when deciding on a network architecture.

Visual inspection of the image reconstructions affirms the result from the image quality metrics. The 3D-UNet was superior in accurately reconstructing anatomical structures and effectively assembling signals into cohesive structures (Fig 4.6). Both networks managed to reduce the overall noise in the images. Despite doing a good job of removing much of the error in the images, none of the networks were perfect. Figure 4.5 demonstrate that both networks struggled the most with the edges between tissues with different intensities. The figure shows that areas like the ventricles and folds have large errors in the DL reconstructions. However, as before, the 3D-UNet was better at reducing the error in large structures. This is visible in figure 4.5 where the error in the ventricles is less visible in the residual plots from the 3D-UNet reconstruction. Furthermore, a lot of the fine detail from the ground truth data was lost after k-space downsampling and has not been recovered by the networks. The result of this is that the images reconstructed by the networks have been smoothed. Both networks also struggled to reconstruct low-contrast differences in the images, as much of this has been conceived as noise and removed. These issues may be attributed to the poor quality of the Looping Star images that make up the training set.

The 2D-UNet only performed better than the 3D-UNet on the SNR, which does not compare the images to the ground truth data. On this metric, the 2D-UNet even performed better than the ground truth data. The 3D-UNet had a performance that was closer to the value of the ground truth. This may suggest that the 3D-UNet network was more adapted to the noise in the image than the 2D-UNet. This was one of the motivations for also training a 3D-UNet for processed data. A network trained on processed data will not have learned EPI artifact patterns and the danger of adding artifacts to the images is mitigated.

5.2.3 Processed UNet evaluation

The 3D-UNet trained on the processed dataset performed similarly to the network trained on unprocessed data. The major difference was the signal-to-noise ratio, where the network trained on processed data performed significantly better (Tab 4.5). This was likely due to the denoising that is performed during the processing of the dataset, as the data used as ground truth during training has practically no noise in the areas outside the brain. Thus the high SNR could be somewhat artificial as the real SNR of the input data is not known, and the denoising performed during the dataset processing may have more of an effect outside the brain.

Many of the same notes on the image quality of the deep learning reconstructions can be made for the network trained on processed data as those trained on unprocessed. Overall image quality was increased (Tab 4.5) and quite a lot of image detail were recovered (Fig 4.9 and Fig 4.10). The residual plots in figure 4.9 demonstrate that the largest reconstruction error was located in high-contrast areas, such as the edges around the ventricles. The highlighted areas in figure 4.10 show that a surprising amount of detail has been recovered. These details are not easily visible in the Looping Star images, and it is difficult to say if these details are learned anatomical structures or if they stem from image contrast that is not visible. The UNet output also appears smoothed compared to the ground truth and some fine detail has been lost (Fig 4.11), as for the unprocessed data.

5.2.4 fMRI evaluation

Very few studies have looked at the effect of deep learning reconstruction on the BOLD variation in fMRI data. Keyword searching in PubMed on the keywords "deep learning fMRI reconstruction" only reveals one study that has looked at using self-supervised learning for accelerated fMRI [68]. PubMed is the largest database of medical research publications. The BOLD signal in fMRI is very small, even <1% in some studies [18]. Therefore, one major worry with using neural networks for an image-to-image reconstruction task was that the networks would remove any voxel variation between the volumes. The mean square error between the original HCP data and Looping Star reconstruction was large (Tab 4.5), making the task of reducing unwanted noise, while still preserving the BOLD variations, challenging. Despite this, the results in section 4.4 show that some BOLD signal has been preserved and even improved in some cases.

Table 4.14 shows that not much of the lost voxel variance information for individual

voxels was recovered by the deep neural network and that the Looping Star reconstruction had a stronger correlation on the majority of the voxel time-series. Figure 4.14 shows that the performance across the testing set was stable, with few outliers in terms of the number of improved voxel variance. Figure 4.15 shows that a lot of the improved correlation was located in the ventricles of the test set participants. The ventricles consist of CSF [67] and contribute nothing to the BOLD signal. Increased correlation in this area can be a result of denoising performed by the 3D-UNet, which shows that the network managed to reduce the noise that was introduced by the downsampling and Looping Star reconstruction. No other clear performance patterns can be observed in figure 4.15, which suggests that the network contained no significant bias toward certain anatomical structures.

