

MSc Applied Mathematics (AM)  
Final Project

# Neural Field and Spline-based Representations for Optimizing the k-space Trajectory for MRI with Deep learning

Tycho de Haan

Supervisors:


Chair: Prof. Dr. Christoph Brune

Daily: Dr. Jelmer M. Wolterink

External: Dr. Silke Glas

Dr. Ir. Frank Simonis

October, 2023



Mathematics of Imaging and AI (MIA)  
Faculty of Electrical Engineering,  
Mathematics and Computer Science  
University of Twente

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>   | <b>3</b>  |
| <b>2</b> | <b>Related works and theory</b>                                 | <b>6</b>  |
| 2.1      | Theoretical framework MRI . . . . .                             | 6         |
| 2.2      | Reconstruction methods . . . . .                                | 8         |
| 2.2.1    | Classical reconstruction methods . . . . .                      | 9         |
| 2.2.2    | Deep learning based reconstruction methods . . . . .            | 9         |
| 2.3      | Sampling trajectory optimization techniques . . . . .           | 10        |
| 2.3.1    | PILOT . . . . .   | 10        |
| 2.3.2    | SPARKLING . . . . .   | 11        |
| 2.3.3    | Hybrid Approach . . . . .                                       | 11        |
| 2.4      | General sampling theory . . . . .                               | 11        |
| <b>3</b> | <b>Method</b>   | <b>16</b> |
| 3.1      | BJORK problem formulation . . . . .                             | 16        |
| 3.2      | Calculating the forward operator . . . . .                      | 17        |
| 3.3      | B-spline sampling layer . . . . .                               | 19        |
| 3.4      | SIREN sampling layer . . . . .                                  | 20        |
| 3.5      | Reconstruction network . . . . .                                | 22        |
| 3.6      | Complete algorithm and dataset . . . . .                        | 23        |
| <b>4</b> | <b>Experiments and results</b>                                  | <b>26</b> |
| 4.1      | Implementation and general setup BJORK . . . . .                | 26        |
| 4.2      | Baseline BJORK results . . . . .                                | 27        |
| 4.3      | Determining the learning rate . . . . .                         | 29        |
| 4.4      | Determining the loss terms . . . . .                            | 31        |
| 4.5      | Warm up reconstruction network . . . . .                        | 35        |
| 4.6      | Training on multi coil compared to single coil data . . . . .   | 35        |
| 4.7      | Testing on datasets containing different anatomies . . . . .    | 36        |
| 4.8      | Testing which sampling patterns SIREN can represent . . . . .   | 37        |
| 4.9      | Optimizing the parameters in the SIREN sampling layer . . . . . | 39        |
| <b>5</b> | <b>Discussion</b>   | <b>45</b> |
| 5.1      | Replicating BJORK . . . . .                                     | 45        |
| 5.2      | Difficulty in determining hyper-parameters . . . . .            | 45        |
| 5.3      | Bi-level optimization problems . . . . .                        | 46        |
| 5.4      | BJORK versus SIREN . . . . .                                    | 47        |
| <b>6</b> | <b>Future work</b>  | <b>49</b> |
| <b>7</b> | <b>Conclusion</b>   | <b>50</b> |

# 1 Introduction

Magnetic resonance imaging (MRI) is one of the most commonly used medical imaging techniques available. In contrast to computed tomography (CT) scans, MRI is safer because it does not use the possibly carcinogenic radiation but magnetization for imaging. Among many things, MRI is used for diagnosing diseases and treatment planning, as it can produce high-quality images with limited motion artifacts. The main drawback of MRI is the time needed to perform a scan, which typically varies from 15 to 90 minutes depending on the size of the area that is being scanned and the number of slices to scan [1]. This limits the total number of scans that can be performed in a given time, making MRI a very costly imaging modality. An important focus of much current research into MRI is to investigate how the total scan time of an MR scanner can be reduced, in order to reduce its cost and increase the total scan capacity, while keeping a reasonable image quality.

An MRI scanner collects measurements in a space of frequencies, which is called the k-space. From these measurements one can reconstruct an image of the object of interest. Samples are collected along a predefined sampling trajectory, which specifies at what time a signal is collected corresponding to a specific spatial frequency, i.e. a specific k-space location. Figure 1 shows the relation between k-space and the image. Sampling the full k-space leads to a perfect reconstruction in the absence of noise. Sampling only the central part of k-space recovers only the low frequency information leading to a reconstruction only containing contrast. Conversely, sampling only the periphery leads to a reconstruction that contains only the high frequency information, which represent the edges of the image. The most commonly used sampling pattern is Cartesian sampling, which uses multiple horizontal lines to sample the entirety of k-space. Since the samples are acquired on a Cartesian grid, reconstruction can be done easily and effectively by applying the inverse fast Fourier transform (IFFT). The downside is that many lines need to be sampled to cover the entire k-space, which is time consuming. Moreover, the sampling pattern does not agree with the results of compressed sensing (CS). This theoretical framework proposes a variable density pattern, where the center of k-space is sampled more densely than the periphery [13]. CS suggests that to minimize the number of measurements needed, one should draw samples from a distribution, where the probability that a given sample is chosen is proportional to this Fourier coefficient. Since the central k-space Fourier coefficients are typically larger than the ones at the periphery, CS suggests to sample the center of k-space more heavily.

Early works that attempt to learn a VDS pattern often use point-based sampling [4], [35]. This method is problematic because it disregards the way an MR scanner works, that is using lines to sample points. By learning a cloud of points, the number of sampled points might be smaller than in conventional Cartesian MR. However, if the number of lines needed to connect the points does not decrease, the total scan time does not decrease as well, since every line costs about the same time to scan. One line can only have a limited number of points, because over time the signal dies out due to spin dephasing. Also the distance between points within a line is limited by the gradient coils maximal amplitude. How to scan a point cloud is not trivial, as there are many ways to divide points into lines. In conclusion, when these methods speak of acceleration in terms of number of points sampled, the actual scan might not be accelerated.

To be able to determine a good sampling pattern there are various aspects to consider. The sampling pattern should be feasible in terms of what the MR scanner can perform and actually improve scan time. In addition, the sampling pattern should find samples that are most informative for reconstructing the image. The reconstruction problem is often posed as an inverse problem. In the variational approach these inverse problems are solved by minimizing a data consistency term and a regularization term. Regularization is needed to prevent overfitting on the noisy data, which leads to noisy reconstructed images. The regularization function, also called a prior, was

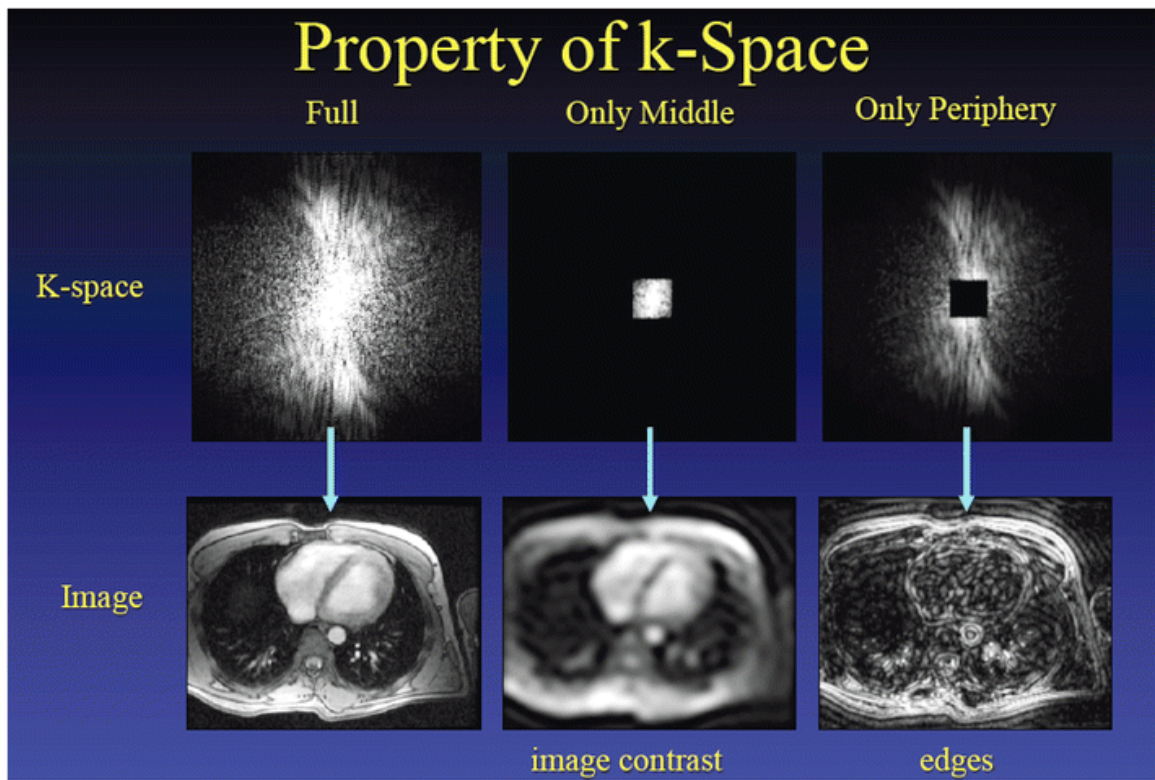


Figure 1: Three figures demonstrating the difference between low frequency and high frequency information in k-space and its influence on the reconstructed image.

traditionally a hand-crafted term, such as an  $L_2$  penalty or total variation (TV) regularization. Due to recent advances in deep learning, the regularizer is often replaced by a deep neural network [25]. The network parameters are optimized by training on a large dataset of MR images in an end-to-end fashion. In this way, the reconstruction algorithm can learn much more informed priors and the overall reconstruction quality greatly improves.

Taking this one step further, algorithms have been developed that attempt to learn the sampling pattern and reconstruction jointly. Early works include the method by Sherry et al [35], which can learn a variable density sampling pattern of scattered points or Cartesian lines jointly with a reconstruction algorithm. Later, SPARKLING [22] and PILOT [46] were introduced which allowed for a wider range of possible k-space trajectories in its optimization. A recent influential work is BJORK [43], which serves as the basis for this work and uses B-splines to parameterize the sampling trajectory. In BJORK, the sampling trajectory and reconstruction network are optimized jointly in an end-to-end fashion. In short, this means that the algorithm starts with an image as input. Next it simulates a k-space based on the current sampling pattern, reconstructs the image and computes the loss, which measures the difference between the reconstructed image and the ground truth. Next, using backpropagation, the weights of the B-splines and the reconstruction network are adapted, such that both the sampling pattern and reconstruction are changed in such a way that the reconstructed image is likely closer to the ground truth. We took this method as a starting point, because it achieves good results and uses penalties to make the sampling trajectory feasible. Moreover, compared to the other influential SPARKLING method [22], BJORK does not require a heuristic sampling density. The sampling pattern is learned fully based on the data it is trained on. This is preferable, as a target sampling density requires expert knowledge and introduces bias.

BJORK uses splines to parameterize the sampling trajectories, but in principle many different methods can be used for parameterization. Wolterink et al. showed the power of neural fields to represent the transformation function in image registration [47]. Neural fields (NFs) are neural networks that parameterize a signal on its domain, for example the pixel locations for images. The neural network architecture acts as regularizer on the object it represents and it turns discrete signals into continuous functions. A good network structure for INR is to use a multilayered perceptron using periodic activation functions, also called sinusoidal representation network (SIREN) [36]. The main advantage of SIREN is that the derivatives of output with respect to input are continuous and can also be represented as SIRENs, since the derivative of a sinusoid is again a sinusoid. Typically, these analytical derivatives are more accurate than the alternative finite difference computations. These properties make SIREN a good candidate to represent not only images but also the k-space sampling pattern.

In this paper we are interested whether representing the k-space sampling pattern using a neural field leads to better results, in terms of reconstruction, but also in terms of convergence behavior. This leads us to the research question:

When jointly optimizing the k-space sampling trajectory and reconstruction network to obtain the best reconstruction results is it better to parameterize the sampling trajectory using B-splines or a SIREN neural field representation?

Chapter 2 discusses the theoretical background of MRI, sampling patterns and reconstruction as well as discussing the related literature. Chapter 3 explains the method used to perform the experiments, including an explanation of BJORK and SIREN. Chapter 4 introduces the experiments that were performed and the results of the experiments. Chapter 5 includes a analysis of the results and discusses research limitations. Chapter 6 discusses future research opportunities. Chapter 7 concludes the paper.

## 2 Related works and theory

### 2.1 Theoretical framework MRI

The image is related to the measurements by the following equation:

$$v = Au + \varepsilon \quad (1)$$

Here  $v \in \mathbb{C}^m$  is a vector of  $m$  measurements,  $u \in \mathbb{C}^n$  the image,  $A$  the system forward operator and  $\varepsilon$  the system noise. The goal is to find  $u$  given  $v$  such that  $v = Au + \varepsilon$ . Problems of these kind are called inverse problems and have been studied extensively in maths and physics for a long time. The system operator  $A$  encodes how the image and measurements are related. This operator describes the entire physical process that generates the measurements. Solving for  $u$  given  $v$  is what we refer to as the reconstruction problem. In this section we only discuss how the MRI inverse problem is defined, not inverse problems in general.

To understand MRI it is essential to understand how the k-space measurements are constructed, as this is what the forward operator does [39]. The k-space is essentially the Fourier transform of the image, which means it transforms the image into the frequency domain. Let  $u = f(x, y)$  be the image with coordinates  $x$  and  $y$ , then its Fourier transform is defined by:

$$v = F(k_x, k_y) = \sum_{i=1}^n \sum_{j=1}^n f(x_i, y_j) e^{-i2\pi(k_x x_i + k_y y_j)} \quad (2)$$

So for a two dimensional image with pixel locations  $x_i$  and  $y_j$ , the k-space is two-dimensional as well, with two axes denoted by  $k_x$  and  $k_y$ , which are the spatial frequencies in  $x$  and  $y$  direction respectively. In general to reconstruct a  $N \times N$  image you need  $N \times N$  samples in k-space. This is called a fully sampled k-space, because for each k-space location  $k_x, k_y$  there is a sample, i.e. the quantity in equation 2 is known. Figure 2 shows the relation between image and k-space in more detail. The  $k_x$  direction is called the frequency encoding direction and  $k_y$  is called the phase encoding direction. The size of the k-space depends on the size of the image, called the field of view (FOV) and the resolution of the image. The size between k-space samples  $\Delta k$  and FOV are related by the formula  $\Delta k = \frac{1}{FOV}$ . The spatial resolution is related to the size of k-space by the formula  $\Delta x = \frac{1}{2k_{max}}$ .

It is impossible to sample all points at once, because the MR signal dies out over time due to the phenomenon of spin dephasing. This means that a scan is partitioned into distinct periods of time, where each period contains a number of points that are being sampled. This collection of points in one partition can be seen as a line in k-space, connecting the various points that are sampled in that partition. These lines in k-space are more commonly referred to as shots. Each shot samples a particular part of the k-space. The collection of shots is referred to as a sampling pattern, denoted by  $\omega$ . The most commonly used sampling pattern is the Cartesian pattern. This pattern just samples horizontal shots in k-space as shown in Figure 3 on the left. This Figure also shows radial and spiral sampling pattern. These are undersampled patterns because they have fewer shots. The total time a scan takes is roughly proportional to the number of shots used. Therefore we are interested in constructing a sampling pattern with fewer shots and radial or spiral-like patterns are a good starting point.

The choice of sampling pattern  $\omega$  determines the forward operator  $A$ , because it determines which k-space measurements are sampled. Based on the measurements and forward operator, we can reconstruct  $u$  such that  $v = A(\omega)u$ . By selecting  $\omega$  in a way that has a lower number of shots, we can save time while still being able to reconstruct  $u$ . The problem of choosing  $\omega$  is called the

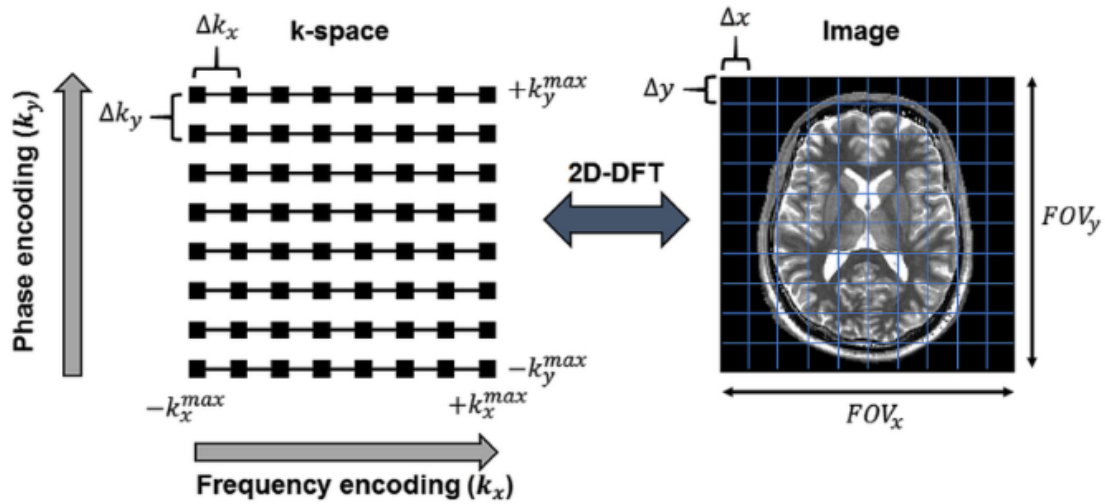


Figure 2: On the left the k-space is shown abstractly as a two dimensional space with frequency encoding direction  $k_x$  and phase encoding direction  $k_y$ . Each k-space measurement has a specific coordinate  $(k_x, k_y)$  in k-space, which specifies which frequency the measurement corresponds to. The right shows the image which is related to the k-space via the two dimensional discrete Fourier transform. The image consists of pixel intensities for a given pixel coordinate  $(x, y)$ . [19]

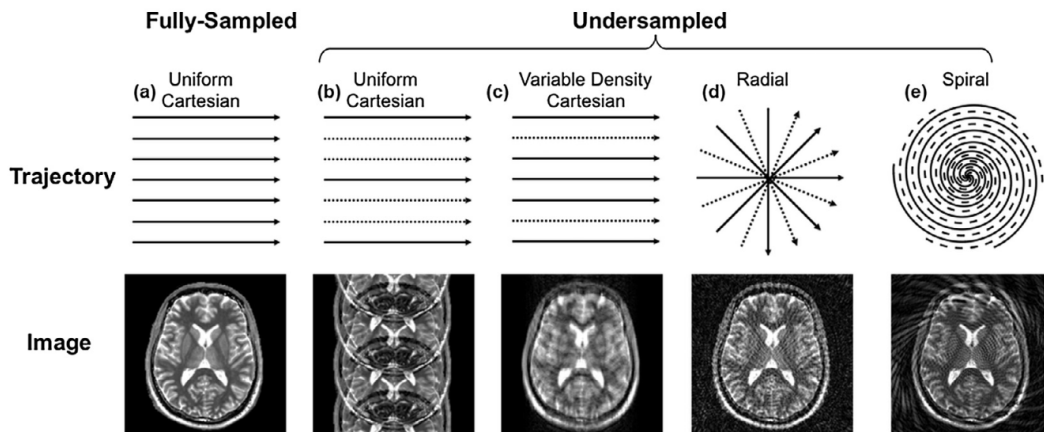


Figure 3: From left to right various sampling trajectories are shown together with the corresponding reconstructed image based on a simple Fourier transformation. From left to right, the uniform Cartesian with perfect reconstruction, the uniform Cartesian with overlapping artifacts, the variable density Cartesian with blurry reconstruction, the radial pattern with streaking artifacts and last the spiral pattern with spiral striking artifacts. [19]

sampling problem. The sampling and reconstruction problems can be solved jointly as a bi-level optimization problem. We want to find the sparser sampling pattern such that we can reconstruct  $u$  in the best way. Here, we should specify what the constraints on  $\omega$  are.

