UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR MEDIZINISCHE INFORMATIK

Fast MRI registration leveraging k-space properties

Schnelle MRT-Registrierung mittels k-Raum Eigenschaften

Masterarbeit

verfasst am
Institut für Medizinische Informatik

im Rahmen des Studiengangs
Medizinische Informatik
der Universität zu Lübeck

vorgelegt von
Jan Meyer

ausgegeben und betreut von
Prof. Dr. Mattias Heinrich

mit Unterstützung von
Ziad Al-Haj Hemidi, Eytan Kats

Lübeck, den 1. Januar 2025

IM FOCUS DAS LEBEN

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Jan Meyer

Zusammenfassung

Irgendwas über MRT und Registration

Abstract

Something about MRI and registration

Acknowledgements

Danke an Mattias, Ziad und Eytan für die gute Betreuung.

Contents

1 Introduction 1

1.1 Contributions of the Thesis 1

1.2 Related Work 1

1.3 Structure of the Thesis 2

2 Basics 3

2.1 Magnetic Resonance Imaging 3

2.2 Image Registration 3

2.3 Deep Learning Architectures 4

2.4 Deep Learning for Image Registration 6

2.5 MRI Motion Reconstruction 7

3 Data 9

3.1 OASIS 9

3.2 CMRxRecon 9

3.3 ACDC 12

4 Network Architectures 14

4.1 Fourier-Net 14

4.2 Fourier Net+ 21

5 Experiments 24

5.1 Parameter Studies on the CMRxRecon Dataset 24

5.2 Parameter Tests on the ACDC Dataset 25

5.3 Data Consistency Experiments 28

5.4 Integration into a Motion-Corrected Reconstruction Pipeline 28

6 Results and Discussion 30

6.1 Parameter Studies on the CMRxRecon Dataset 30

6.2 Further Tests on the ACDC Dataset 32

7 Conclusion 40

Bibliography 41

1

Introduction

MRI is a commonly used medical imaging technique that is non-invasice, radiation-free and has great contrast for soft tissue. However, speed is its main weakness as full-body scan can take up to 30 minutes, which is a burden on patients as well as hindering efficiency. Thus, MRI acquisition is usually accelerated by subsampling the k-space in which the data is recorded. This can, however, lead to problematic image artifacts that hinder many processing algorithms. Another problem of the slow data acquisition are motion artifacts, which are especially common for organs like lung and heart. While breathing can be controlled, cardiac motion is involuntarily and should not be stopped. These challenges have been traditionally addressed with computationally intensive algorithms that usually iterativly solve an optimization problem. New approaches, however, are based around the deep neural networks, which have seen a rise in popularity in the recent years in many fields including image processing. In this thesis, the possibility of using an unsupervised deep learning approach to align subsampled MRI data as well its potential usage in a motion reconstruction pipeline will be investigated.

1.1 Contributions of the Thesis

We managed to implement an unsupervised approach to correct motion in subsampled MRI images and much more.

1.2 Related Work

There are many papers on image registration in general, however in the context of medical image registration with deep learning their number is reduced due to the specialized nature of the subject at hand. Yet, there are a couple of papers that give a good overview of the topic. They give a brief overview of registration methods, basics of deep learning with already existing networks for image registration as well as covering potential applications and challenges [1, 2, 3, 4, 5]. A lot of different approaches for medical image registration are based on *VoxelMorph* [6], such as both *Fourier-Net* [7] and its successor *Fourier-Net+* [8].

1.3 Structure of the Thesis

In chapter 2, the general basis of the thesis containing descriptions of MRI in section 2.1, Image registration in section 2.2 and deep learning architectures as well as their use in image registration in sections 2.3 and 2.4. A special emphasize is put on motion reconstruction in MRI as described in section 2.5.
The data used in this thesis is described in Chapter 3 with details concerning the *OASIS* and *CMRxRecon* datasets in sections 3.1 and 3.2 respectively.
In chapter 4, the networks used in this thesis, namely *Fourier-Net* and *Fourier-Net+*, are described in section 4.1 and 4.2.
The experiments conducted in the making of this thesis are explained in chapter 5 with results and discussion following in chapter 6.
Lastly, the thesis is briefly summarized in chapter 7.