The fMRI analysis of a seed in the default mode network (posterior cingulate cortex) shows that the neural networks preserved enough of the BOLD variation to still show activation in a single participant analysis (Fig 4.18). One interesting observation is that for the Looping Star, the activated area close to the PCC shows correlated activity in more voxels than the ground truth data (Fig 4.16 and 4.17). This is likely attributed to an effect of the Looping Star reconstruction as the areas did not show a correlation in the fully sampled ground truth data. As mentioned in section 4.1, the signal from high-intensity voxels in the Looping Star images appears to bleed into neighboring voxels. This may be the cause of the increased activation. The 3D-UNet managed to suppress this effect, which is visible in the single participant analysis of the 3D-UNet reconstruction in figure 4.18. It is difficult to say if this is attributed to the learned patterns in the network or if it has resulted from the general smoothing of the 3D-UNet reconstructed images. Either way, it can be considered a perk that the network manages to reduce unwanted noise in the time-series.

Figure 4.16, 4.17 and 4.18 show that the 3D-UNet reconstruction produced a more similar activation pattern to the HCP ground truth data compared to the Looping Star reconstruction. This is mainly due to the reduction of the large activated area in the Looping Star reconstruction. The 3D-UNet reconstruction also managed to reduce activation in areas in the Looping Star reconstruction that were not activated in the ground truth data. In a few cases, the 3D-UNet even managed to produce activations in areas where it had been lost between ground truth data and Looping Star reconstruction. These results suggest that the 3D-UNet manages to suppress some of the time-series noise in the Looping Star reconstruction while still preserving the BOLD variations. However, while it managed to recover some lost signal, the majority of the signal loss

after downsampling is not improved by the network as it fails to recover many of the activated areas.

The group-level analysis of the 5 participants in the test set show that the 3D-UNet reconstructions detected fewer connected voxels across the test set participants. The bleeding of voxel signal in the Looping Star reconstruction is suppressed in the group-level analysis (Fig 4.20). This suggests that this effect is less of an issue for group-level analyses as it manages to distinguish which voxels that are actually correlated on the group level. The suppression of the increased activation in the Looping Star data was the most prominent effect of the 3D-UNet reconstruction in the single-participant analysis. As this effect is mitigated in the group-level analysis, the 3D-UNet reconstruction shows no clear advantages over the Looping Star reconstruction on the group level.

The networks in this project were trained with no reward for accurate reconstruction of temporal voxel variations. The only temporal representation during training was the 50 different volumes per participant. These volumes were randomly loaded to the network during training and the temporal variations of individual voxel values were therefore not known to the network. The results from the fMRI analyses show that despite this, a lot of the BOLD signals were conserved by the network. In addition, unwanted noise that was introduced by the Looping Star reconstruction was suppressed. This is especially visible for the seed-to-voxel analysis for a single participant. However, as the results show, the network did not manage to improve upon the correlations in the group-level analysis from the Looping Star reconstruction. This suggests that the methodology that has been proposed in this project has limitations, and any potential users should be aware of this. However, the fact that a lot of the BOLD variation is conserved suggests that different methodologies that focus on rewarding the preservation of correct signal variation could be implemented to improve performance.

Chapter 6

Conclusion and future work

In this thesis, a neural network for the reconstruction of Looping Star fMRI images was developed and evaluated. The main findings of this study revealed that a 3D-UNet outperformed a 2D-UNet despite having fewer model parameters. The utilization of 3D convolutions in the 3D-UNet allowed for a better reduction of the global artifacts and noise patterns that are produced by the reconstruction of a Looping Star k-space trajectory. One conclusion that can be drawn from this is that the shape of the k-space trajectory and the artifacts it produces should be considered when deciding on a network architecture. fMRI analysis of the deep learning reconstructions also revealed that the deep neural network managed to retain a lot of the fMRI BOLD variations and remove unwanted noise despite the large error between network input and ground truth data. The network managed this despite being trained with no reward for accurate BOLD variation recovery. This shows promise for the use of deep learning reconstruction for fMRI data, which is an unexplored research field.

As the field of deep learning reconstruction for the Looping Star sequence has not been explored before, the work presented in this thesis is of novelty and value on its own. However, there is also potential for future work and improvements.

The work in this thesis has not involved testing and evaluation of the network performance on real Looping Star images. The only data that has been tested was the images created through emulation of the k-space acquisition. Testing and evaluating the network on real Looping Star images are essential to assess its performance in reducing noise and artifacts in an actual clinical setting. Comparing the network's performance on emulated and real data can uncover potential limitations in the methodology.