Samples in k-space are collected along parameterized lines  $s : [0, T] \rightarrow \mathbb{R}^2$  [8]. The gradient waveform needed to produce a sampling curve  $s$  is defined by:  $g(t) = \gamma^{-1}\dot{s}(t)$  or in other words  $s(t) = \gamma \int_0^t g(\tau) d\tau$ , where  $\gamma$  denotes the gyromagnetic ratio. This gradient waveform is produced by running an electric current through the gradient coils in the MRI scanner. Because of hardware limitations, the amplitude of this electric current is bounded and can not change too quickly. These two constraints are referred to as the maximal gradient and slew rate respectively and are denoted as:

$$\|\dot{s}\|_{\infty,2} \leq \gamma G_{max} \quad \|\ddot{s}\|_{\infty,2} \leq \gamma S_{max}$$

This norm is the  $l^{\infty,2}$  norm which is defined by:

$$\|f\|_{\infty,2} = \sup_{t \in [0, T]} \left( \sum_{i=1}^d |f_i(t)|^2 \right)^{\frac{1}{2}}$$

and the dot denotes the time derivative:

$$\dot{s} = \frac{s(t_i) - s(t_{i-1})}{\Delta t} \quad (3)$$

here  $\Delta t$  denotes the raster time, which is the time between subsequent samples in k-space. Additionally, the signal intensity decays due to transverse relaxation of spins. Therefore there is a maximal time the MR signal is available.

## 2.2 Reconstruction methods

The reconstruction problem is typically formulated in the variational way:

$$\min_x \{D(Ax, y) + R(x)\} \quad (4)$$

Here  $D(\cdot)$  is the data consistency function, which ensures the image in the measurement domain is close to the measured data. The function  $R(\cdot)$  is the regularization function, which acts to make the image more regular, that is, it should have desirable properties that make it more likely to be the image that we are looking for. The choice of  $R(\cdot)$  reflects which properties we want the image  $x$  to have.

There are various aspects to solving this reconstruction problem. The main aspects that we are going to consider are: the choice of data consistency term, the choice of regularizer and the choice of minimization technique. The choice of data consistency is typically mean squared error or mean absolute error. Combinations of these terms are also possible. The choice of regularizer and of optimization algorithm are both very important and also connected. For example, if the regularization function is not differentiable, one can not use methods that need an explicit gradient such as gradient descent. Of course the data consistency also influences the choice of optimization, but the choice for data consistency terms is more limited. Therefore we focus on the choice of regularizer and optimization method in the next sections and how the two choices are connected.



### 2.2.1 Classical reconstruction methods

The earliest reconstruction techniques are based on regridding and were developed in the 80s and 90s [23], [33]. Later compressed sensing techniques were developed, which are based on  $l_1$  regularization of the wavelet coefficients. Here we want to minimize [24]:

$$\min_x \|A(\omega)u - v\| + \lambda \|\Psi u\|_1 \quad (5)$$

where  $A(\omega)$  is the undersampled Fourier operator corresponding to the sampling pattern  $\omega$  and  $\Psi$  is some transformation function that is supposed to transform the image into some sparse domain. If the sparsifying transform is chosen to be the finite difference operator, this is called total variation (TV) regularization. Alternatively, the orthogonal wavelet synthesis operator can be used. The  $l_1$  norm is taken to enforce sparsity, as an approximation to the  $l_0$  norm. The  $l_0$  norm is not taken because it would make the problem a NP-hard optimization problem.

There are various ways to solve a function such as in equation 5. There is the first order primal dual algorithm by Chambolle and Pock [12]. This algorithm was shown to work on a variety of imaging inverse problems. There is also the Fast Iterative Soft Thresholding algorithm (FISTA) [5] and the Alternating direction method of multipliers (ADMM) [7] which are commonly used to solve such problems.

### 2.2.2 Deep learning based reconstruction methods

A starting point for a reconstruction method is to use an inverse Fourier transform (IFT). Such a reconstruction can be improved by introducing a denoiser network. The parameters of this network can be trained on a dataset, making it a deep learning based reconstruction method. Since we are dealing with images the denoiser is usually a convolutional neural network (CNN). More specifically, as a CNN the U-net is commonly used [31]. This deep convolutional neural network was originally designed to perform segmentation of images, but turned out to be useful for a wide variety of tasks in imaging. The U-net is effective because its structure of downsampling and upsampling allows it to learn features at various resolutions. It can be used to denoise the k-space before applying the IFT, or to denoise the image after the IFT has been applied. Alternatively it can also be trained to approximate the IFT and act as a denoiser at the same time. The U-net architecture is also part of many more involved reconstruction algorithms.

One of the earliest attempts to reconstruct MR images using deep learning was the AUTOMAP algorithm introduced in [50]. This paper uses a deep convolutional network to act as the mapping from k-space data to image directly. It does not consider what is known about the inverse operator in MRI but instead learns this on its own. This means the method needs a lot of data, which is a large drawback. The architecture used is a couple of fully connected layers, which are used to approximate the Fourier transform, followed by convolutional layers, which are supposed to extract high level features and obtain a sparse representation of the image. Their method outperforms conventional techniques such as compressed sensing reconstruction and conjugate-gradient SENSE reconstruction. Nonetheless, it is outperformed by later methods which do consider knowledge about the forward operator in their solution.

One less conventional algorithm which has been used to reconstruct images is Deep Image Prior [42]. This method uses the structure of a deep neural network to regularize images, but not in the conventional end-to-end training manner. The authors of this paper argue that a great deal of image statistics are captured by the structure of a convolutional network independent of learning. While this method does not provide a state-of-the-art reconstruction network, it provides an important idea when investigating reconstruction methods.

The Plug and Play algorithm is a method based on the proximal gradient method [38]. Proximal operators appear in proximal gradient methods such as the Alternating Direction Method of Multipliers (ADMM). The proximal operator is defined as an optimization problem and it is often difficult or computationally expensive to find a closed form. This method replaces the proximal operator by a trained denoiser, such as a convolutional neural network, but it could be any algorithm. By replacing the proximal operator with a neural network, this makes proximal methods much easier to perform. An additional benefit is that there is more flexibility in choosing operators.

The most recent and most effective type of reconstruction method are the unrolled neural network (UNN) methods. These methods combine classical methods like the primal dual gradient method with deep learning methods, such that we have a model specific part and a data specific part. The model specific part makes sure that when the forward operator is applied to the image, the results are close to the measurements. The data specific part makes sure the reconstruction images have similar features as the other images in the dataset. In this way the data is corrected both in the measurement domain (k-space) and in the image domain. One highly relevant example of an UNN is the Model-Based Deep learning (MoDL) architecture [3], which serves as the basis for the reconstruction network used in this work. In short, data consistency is enforced by numerical optimization blocks, which are optimized using the conjugate gradient method. Prior learning is done via a deep convolutional neural network. The DC block and CNN block are applied consecutively for a number of iterations. The weights of the CNN network are shared across these iterations, as this leads to better performance. Different examples of UNNs are KIKI [15], LPD-net [2],  $\Sigma$ -Net, End-to-End VarNet [37] and NC-PDNet [30]. These are the most effective reconstruction methods available, many of them performing very well on the various fastMRI challenges.

## 2.3 Sampling trajectory optimization techniques

### 2.3.1 PILOT

The first method to learn k-space trajectories and reconstruction parameters simultaneously with deep learning was PILOT [46]. PILOT stands for Physics-Informed Learned Optimized Trajectories for Accelerated MRI. PILOT validates its framework retrospectively. This means that k-space measurements are simulated instead of measured with an actual MRI scanner. The PILOT method is one deep learning pipeline consisting of multiple parts. Images are subsampled, which means applying the non-uniform fast Fourier transform (NUFFT) to get simulated measurement data according to some specified trajectory. Next, the measurements are regridded onto a Cartesian grid and the adjoint FFT is applied to obtain distorted images. Subsequently, these images are denoised via a U-net.

One issue with this method is that its performance is heavily dependent on how the trajectory is initialized. The PILOT traveling salesman problem (TSP) method is proposed, which samples k-space locations from a normal distribution and uses a TSP solver to obtain a feasible trajectory based on these points. The TSP solver finds a heuristically short path connecting the points. In later versions of this work, cubic B-splines are also used to perform multi-scale optimization. This method is called multi-scale PILOT and improves performance significantly. Learned trajectories do not deviate much from the initial trajectory. This suggests the algorithm gets stuck in a local optimum and does not learn much, which is a large drawback of this method. Still it was the first method to optimize sampling and reconstruction in a fully data driven way and served as a good starting point.

### 2.3.2 SPARKLING

SPARKLING is a  $k$ -space sampling method in the context of compressed sensing [22]. SPARKLING stands for Spreading Projection Algorithm for Rapid  $K$ -space Sampling. Two central ideas when finding the optimal sampling pattern are the use of a variable density, with more sampling in the center of  $k$ -space and getting a locally uniform coverage of  $k$ -space. Poisson disk sampling can do this but does not consider hardware constraints [9]. The idea of SPARKLING is to find a sampling density that is close to some target sampling density but also satisfies hardware constraints. This is achieved by combining gradient descent and projection onto the set of viable sampling densities. The objective function can be written as two terms: First, the attractive term, making sure the target density and obtained density are close. Second, a repulsive term making sure  $k$ -space samples do not form clusters.

The downside of this method is that the target anatomy is not considered at all [43]. The only data that is used is the target density, which was handcrafted based on ideas about what the optimal sampling should be like. As an example, the radial symmetry of the density is not realistic given that organs are in general not symmetric. Also the sampling pattern might not be suited for the reconstruction algorithm used, as the reconstruction is not part of the optimization pipeline.

### 2.3.3 Hybrid Approach

A different hybrid approach was introduced in [10]. This method introduces a new method to jointly optimize trajectories and reconstruction. In particular, the method addresses whether the trajectory and reconstruction network should be optimized at the same time, which is joint learning such as in PILOT and BJORK. A problem with this method is that the reconstruction network will initially have difficulties with dealing with different trajectories. This could lead to trajectory updates in the wrong direction, as it unfairly classifies some trajectories as giving bad reconstruction. The second option is alternating descent. In this case the trajectory is first optimized with a parameter free reconstruction algorithm. After learning the trajectory this way the reconstruction algorithm is optimized, while holding the trajectory fixed. The hybrid learning approach, combines the previous two methods by moving from alternative descent to joint learning.

Figure 4 compares the different sampling methods discussed in this section. It shows that PILOT does not deviate much from the initial radial trajectory and hence does not learn much. BJORK learns quite a lot, while keeping the initial radial structure, because the lines fluctuate but do not cross each other often. On the other hand, SPARKLING and hybrid learning are quite unstructured and their lines cross very often. These two methods have higher gradient and slew rate, which suggests they use more of their freedom. Still, we have to look at the reconstruction results whether these wilder patterns lead to better results.

## 2.4 General sampling theory

For Cartesian sampling the most important theorem is the Shannon-Nyquist sampling theorem. It gives a necessary sampling rate for optimal reconstruction. It states the following [34]:

**Theorem 1** *If a function  $f(t)$  contains no frequencies higher than  $W$  cps (Hertz), it is completely determined by giving its ordinates at a series of points spaced  $\frac{1}{2W}$  seconds apart.*

This theorem gives a sufficient sampling rate to reconstruct a function with bounded frequency. This minimum sampling rate is twice the maximum frequency  $r_n = 2W$ . Subsampling below this rate can lead to aliasing effects. Other techniques must be employed to be able to subsample this much and still get a good reconstruction.

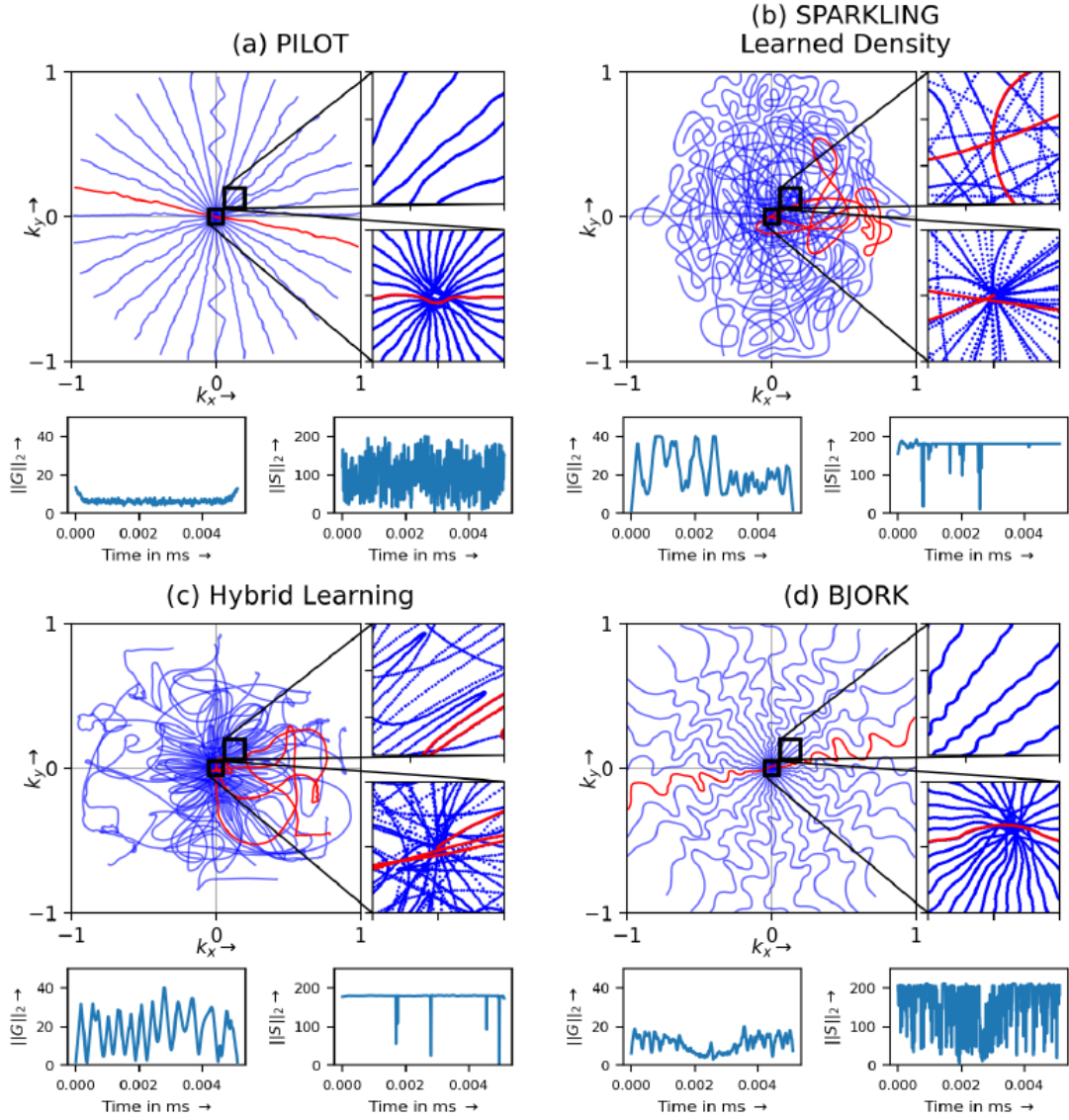


Figure 4: Shows the sampling patterns for PILOT, SPARKLING, hybrid learning method and BJORK in subfigures a-d. Also shows for each pattern the gradient amplitude  $\|G\|_2$  and slew rate amplitude  $\|S\|_2$ . Figure taken from [29]

In this section, we discuss two major subsampling methods that were introduced to speed up MRI. The first is parallel imaging which was introduced in the 90s. Parallel imaging uses multiple receiver coils to measure slightly different views of the imaged volume. These different views are the result of each coil having a different coil sensitivity, which specifies how strongly the coil measures each part of the image more or less strongly. This means we can represent the k-space data of coil  $i$  denoted  $v_i$  as the Fourier transform of the image  $u$  multiplied pointwise by a complex-valued position dependent coil sensitivity map  $S_i$  [49]:

$$v_i = F(S_i u) + \varepsilon \quad (6)$$

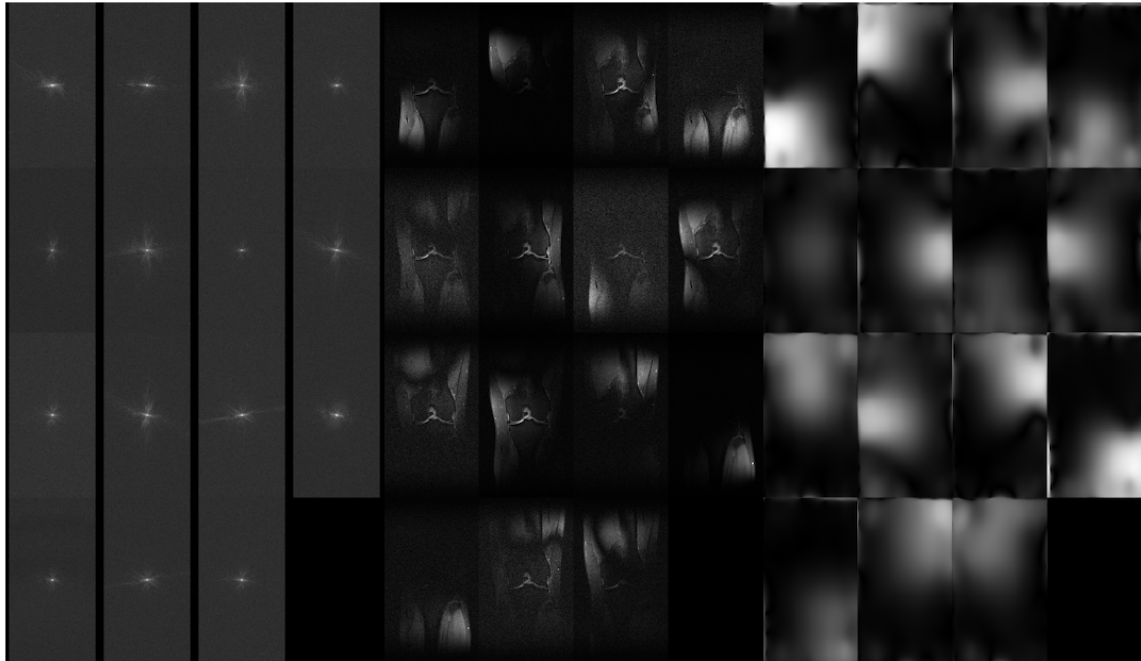
Figure 5 visualizes what happens in equation 6. Figure 5a shows the k-space data  $v_i$  corresponding to each of the 15 coils. Next, Figure 5c shows the estimated sensitivity maps, which were estimated using the ESPIRiT method [41]. Finally, Figure 5b shows the quantity  $S_i m$  which is the part of the image ‘seen’ by coil  $i$  for all coils, also called the individual coil images. The collection of k-spaced data corresponding to each coil is used to get the final reconstruction. Intuitively this makes sense as putting all the individual coil images together results in the full image. Parallel imaging is used to accelerate MR imaging by subsampling each coil’s k-space. Because many coils are used this does not lead to lower quality resolutions, as long as the total number of k-space measurements for all coils is larger than the total number of voxels in the image.

The second important subsampling method is compressed sensing. Compressed sensing is a general methodology which can be used to recover a signal based on fewer measurements by exploiting the sparsity of the signal. The sparsity of the signal reduces the number of unknowns that need to be determined, which decreases the number of measurements needed. Applying CS to MRI was first done by Lustig et al [24]. It achieves this reduction in scanner time by exploiting the inherent structure of MR images. The information needed to represent an image can be reduced by compressing the signal. This is done by applying some transform to the image, such that the signal in the transform domain admits a sparse representation. This sparse representation contains the most crucial information of the image. This means that by taking measurements in a subset of k-space that correspond to this compressed information, we need fewer measurements and are still able to reconstruct the image.