2

Basics

In this chapter the basics of the thesis are explained.

2.1 Magnetic Resonance Imaging

Here MRI is described.

2.2 Image Registration

Image registration is a challenging, yet important task for image processing. It can be described as the process of transforming different image datasets into one coordinate system with matched imaging contents [2]. In the medical field this can be used for clinical applications such as disease diagnosis and monitoring, image-guided treatment delivery, and post-operative assessment. Medical image registration is typically used to pre-process data for tasks like object detection (for e.g. tumor growth monitoring) and segmentation (for e.g. organ atlas creation) where variation in spatial resolution is common between modalities like CT and MRI and patients. Thus the performance of these methods is dependent on the quality of image registration [1].

Medical image registration was often done manually by clinicians, however, registration tasks are often challenging and the quality of manual alignments is dependent on the expertise of the user. These manual registrations are thus not only time consuming, but also hardly reproducible leading to high interobserver-variability. The need for automatic registration is very much apparent, but this task remained hard to solve for a long time, requiring a lot of computational power and time for computer algorithms to solve the problem. While neural networks also require a lot of computational power and time to train, they promise fast execution after training. With the rise of deep learning these network gained popularity and now pose a real alternative to conventional algorithms and manual registration [2]. We will discuss these new approaches in the next section, but first we need to formally define our problem.

In pair-wise image registration two images (F and M) are to be aligned, with F denoting the fixed and M the moving image. T is the desired spatial transformation that aligns the two images. This can be posed as an optimization problem:

$$T' = \arg\max S(F, T(M)), \tag{2.1}$$

with T' being the best transformation that maximizes the similarity S between the two images. This process is done iteratively improving estimates for the desired T, such that the defined similarity in the cost function is maximized [1].

Transformations can be categorized as rigid, affine, and deformable. A rigid transformation consists of rotation and translation; an affine transformation includes translations, rotations, scaling, and sheering; the two kinds of transformations are described as a 2D single matrix. Unlike rigid and affine transformation, deformable transformation is a high-dimension problem that we need to formulate by a 3D matrix for 2D deformable registration i.e., a so-called deformation field. While rigid and affine registration algorithms have already achieved good performance in many applications, deformable registration is still a challenging task due to its intrinsic complexity, particularly when the deformation is large. However, these are also the transformations most likely encountered in clinical practice as it can be utilized to fuse information from different modalities such as MRI and CT [4]. Additionally, deformable image registration can also be utilized for various computer-assisted interventions like biopsy [9] and (MRI-guided) radiotherapy [10, 11].

Intuitively, deformable image registration is an ill-posed problem, making it fundamentally different from other computer vision tasks such as object localization, segmentation or classification. Given two images, deformable image registration aims to find a spatial transformation that warps the moving image to match the fixed image as closely as possible. However, there is no ground-truth available for the desired deformation field and without enforcing any constraints on the properties of the spatial transformation, the resulting cost function is ill-conditioned and highly non-convex. In order to address the latter and ensure tractability, all image registration algorithms regularize the estimated deformation field, based on some prior assumptions on the properties of the underlying unknown deformation [1].

Many methods have been proposed for medical image registration to deal with the complex challenges of this task. Popular conventional registration methods include optical flow [12], demons [13] and many more. However, most of these still lack accuracy and computation speed, which makes newer deep learning approaches all the more interesting [3].

2.3 Deep Learning Architectures

Neural networks, despite the theoretical concepts being around for decades, have seen a meteoric rise in popularity over the last few years as constraints on computational power have been alleviated. Especially deep neural networks, which are often summarized under the term deep learning (DL). Recent years have witnessed an almost exponential growth in the development and use of DL algorithms, sustained thus far by rapid improvements in computational hardware (e.g. GPUs). Consequently, clinical applications requiring image classification, segmentation, registration, or object detection/localization, have

witnessed significant improvements in algorithmic performance, in terms of accuracy and/or efficiency [1]. The following network architecture are widely used for different tasks including medical image registration.
Some basic stuff about network training, testing and different architectures that are relevant for the later Chapters.

Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of deep neural networks with regularized multilayer perceptron, which are mainly used for image processing. CNNs use convolution operations instead of general matrix multiplications in typical neural networks. These convolutional filters make CNNs very suitable for visual signal processing. Because of their excellent feature extraction ability, CNNs are some of the most successful models for image analysis. Different variants of CNN have been proposed and have achieved the-state-of-art performances in various image processing tasks. A typical CNN usually consists of multiple convolutional layers, max pooling layers, batch normalization layers, sometimes dropout layers, a sigmoid or softmax layer. In each convolutional layer, multiple channels of feature maps are extracted by sliding trainable convolutional kernels across the input feature maps. Hierarchical features with high-level abstraction are extracted using multiple convolutional layers. These feature maps usually go through multiple fully connected layer before reaching the final decision layer. Max pooling layers are often used to reduce the image sizes and to promote spatial invariance of the network. Batch normalization is used to reduce internal covariate shift among the training samples. Weight regularization and dropout layers are used to alleviate data overfitting [3]. The loss function is often defined as the difference between the predicted and the target output. CNNs are usually trained by minimizing the loss via gradient back propagation using optimization methods like Adam [14].

U-Net

The U-Net [15] architecture is an extension of the typical CNNs structure typically used for image segmentation, however it can also be used for image registration tasks. It adopts symmetrical contractive and expansive paths with skip connections between them. The encoding blocks on the left extract important features from the image using convolution layers and max pooling, which are then stored in the latent space in the middle. From there it is reconstructed using upsampling and convolutions in the decoding blocks on the right. Additionally, skip connections are used to improve the spatial resolution of the segmentation. This architecture allows effective feature learning from a small number of training datasets [3].

Autoencoders

An autoencoder (AE) is a type of CNNs that learns to reconstruct an image from its input without supervision. AEs usually consists of an encoder which extracts the input features, which are stored a low-dimensional latent state space, similar to a U-Net, and a decoder which restore the original input from the latent space. To prevent an AE from learning an identity function, regularized autoencoders were invented, which can be used for e.g. denoising AEs. Variational AEs (VAEs) are generative models that learn latent representation using a variational approach, which constrains the variability of the outputs. VAEs can been used for anomaly detection and image generation [3].

Generative Adversarial Networks

Generative adversarial networks (GANs) consist of two competing networks, a generator and a discriminator. The generator is trained to generate artificial data that approximate a target data distribution from a low-dimensional latent space similar to an AE. The discriminator is trained to distinguish the artificial data from actual data. The discriminator encourages the generator to predict realistic data by penalizing unrealistic predictions via learning. Therefore, the discriminative loss could be considered as a dynamic network-based loss term. The generator and discriminator both are getting better during training to reach Nash equilibrium. In medical imaging, GANs have been used to perform image synthesis for inter- or intra-modality, such as MRI to synthetic CT and vise versa. In medical image registration, GANs are usually used to either provide additional regularization or translate multi-modal registration to uni-modal registration [3].

2.4 Deep Learning for Image Registration

Recently, there has been a surge in the use of deep learning based approaches for medical image registration. Their success is largely due to their ability to perform fast inference, and the flexibility to leverage auxiliary information such as anatomical masks as part of the training process. The most effective methods, such as *VoxelMorph* [6], typically employ a U-Net style architecture to estimate dense spatial deformation fields. These methods require only one forward pass during inference, making them orders of magnitude faster than traditional iterative methods. Following the success of *VoxelMorph*, numerous deep neural networks have been proposed for various registration tasks [8]. Other approaches also utilize CNNs, AEs and GANs. Typical strategies are discussed in more detail in the following sections.

Supervised Registration

Supervised registration describes training a network with a ground truth displacement field that is either real (created by hand) or synthetic (generated via traditional iterative registration algorithms). Thus the loss can easily be calculated as the difference in the displacement fields of the network prediction and the ground truth. These methods have achieved notable results with real displacement fields as supervision. However, this approach is very limited by the size and the diversity of the dataset. As the displacement fields are often calculated by conventional algorithms their effectiveness might be limited for difficult problems with which the traditional algorithms struggle. Fully supervised methods are widely studied and have notable results, but the generation of real or synthetic displacement fields is hard, and these displacements fields might be different from the real ground truth, which can impact the accuracy and efficiency of these kinds of methods [4]. Notable approaches include *BIRNet* [16].