To further improve the reconstruction results and explore the field of deep learning

reconstruction, other reconstruction methods could be explored. Using different non-Cartesian reconstruction methods, such as gridding, can create new training datasets that allow for deep learning k-space interpolation which has shown promise [61]. Exploring different datasets, or modifying the data preprocessing procedure, layer structure and network depth of the 3D-UNet, are potential avenues for improving model performance. Additionally, increasing the network complexity through cascaded networks, which have gained popularity in image reconstruction, could offer even better model performance [68][8]. These future endeavors will contribute to advancing the field and unlocking the full potential of deep learning reconstruction for Looping Star fMRI imaging.

Appendix A

Source code

The code used in this thesis is available on Github:
https://github.com/MariusRusaas/MasterThesis_LS_DeepLearning

This code contains both the MatLab code used for downsampling and calculation of test metrics, Python and PyTorch code used for dataset preparation and training as well as the code from Ray Tune used for hyperparameter selection. Test set metrics, training metrics and metrics from the hyperparameter optimization are saved in e.g. JSON or MAT files.

A.1 Folders

A.1.1 DeepLearning

- **dataset_management.py**: Dataset classes used for generating the data loaders.
- **2D-UNet**: Folder containing the utilities used for training a 2D-UNet model as well as a notebook detailing the training process.
- **3D-UNet**: Folder containing the utilities used for training a 3D-UNet model as well as notebooks detailing the training process on both processed and unprocessed data. This folder also contains the RayTune model selection experiment on processed data.

A.1.2 LS_downsampling

- **downsampling_wDCF.m**: MATLAB script that demonstrates the usage of the MIRT non-Uniform k-space tool to downsample a full dataset.

- **downsampling_utils**: Folder containing MATLAB functions used during the downsampling, e.g. the image squaring, data saving and downsampling functions.
- **test_subj.mat**: MATLAB file containing the subject IDs of the test set participants.

A.1.3 test_set_evaluation

- SnR.m: MATLAB function for calculating the signal-to-noise ratio in one volume.
- **test_metrics.m / test_metrics_proc.m**: MATLAB scripts that calculate the metrics for all the subjects in the test set.
- **test_metrics_proc.mat / test_metrics_unproc.mat**: MATLAB files with the calculated metrics for all test set subjects.
- **Voxel_evaluation**: A folder containing the MatLab scripts used when calculating the correlations between voxel time-series.

Bibliography

[1] Nikou L. Damestani et al. “Revealing the mechanisms behind novel auditory stimuli discrimination: An evaluation of silent functional MRI using looping star”. In: *Human Brain Mapping* 42.9 (2021), pp. 2833–2850. DOI: https://doi.org/10.1002/hbm.25407.

[2] Beatriz Dionisio-Parra et al. “Looping Star fMRI in Cognitive Tasks and Resting State”. In: *Journal of Magnetic Resonance Imaging* 52.3 (2020), pp. 739–751. DOI: https://doi.org/10.1002/jmri.27073.

[3] Adriaan Moelker and Peter M.T. Pattynama. “Acoustic noise concerns in functional magnetic resonance imaging”. In: *Human Brain Mapping* 20.3 (2003), pp. 123–141. DOI: https://doi.org/10.1002/hbm.10134.

[4] nordicAudio. *nordicAudio - More sound. less noise.* URL: https://www.nordicneurolab.com/nordicaudio.

[5] Jana Hutter et al. “Quiet echo planar imaging for functional and diffusion MRI”. In: *Magnetic Resonance in Medicine* 79.3 (2018), pp. 1447–1459. DOI: https://doi.org/10.1002/mrm.26810.

[6] Florian Wiesinger, Anne Menini, and Ana Beatriz Solana. “Looping Star”. In: *Magnetic Resonance in Medicine* 81.1 (2019), pp. 57–68. DOI: https://doi.org/10.1002/mrm.27440.

[7] Siemens Healthineers. *Deep Resolve, MRI - faster than ever before*. URL: https://www.siemens-healthineers.com/magnetic-resonance-imaging/technologies-and-innovations/deep-resolve.

[8] *Philips SmartSpeed: No compromise - Image quality and speed at your fingertips*. URL: https://www.philips.no/healthcare/resources/landing/smartspeed.

[9] GE Healthcare. *MR image reconstruction with AIR Recon DL*. URL: https://www.gehealthcare.com/products/magnetic-resonance-imaging/air-technology/air-recon-dl.