In general, to apply compressed sensing the reconstruction problem must satisfy three conditions:

- The signal must be compressible. This means it must allow a sparse encoding in some transformation domain. MRI satisfies this by for example applying a wavelet transformation to the image.
- The undersampling artifacts must be incoherent. This means that the artifacts created by undersampling the k-space should behave like noise in the sparsifying transformation domain.
- The image should be reconstructed by an algorithm that promotes consistency with the measurements and sparsity in the transformation domain. This is possible with MRI reconstruction algorithms.

The second condition is not trivially true and must be enforced by the sampling strategy. This condition must not be pushed too far because a completely random sampling strategy would give very low coherence but is not useful in practice. The coherence condition does not take into account the energy distribution of MR images in k-space, which is more dense close to the center. So the optimal sampling strategy should conform with this variable density. Also it is very difficult to actually perform a completely random sampling pattern because you need rapid switching of the gradients. This results in eddy currents and artifacts.



(a) k-space data from 15 coils      (b) Individual coil spatial images from fully sampled data      (c) Coil sensitivity map magnitudes given by ESPIRiT

Figure 5: Demonstration of parallel imaging with 15 coils. Subfigure a shows the k-space for each coil separately. Subfigure c shows the sensitivity map corresponding to each coil. Subfigure b shows the individual reconstructions based on the sensitivity map and k-space data. Figure taken from [49]

Incoherence is measured as follows: suppose we sample a subspace  $S$  of the k-space. By  $F_S$  we denote the Fourier transform evaluated only at frequencies in  $S$ . Define the point spread function  $PSF(i, j) = (F_S^* F_S)(i, j)$ . Cartesian sampling gives a point spread function equal to the identity mapping. Subsampling results in non-zero off diagonal entries. The point spread function indicates interference between pixel  $i$  and  $j$ . This leads to blurring or aliasing in the reconstructed image. The coherence is defined as the largest off diagonal entry of the normalized PSF. The PSF should be adapted to the transformed PSF if the signal is sparse in some transformed domain. So this point spread function can be used to find a sampling trajectory with low coherence.

The way compressed sensing is used in MRI differs quite a lot from how it was proposed in theory [8]. This is mainly because a cloud of sampling points might work well in theory but is difficult to sample in practice. As discussed before, sampling patterns need to be continuous trajectories, which need to satisfy gradient and slew rate constraints to be physically feasible. This is the main drawback of compressed sensing and shows a discrepancy between what was conceived theoretically and can actually be done practically. Researchers have attempted to implement compressed sensing MRI in various ways such as radial patterns, spiral patterns, noisy spiral patterns and Poisson disc sampling. These patterns are easy to implement and consist of lines which are physically feasible for the MRI scanner.

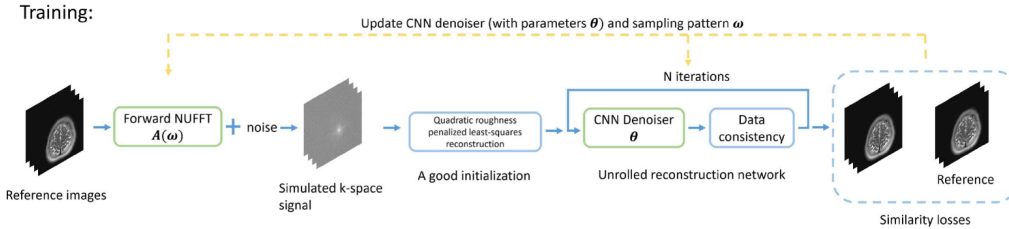


Figure 6: Shows the BJORK training loop. The training takes as input a batch of reference images. To these images the forward NUFFT is applied to obtain the corresponding simulated k-space signals. An initial reconstruction is calculated based on the quadratic roughness penalized least-squares reconstruction of these k-space signals. Next, during  $N$  iterations, a CNN denoiser and data consistency term in the form of conjugate gradient algorithm are applied to this initial reconstruction. The final reconstruction is compared to the reference images and a loss is calculated. By backpropagating the loss, the parameters of the CNN denoiser  $\theta$  and sampling pattern  $\omega$  are updated. Figure taken from [43]

### 3 Method

#### 3.1 BJORK problem formulation

As a starting method for joint optimization of sampling pattern and reconstruction algorithm we used BJORK [43]. The idea of BJORK is to optimize the k-space sampling trajectory and image reconstruction jointly in a supervised manner. BJORK stands for B-spline parameterized Joint Optimization of Reconstruction and k-space trajectory. The B-spline parameterization helps to make the optimization easier, by reducing the degrees of freedom compared to optimizing the sampling points directly. The BJORK training pipeline is shown in Figure 6.

We want to optimize  $\omega \in \mathbb{R}^{N_h \times N_s \times N_d}$ , a sampling pattern and  $\theta \in \mathbb{R}^M$ , the  $M$  parameters of the trainable reconstruction model. Here  $N_h$  is the number of lines, which are called shots,  $N_s$  the number of samples per shot and  $N_d$  the dimensionality of the image, which we assume to be  $N_d = 2$ . The reconstruction parameters  $\theta$  are the parameters of a neural network  $f_\theta$ .

The training set  $X$  consists of training samples  $\bar{x} \in \mathbb{C}^{N_v}$ , which are fully sampled reference images with  $N_v$  voxels. Based on these images, the k-space data is retrospectively sampled by applying the forward operator  $A(\omega)$  to this image, such that  $y = A(\omega)\bar{x}$ . Here  $A(\omega) \in \mathbb{C}^{N_s N_c \times N_v}$  is the forward operator, where  $N_c$  is the number of coils used to measure k-space data. In the case of simulated data there are no actual coils that measure data, but we simulate them. In the case of multicoil data, i.e.  $N_c > 1$ , this operator is a SENSE operator [27], which uses the sensitivity maps of the image, multiplied pointwise by a non-uniform Fourier transform (NUFFT) which takes the trajectory  $\omega$  as input. Each coil has its own sensitivity map which is usually estimated from a small part of k-space called the autocalibration (AC) region. We obtain the sensitivity mappings from the fully sampled k-space data using the ESPIRiT method [41].

The reconstruction network  $f_\theta$  takes as input the k-space data  $y$  and outputs a reconstructed image  $\hat{x} = f_\theta(y)$ . The goal is to get a good sampling pattern  $\omega$  and a good reconstruction algorithm  $f_\theta$  such that the reconstructed image  $\hat{x}$  is on average close to the ground truth image  $x$  for  $x$  taken from a training set  $\mathcal{X}$ . The similarity between  $\hat{x}$  and  $\bar{x}$  is measured by a loss function  $l$ , which the authors of BJORK take as a combined L1 and L2 loss:  $l(\hat{x}, \bar{x}) = \|\bar{x} - \hat{x}\|_2^2 + \|\bar{x} - \hat{x}\|_1$

To ensure the learned sampling pattern  $\omega$  is physically feasible, i.e. can be performed by an



actual MRI scanner, two constraints need to be enforced: the maximal gradient and maximal slew rate. The optimization problem with these constraints looks as follows:

$$\arg \min_{\omega \in \mathbb{R}^{N_s \times N_d}, \theta \in \mathbb{R}^M} \mathbb{E}_{x \in \mathcal{X}} [l(f_\theta(\omega; A(\omega)x + \varepsilon), x)]$$

subject to:

$$\begin{aligned} \|D_1 \omega^d\|_\infty &\leq \gamma \Delta t g_{max} \\ \|D_2 \omega^d\|_\infty &\leq \gamma \Delta t^2 s_{max}, d = 1, 2 \end{aligned}$$

In the constraints,  $D_1$  and  $D_2$  denote the first and second-order derivatives.  $\Delta t$  is the raster time and  $\gamma$  the gyromagnetic ratio. For multiple shots, the difference operator is applied for each shot separately. The constraints are included in the optimization function by using a penalty function:  $\phi_\lambda(|x|) = 1^T \max.(|x| - \lambda, 0)$  so the extra term is:

$$\mu_1 \phi_{\gamma \Delta t g_{max}}(|D_1 \omega|) + \mu_2 \phi_{\gamma \Delta t^2 s_{max}}(|D_2 \omega|) \quad (7)$$

Here the norm is taken with respect to  $N_d$ , so we can write this as:

$$\begin{aligned} &\mu_1 \sum_{i=1}^{N_s} \max(|D_1 \omega_i| - \gamma \Delta t g_{max}) \\ &= \mu_1 \sum_{i=1}^{N_s} \max(\sqrt{(D_1 \omega_i^x)^2 + (D_1 \omega_i^y)^2} - \gamma \Delta t g_{max}) \\ &= \mu_1 \sum_{i=1}^{N_s} \max(\sqrt{(\omega_i^x - \omega_{i-1}^x)^2 + (\omega_i^y - \omega_{i-1}^y)^2} - \gamma \Delta t g_{max}) \end{aligned}$$

and similarly for the second term.

### 3.2 Calculating the forward operator

The forward operator can be seen as a sampling operator  $A$  that maps the image  $x$  to the k-space samples  $y = A(x) = (\hat{x}(s(j\Delta t)))_{0 \leq j \leq m-1}$ . This operator maps a complex  $n$ -dimensional vector to a complex  $m$ -dimensional vector. We assume the image  $x$  can be decomposed as:  $x = h * v$ ,  $h$  being an interpolation kernel and  $v = \sum_{0 \leq i_1, i_2 \leq \sqrt{n}-1} v_{i_1, i_2} \cdot \frac{\delta(i_1, i_2)}{\sqrt{n}}$  being the interpolation coefficients with pixel coordinates  $i_1$  and  $i_2$ . Using this representation for  $x$  the k-space data can be expressed as:

$$\hat{x}(\xi) = \hat{h}(\xi) \cdot \left( \sum_{0 \leq i_1, i_2 \leq \sqrt{n}-1} v_i \cdot \exp\left(-\frac{2i\pi}{\sqrt{n}} \langle \xi, I \rangle\right) \right)$$

where  $I = (i_1, i_2)$  and  $\xi$  is the two dimensional frequency input. Usually  $h = \delta$  is taken, which is reasonable when working with discretized images. Here the k-space locations lie on the trajectories specified by  $s$ , so  $\xi \in \{s(j\Delta t), 0 \leq j \leq m-1\}$ . To compute  $\hat{x}$  naively would take  $O(mn)$  steps. If the sample locations lie on a Cartesian grid the fast Fourier Transform can be used which takes  $O(n \log n)$  operations. In general the sampling locations are non-Cartesian and the Non Uniform

Fast Fourier transform (NUFFT) needs to be applied. Here we describe the NUFFT method by Fessler [17]. The Fourier transform of a one dimensional signal  $x_n$  is defined as:

$$X(\omega) = \sum_{n=0}^{N-1} x_n e^{-i\omega n}$$

The frequency locations are non-uniformly spaced locations  $\omega_m$ .

$$X_m := X(\omega_m) = \sum_{n=0}^{N-1} x_n e^{-i\omega_m n}$$

As a first step we perform an oversampled FFT, where  $K \geq N$ :

$$Y_k = \sum_{n=0}^{N-1} s_n x_n e^{-2i\pi kn/K}$$

where  $s = (s_1, \dots, s_N)$  are scaling factors. This step takes  $O(K \log N)$  iterations. The next step is to approximate  $X_m$  by interpolating  $Y_k$  using the neighboring locations of  $\omega_m$ :

$$\hat{X}(\omega_m) = \sum_{k=0}^{K-1} v_{mk}^* Y_k = \langle Y, v_m \rangle, m = 1, \dots, M$$

Here  $v_{mk}$  denote the interpolation coefficients, which is a vector  $v_m$  for every k-space point  $m$ . This vector is restricted to have at most  $J$  nonzero elements for computational efficiency. These points are chosen as the  $J$  nearest neighbors of the frequency  $\omega_m$ .

The interpolation operator can be written as:

$$\hat{X}(\omega_m) = \sum_{j=1}^J Y_{k_m+j_K} u_j^*(\omega_m)$$

where  $u_j(\omega_m)$  denotes the  $J$  nonzero entries of the interpolation vector  $v_m$  and  $k_m$  denotes the integer offset, which makes sure the right  $Y$  point is chosen. This integer offset is defined as:

$$k_0(\omega) = \begin{cases} (\arg \min_{k \in \mathbb{Z}} |\omega - \gamma k|) - \frac{J+1}{2}, J \text{ odd} \\ (\max\{k \in \mathbb{Z} : \omega \geq \gamma k\}) - \frac{J}{2}, J \text{ even} \end{cases}$$

The interpolation vector is chosen as:

$$u(\omega) = \Lambda'(\omega)[C' S S' C]^{-1} C' S b(\omega)$$

here we have a variety of symbols:

$$\Lambda_{jj}(\omega) = e^{-i[\omega - \gamma(k_0(\omega) + j)]\eta_0}$$

$$C_{nj} = \frac{e^{i\gamma j(n-\eta_0)}}{\sqrt{N}}$$

$$S = \text{diag}(s_n)$$

$$b_n(\omega) = \frac{e^{i(\omega - \gamma k_0(\omega))(n - \eta_0)}}{\sqrt{N}}$$

The above expressions need to be calculated at every frequency location  $\omega_m$ . To extend this one dimensional NUFFT to two dimensions is straightforward. With a 2d-NUFFT frequency space is two dimensional so the 2-D FFT needs to be oversampled in these two directions. Also the interpolator for each frequency location needs to be calculated using the nearest  $J \times J$  sample locations. The interpolator can be calculated by taking the Kronecker product of 1-D interpolators.

As discussed before, when dealing with multicoil data the sensitivity maps also need to be calculated, as they are part of the forward operator. This can be done with the ESPiRiT method [41]. This method uses a central region of k-space, also called the autocalibration region to estimate the sensitivity maps.

### 3.3 B-spline sampling layer

The sampling pattern is parameterized using second-order quadratic B-splines. This means the trajectory line is represented as a linear combination of time shifted quadratic functions, the basis functions [48]. For example, the  $x$  coordinate of one k-space trajectory line can be represented as:

$$k_x(t_j, c_x) = \sum_{l=1}^L c_{x,l} h_x(t_j - l\Delta t) \quad (8)$$

where  $t_j$  is time,  $\Delta t$  the raster time and the vector  $c_x$  are the  $L$  basis coefficients. These coefficients will be taken as parameters and optimized in BJORK. The function  $h_x$  are the quadratic basis functions. In matrix form this can be written as:

$$k_x(c_x) = H_x c_x \quad (9)$$

where the matrix  $H_x$  has as columns the time shifted basis functions. These time shifted B-spline basis functions are shown in Figure 7 on the left. The other two plots show the derivative and second derivative of the quadratic B-spline kernel, which are linear and piecewise constant respectively. These basis function make it easier to check the constraints, because the first and second regular have a very regular shape [20]. Normally we would have to check for all  $N_t$  timepoints whether the gradient and slew rate exceed their limit. The second derivative consists of rectangular functions, which have one maximum. Therefore, we only need to check each rectangular function, instead of all time points. Likewise, the gradient only attains a maximum when the second derivative zero-crosses, which is also less than all timepoints. This property only holds for quadratic B-spline and not for higher order B-splines.

Additionally, this approach reduces the number of parameters needed to represent the trajectory. Mathematically, the sampling  $\omega$  is parameterized as  $\omega = Bc$  where  $B \in \mathbb{R}^{N_s \times L}$  is the interpolation matrix and  $c \in \mathbb{R}^{L \times N_d}$  are the coefficients. Here  $N_s$  is the number of k-space samples,  $N_d$  the number of dimensions and  $L$  is the number of interpolation kernels. The number of kernels is determined by a parameter called the decimation rate, which is defined as the ratio:  $decim = \frac{N_s}{L}$ . This parameter denotes how many k-space points are controlled by one B-spline kernel. The algorithm starts with a decimation rate of 32, which means many points are controlled by the same kernel. This makes the optimization act globally. After optimizing the sample trajectory and reconstruction algorithm the decimation rate is halved and again the trajectory and reconstruction are optimized. This lets the algorithm perform more local adjustments, since there are more B-spline kernels controlling a fewer number of points. This is done for a number of steps. This is a form of multi level optimization and it has been shown to help to escape sub optimal local minima.

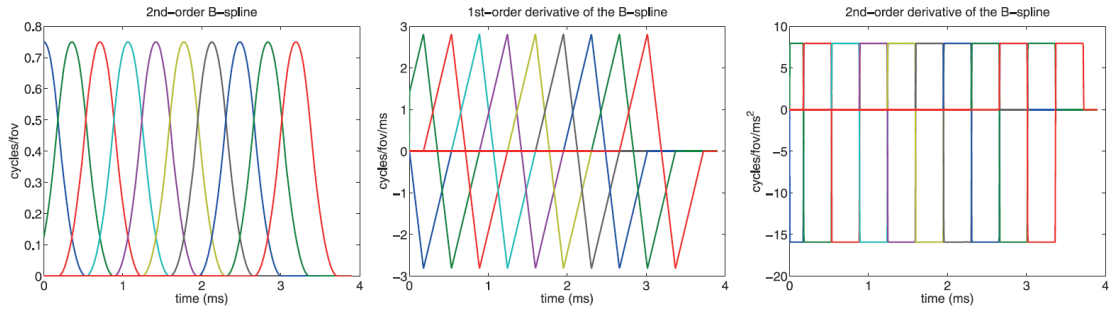


Figure 7: B-spline function basis and 1st and second derivative [20]

### 3.4 SIREN sampling layer

We want to represent a sampling pattern using the SIREN sampling layer, similar to how the B-splines could represent it. It is called the SIREN sampling layer because it uses the SIREN network [36]. The entire sampling layer that partly uses this network is referred to as the SIREN sampling layer. We mention three reasons why using a SIREN sampling layer might be useful. First, the SIREN sampling layer is continuous, which means it can produce sampling lines of arbitrary resolution. In BJORK, the interpolation matrix  $B$  is determined based on a fixed number of points per shot. The SIREN sampling layer parameterizes continuous lines, so an arbitrary number of points can be generated. Secondly, the gradient and slew rate can be calculated analytically from the SIREN sampling layer instead of having to rely on finite difference calculation in BJORK, making it more accurate. Thirdly, the SIREN sampling layer has the potential to apply multi-level optimization, without actually increasing the number of parameters. In BJORK, one can add more B-spline kernels, which increases the number of control points and makes the optimization more local. In this way it can move from less kernels and global optimization to more kernels and local optimization. The SIREN can do this by increasing a parameter called  $\omega$  which increases the periodicity of the output of the SIREN sampling layer. This does not increase the number of parameters, but does allow for more local changes in the pattern. So by gradually increasing the  $\omega$  parameter, it can mimic BJORK’s multilevel optimization strategy.

To implement the SIREN sampling layer, we use the pipeline shown in Figure 8. The SIREN sampling layer calculates for a given shot number and sample number the corresponding  $x$  and  $y$  coordinates of the sampling point. The shot number indicates which line is selected and the sample number indicates which point on this line is selected. The input coordinates are normalized between 0 and 1 to make it easier for the network to train. The network is a SIREN [36], that is a fully connected neural network with sinusoidal activation functions. This network outputs  $x$  and  $y$  coordinates and is followed by a rotation matrix. This rotates the given vector by an angle  $\phi$ , which depends on the particular shot number. In this way the SIREN does not have to learn the angle of each line, only how far it deviates from a predefined angle. This leads to faster training and makes it less likely for lines to cross or move into a completely different part of k-space, away from the predefined angle. The entire sampling pattern is obtained by inputting a batch of all shot numbers and all corresponding sample numbers.

We now discuss the SIREN architecture in detail. The SIREN is a fully connected neural network with periodic activation function. Other methods have used ReLU based neural networks, but these fail to represent fine details and have difficulty representing the derivative of signals well.