Unsupervised Registration

As the preparation of the ground truth displacement field for supervised methods is inconvenient, limitations in generalizing results in different domains and various registration tasks are inevitable. Thus, unsupervised registration has a more convenient training process with paired images as inputs, but without a ground truth. Generally, unsupervised learning consists of similarity-based and GAN-based methods, where the loss function computes the similarity between the aligned images and the smoothness of the displacement field, rather than the difference to a ground truth [4]. Well known example are *IC-Net* [17], *VoxelMorph* [6], *TransMorph* [18] and *SYMNet* [19].

2.5 MRI Motion Reconstruction

Patient motion during acquisition is one of the major impediments of high-quality MRI scans. This is especially true for thoracic and abdominal imaging, as organs move during breathing. This motion can induce several consequences on MR signal formation. Intraview and interview motion have to be distinguished between: motion is intraview when occurring during individual MR experiments (between RF excitation and echo formation), whereas motion is interview when occurring between individual MR experiments. Whenever the period of motion is slow compared to the period of MR acquisition defined by the repetition time TR, the assumption can be made that motion is interview. This is often a reasonable assumption when considering pseudo-periodic motion induced by respiration, and also possibly by cardiac contraction, which are the two common sources of motion in cardiac and abdominal imaging (typically, the adult respiratory period is about 4–5 s, and $TR \approx 10ms$ for fast imaging). Interview motion results in spatial encoding inconsistencies, and thus in image deterioration which can take complex forms (blurring/ghosting artifacts) as acquisition is performed in a Fourier space. Several strategies can be employed in order to handle patient motion better. Patient cooperation is the most commonly used method. However, breath-holds cannot last much longer than $20s$ and physiological drifts cannot be completely avoided. This leads to a limitation on the time-period of signal recording and thus, signal-to-noise ratio (SNR). Moreover, the position of organs in successive breath-holds may not be reproducible. Synchronization techniques are well-established and systematically used in clinical protocols, but they require a high-level of motion reproducibility. This is often a limiting factor considering heart rate variability (whether in free breathing or during a breathhold), and respiratory variability in terms of amplitude and frequency. This is why alternative techniques have been put forward with the aim of inverting the process of spatial encoding of moving structures that underlies artifacts. While it is possible to correct for motion prospectively, by modulating the magnetic field gradients and RF fields in order to cancel the effect of motion in the Bloch equations, the method is limited to correcting of, at

best, affine motion, due to magnetic field gradient systems being linear. Motion can also be compensated for in reconstruction, however, many methods are restricted to rigid or affine motion due either to computational issues or to the difficulty of modeling complex displacement fields [20]. The latter however seems to become less of a problem as image registration improves with e.g. the help of deep learning.
Motion-resolved data acquisition for these applications are usually accelerated by Parallel Imaging or Compressed Sensing techniques yielding sub-Nyquist sampled (subsampled) k-space data. In order to reconstruct aliasing-free images, these methods rely on reconstruction schemes that, for example incorporate sparsity or low-rank constraints to solve the ill-posed problem. Fixed sparsity assumptions in Compressed Sensing are often too restrictive and incapable of fully modeling spatio-temporal dynamics. Careful fine-tuning between regularization and data consistency is required and especially in highly subsampled cases residual aliasing may remain in the image (under-regularization) or staircasing and blurring artifacts can occur (over-regularization) which affect the image registration. After reconstruction, non-rigid motion fields can be estimated in image space from reconstructed images by solving a registration problem. A particular interest and challenge lies in the derivation of reliable motion fields which capture the spatio-temporal non-rigid deformations, such as respiratory or cardiac movement. Instead of performing these two steps sequentially, motion-compensated image reconstruction schemes like *GRICS* [20] integrate both motion field estimation and motion correction into the reconstruction process. These methods require reliable motion-resolved images from which the motion fields can be estimated. Motion field estimation can be controlled or supported by external motion surrogate signals, initial motion field estimates, from motion-aliased images or low-frequency image contents. Moreover, spatio-temporal redundancies can be exploited to achieve an aliasing-free image. While these methods have been proven to be more robust against registration errors, they can require a significantly increased computational demand and/or limit imaging acceleration [21].