[10] Yoseob Han et al. “Deep learning with domain adaptation for accelerated projection-reconstruction MR”. In: *Magnetic Resonance in Medicine* 80.3 (2018), pp. 1189–1205. DOI: https://doi.org/10.1002/mrm.27106.

[11] Leyla Loued-Khenissi, Olivia Döll, and Kerstin Preuschoff. “An Overview of Functional Magnetic Resonance Imaging Techniques for Organizational Research”. In: *Organizational Research Methods* 22.1 (2019), pp. 17–45. DOI: 10.1177/1094428118802631.

[12] Atle Bjørnerud. *The Physics of Magnetic Resonance Imaging*. 2006.

[13] Stuart Currie et al. “Understanding MRI: basic MR physics for physicians”. In: *Postgraduate Medical Journal* 89.1050 (Dec. 2012), pp. 209–223. ISSN: 0032-5473. DOI: 10.1136/postgradmedj-2012-131342.

[14] Ronald Bracewell. *The Fourier transform and its applications*. McGraw-Hill, 1986.

[15] John W. Tukey James W. Cooley. “An algorithm for the machine calculation of complex Fourier series”. In: *Mathematics of Computation* (1965), pp. 297-301–1054. DOI: 10.1090/S0025-5718-1965-0178586-1.

[16] Donald B. Plewes and Walter Kucharczyk. “Physics of MRI: A primer”. In: *Journal of Magnetic Resonance Imaging* 35 (5 2012), pp. 1038–1054. ISSN: 10531807. DOI: 10.1002/jmri.23642.

[17] John Pauly. *Non-Cartesian Reconstruction*. 2005. URL: https://mri-q.com/uploads/3/4/5/7/34572113/pauly-non-cartesian_recon.pdf.

[18] Gary H. Glover. “Overview of Functional Magnetic Resonance Imaging”. In: *Neurosurgery Clinics of North America* 22.2 (2011). Functional Imaging, pp. 133–139. ISSN: 1042-3680. DOI: https://doi.org/10.1016/j.nec.2010.11.001.

[19] Robert Turner et al. “Functional magnetic resonance imaging of the human brain: data acquisition and analysis”. In: 123 (1998), pp. 5–12. DOI: 10.1007/s002210050538.

[20] Linus Pauling and Charles D. Coryell. “The Magnetic Properties and Structure of Hemoglobin, Oxyhemoglobin and Carbonmonoxyhemoglobin”. In: *Proceedings of the National Academy of Sciences* 22 (1936), pp. 210–216. ISSN: 0027-8424. DOI: 10.1073/PNAS.22.4.210.

[21] Seiji Ogawa et al. “Oxygenation-sensitive contrast in magnetic resonance image of rodent brain at high magnetic fields”. In: *Magnetic Resonance in Medicine* 14.1 (1990), pp. 68–78. DOI: https://doi.org/10.1002/mrm.1910140108.

[22] Seiji Ogawa and Tso-Ming Lee. “Magnetic resonance imaging of blood vessels at high fields: In vivo and in vitro measurements and image simulation”. In: *Magnetic Resonance in Medicine* 16.1 (1990), pp. 9–18. DOI: https://doi.org/10.1002/mrm.1910160103.

[23] Seiji Ogawa et al. "Brain magnetic resonance imaging with contrast dependent on blood oxygenation." In: *Proceedings of the National Academy of Sciences* 87.24 (1990), pp. 9868–9872. DOI: 10.1073/pnas.87.24.9868.

[24] Seiji Ogawa et al. "Intrinsic signal changes accompanying sensory stimulation: functional brain mapping with magnetic resonance imaging." In: *Proceedings of the National Academy of Sciences* 89.13 (1992), pp. 5951–5955. DOI: 10.1073/pnas.89.13.5951.

[25] Karl Friston et al., eds. *Statistical Parametric Mapping: tha analysis of functional brain images*. London: Academic Press, 2007. ISBN: 978-0-12-372560-8. DOI: https://doi.org/10.1016/B978-0-12-372560-8.50052-8.

[26] Seong-Gi Kim and Seiji Ogawa. "Biophysical and Physiological Origins of Blood Oxygenation Level-Dependent fMRI Signals". In: *Journal of Cerebral Blood Flow & Metabolism* 32.7 (2012). PMID: 22395207, pp. 1188–1206. DOI: 10.1038/jcbfm.2012.23.