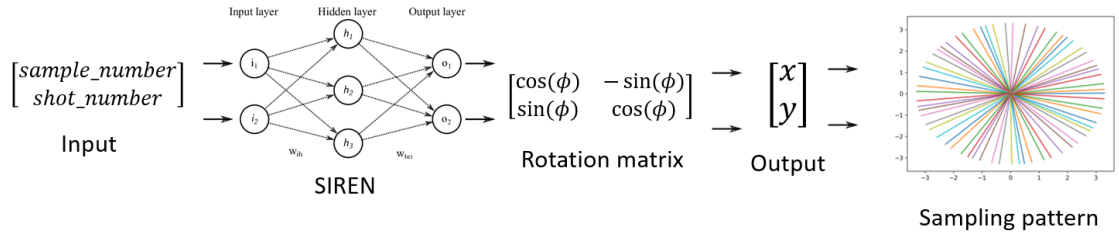


Figure 8: Shows the SIREN sampling layer. The input consisting of normalized sample and shot number are fed into the SIREN network, which outputs a two dimensional vector. This vector is rotated by the angle  $\phi$  by applying the rotation matrix. The output is the sampling coordinate corresponding to the specific line and point on the line that were used as input. By combining all inputs ranging over the number of lines and number of points per line, we can generate the entire sampling pattern as shown in the last step.

This happens because the second derivative of the ReLU function is zero, so the complexity of the representations is limited. Other nonlinear activation functions might have interesting derivatives but are shown to have undesirable training behavior and difficulty to represent details as well.

The output of a SIREN is simply a composition of sinusoidal functions:

$$\Phi(x) = W_n(\phi_{n-1} \circ \phi_{n-2} \circ \dots \circ \phi_0)(x) + b_n \quad (10)$$

where the  $i$ -th layer of the network  $\phi_i : \mathbb{R}^{M_i} \rightarrow \mathbb{R}^{N_i}$  is:

$$\phi_i(x_i) = \sin(W_i x_i + b_i) \quad (11)$$

Here the input vector  $x_i \in \mathbb{R}^{M_i}$  is linearly scaled by the weight matrix  $W_i \in \mathbb{R}^{N_i \times M_i}$  and translated by the bias vector  $b_i \in \mathbb{R}^{N_i}$ . Now we have described all parts of the pipeline in Figure 8. We introduce the following notation: the sample number is denoted as  $i_s$  and the shot number as  $i_n$ . The function which applies the SIREN network followed by the rotation matrix is denoted as  $S_c$ , where  $c$  denotes the parameters of the SIREN. To represent a pattern  $\omega$ , we can optimize the parameters  $c$  such that  $\omega_{x,y} = S_c(i_s, i_n)$ .

The SIREN architecture can be modified in various ways. First we can choose the number of layers and the number of features per layer. Secondly, we can choose how to initialise the weights. The chosen initialisation scheme for the weights is very important for effective training of the SIREN. The main point of the initialisation is keeping the distribution of activation values stable throughout the network, instead of depending on the number of layers used.

Consider a single sine neuron  $y = \sin(ax + b)$  where  $x$  is uniformly distributed from -1 to 1. Then it can be shown that its output has arcsin(-1,1) distribution as long as the function spans at least half a period, which is when  $a > \frac{\pi}{2}$ . Suppose we now look at a SIREN layer with  $n$  features and one output:  $y = \sin(w^T x + b)$ . Suppose this is not the first layer, then the input  $x$  is arcsin distributed. If we assume the weights are uniformly distributed,  $w_i \sim U(-\frac{c}{\sqrt{n}}, \frac{c}{\sqrt{n}})$  for some  $c \in \mathbb{R}$  then  $w^T x \sim N(0, \frac{c^2}{6})$ . Putting the normally distributed variable in the sinus function gives an output that is again arcsin distributed as long as  $c > \sqrt{6}$ . Sitzmann et al. [36] propose to take  $c = \sqrt{6}$  such that  $w^T x$  has standard normal distribution. This standard normal distribution ensures that  $|w^T x|$  rarely exceeds  $\pi$ , which means the frequency throughout the SIREN grows slowly. For the first layer a new parameter  $\Omega_0$  is introduced to make sure the initial output  $y = \sin(\Omega_0 W x + b)$

spans multiple periods over  $[-1, 1]$ . Additionally, we also scale the intermediate layers input with some parameter  $\Omega$ , such that the  $i$ -th layer of the network is:

$$\phi_i(x_i) = \sin(\Omega W_i x_i + b_i) \quad (12)$$

The gradient and slew rate can be calculated easily because the SIREN is differentiable. This is the case as it is a composition of sinusoids which are differentiable. By taking the derivative of the output with respect to the input sample number we obtain the gradient. By taking the derivative of the gradient w.r.t. the input we obtain the slew rate. In this way the gradient and slew rate are expressed analytically in terms of the network weights. This is preferable to finite differences which are used in BJORK, because this has much penalty terms which are more complicated to calculate.

For the SIREN sampling network, we try to imitate the multilevel optimization strategy that was part of BJORK. In BJORK at every level the number of coefficients is doubled, such that as the levels progress the sampling pattern is optimized at a finer scale. This is done by considering a strategy in which  $\Omega_0$  can increase over time. By increasing  $\Omega_0$  over time, we hope to increase the networks ability to represent lines with higher periodicity. This higher periodicity can account for more detail, so in this way it is similar to BJORK’s multilevel strategy. In particular, at every level we multiply  $\Omega_0$  by some parameter  $k_0$ . The appropriate value for  $k_0$  is determined in section 4.9.

### 3.5 Reconstruction network

A model-based unrolled neural network is chosen as the reconstruction algorithm. Typically, the MR image reconstruction problem consists of data consistency and regularization:

$$\hat{x} = \arg \min_x \|Ax - y\|_2^2 + \mathcal{R}(x)$$

Here  $y$  is the simulated k-space data. The goal of the regularizer is to reduce noise and aliasing artifacts. To unroll this equation we introduce an auxiliary variable  $z$  and a positive penalty parameter  $\mu > 0$ :

$$\hat{x} = \arg \min_x \min_z \|Ax - y\|_2^2 + \mathcal{R}(z) + \mu \|x - z\|_2^2$$

We can solve this problem using an alternating minimization approach:

$$x_{i+1} = \arg \min_x \|Ax - y\|_2^2 + \mu \|x - z_i\|_2^2$$

$$z_{i+1} = \arg \min_z \mathcal{R}(z) + \mu \|x_{i+1} - z\|_2^2$$

The  $x$  update can be found by solving the following equation with the conjugate gradient approach:

$$(A'A + \mu I)x_{i+1} = A'y + \mu z_i$$

The update for  $z$  works as a proximal operator. We substitute this with a CNN-based denoiser  $z_{i+1} = D_\theta(x_{i+1})$ . As a denoiser, the Deep Iterative Down-Up CNN is used. This method is similar to a U-net but is more data efficient. As an initial point for the conjugate gradient method, we need a good initial estimate. To get this estimate, the quadratic roughness penalty approach is used:

$$x_0 = \arg \min_x \|Ax - y\|_2^2 + \lambda \|Rx\|$$

here  $R$  denotes the 2-dimensional finite difference operator. This minimization problem is again solved with the conjugate gradient method:

$$x_0 = (A'A + \lambda R'R)^{-1} A'y$$

As an initial point for this CG algorithm the adjoint reconstruction is used  $x'_0 = A'y$ . The conjugate gradient algorithm works as follows [26]: we want to solve an equation of the form  $Ax = b$ , where  $A$  is a symmetric, positive definite matrix. Solving this equation is equivalent to minimizing the strictly convex quadratic function:

$$q(x) = -b^T x + \frac{1}{2}x^T Ax$$

The method works by minimizing the residual  $r = b - Ax$ , which is equal to the gradient of  $q$  wrt  $x$ . This gradient is zero, i.e. the residual is zero if  $Ax = b$ . The method starts by choosing some initial  $x_0$ . The initial residual is:  $r_0 = Ax_0 - b$  and initial direction  $d_0 = -r_0$ . Every iteration  $i$ :

$$\alpha_i = \frac{-r_i^T d_i}{d_i^T A d_i}$$

$$x_{i+1} = x_i + \alpha_i d_i$$

$$r_{i+1} = Ax_{i+1} - b$$

$$\beta_i = \frac{\|r_{i+1}\|}{\|r_i\|}$$

$$d_{i+1} = -r_{i+1} + \beta_i d_i$$

This process continues until either the residual is small enough or some predefined maximum number of iterations is reached.

### 3.6 Complete algorithm and dataset

The complete algorithm is given in algorithm 1. It contains a total of four nested for loops. The outer loop loops over the four levels. At each level the number of B-spline kernels is doubled. This allows the algorithm to optimize the sampling pattern more locally at each level, the multi-level optimization. For each level there are multiple epochs. For each epoch, mini-batches of MR images are taken from the training set. These images are used for the optimization pipeline as explained in this chapter. The most inner loop are the iterations of the unrolled reconstruction network.

For training, we use the FastMRI knee from NYU school of medicine and Meta [21]. This is one of the largest public datasets containing MR images with k-space data. The dataset was introduced to stimulate research in accelerated MRI, specifically using deep learning methods.

The fastMRI dataset consists of images of three anatomies: knees, brains and prostates. For the knee and brain anatomies there are four types of data: the raw multi-coil k-space data, the emulated single-coil k-space data, the ground truth images and the DICOM images, which we do not use. Using this data two types of task can be performed: the single-coil and multi-coil reconstruction task. For the single coil task, ground truths are calculated based on the emulated single-coil k-space data, while for the multi-coil task the ground truth images from fully-sampled multi-coil acquisitions is used. For each task and anatomy, the data is split into training and validation subsets. There are also test and challenge subsets, but these are not relevant for us as undersampling masks are applied to that data, which are specific to the 2019 FastMRI challenge. Figure 9 shows two example images from the knee dataset, one with fat suppression and one without.

The entire knee dataset contains 1,594 scans which were either acquired on one of three clinical 3T system or one clinical 1.5T system. Data was acquired using a 15 channel knee coil array and conventional Cartesian 2D turbo spin echo (TSE) protocol. Two pulse sequences were used, giving coronal proton-density weighting with fat suppression for 798 scans and without fat suppression

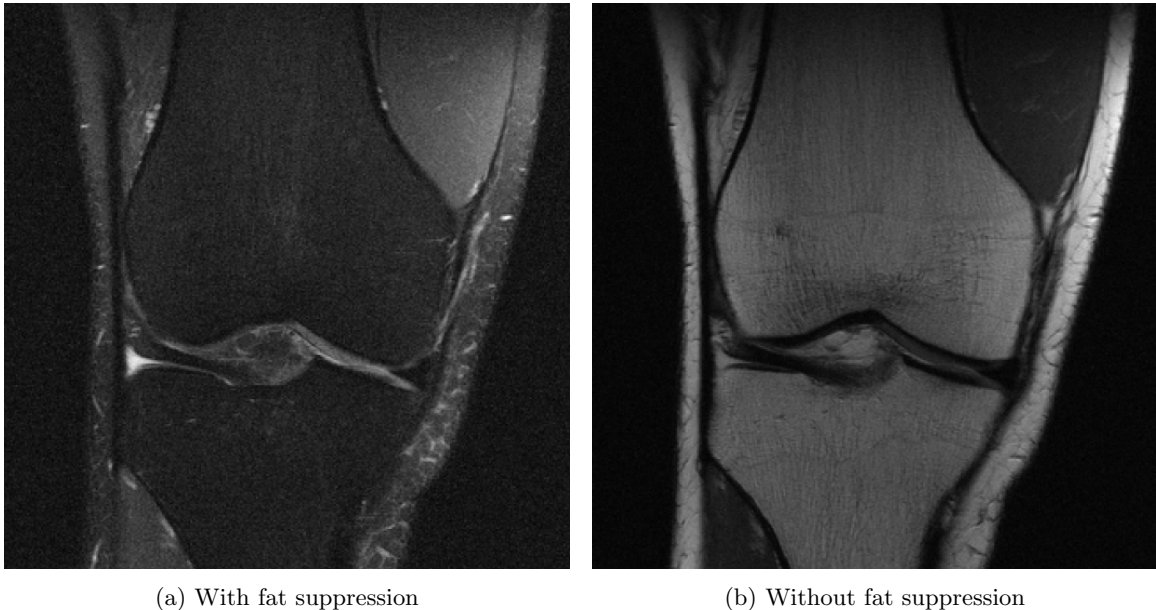


Figure 9: Two proton-density weighted images, one with fat suppression and one without. The fat suppression can help make details more visible, but it often increases noise

for 796 scans. Concerning the sequence parameters, the echo train length was 4, the matrix size  $320 \times 320$ , in-plane resolution  $0.5\text{mm} \times 0.5\text{mm}$ , slice thickness 3mm with no gaps between slices. The repetition time (TR) varies between 2200 and 3000 milliseconds. The echo time (TE) varied between 27 and 34 milliseconds.

The brain dataset contains 6970 scans, which were acquired using 11 magnets using 5 different clinical locations with 1.5T and 3T field strengths. The dataset contains axial T1 weighted, T2 weighted and FLAIR images.

It is important to consider in detail how the ground truth images were derived from the k-space measurements. The multi-coil ground truth was derived using the root-sum-of-squares approach, which is commonly used to combine information from multiple coils into one image. First, for each coil the inverse Fourier transform is applied to the individual coil data:

$$\tilde{m}_i = \mathcal{F}^{-1}(y_i) \quad (13)$$

This calculates the  $i$ th coil image from the k-space data  $y_i$ . The RSS reconstruction is then calculated from the  $n_c$  coils:

$$\tilde{m}_{rss} = \left( \sum_{i=0}^{n_c} |\tilde{m}_i| \right)^{\frac{1}{2}} \quad (14)$$

For the knee dataset, the training set contains 973 volumes, which consist of 34,742 slices. The validation set contains 199 volumes, which consist of 7135 slices. This entire dataset is referred to as the large dataset. We construct a middle sized dataset, which contains 200 volumes with 7049 slices. We also construct a small dataset with 10 volumes and 332 slices.

To evaluate performance we use two metrics. First, we use the structural similarity index measure (SSIM). This measure gives a score between 0 and 1 for similarity, where higher is better. Secondly,



we use the peak signal to noise ratio (PSNR), which is defined for two images  $f$  and  $g$  as:

$$PSNR(f, g) = 10 \log_{10}(\max(f)^2 / MSE(f, g)) \quad (15)$$

The MSE denotes the mean squared error between the two images. The PSNR approaches infinity as the MSE goes to zero. Lower MSE is better, so a higher PSNR means more similarity. Both measures are generally accepted as good measures for image similarity.

---

**Algorithm 1** Training algorithm for BJORK/SIREN

---

**Require:** Training set  $X$ ; denoiser  $D_\theta$  for initial CNN weights  $\theta_0$ ; initial trajectory  $\omega_0$ ; levels of optimization  $N_{level}$ ; number of epoch  $N_{epoch}$ ; step size of denoiser update  $\eta_D$ ; step size of trajectory update  $\eta_\omega$ ; penalty parameter for gradient/slew rate constraint  $\mu_1$  and  $\mu_2$

**Ensure:**  $\omega = Bc$  (BJORK) or  $\omega = SIREN(sample, shot)$

$\theta \leftarrow \theta_0$   
 $\omega \leftarrow \omega_0$   
 Pre-train  $D_\theta$  with fixed  $\omega$   
**for**  $l = 1$  to  $N_{level}$  **do**  
   BJORK:  
   Initialize new coefficient matrix  $B_l$   
   Initialize new coefficient  $c_l^0$  with  $\omega_{l-1} \approx B_l c_l^0$   
   SIREN:  
   Fit SIREN network weights  $c$  such that  
    $\omega_{l-1} \approx S_c(i_s, i_h) \forall i_s \in (1, \dots, N_s) \forall i_l \in (1, \dots, N_h)$   
   **for**  $j = 1$  to  $N_{epoch}$  **do**  
     **for** training batches  $x^K$  in  $X$  **do**  
       Simulate the k-space w.r.t.  $\omega_l$ :  
        $y^K = A(\omega_l^K) x^K + \varepsilon$   
       Reconstruction with UNN  
       Reconstruct initial images using  $x_0 = (A'A + \lambda R'R)^{-1} A'y$   
       **for**  $i = 1$  to  $N_{iter}$  **do**  
          $x_{i+1}$ : UNN reconstruction update of  $z_i$  using:  
          $x_{i+1} = (A'A + \mu I)^{-1} (A'y + \mu z_i)$   
         apply CNN:  $z_{i+1} = D_\theta(x_{i+1})$   
       **end for**  
       Calculate loss function:  
        $L = l(x^K, x^K) + \mu_1 \phi_{\gamma \Delta t g_{max}}(|D_1 \omega_l^K|) + \mu_2 \phi_{\gamma \Delta t^2 s_{max}}(|D_2 \omega_l^K|)$   
       Update denoiser and trajectory:  
        $\theta^K = \theta^{K-1} - \eta_D \nabla_{\theta^{K-1}} L$   
        $\omega^K = \theta^{K-1} - \eta_\omega \nabla_{\omega^{K-1}} L$   
     **end for**  
**end for**  
**end for**

---

## 4 Experiments and results

This section is structured as follows. Sections 4.1 up to 4.7 contain results relating to BJORK and sections 4.8 to 4.9 results relating to SIREN. We start in subsection 4.1 by discussing how we implemented BJORK. This is relevant because full plug-and-play code for BJORK is not available. Instead there are parts of the BJORK training pipeline available online, which we used as a starting point for our implementation. Moreover, some parameter values are not mentioned in the original BJORK paper even though we need them. We mention the various choices for parameter values, where we tried to replicate the original method as best as possible. In section 4.2, we report the results of this ‘baseline’ version of BJORK. Here, we compare our results to the results in the original BJORK paper and mention possible reasons why the results differ. We also compare the baseline BJORK results to training only the reconstruction network, without optimizing the sampling trajectory. This is to evaluate the added value of optimizing the sampling trajectory in our baseline implementation. Next, we are interested in ways to improve the baseline BJORK implementation. In section 4.3 we investigate the influence of the learning rate of both the sampling layer and reconstruction network on performance. In section 4.4 we investigate the influence of the loss function on performance. Here we consider L1, L2 and SSIM loss functions as well as combinations of loss functions. In section 4.5, we test the added effect of training the reconstruction network on its own without training the sampling layer, referred to as warming up the reconstruction network. The last two sections on BJORK are not on improving performance but on testing and understanding the method in a different context. In section 4.6 we compare the performance when training on multi-coil data versus performance when training on single coil data. In section 4.7 we train BJORK on the brain dataset and test its performance on the knee dataset. After BJORK we discuss the results on the SIREN sampling layer. First, in section 4.8 we investigate whether the SIREN sampling layer can represent a trajectory found by BJORK. Next, in section 4.9 we look at the results of jointly optimizing the SIREN sampling layer and reconstruction network.

### 4.1 Implementation and general setup BJORK

To implement BJORK we used the repository <https://github.com/guanhuaw/Bjork> as a starting point. This repository provides a PyTorch-based implementation of BJORK [43], but it only contains the key components. These are the system operators and B-spline parameterization of the sampling trajectory. The system operators are the forward operator, that is, the NUFFT operator, the adjoint operator and the conjugate gradient method to compute the inverse operator. The implementation uses the MRI system matrix from the MIRTorch package [45]. The B-spline code can take as input a trajectory and calculate the fitted B-spline trajectory, that is, the coefficients and interpolation matrix. It can also calculate the gradient and slew rate of this pattern.