There are also approaches for learning new subsampling strategies in a data-driven manner (pruning unimportant k-space frequencies) [22] as well as deep learning based radial [23] and non-Cartesian [24] subsampling for MRI acceleration.

3

Data

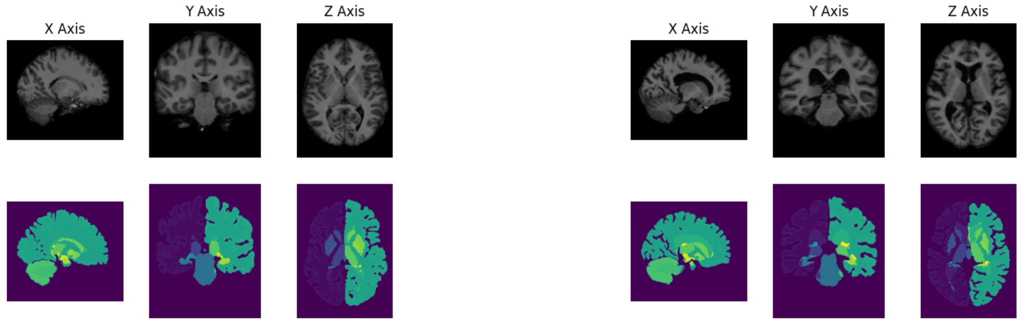
In the following chapter, the datasets used in this thesis are presented and potential preprocessing steps, as well as uses, discussed. The first dataset called *OASIS*, containing MRI brain scans, was used in the original publications for training and evaluation of both *Fourier-Net* and *Fourier-Net+*. It was thus used to get familiar with these networks and their respective work-flows as well as reproducing some results from the paper to ensure that the code works as intended.
For our major experiment the *CMRxRecon* dataset was used. The cardiac MRI k-space data was used for extending the networks to frame-to-frame registration in order for them to be integrated into a motion reconstruction pipeline. It already contains subsampled data, but does not provide segmentations for multi-coil data.
Thus a third dataset, the *ACDC* cardiac dataset, was used. While it contains no k-space data segmentations are given for end-systolic and end-diastolic frames, which can thus be used for the evaluation of registration performance. Most parameter studies and optimizations of the networks were done with this dataset.

3.1 OASIS

The *OASIS-1* dataset [25] contains T1-weighted MRI brain scans from 454 subjects and was mainly used for training and testing the original *Fourier-Net*/*Fourier-Net+*. The brain scans were further pre-processed by [26] for the *Learn2Reg*-Challenge [27]. This enables subject-to-subject brain registration, as all MRI scans were bias-corrected, skull-stripped, aligned, and cropped to the size of $160 \times 192 \times 224$. The images were stored in the *NIFTI* format, which makes it really easy to use. Examples can be seen in Figures 3.1b and 3.1a with slices from the center of the x-, y- and z-axis.

3.2 CMRxRecon

The *CMRxRecon* dataset [28] from the *CMRxRecon2023* challenge specializes in Cardiac magnetic resonance imaging (CMR). The dataset includes fully sampled and subsampled multi-coil k-space data, as well as auto-calibration lines. This includes imaging of differ-



(a)

(b)

Figure 3.1: Example images (upper row) from the *OASIS* dataset with corresponding labels (bottom row).

ent anatomical views like long-axis (2-chamber, 3-chamber, and 4-chamber) and short-axis (SAX). There is a total of 120 training data, 60 validation data, and 120 test data from healthy volunteers. One of the goals of the challenge is the reconstruction from subsampling and motion correction for the heart movement.