[27] Alfonso Nieto-Castanon. *Handbook of functional connectivity Magnetic Resonance Imaging methods in CONN*. Feb. 2020. ISBN: 978-0-578-64400-4. DOI: 10.56441/hilbertpress.2207.6598.

[28] H. Lv et al. "Resting-state functional MRI: Everything that nonexperts have always wanted to know". In: *American Journal of Neuroradiology* 39 (8 Aug. 2018), pp. 1390–1399. ISSN: 1936959X. DOI: 10.3174/ajnr.A5527.

[29] Bharat Biswal et al. "Functional connectivity in the motor cortex of resting human brain using echo-planar mri". In: *Magnetic Resonance in Medicine* 34.4 (1995), pp. 537–541. DOI: https://doi.org/10.1002/mrm.1910340409.

[30] David P. Madio and Irving J. Lowe. "Ultra-fast imaging using low flip angles and fids". In: *Magnetic Resonance in Medicine* 34.4 (1995), pp. 525–529. DOI: https://doi.org/10.1002/mrm.1910340407.

[31] Aston Zhang et al. *Dive into Deep Learning*. 2023. DOI: https://doi.org/10.48550/arXiv.2106.11342. arXiv: 2106.11342 [cs.LG].

[32] Xue Ying. "An Overview of Overfitting and its Solutions". In: *Journal of Physics: Conference Series* 1168.2 (2019), p. 022022. DOI: 10.1088/1742-6596/1168/2/022022.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[34] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com.

[35] Gabriel Goh. "Why Momentum Really Works". In: *Distill* (2017). DOI: 10.23915/distill.00006.

[36] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[37] Stefan Falkner, Aaron Klein, and Frank Hutter. *BOHB: Robust and Efficient Hyperparameter Optimization at Scale*. 2018. arXiv: 1807.01774 [cs.LG].

[38] Matthias Feurer and Frank Hutter. "Hyperparameter Optimization". In: *Automated Machine Learning: Methods, Systems, Challenges*. Ed. by Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Cham: Springer International Publishing, 2019, pp. 3–33. ISBN: 978-3-030-05318-5. DOI: 10.1007/978-3-030-05318-5_1.

[39] Lisha Li et al. *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*. 2018. arXiv: 1603.06560 [cs.LG].

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. DOI: https://doi.org/10.48550/arXiv.1505.04597. arXiv: 1505.04597 [cs.CV].

[41] Özgün Çiçek et al. *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*. 2016. DOI: https://doi.org/10.48550/arXiv.1606.06650. arXiv: 1606.06650 [cs.CV].

[42] Shekhar S Chandra et al. "Deep learning in magnetic resonance image reconstruction". In: *Journal of Medical Imaging and Radiation Oncology* 65.5 (2021), pp. 564–577. DOI: https://doi.org/10.1111/1754-9485.13276.

[43] H. Fischer and R. Ladebeck. "Echo-Planar Imaging Image Artifacts". In: *Echo-Planar Imaging: Theory, Technique and Application*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 179–200. ISBN: 978-3-642-80443-4. DOI: 10.1007/978-3-642-80443-4_6.

[44] Matthew F. Glasser et al. "The minimal preprocessing pipelines for the Human Connectome Project". In: *NeuroImage* 80 (2013). Mapping the Connectome, pp. 105–124. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2013.04.127.

[45] Jeffrey A Fessler. *Michigan Image Reconstruction Toolbox*. Sept. 9, 2022. URL: https://web.eecs.umich.edu/~fessler/code/index.html.

[46] *MATLAB version 9.13.0.2105380 (R2022b) Update 2*. The Mathworks, Inc. Natick, Massachusetts, 2022.

[47] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[48] Richard Liaw et al. “Tune: A Research Platform for Distributed Model Selection and Training”. In: *arXiv preprint arXiv:1807.05118* (2018).

[49] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[50] Milesial. *PyTorch-UNet*. Nov. 25, 2022. URL: https://github.com/milesial/Pytorch-UNet.

[51] Adrian Wolny et al. “Accurate and versatile 3D segmentation of plant tissues at cellular resolution”. In: *eLife* 9 (2020). Ed. by Christian S Hardtke et al., e57613. ISSN: 2050-084X. DOI: 10.7554/eLife.57613.