The repository does not contain a training loop, so we had to implement this ourselves. While the gradient and slew rate calculations were implemented, the penalty calculation was not. We implemented this penalty in the way that it was discussed in section 3.1. To obtain a good initialization, we had to compute the quadratic roughness penalty least squares reconstruction (QRPLSR). This is done by using the *Diff2dgram* operator from MIRTorch, which computes  $R'R$  where  $R$  is the finite difference operator. Next, we compute the operator  $A'A + \lambda R'R$  and use the conjugate gradient method to calculate the QRPLSR  $(A'A + \lambda R'R)^{-1}A'k$ , where  $A$  is the forward operator and  $k$  the simulated k-space. To initialize the CG algorithm we use the adjoint reconstruction  $A'k$ . The initial reconstruction is saved in a variable  $x_0$ . We apply the denoiser network  $D_\theta$  to obtain  $z_0 = D_\theta(x_0)$ , the denoised  $x_0$ . The denoiser architecture is a DIDN, the code for this network is taken from <https://github.com/guanhuaw/SNOPY> which is the code for SNOPY [44]. Next, for

$N_{iter}$  iterations we calculate:  $x_{i+1} = (A'A + \mu I)^{-1}(A'k + \mu z_i)$ ,  $z_{i+1} = D_\theta(x_{i+1})$ , where we use the *Gram\_inv* function from the BJORK repository and again the denoiser. The loss is calculated using the final reconstruction  $x_{N_{iter}}$  and the ground truth image. Finally, the loss is back-propagated and the weights are updated using the ADAM optimizer.

We chose the parameter values in the algorithm based on the values chosen by the authors of BJORK, whenever they were available. First, regarding the sampling pattern, as an initial sampling pattern we take a radial sampling pattern. For this pattern the number of shots is 32 and there are 1280 samples per shot. The initial decimation rate is 32, which means that 40 samples per shot are controlled by one B-spline kernel. The regularization parameters for the gradient and slew rate  $\mu_1$  and  $\mu_2$  are chosen as 1 and 0.01 respectively. These numbers were not specified in BJORK so it is unclear what values they used. We chose these numbers heuristically. The maximal gradient and maximal slew rate are 45 mT/m and 200 T/m/s respectively. These values were chosen to be compatible with the MR scanner in the TechMed Centre at the University of Twente.

Regarding the training, the learning rate for the sampling layer is 1e-3 and for the reconstruction network is 1e-5. The number of levels is chosen to be one. This partially defeats the purpose of the multi-level optimization strategy. However, this is done because most of the change in sampling pattern and also the greatest improvement in reconstruction quality occurs in the first level. Using only one level also saves a great deal of time. The number of epochs per level is equal to 3. We use 1 epoch to warm up the reconstruction network, so training without optimizing the trajectory. The training batch size is 4. The regularization parameter for the initial reconstruction  $\lambda$  is 0.1. The regularization parameter  $\mu$  is taken to be 10. For the CNN network we take 4 reconstruction blocks with 32 channels each. The conjugate gradient algorithm uses a maximum of 10 iterations to converge. The entire reconstruction network consists of 6 iterations of consecutive CG algorithm and CNN denoising.

## 4.2 Baseline BJORK results

In this section, we test the BJORK algorithm using the parameter settings described in the previous section. First, we perform an experiment where we only optimize the BJORK reconstruction network, so the sampling pattern stays the same radial pattern. We evaluate the model by calculating the average PSNR and SSIM on the entire validation set. The average PSNR is 35.1 and the average SSIM is 0.79. Second, we test the ‘baseline’ BJORK model, where we optimize both the reconstruction algorithm and the sampling pattern. Again we evaluate on the entire validation set. The average PSNR is 34.6 and the average SSIM is 0.82. These results are very close together and it shows that the sampling network only improves reconstruction marginally. As an example, Figure 10 shows a typical reconstruction for both cases. The BJORK reconstruction is slightly better than using only the reconstruction network, but this difference is too small to be noticeable.

To check whether our results make sense, we need to compare our results to the results reported in the original BJORK paper [43]. We want to check the sampling trajectory and the reconstruction quality, both quantitatively and qualitatively. First, we compare the trajectories. Figure 11 compares the sampling pattern we found to the sampling trajectory found in the original BJORK paper when optimizing on the knee dataset. Compared to our sampling pattern, this pattern looks very smooth and regular. It covers almost every part of k-space, with the center sampled more densely than the periphery. Our pattern looks similar but the lines do not cover k-space as well.

Next, we investigate how much the sampling pattern contributes to the reconstruction quality in the original BJORK paper. Important to consider here is that the BJORK paper only reports on reconstruction scores for the brain dataset, since this was their main training dataset. The results for the knee dataset are likely to be similar but this difference is important to keep in mind.

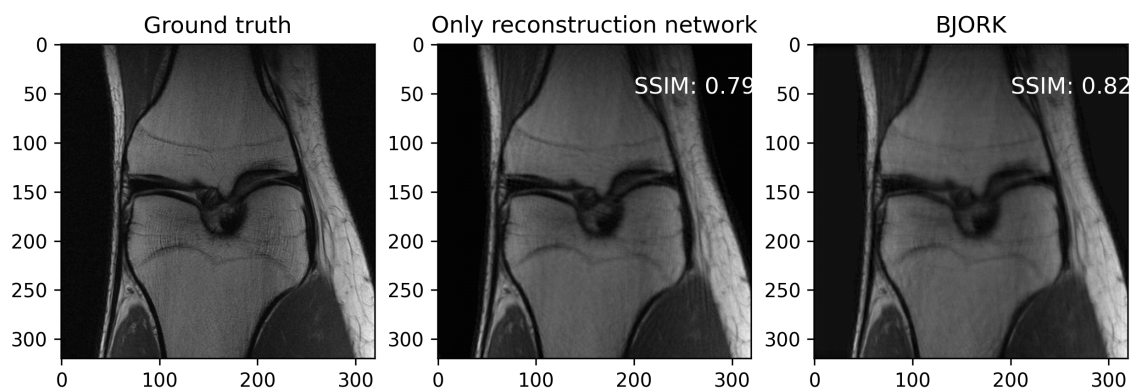


Figure 10: For one validation slice, we compare the reconstruction when optimizing only the reconstruction network (middle) to optimizing both reconstruction and sampling pattern (right). The ground truth is shown on the left.

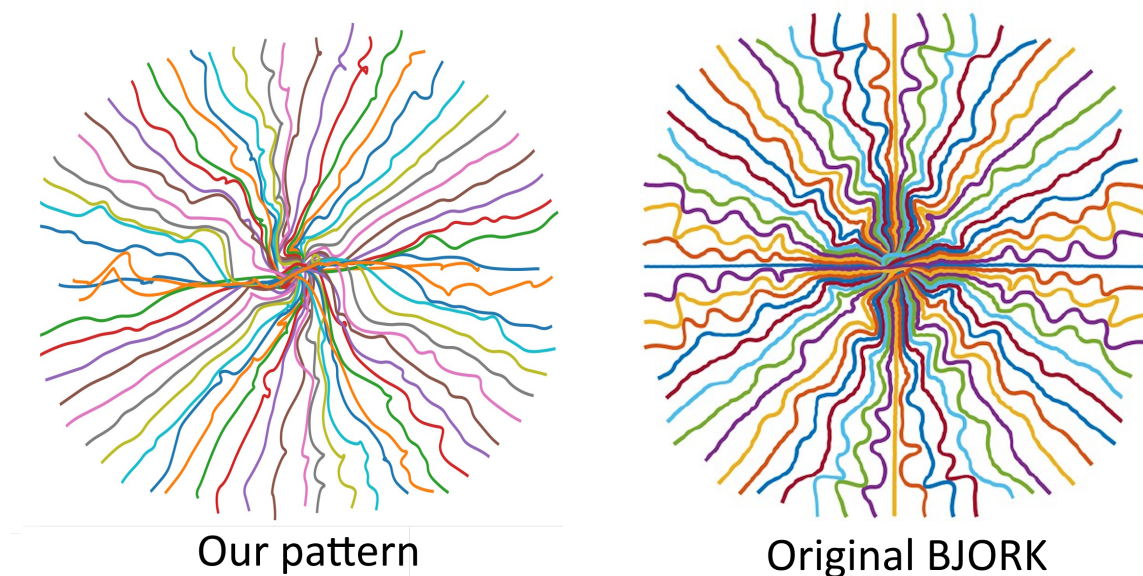


Figure 11: Our sampling pattern for the knee dataset compared to the sampling trajectory optimized on the fastMRI knee dataset reported in original BJORK paper [43]

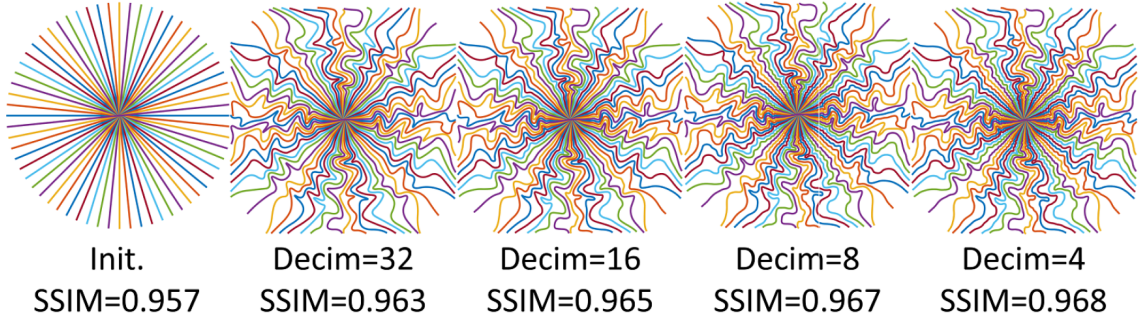


Figure 12: Figure showing how the sampling pattern changes during training in the original BJORK paper for the brain dataset. The pattern is initialized as a radial pattern showed first. The parameter ‘Decim’ denotes the ratio between the number of sampled per line, in this case 1280, and the number of B-spline kernels. So in the second plot there are 40 kernels, while in the last plot there are 160 kernels. More kernels means higher complexity so an increased computation time. The SSIM shown is the average reconstruction SSIM on the test set with the corresponding sampling pattern. Figure taken from [43].

Figure 12 shows how the pattern trained on the brain dataset develops during training and how the reconstruction quality improves. The most important insight is that the reconstruction quality is already very high for the initial un-optimized radial trajectory. The reconstruction quality does improve only marginally by optimizing the trajectory. This is in line with what we found in our results.

### 4.3 Determining the learning rate

#### Experimental setup

The learning rate is a very important parameter in machine learning. It determines the convergence behavior of the loss function. Setting the learning rate too low might lead to very slow convergence, while setting it too high might prevent the optimizer from finding a local minimum. This happens because the optimizer will take too large gradient descent steps, which leads to overshooting the minimum. Therefore, setting the appropriate learning rate is crucial to getting good results. One complicating factor is that we have to determine two learning rates. Moreover, since we are dealing with a bi-level optimization problem, the two optimization problems are related, so these learning rates cannot be chosen independently. For example, if we choose a higher learning rate for the sampling network, the learning rate of the reconstruction network might also have to be increased, because it must adapt to the rapidly changing sampling pattern. Therefore, many combinations of learning rates may need to be tried to get good convergence results. The appropriate learning rate likely also depends on the loss function that is being used. However, it would be too involved to optimize the learning rate and loss function simultaneously. Besides, other parameters such as number of epochs also play a role. Therefore, we first optimise the learning rate using a combination of L1 and L2 loss function, which was the loss function used in the original BJORK implementation. When we compare the different loss functions in section 4.3, we also validate whether the learning rates found in this section are correct for other loss functions as well.

To determine the right learning rate parameters we perform a sweep using Weights and Biases.

| Learning rate sampling | learning rate reconstruction | average SSIM | average PSNR |
|------------------------|------------------------------|--------------|--------------|
| 1e-4                   | 1e-5                         | 0.41         | 22.8         |
| 1e-4                   | 1e-4                         | 0.74         | 30.8         |
| 5e-4                   | 1e-5                         | 0.43         | 23.3         |
| 5e-4                   | 1e-4                         | 0.79         | 32.8         |
| 1e-3,                  | 1e-5                         | 0.68         | 30.4         |
| 1e-3                   | 1e-4                         | 0.78         | 30.3         |
| 5e-3                   | 1e-5                         | 0.69         | 27.3         |
| 5e-3                   | 1e-4                         | 0.72         | 28.3         |
| 1e-2                   | 1e-5                         | 0.74         | 30.2         |
| 1e-2                   | 1e-4                         | 0.79         | 31.6         |

Table 1: Table summarizing the sampling layer and reconstruction network learning rates sweep trained on the small sized dataset

Weights and Biases is an AI developer platform, which can track useful information from your training runs, such as loss function values and reconstructed images. During the sweep, runs are performed with a variety of learning rates. The learning rates for the sampling pattern are chosen from the range of values ( $1e-4, 5e-4, 1e-3, 5e-3, 1e-2$ ) and the reconstruction network learning rates from the range of values ( $1e-5, 1e-4$ ). We perform a grid sweep which means every combination of values is considered from these ranges. A total of 10 runs is performed using the small data set. This smaller data set is used because the sweep performs many runs, so with larger datasets it would take too long. Still, we expect to see notable differences on the small datasets as it contains a reasonable number of training slices. To judge the performance of the trained model we calculate the average SSIM and PSNR score on 400 (different) randomly selected validation slices from the validation set.

## Results

There are only 10 combinations of learning rates so we can summarize the results in a table. Table 1 shows the results. Looking at the reconstruction networks learning rate, we can see that the higher value  $1e-4$  outperforms  $1e-5$  both in terms of SSIM and PSNR. For the sampling layer learning rate the results are less conclusive. Overall, a higher learning rate for the sampling layer seems to be better, but the best performance is realized with  $5e-4$ . For this small dataset it seems to be better to have higher learning rates, because otherwise the algorithm does not have enough time to converge. One point of criticism is that a small dataset is used in this experiment and it is not clear to what extent these results will generalize to the larger datasets.

Therefore, we replicate the experiment but now for the middle sized dataset. The results are summarized in Table 2. These results are very different from the small dataset results. For the reconstruction network the value  $1e-5$  seems to perform better than  $1e-4$ , except for the sampling layer learning rate  $5e-4$ . For the sampling layer learning rate, the best values are for  $1e-3$  and lower. For these values the SSIM and PSNR are very close together. We stick with the combination  $1e-5$  and  $1e-3$  for the sampling layer and reconstruction learning rate respectively, since it was suggested by the paper and works comparably well.

| Learning rate sampling | learning rate reconstruction | average SSIM | average PSNR |
|------------------------|------------------------------|--------------|--------------|
| 1e-4,                  | 1e-5                         | 0.82         | 35.9         |
| 1e-4                   | 1e-4                         | 0.74         | 34.9         |
| 5e-4                   | 1e-5                         | 0.81         | 34.5         |
| 5e-4                   | 1e-4                         | 0.82         | 36.0         |
| 1e-3                   | 1e-5                         | 0.80         | 35.5         |
| 1e-3                   | 1e-4                         | 0.78         | 34.9         |
| 5e-3                   | 1e-5                         | 0.78         | 32.7         |
| 5e-3                   | 1e-4                         | 0.78         | 32.4         |
| 1e-2                   | 1e-5                         | 0.80         | 34.3         |
| 1e-2                   | 1e-4                         | 0.79         | 29.9         |

Table 2: Table summarizing the sampling layer and reconstruction network learning rates sweep performance trained on the middle sized dataset

## 4.4 Determining the loss terms

### Experimental setup

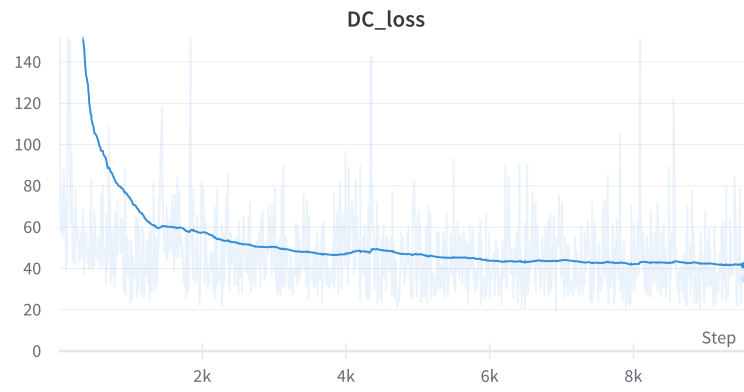
Using the information about learning rates from the previous section, we want to consider the loss function in this experiment. We consider L1 loss, L2 loss and SSIM loss. These loss terms are compared in three experiments, where we use the respective loss function in training. We still optimize both the sampling trajectory and reconstruction network. This is done because different loss functions might have different training behavior not only in terms of reconstruction, but also in terms of what kind of sampling trajectories are learned and how that again influences reconstruction performance. It is important to consider how the sampling problem is closely related to the reconstruction problem and therefore we should not disregard it, even though the loss function at first glance seems to be unrelated to the sampling problem. For training, the middle sized training set is used. This set is used because the number of experiments is limited and we want to see convergence behavior over a larger number of iterations with more training opportunity. We again evaluate the model performance by calculating the average SSIM on the entire validation set.

### Results

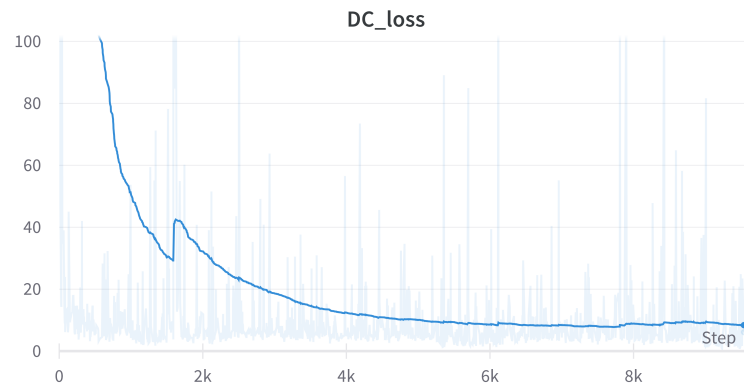
The average SSIM scores are 0.8, 0.81 and 0.78 for the L1, SSIM and L2 loss respectively. The average PSNR scores are 35.6, 35 and 35.1 for the L1, SSIM and L2 loss respectively. The results are quite close but L1 and SSIM are clearly better than L2 in terms of SSIM, while in terms of PSNR the L1 loss beats the other two by a small margin. We will now look at some aspects of the loss functions separately.

First the convergence of the loss functions shown in Figures 13a, 13b and 13c. The L1 loss function has the smoothest convergence, with the loss decreasing until the very end. The L2 loss is also relatively smooth, but in the end the loss function slightly increases again, suggesting slightly worse model performance. The SSIM convergence takes longer and has a more unusual path. Still it reaches a good performance and the loss curve suggests that more training might even lead to better results as it look likes the loss was still decreasing in the end. Figures 14 and 15 show example reconstruction on the validation set for the different loss terms.