Image Reconstruction

As the data was recorded in the k-space (see Figure 3.3) and stored as *.mat* files we first need to reconstruct the images to use them for training and testing of *Fourier-Net/Fourier-Net+*. For this, multi-coil reconstruction algorithms like *SENSE* [29] can be used. The sensitivity of the different coils can be seen in Figure 3.2, where each coil focuses on a specific area of the image. These images are then stitched together in reconstruction to produce a combined image with great overall contrast. The dataset, as mentioned before, contains fully sampled as well as subsampled k-space data (see Figure 3.3a and Figure 3.3b), with the latter being typically used to accelerate the MRI acquisition process. This is done by not using all of the available k-space data, but rather masking the signal to achieve the subsampling. The center region of the k-space is always fully sampled, but the outer regions are subsampled depending on the sampling strategy [30]. The dataset contains subsampled k-space data for 4x and 8x acceleration with the latter of course leading to more distortions in the reconstructed image as the subsampling can induce image reconstruction artifacts. This can be seen in Figure 3.3d, where the image reconstructed from 4x accelerated k-space data seems blurred when compared to the fully sampled one in Figure 3.3c. For all experiments the short-axis view data was used. As image sizes between patients can vary slightly, interpolation was used to standardize the images to a size of 246×512 to avoid further problems with e.g. loss calculation. Additionally, min-max-normalization is used to standardize the data range for all images to $[0, 1]$ as this was also varying. The reconstructed images were stored for every image slice for every patient. Thus, all frames for a specific image slice were in a single folder to enable easy access for later data load-in for frame-to-frame registration.

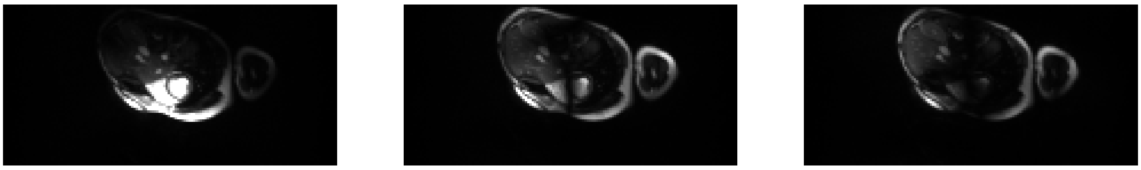
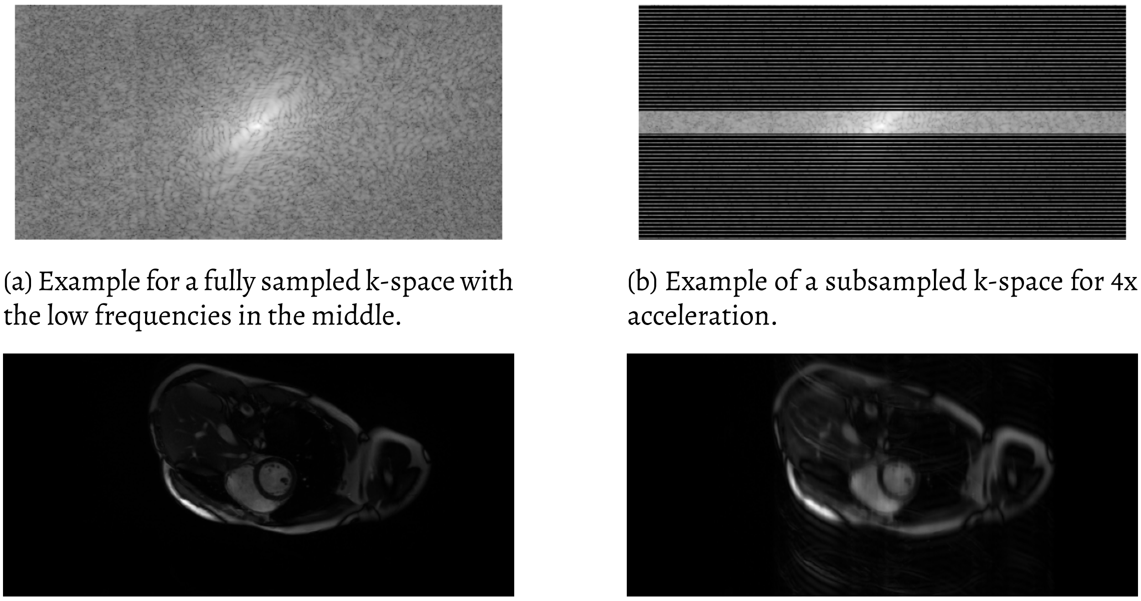


Figure 3.2: MRI images for three different coils from the *CMRxRecon* dataset [28].