[52] Yoseo Han, Leonard Sunwoo, and Jong Chul Ye. “*k* -Space Deep Learning for Accelerated MRI”. In: *IEEE Transactions on Medical Imaging* 39.2 (2020), pp. 377–386. DOI: 10.1109/TMI.2019.2927101.

[53] Zilu Ma and Nanyin Zhang. “Chapter 22 - Brain-wide connectivity architecture: developmental aspects”. In: *Factors Affecting Neurodevelopment*. Ed. by Colin R. Martin, Victor R. Preedy, and Rajkumar Rajendram. Academic Press, 2021, pp. 247–257. ISBN: 978-0-12-817986-4. DOI: https://doi.org/10.1016/B978-0-12-817986-4.00022-5.

[54] Susan Whitfield-Gabrieli and Alfonso Nieto-Castanon. “Conn: A Functional Connectivity Toolbox for Correlated and Anticorrelated Brain Networks”. In: *Brain Connectivity* 2.3 (2012). PMID: 22642651, pp. 125–141. DOI: 10.1089/brain.2012.0073.

[55] A. Nieto-Castanon and S. Whitfield-Gabrieli. *CONN functional connectivity toolbox (RRID:SCR_009550)*. Version version 22. Nov. 30, 2022. DOI: doi:10.56441/hilbertpress.2246.5840. URL: https://web.conn-toolbox.org/.

[56] John Ashburner and Karl J. Friston. “Unified segmentation”. In: *NeuroImage* 26.3 (2005), pp. 839–851. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2005.02.018.

[57] John Ashburner. “A fast diffeomorphic image registration algorithm”. In: *NeuroImage* 38.1 (2007), pp. 95–113. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2007.07.007.

[58] Rahul S. Desikan et al. “An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest”. In: *NeuroImage* 31.3 (2006), pp. 968–980. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2006.01.021.

[59] Matt A. Bernstein, Kevin F. King, and Xiaohong Joe Zhou. "Chapter 16 - Echo Train Pulse Sequences". In: *Handbook of MRI Pulse Sequences*. Ed. by Matt A. Bernstein, Kevin F. King, and Xiaohong Joe Zhou. Burlington: Academic Press, 2004, pp. 702–801. ISBN: 978-0-12-092861-3. DOI: https://doi.org/10.1016/B978-012092861-3/50023-6.

[60] Florian Wiesinger. Tobias C Wood Emil Ljungberg. *Radial Interstices Enable Speedy Low-Volume imagING*. URL: https://github.com/spinicist/riesling.

[61] Tianming Du et al. "Adaptive convolutional neural networks for accelerating magnetic resonance imaging via k-space data interpolation". In: *Medical Image Analysis* 72 (2021), p. 102098. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2021.102098.

[62] Chang Min Hyun et al. "Deep learning for undersampled MRI reconstruction". In: *Physics in Medicine Biology* 63.13 (2018), p. 135007. DOI: 10.1088/1361-6560/aac71a.

[63] Ke Wang et al. "High Fidelity Deep Learning-based MRI Reconstruction with Instance-wise Discriminative Feature Matching Loss". In: (2021). arXiv: 2108.12460 [eess.IV].

[64] Kyong Hwan Jin et al. "Deep Convolutional Neural Network for Inverse Problems in Imaging". In: *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4509–4522. DOI: 10.1109/TIP.2017.2713099.

[65] Thomas Küstner et al. "CINENet: deep learning-based 3D cardiac CINE MRI reconstruction with multi-coil complex-valued 4D spatio-temporal convolutions". In: *Scientific Reports* 10.1 (Aug. 2020), p. 13710. ISSN: 2045-2322. DOI: 10.1038/s41598-020-70551-8.

[66] J. Levi Chazen et al. "Rapid lumbar MRI protocol using 3D imaging and deep learning reconstruction". In: *Skeletal Radiology* (Jan. 2023). ISSN: 1432-2161. DOI: 10.1007/s00256-022-04268-2.

[67] Vladimir Korzh. "Development of brain ventricular system". In: *Cellular and Molecular Life Sciences* 75.3 (Feb. 2018), pp. 375–383. ISSN: 1420-9071. DOI: 10.1007/s00018-017-2605-y.

[68] Omer Burak Demirel et al. "20-fold Accelerated 7T fMRI Using Referenceless Self-Supervised Deep Learning Reconstruction". In: *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*. 2021, pp. 3765–3769. DOI: 10.1109/EMBC46164.2021.9631107.