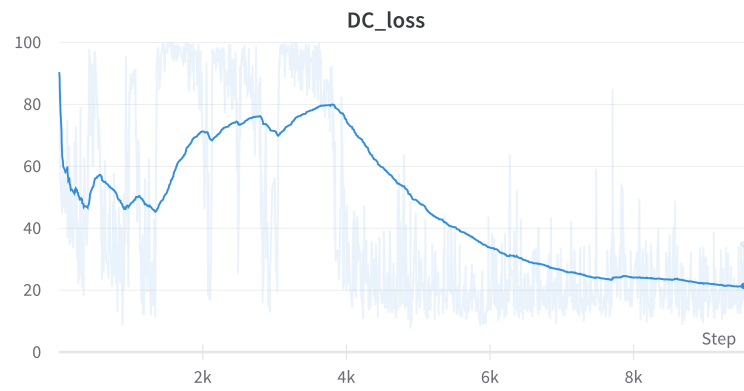
It is also possible to combine loss terms. Since L1 seems to be the best option we investigate the combination of L1 with SSIM and the combination of L1 with L2. How to weigh the two terms can



(a) L1



(b) L2



(c) SSIM

Figure 13: Comparison loss functions



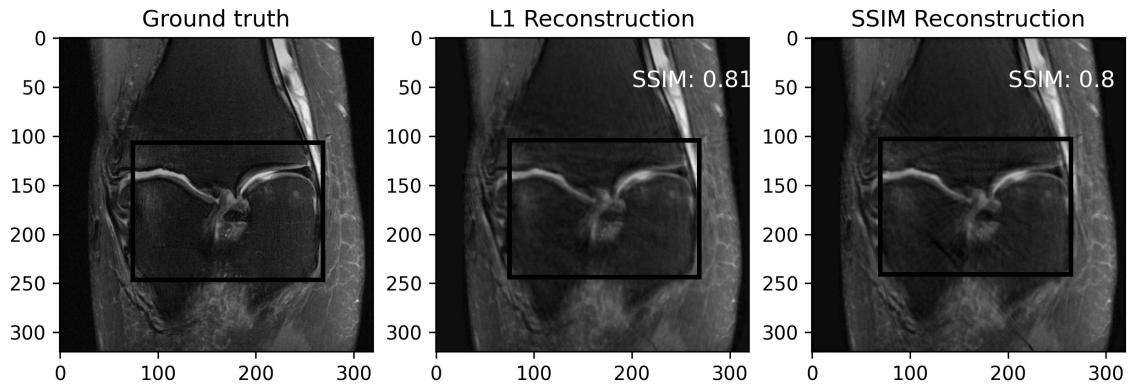


Figure 14: Comparison between L1 and SSIM reconstruction on one validation slice. The L1 reconstruction is slightly better than SSIM, but it is not noticeably different.

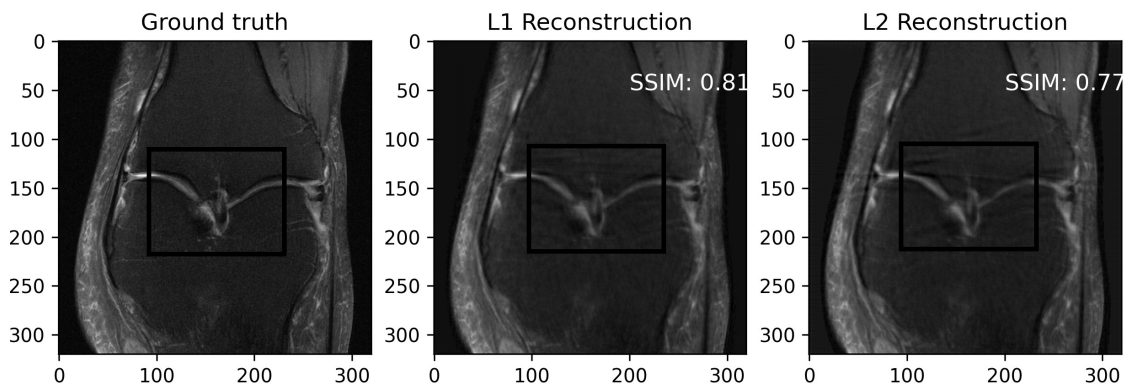
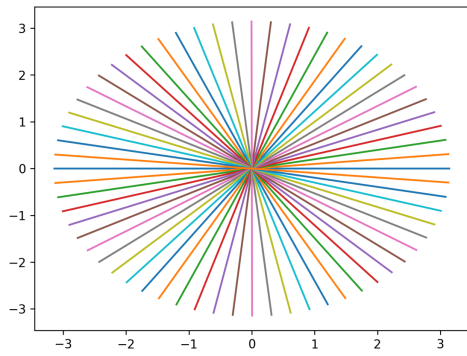
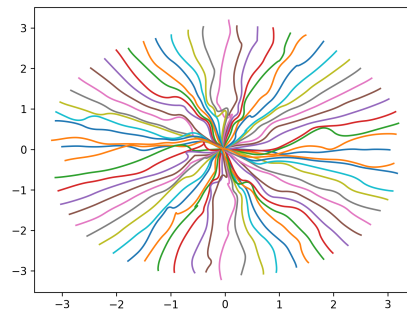


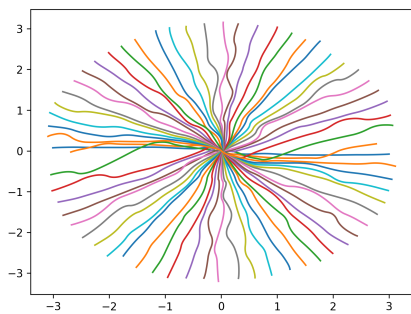
Figure 15: Comparison between L1 and L2 reconstruction. The L2 reconstruction has more streaking artifacts which can be seen in the middle of the image



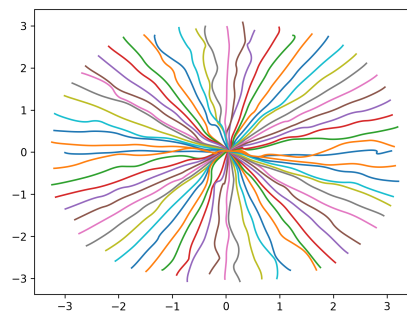
(a) Radial pattern



(b) L1



(c) L2



(d) SSIM

Figure 16: Comparison between sampling patterns which were trained using L1,L2 and SSIM loss functions. The radial pattern is shown for reference since all sampling patterns are initialized as radial pattern.

be optimized, but in this investigation we just picked a value that lets neither terms dominate the learning behavior.

Combining L1 and SSIM gives an average SSIM score on the validation (sub)set of 0.81 and an average PSNR of 35.4. In terms of SSIM, this is tied with the SSIM loss and it beats L1 and L2 loss. In terms of PSNR, it beats both SSIM and L2 loss, but is slightly worse than using only L1 loss.

Combining L1 and L2 gives an SSIM of 0.82 and an average PSNR of 34.5. This is better than using L1 and L2 loss separately in terms of SSIM but not in terms of PSNR, where it is worse than L1,L2 and SSIM loss as well as the combination of L1 and SSIM loss.

Overall, we give preference to combining L1 and SSIM loss, because it has both a high SSIM and high PSNR performance. Therefore, we continue to use this combination of loss terms in the following experiments.

## 4.5 Warm up reconstruction network

As mentioned in section 4.1, we perform one iteration of training to warm up the reconstruction network, without training the sampling network. We suspect this improves performance as it was suggested by the authors of BJORK. In this experiment we perform the simple test of leaving out the warming up phase of the reconstruction network and immediately go to optimizing reconstruction and sampling network simultaneously. The test is done with a combination of L1 and L2 loss and trained on the middle sized dataset. The result is an average SSIM of 0.8 on the entire validation set. This is slightly worse than the run with warm up which had an average SSIM of 0.82. In terms of PSNR the two runs have virtually the same performance. This shows there is utility in using the warm up, although the improvement in results is slight. Since warming up the reconstruction network takes only 1 extra iteration of training, the additional time is worthwhile and we continue to use the warm-up phase in the other experiments.

## 4.6 Training on multi coil compared to single coil data

In the previous experiments we simulated multicoil k-space data. Here we used the images which were reconstructed from multicoil Cartesian k-space data. This data is needed because we need to estimate the sensitivity maps from the 15 coils used to measure the data. Using multiple coils for MRI is called parallel imaging and has greatly improved image quality and decreased scan time [19]. Most modern MR scanners use multiple coils to measure the k-space data, but the older single coil scanners are still used in some places. Therefore, it is interesting to look at the single coil case as well. The FastMRI dataset also contains single coil data, which was not actually measured using one coil but simulated from multicoil data to act as if it was measured using only one coil. This is referred to as an emulated single coil (ESC) methodology. In the FastMRI the ESC method by Tygert et al. is used [40]. This method computes a linear combination of the k-space data from multiple coils, where the weights are optimized to match the ground truth reconstruction using least squares. We use this ESC data to perform the comparison between the two types of scans.

The single coil run was evaluated on 200 slices of the validation set. The average SSIM on this set was 0.58, which is significantly lower than 0.81 on the multicoil data. Figure 17 shows one example of a single coil reconstruction compared to the multi coil reconstruction. The general shape is clearly visible, but much of the detail is lacking. This shows multicoil gives much better reconstruction, which was expected. Still, the single coil reconstruction still has reasonable quality, considering how little data is available. These results can be compared to submissions to the 2019 fastMRI challenge for knee images. The best performing method in the single coil 8 times acceleration category was invertible Recurrent Inference Machine (i-RIM) [28]. They achieved an average validation SSIM

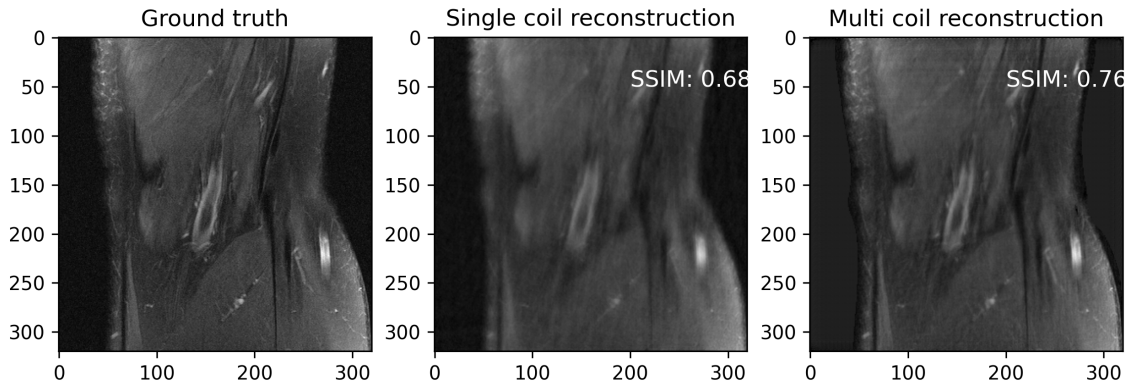


Figure 17: Example single coil and multi coil reconstruction

score of 0.69. This result is not completely comparable to our result, because we use 10 times acceleration and non Cartesian shots instead of Cartesian shots. Still, it indicates that achieving a good SSIM with high acceleration factor on single coil data is hard, which puts our score of 0.58 into context.

#### 4.7 Testing on datasets containing different anatomies

The previous experiments have exclusively been performed on the FastMRI knee dataset. However, as mentioned in section 3.6, we also have access to the FastMRI brain dataset. The first question that is interesting to consider is how BJORK performs on the brain dataset and what kind of patterns we obtain. The brain anatomy is very different from the knee anatomy, so we expect that different areas of k-space are more relevant, which is expected to lead to different sampling patterns. Secondly, it is interesting to consider how well BJORK performs on anatomies that were not considered during training. For example, suppose we train BJORK on the brain dataset, we can see how it performs when reconstructing knee images. In other words, we want to check how well BJORK generalizes to anatomies it has not seen yet. To perform this experiment, we train BJORK on the brain dataset using the same settings as for the knee dataset. We use a combination of L1 and SSIM loss. Because the brain dataset has scans with varying number of coils used, we use a batch size of 1 to prevent issues with batch stacking. The model is validated on 1000 random slices from the brain validation set.

The average SSIM on the brain validation dataset is 0.79. The learned sampling layer is shown in Figure 18. This sampling pattern differs notably from the knee sampling pattern. Here, the different batch sizes that were used have to be taken into account. Since we used a batch size of 1, the optimization only considers one slice at a time, which leads to less generalization over multiple slices compared to taking a batch size of 4. This probably lead to the more erratic pattern compared to the knee sampling pattern. Besides, we expected differences between the brain and knee sampling pattern as they are different anatomies and this can be seen when comparing the Figures. Using the learned sampling pattern and reconstruction network we trained on the brain dataset, we reconstructed images from the knee validation dataset. This led to an average SSIM score of 0.67 and an average PSNR of 32. This is significantly worse than when the knee dataset was used for training. Figure 19 shows an example of a knee MR images which is reconstructed using the training

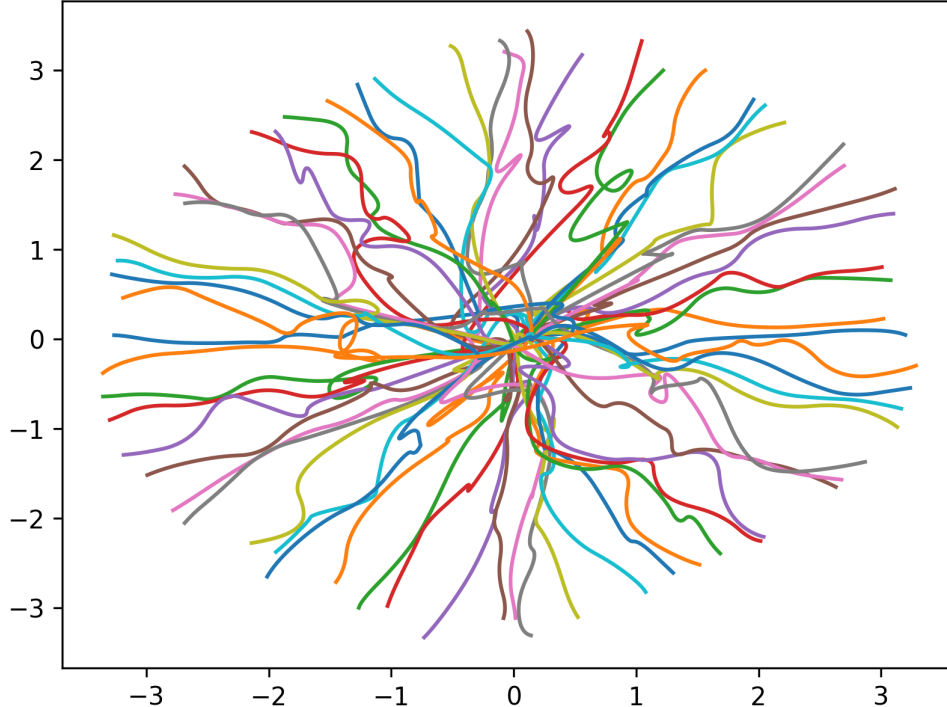


Figure 18: Sampling pattern obtained by training BJORK on the brain dataset. This pattern differs substantially from the pattern in Figure 12 because we use a batch size of one and not all data is used.

results from using the brain dataset. This shows training on one anatomy does not generalize well to another anatomy. This is likely because the pattern is different and the reconstruction network has specialized on the specific anatomy it was trained on.

#### 4.8 Testing which sampling patterns SIREN can represent

In this section, we want to verify whether the SIREN sampling layer can represent the types of patterns we have represented with the BJORK sampling layer. To fit the SIREN to a given pattern we minimize the sum of absolute difference between the ground truth pattern and the pattern obtained from the SIREN sampling layer. First, we try to fit a simple radial sampling pattern, which was used as an initial pattern for BJORK. The fitted pattern is shown in Figure 20. The lines in the pattern are not entirely straight, which shows that the pattern slightly deviates from the radial pattern. This can also be seen from the objective function, which is low for a good fit. The objective value is 0.44, which is the total absolute difference between the pattern in Figure 20 and the radial sampling pattern.

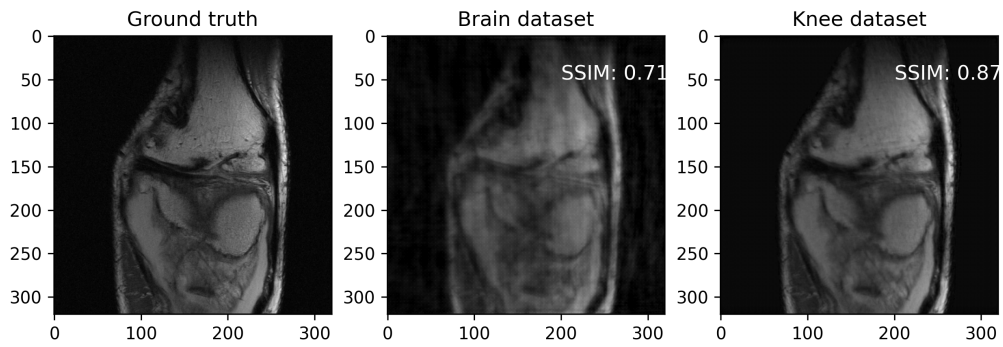


Figure 19: Reconstruction of knee image using parameters trained on the brain dataset compared to using parameters trained on the knee dataset.

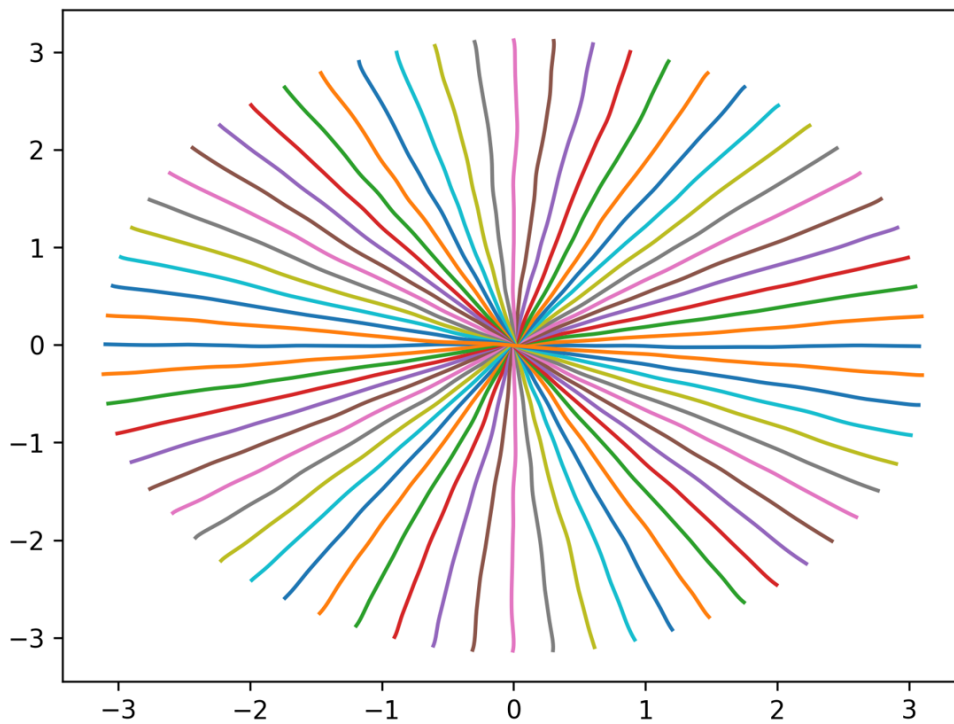


Figure 20: SIREN sampling layer with  $\Omega_0 = 30$ ,  $\Omega = 1$ , 4 hidden layers with 128 hidden features applied to a radial sampling pattern with 32 shots and 1280 samples per shot.

Next, we represent a pattern obtained by training the BJORK algorithm using the SIREN sampling layer. The result is shown in Figure 21. The objective value for this fitted pattern is 0.68, which is worse than the fitted radial pattern. The discrepancy between ground truth and fitted pattern is more clearly visible. The SIREN representation is much smoother due to it being composed of sinusoid functions. Especially at points where the BJORK pattern is more irregular, the SIREN represents this more smoothly. Still, the patterns are very similar, so we can say the SIREN sampling layer can represent BJORK patterns reasonably well.

#### 4.9 Optimizing the parameters in the SIREN sampling layer

When designing the SIREN sampling layer architecture there are a lot of parameters we can tune. The SIREN sampling layer is different from BJORK sampling layer, so the same learning rates may not apply and also the best loss function could be different. We assume that the combination of L1 and SSIM loss works well for SIREN as well and we can use this. The learning rate might need additional tweaking because SIREN probably has very different learning behavior. The number of layers and number of features per layer need to be determined. Additionally we need to determine  $\Omega_0$ , the parameter of the first layer,  $\Omega$  the parameter of all other layers and  $k_0$  the scaling factor of  $\Omega_0$ .

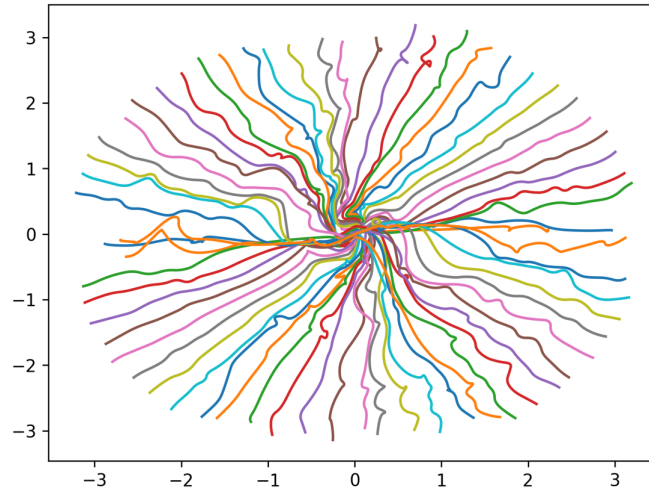
##### Tuning the sampling learning rate and $\Omega_0$

In the first experiment we want to determine the right learning rate and  $\Omega_0$  value. We perform a sweep to optimize the learning rate,  $\Omega_0$  and  $k_0$  simultaneously, as they are expected to be strongly dependent on each other. The learning rate we choose in the range  $(1e-5, 5e-5, 1e-4)$ , because preliminary tests showed that higher values lead to unstable learning. The SIREN paper suggests to use  $\Omega_0 = 30$ , so we choose values around 30, specifically  $(20, 30, 40)$ . For  $k_0$  we take the range  $(1, 1.05, 1.1, 1.2)$ , where we also consider the option to not increase  $\Omega_0$ . We train on the small knee dataset. The performance is evaluated by calculating the average SSIM on 200 slices of the validation set.

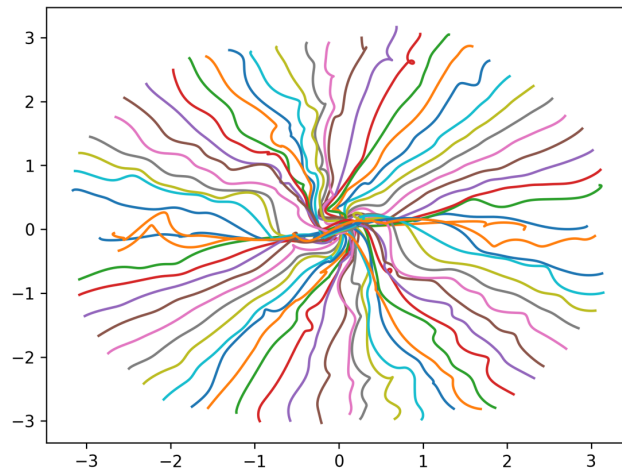
The results of this first experiment is shown in Figure 22. It is quite difficult to draw conclusions based on this Figure because the SSIM scores are quite close together for most runs. One clear observation we can make is that the learning rate should not be  $1e-4$ , because that leads to a lower SSIM score. Looking at one result from this group reveals what is happening. Figure 23 shows the sampling pattern for a run with this learning rate. The shots in the pattern often cross themselves and other shots, which indicates redundancy in the pattern. The average DC loss for this run is increasing over time, which shows the learning is unstable and the learning rate is too high. This is confirmed by the negative correlation of -0.22 between the validation SSIM and sampling learning rate in this sweep.

Investigating what the values for  $\Omega_0$  and  $k_0$  should be from Figure 22 is very difficult, as both high and low  $\Omega_0$  and  $k_0$  values lead to good reconstruction quality and their results are very close together. Therefore, we calculate the average SSIM per  $\Omega_0$  value, which gives 0.77 for  $\Omega_0 = 20$ , 0.77 for  $\Omega_0 = 30$  and 0.78 for  $\Omega_0 = 40$ . These values suggest that in terms of performance  $\Omega_0$  is not an important parameter. For  $k_0$  the average SSIM scores are 0.76, 0.80, 0.76 and 0.77 for  $k_0 = (1, 1.05, 1.1, 1.2)$  respectively, so a slight advantage for  $k_0 = 1.05$ .

Besides the quantitative results we want to look at at the sampling patterns for each  $\Omega_0$  value qualitatively. Figure 24 shows this comparison where  $k_0$  is kept constant. The learning rate is not the same across these runs but we hypothesise the main difference in these plots to be due to the difference in  $\Omega_0$ . Further experiments should point out whether this is truly the case and more importantly whether this improves the reconstruction quality.



(a) BJORK pattern



(b) SIREN representation of BJORK

Figure 21: Comparison BJORK pattern and fitted BJORK pattern with SIREN with  $\Omega_0 = 30$ ,  $\Omega = 1$ , 4 hidden layers with 128 hidden features applied to a radial sampling pattern with 32 shots and 1280 samples per shot.



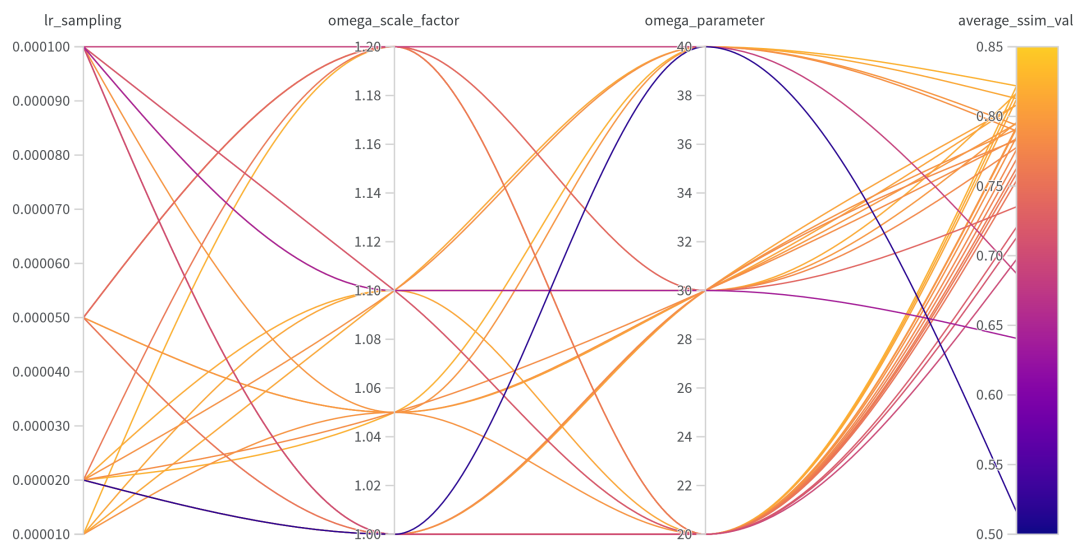


Figure 22: Results of sweep in section 4.9. The parameters involved are the learning rate for the sampling pattern, the scale factor  $k_0$  and  $\Omega_0$ .

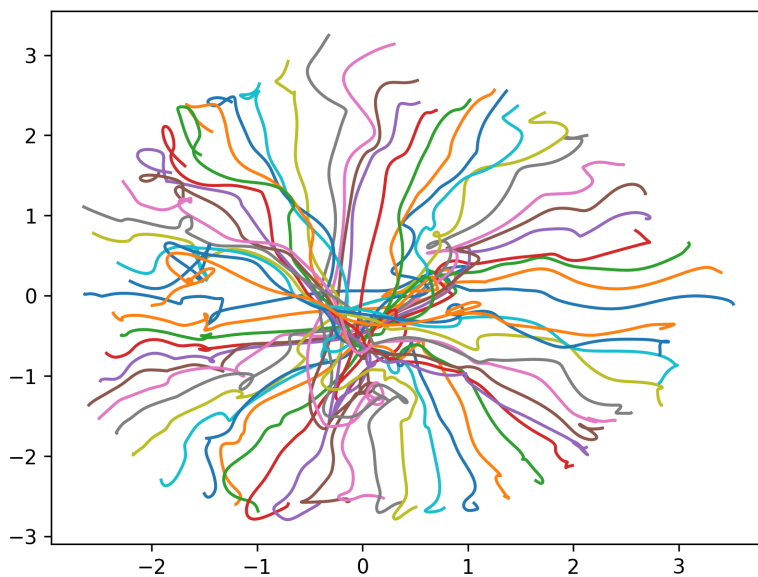
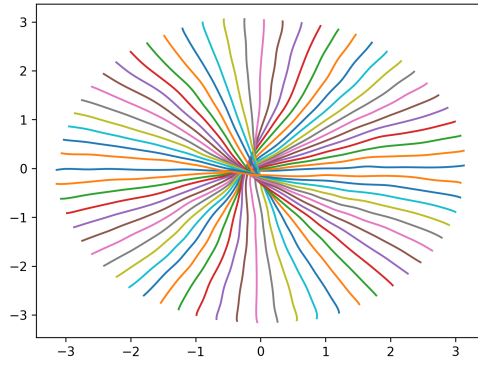
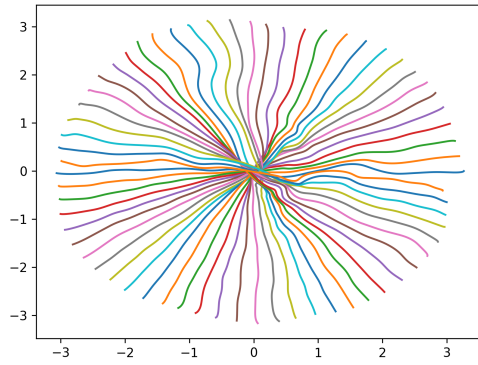


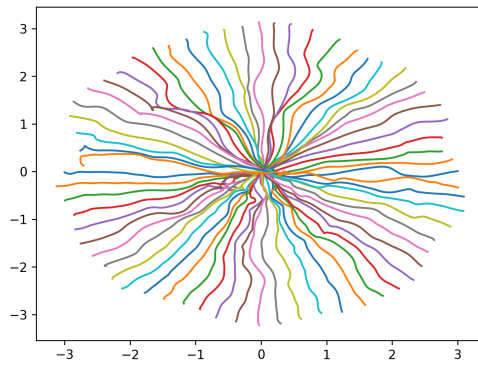
Figure 23: Sampling pattern for SIREN network with high learning rate  $1e-4$



(a)  $\Omega_0 = 20$



(b)  $\Omega_0 = 30$



(c)  $\Omega_0 = 40$

Figure 24: Comparison of the SIREN sampling pattern for different  $\Omega_0$  values for a fixed  $k_0 = 1.1$ . It can be observed that a higher  $\Omega_0$  leads to a pattern that is significantly different from the initial pattern.

## Tuning $k_0$

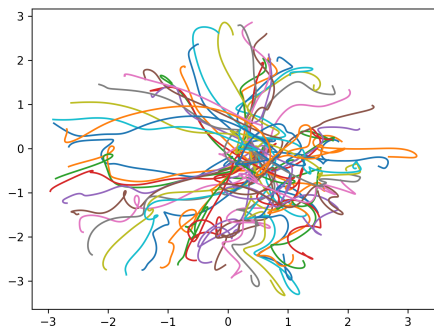
To test the influence of the  $k_0$  variable, we perform two experiments, one with  $k_0 = 1$ , so no change in  $\Omega_0$ , and the other with  $k_0 = 1.1$ . We choose  $\Omega_0 = 40$  since that pattern showed more change in the previous experiment. These experiments are performed on the middle sized dataset as we want to see whether the results found in the sweep for the small dataset also generalise to the larger dataset.

First, we investigate the case  $k_0 = 1$ . The most important issue with this run is that the training behavior on the middle sized dataset is much different from the behavior on the small dataset. Whereas on the smaller dataset the sampling pattern changed relatively little from the original radial pattern, on the middle sized dataset the sampling pattern changes dramatically with the same parameter settings. This shows that the learning behavior of the SIREN sampling layer is very sensitive to the amount of training, which is what we also saw with BJORK. The learned trajectory is shown in Figure 25a. Such an irregular sampling pattern is not bad by definition. In fact, the gradient and slew rate constraints are satisfied by this learned pattern, so it is physically feasible. However, the reconstruction quality is not great with an average SSIM of 0.75 on the validation set. The reason why this pattern does not perform great can be concluded from the loss curves, which are shown in Figure 26. It can be seen that the data consistency is initially decreasing until around iteration 8000. After this point the gradient penalty loss increases. While the algorithm tries to suppress the pattern to satisfy the gradient constraint, the data consistency (DC) loss starts increasing. This pattern continues leading to an unstable DC loss curve. The reason for this instability is likely twofold. First, the sampling pattern is changing too rapidly which leads to difficulties in the reconstruction network to adapt to the new sampling pattern. Second, the gradient and slew rate terms are not penalized enough, leading to even wilder patterns further exacerbating the first problem. The solution to these problems is to lower the sampling pattern learning rate and increase the penalty terms for gradient and slew rate. Another reason for this training behavior could be due to the wrong SIREN architecture. Our choice of  $\Omega_0$  could be right for the small dataset, but wrong for the larger one. Figure 25b shows the sampling pattern for  $k_0 = 1.1$ , which omits a large part of ‘northern’ k-space. This suggests the choice of  $k_0$  has a significant effect on the sampling pattern and that in general  $\Omega_0$  can be lowered to have the sampling trajectory behave more regularly. This confirms what we saw in the previous experiment that  $\Omega_0$  regulates the periodicity of the pattern and therefore how wildly it grows during training. Interestingly, the average SSIM for this pattern is 0.77 which is slightly better than  $k_0 = 1$ . However, leaving out such a large part of k-space is very undesirable behavior. Therefore, in the next experiment we try to get a more regular pattern.

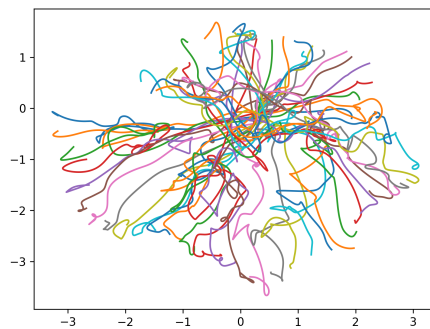
## Lower learning rate and two $\Omega$ values

To get better learning behavior the sampling layer learning rate is chosen to be the lowest value  $1e-5$  in this experiment. Additionally, the gradient penalty parameter is increased to 10 and slew rate penalty parameter to 0.1. The  $\Omega_0$  parameter is lowered from 40 to 30. We investigate two values of  $\Omega$ : 2 which was used in previous experiments and 1, to check whether the learning behavior improves. Performance is evaluated by testing on 1000 slices of the validation set.

The result is an average SSIM of 0.76 and 0.79 for  $\Omega = 2$  and  $\Omega = 1$  respectively. This is a clear improvement over the previous experiment, but it is not on the same level as the BJORK experiments. The sampling patterns are less erratic than in previous experiments as can be seen in Figure 27, where we compare the patterns for the two  $\Omega$  values. We again observe that the pattern that leaves out a part of k-space in Figure 27a performs better. Increasing the penalty parameters, decreasing the learning rate and  $\Omega$  parameters has been effective at reducing the variability of the sampling pattern and improving the reconstruction performance.

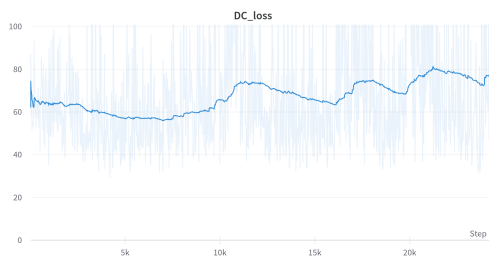


(a) Sampling trajectory for  $k_0 = 1$

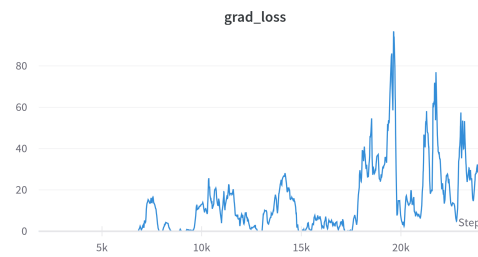


(b) Sampling trajectory for  $k_0 = 1.1$

Figure 25: Comparison sampling patterns for two  $k_0$  values. The right plot with higher  $k_0$  value has a large part of k-space missing, which suggests that choosing  $k_0$  too high leads to a bad sampling pattern.

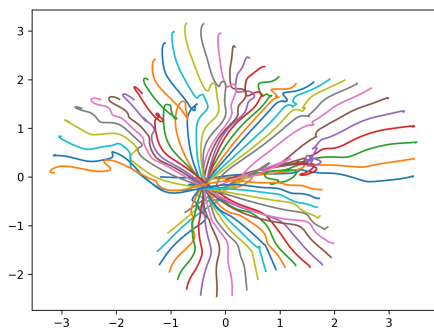


(a) Data consistency loss for  $k_0 = 1$  experiment

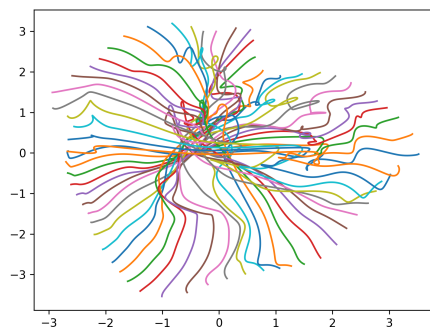


(b) Gradient penalty loss for  $k_0 = 1$  experiment

Figure 26: Unstable loss curve behavior for SIREN sampling layer



(a) Sampling pattern for  $\Omega = 1$



(b) Sampling pattern for  $\Omega = 2$

Figure 27: Comparing sampling patterns for two  $\Omega$  values

## 5 Discussion

The main goal of this thesis was to establish whether BJORK or SIREN is a better option for the optimization of k-space sampling trajectories. This turns out to be pretty difficult to compare fairly because there are a lot of contributing factors to the performance of the algorithm. In this section we try to address the most important contributing factors and give a general conclusion. These factors are possible reasons which make drawing a general conclusion more difficult and could be seen as general limitations of the research.

### 5.1 Replicating BJORK

To test which of the two sampling methods is best we tried to replicate BJORK as accurately as possible. Here we ran into various problems. First of all, there was some code available on GitHub but it was far from complete. It contained large parts of the sampling layer and of the reconstruction network, but their combined implementation together with other things such as data loading, data preprocessing, loss curves and training procedures were missing. Moreover, the code that was available was largely undocumented so it took time to get to know what the code was doing and to determine which parts were still missing.

The most important issue is determining the right hyper-parameters. For many parameters used the authors of BJORK suggest values, but for many their values are simply not mentioned in the paper or in the code. These are for example the regularization parameters of the gradient and slew rate penalty terms,  $\mu_1$  and  $\mu_2$ . These are very important parameters as they gauge how much ‘freedom’ the sampling pattern has to violate the constraints, which has direct effect on the sampling patterns and the corresponding reconstruction network. Similarly, it is mentioned that a combination of L1 and L2 loss terms is used, but it is not mentioned how these terms are scaled. It can be assumed that they are normalized, but this is not clear from the paper. This is partly why we performed an experiment to determine the right loss terms. For the reconstruction network it is mentioned that the DIDN network is used, but it is not specified how many layers and how many features per layer are used.

Another very important difference is the way the authors deal with multicoil data. Authors of the original BJORK paper use the ESPiRiT method to estimate the sensitivity maps, but do not mention the parameters used in this algorithm, which have to be adapted per dataset. Even more importantly, instead of using the ground truth in the FastMRI dataset, they used the conjugate phase reconstruction as ground truth. We have used the FastMRI reconstruction as ground truth, because it is easier and even if there is slight error in our estimation of sensitivity maps, the ground truth is at least accurate.

### 5.2 Difficulty in determining hyper-parameters

There were many unknown parameters in BJORK. Moreover, it also seemed a good idea to alter parameters that were specified in the BJORK paper such as the learning rates and the loss function weights, as our implementation of BJORK might require a different value for these parameters. Finding the optimal hyper-parameter value is often very difficult for various reasons. First of all it depends on the problem at hand and the dataset that is used. Secondly, the parameters are often highly dependent on each other. The learning rate might depend on the loss function that is being used and the number of epochs needed will depend on the learning rate. These hyper-parameters can not be optimized in the same way the normal parameters such as the reconstruction network weights are optimized with a gradient descent algorithm, because it is impossible to calculate the derivative

of the loss with respect to for example the learning rate. Hence, usually the only possible approach is trial and error. This means simply choosing values for the hyper-parameters and tweaking until they give good performance. We have tried to do this more systematically by using the sweeps and corresponding correlation analysis, but it is still mainly trial and error. We have used the random sweep option, that means parameters were taken from a given list at random, as opposed to grid search which just considers every option sequentially. Bergstra et al. have shown that this random approach works better than the grid approach [6].

There are also other systematic approaches that can be taken to optimize hyper-parameters. Weights and biases also provides the option for Bayesian parameter search which uses the work by Falkner et al. [16]. Any hyper-parameter optimization method has to deal with that training the model is often costly in terms of time and resources. It should consider this cost when finding the parameter and spend its time wisely. Bayesian performance can be advantageous over random methods because it is a guided method. It considers a metric of performance in its decision process which parameters to consider next, and in this way the parameter search is guided. As with many optimization problems there is a trade-off between exploration and exploitation. Random search is exclusively exploration. Bayesian search is guided and hence more exploitation, at the risk of not exploring enough of the parameter space. Additionally, Bayesian search has problems with scalability because it used Gaussian processes for uncertainty estimates, but these do not scale well. The work by Falkner et al. [16] uses the Hyperband method which stops poorly performing configurations saving time. Future work could try to optimize the hyper-parameters in a Bayesian approach, but it is not certain whether this would find a better configuration faster.

### 5.3 Bi-level optimization problems

The problem of optimizing sampling and reconstruction simultaneously is quite unconventional compared to common machine learning problem, which usually only focuses on the reconstruction or the sampling. We are effectively dealing with a bi-level optimization problem and in the results we found out that it is often difficult to correctly balance the two parts of the problem. Sometimes the reconstruction network would have difficulty keeping up with the changing pattern. Alternatively, the reconstruction can improve quickly and get stuck in a local minimum, such that the sampling pattern changes very little. This interaction between two networks resembles the way generator and discriminator can interact in Generative Adversarial Networks. Similarly, one has to think about a strategy to make neither network act more strongly than the other.

Salehi et al. suggest a more theoretical approach to solving bi-level optimization problems [32]. They consider problems of the form:

$$\min_{\theta \in \mathbb{R}^d} \left\{ \frac{1}{M} \sum_{i=1}^M g_i(\hat{x}_i(\theta)) + r(\theta) \right\} \quad (16)$$

$$s.t. \hat{x}_i(\theta) = \arg \min_{x \in \mathbb{R}^N} \Phi_i(x, \theta) \quad (17)$$

We can write this in terms of the sampling parameters  $\omega$ , which are the B-spline coefficients in BJORK or SIREN weights, and the reconstruction parameters  $\theta$ :

$$\min_{\omega \in \mathbb{R}^d} \left\{ \frac{1}{M} \sum_{i=1}^M g_i(\hat{\theta}_i(\omega)) + r(\omega) \right\} \quad (18)$$

$$s.t. \quad \hat{\theta}_i(\omega) = \arg \min_{\theta \in \mathbb{R}^N} \Phi_i(\omega, \theta) \quad (19)$$

Now we can set the function  $g = 1$  and the function  $r$  equal to the gradient and slew rate:

$$r(\omega) = \mu_1 \|D_1 \omega\| + \mu_2 \|D_2 \omega\| \quad (20)$$

and the function  $\Phi$  equal to the reconstruction loss of image  $x_i$ :

$$\Phi_i(\omega, \theta) = \|f_\theta(A(\omega)x_i + \varepsilon) - x_i\| \quad (21)$$

The problem with this formulation is that it is required that  $\Phi$  is strongly convex in  $\theta$  and that  $r$  is convex. These conditions are very likely not satisfied in our case. These convexity assumptions are needed to ensure that a gradient descent or quasi-Newton algorithm attains a global minimum or that at least the distance from the global optimum can be estimated. These estimates are needed to guarantee convergence of the algorithm. If we would consider a simple least squares reconstruction without a neural network the formulation would work. For our problem the work can not directly be applied. Chen et al. provide an extensive survey on gradient based Bi-level optimization for deep learning [14]. This survey is very general and does not focus on any imaging problem such as MRI reconstruction and sampling. This survey instead focuses on general techniques for solving bi-level optimization problem such as explicit gradient update and proxy update. Even though the formulations do not match our problem exactly, they still provide important insights in how bi-level optimization problems might be solvable in a more rigorous manner.

## 5.4 BJORK versus SIREN

Comparing the best version of BJORK to the best version of SIREN, we find better results for BJORK with an SSIM of 0.82 compared to 0.79 for SIREN. Both algorithms were optimized on the middle sized dataset. So in terms of results BJORK outperforms SIREN by a slight margin. BJORK's other main advantage has to do with the amount of parameter tweaking that was necessary to obtain these results. In terms of sampling layer the B-splines has as parameters the number of coefficients, which was provided by the authors of BJORK. SIREN, on the other hand, has the parameters  $\Omega_0$  and  $\Omega$ , which are the number of layers and the number of features per layer. It is much more work to find reasonable values for these parameters. Moreover, the SIREN sampling layer is very sensitive to the specific choice of parameters and even after careful consideration the learning behavior is still very erratic. There might be a parameter setting that make SIREN work better than BJORK, but finding these will require a lot more tweaking.

The main argument against BJORK was that the B-splines might be restrictive in terms of what kind of patterns it can represent. As we saw, SIREN is able to represent the BJORK patterns and many other types of wild patterns. However, it is not clear whether the added flexibility of SIREN is actually useful. The SIREN sampling layer has the benefit that the gradient and slew rate penalties work directly on the network weights, which would make optimization easier. However, there is no notable difference between BJORK and SIREN optimization times, or gradient and slew rate quality. Both methods find patterns that satisfy the constraints so in terms of penalties they are equal.

While in SIREN can represent BJORK patterns and many other types of patterns, the optimization using the SIREN sampling layer does not work well. This is likely not a problem of SIREN, but the result of the joint optimization problem being inherently difficult to solve. Optimization relies on gradient descent and there are many reasons why this can become unstable. The SIREN sampling layer cannot be written off, but a lot more research is needed to make it stable and reliable. Next to additional parameter tweaking, it would be a good idea to look at the optimization methods themselves in more detail.



## 6 Future work

There is a lot of opportunity for future research. Joint optimization of MR sampling and reconstruction with deep learning is a relatively new research field, so there is a lot of room for experimentation. In the course of writing this thesis, the BJORK’s follow-up method, the stochastic optimization of 3D Non-Cartesian Sampling Trajectory (SNOPY) method was introduced [44]. The most notable differences between SNOPY and BJORK are the extension to 3D MRI reconstruction and the use of stochasticity in gradient descent optimization. However, the method also improves BJORK in other ways, which are also relevant for the 2D case. This method can combine multiple optimization objectives such that the obtained sampling trajectory has the desired properties. This extends the objective function of BJORK which only contained image quality and hardware constraints. For example, this method introduces a penalty for peripheral nerve stimulation (PNS), since this effect is stronger in 3D MRI imaging. It also introduces a penalty to let the trajectory cross the center of k-space at a collection of gradient time points. This should help with improving image contrast. Parameterization can be done with B-splines, like in BJORK. It can be challenging to implement a BJORK trajectory on an actual MRI scanner because many different gradient waveforms need to be implemented. Another approach is to parameterize the trajectory using a vector of coefficients  $c$  which consists of attributes of existing trajectories, for example rotation angles of a radial trajectory. The sampling trajectory  $\omega(c)$  is a nonlinear function of  $c$ . The trajectory should be differentiable in  $c$  to make optimization possible. Using this different parameterization can also make it easier to integrate the new trajectories into existing workflows. The optimization is done via stochastic gradient Langevin dynamics (SGLD). This introduces randomness in the gradient update step, which improves convergence speed. Optimizing the trajectory with SNOPY helps with structural integrity in the reconstruction even with the simpler reconstruction method like U-net. This suggests the sampling is more important than exactly which reconstruction algorithm is used. Other examples of recent methods for jointly optimizing sampling trajectory and reconstruction network are the 3D SPARKLING method [11] and Bayesian optimization techniques [18].

One could try to adapt the SIREN into a version that leads to higher reconstruction quality. This could be by tweaking the parameters, but could also be by introducing more regularization terms acting on the sampling lines. These regularization terms would penalize for example overlap between lines or lines deviating too much from a predefined area of k-space.

Alternatively, more research can be done on the choice of initial sampling pattern. In this work, we used as an initial pattern the radial sampling pattern, as done in the original BJORK paper. This choice makes sense as it already incorporates the idea of sampling the center more densely as suggested by the area of variable density sampling. However, it is also interesting to consider other initial sampling patterns like spiral or even Cartesian patterns. If we obtain the same patterns even when starting with these other initial patterns, this would support the validity of the found patterns even more.

This work also demonstrates the necessity for rigorous analysis of bi-level optimization problems. We used BJORK’s approach of using gradient descent for both sampling layer and reconstruction network at the same time. However, one could also optimize only the sampling layer or reconstruction network in phases, which could lead to better learning behavior, such as is done by Chaithya et al. [10]. We did not consider analytical convergence behavior in this work. Although it might be necessary to simplify the problem in order to perform more rigorous analysis, it might also give a lot of insight into what works well, such as is done in Salehi et al. [32]. This can again be used to solve the more difficult problem, which we now mostly try to optimise heuristically. There is a large literature on bi-level optimization theory, but we think currently the main challenge is to formulate the problem in a way that allows for this more rigorous analysis.

## 7 Conclusion

In this thesis we tried to compare the B-spline sampling layer and the SIREN sampling layer in terms of their ability to represent k-space sampling trajectories. Moreover, we optimized this k-space sampling trajectory and the reconstruction network jointly. A large part of this thesis has been dedicated to understanding and implementing the BJORK method. We performed multiple experiments to determine the learning rate, which loss function to use and whether to use a warming up for the reconstruction network. For the SIREN sampling layer, we performed various experiments as well. The results from these experiments indicate that BJORK outperforms SIREN for the various parameter settings that we used for both methods. For BJORK it is easier to obtain a well-performing sampling pattern than for SIREN, because there are fewer parameters and BJORK is more limited in terms of what it can represent. The SIREN sampling layer is more flexible in what it can represent, but most erratic sampling patterns are not useful for reconstruction. Under the parameter configurations that were considered, BJORK has better learning behavior and is therefore preferable to use.

## References

- [1] How its performed, mri scan. <https://www.nhs.uk/conditions/mri-scan/what-happens>.
- [2] J. Adler and O. Öktem. Learned primal-dual reconstruction. *IEEE transactions on medical imaging*, 37(6):1322–1332, 2018.
- [3] H. K. Aggarwal, M. P. Mani, and M. Jacob. Modl: Model-based deep learning architecture for inverse problems. *IEEE transactions on medical imaging*, 38(2):394–405, 2018.
- [4] C. D. Bahadir, A. Q. Wang, A. V. Dalca, and M. R. Sabuncu. Deep-learning-based optimization of the under-sampling pattern in mri. *IEEE Transactions on Computational Imaging*, 6:1139–1152, 2020.
- [5] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.
- [6] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.
- [7] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.
- [8] C. Boyer, N. Chauffert, P. Ciuciu, J. Kahn, and P. Weiss. On the generation of sampling schemes for magnetic resonance imaging. *SIAM Journal on Imaging Sciences*, 9(4):2039–2072, 2016.
- [9] R. Bridson. Fast poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches*, 10(1):1, 2007.
- [10] G. Chaithya, Z. Ramzi, and P. Ciuciu. Hybrid learning of non-cartesian k-space trajectory and mr image reconstruction networks. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5. IEEE, 2022.
- [11] G. Chaithya, P. Weiss, G. Daval-Fr erot, A. Massire, A. Vignaud, and P. Ciuciu. Optimizing full 3d sparkling trajectories for high-resolution magnetic resonance imaging. *IEEE Transactions on Medical Imaging*, 41(8):2105–2117, 2022.
- [12] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40:120–145, 2011.
- [13] N. Chauffert, P. Ciuciu, J. Kahn, and P. Weiss. Variable density sampling with continuous trajectories. *SIAM Journal on Imaging Sciences*, 7(4):1962–1992, 2014.
- [14] C. Chen, X. Chen, C. Ma, Z. Liu, and X. Liu. Gradient-based bi-level optimization for deep learning: A survey. *arXiv preprint arXiv:2207.11719*, 2022.
- [15] T. Eo, Y. Jun, T. Kim, J. Jang, H.-J. Lee, and D. Hwang. Kiki-net: cross-domain convolutional neural networks for reconstructing undersampled magnetic resonance images. *Magnetic resonance in medicine*, 80(5):2188–2201, 2018.
- [16] S. Falkner, A. Klein, and F. Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pages 1437–1446. PMLR, 2018.

- [17] J. A. Fessler and B. P. Sutton. Nonuniform fast fourier transforms using min-max interpolation. *IEEE transactions on signal processing*, 51(2):560–574, 2003.
- [18] A. Gossard, F. de Gournay, and P. Weiss. Bayesian optimization of sampling densities in mri. *arXiv preprint arXiv:2209.07170*, 2022.
- [19] J. Hamilton, D. Franson, and N. Seiberlich. Recent advances in parallel imaging for mri. *Progress in nuclear magnetic resonance spectroscopy*, 101:71–95, 2017.
- [20] S. Hao, J. A. Fessler, D. C. Noll, and J.-F. Nielsen. Joint design of excitation k-space trajectory and rf pulse for small-tip 3d tailored excitation in mri. *IEEE transactions on medical imaging*, 35(2):468–479, 2015.
- [21] F. Knoll, J. Zbontar, A. Sriram, M. J. Muckley, M. Bruno, A. Defazio, M. Parente, K. J. Geras, J. Katsnelson, H. Chandarana, et al. fastmri: A publicly available raw k-space and dicom dataset of knee images for accelerated mr image reconstruction using machine learning. *Radiology: Artificial Intelligence*, 2(1):e190007, 2020.
- [22] C. Lazarus, P. Weiss, N. Chauffert, F. Mauconduit, L. El Gueddari, C. Destrieux, I. Zemmoura, A. Vignaud, and P. Ciuciu. Sparkling: variable-density k-space filling curves for accelerated t2\*-weighted mri. *Magnetic resonance in medicine*, 81(6):3643–3661, 2019.
- [23] R. M. Lewitt. Reconstruction algorithms: transform methods. *Proceedings of the IEEE*, 71(3):390–408, 1983.
- [24] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly. Compressed sensing mri. *IEEE signal processing magazine*, 25(2):72–82, 2008.
- [25] M. T. McCann, K. H. Jin, and M. Unser. Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine*, 34(6):85–95, 2017.
- [26] J. L. Nazareth. Conjugate gradient method. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):348–353, 2009.
- [27] K. P. Pruessmann, M. Weiger, M. B. Scheidegger, and P. Boesiger. Sense: sensitivity encoding for fast mri. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 42(5):952–962, 1999.
- [28] P. Putzky, D. Karkalousos, J. Teuwen, N. Miriakov, B. Bakker, M. Caan, and M. Welling. i-rim applied to the fastmri challenge. *arXiv preprint arXiv:1910.08952*, 2019.
- [29] C. G. Radhakrishna and P. Ciuciu. Jointly learning non-cartesian k-space trajectories and reconstruction networks for 2d and 3d mr imaging through projection. *Bioengineering*, 10(2):158, 2023.
- [30] Z. Ramzi, G. Chaithya, J.-L. Starck, and P. Ciuciu. Nc-pdnet: A density-compensated unrolled network for 2d and 3d non-cartesian mri reconstruction. *IEEE Transactions on Medical Imaging*, 41(7):1625–1638, 2022.
- [31] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

- [32] M. S. Salehi, S. Mukherjee, L. Roberts, and M. J. Ehrhardt. Dynamic bilevel learning with inexact line search. *arXiv preprint arXiv:2308.10098*, 2023.
- [33] H. Schomberg and J. Timmer. The gridding method for image reconstruction by fourier transformation. *IEEE transactions on medical imaging*, 14(3):596–607, 1995.
- [34] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.
- [35] F. Sherry, M. Benning, J. C. De los Reyes, M. J. Graves, G. Maierhofer, G. Williams, C.-B. Schönlieb, and M. J. Ehrhardt. Learning the sampling pattern for mri. *IEEE Transactions on Medical Imaging*, 39(12):4310–4321, 2020.
- [36] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.
- [37] A. Sriram, J. Zbontar, T. Murrell, A. Defazio, C. L. Zitnick, N. Yakubova, F. Knoll, and P. Johnson. End-to-end variational networks for accelerated mri reconstruction. In *Medical Image Computing and Computer Assisted Intervention—MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II 23*, pages 64–73. Springer, 2020.
- [38] Y. Sun, B. Wohlberg, and U. S. Kamilov. An online plug-and-play algorithm for regularized image reconstruction. *IEEE Transactions on Computational Imaging*, 5(3):395–408, 2019.
- [39] D. B. Twieg. The k-trajectory formulation of the nmr imaging process with applications in analysis and synthesis of imaging methods. *Medical physics*, 10(5):610–621, 1983.
- [40] M. Tygert and J. Zbontar. Simulating single-coil mri from the responses of multiple coils. *Communications in Applied Mathematics and Computational Science*, 15(2):115–127, 2020.
- [41] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig. Espirit—an eigenvalue approach to autocalibrating parallel mri: where sense meets grappa. *Magnetic resonance in medicine*, 71(3):990–1001, 2014.
- [42] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.
- [43] G. Wang, T. Luo, J.-F. Nielsen, D. C. Noll, and J. A. Fessler. B-spline parameterized joint optimization of reconstruction and k-space trajectories (bjork) for accelerated 2d mri. *IEEE Transactions on Medical Imaging*, 41(9):2318–2330, 2022.
- [44] G. Wang, J.-F. Nielsen, J. A. Fessler, and D. C. Noll. Stochastic optimization of 3d non-cartesian sampling trajectory (snopy). *arXiv preprint arXiv:2209.11030*, 2022.
- [45] G. Wang, N. Shah, K. Zhu, D. C. Noll, and J. A. Fessler. Mirtorch: A pytorch-powered differentiable toolbox for fast image reconstruction and scan protocol optimization.
- [46] T. Weiss, O. Senouf, S. Vedula, O. Michailovich, M. Zibulevsky, and A. Bronstein. Pilot: Physics-informed learned optimized trajectories for accelerated mri. *arXiv preprint arXiv:1909.05773*, 2019.

- [47] J. M. Wolterink, J. C. Zwienenberg, and C. Brune. Implicit neural representations for deformable image registration. In *International Conference on Medical Imaging with Deep Learning*, pages 1349–1359. PMLR, 2022.
- [48] C.-Y. Yip, W. A. Grissom, J. A. Fessler, and D. C. Noll. Joint design of trajectory and rf pulses for parallel excitation. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 58(3):598–604, 2007.
- [49] J. Zbontar, F. Knoll, A. Sriram, T. Murrell, Z. Huang, M. J. Muckley, A. Defazio, R. Stern, P. Johnson, M. Bruno, et al. fastmri: An open dataset and benchmarks for accelerated mri. *arXiv preprint arXiv:1811.08839*, 2018.
- [50] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487–492, 2018.