(a) Example for a fully sampled k-space with the low frequencies in the middle.

(b) Example of a subsampled k-space for 4x acceleration.

(c) Corresponding image reconstructed from the fully sampled k-space.

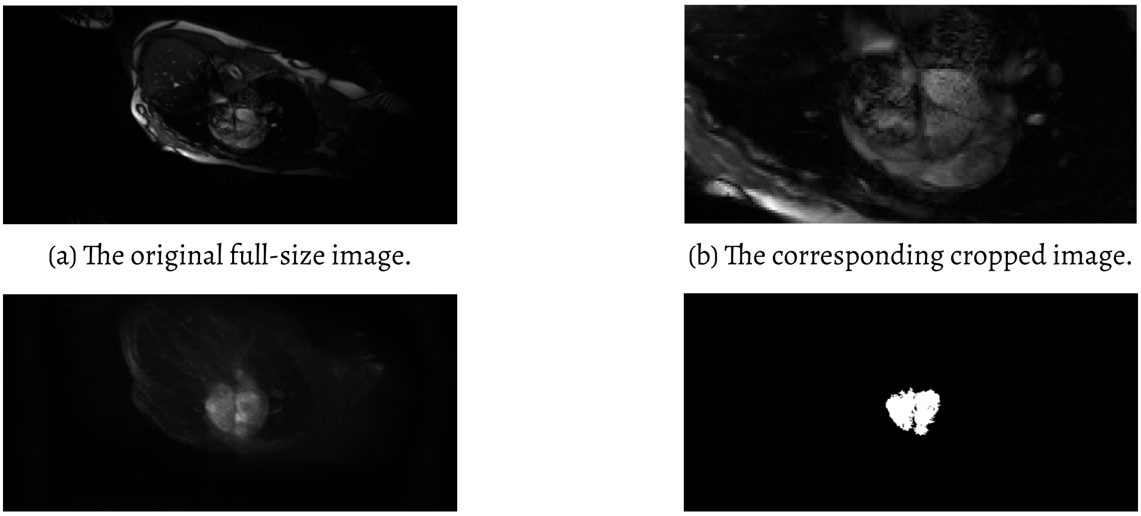
(d) Corresponding image reconstructed from the subsampled k-space.

Figure 3.3: Fully sampled and subsampled k-space data from the from the *CMRxRecon* dataset [28] with corresponding images.

Pre-Processing

As seen before, a large part of the image is just black background that contains no usable information. The interesting cardiac region in the image, however, is much smaller and thus image cropping can be used to extract it. The fairly large image size can thus be cut down without losing important image information, which also helps with memory efficiency. One problem however remains: image alignment. Whereas in the *OASIS* dataset the brain scan were already aligned, the cardiac data in the *CMRxRecon* dataset is not. Thus, one cannot simply center-crop all images as the cardiac region might be located in very different parts of the image. To resolve this issue, adaptive cropping with automatic detection of the cardiac region in the images is needed.
To begin, some assumption about the data needs to be made: First, the cardiac region is at the same position for all slices and frames from one patient. Second, most of the movement happens in the cardiac region between frames. From the latter we can conclude that we need to examine the frames, thus the difference between consecutive frames is

calculated and averaged for each slice. According to the first assumption this process is repeated for all slices and their results summed to create a sort of heatmap as seen in Figure 3.4c with a clearly higher intensity in the cardiac region compared to the background. To further isolate this region a simple thresholding is performed followed by an opening operation, which removes any smaller fragments still left in the image. The cardiac region is now clearly separated from the rest of the image as seen in Figure 3.4d. The only step left is to calculate the approximate center of this region and create a crop around it. To drastically reduce the image size a cropping factor of one third was used, reducing the image size from 246 × 512 to 82 × 170. The result of this drastic crop can be seen in Figure 3.4b when compared to the original image in Figure 3.4a. Note, however, that the cropped image in Figure 3.4 appears larger than it actually is, when compared to the original, to allow for a more detailed look. Thus both images are not presented to scale, but in order to show the most image detail.



(a) The original full-size image.

(b) The corresponding cropped image.

(c) Sum of the differences between slices and their respective frames with the cardiac region in the middle.

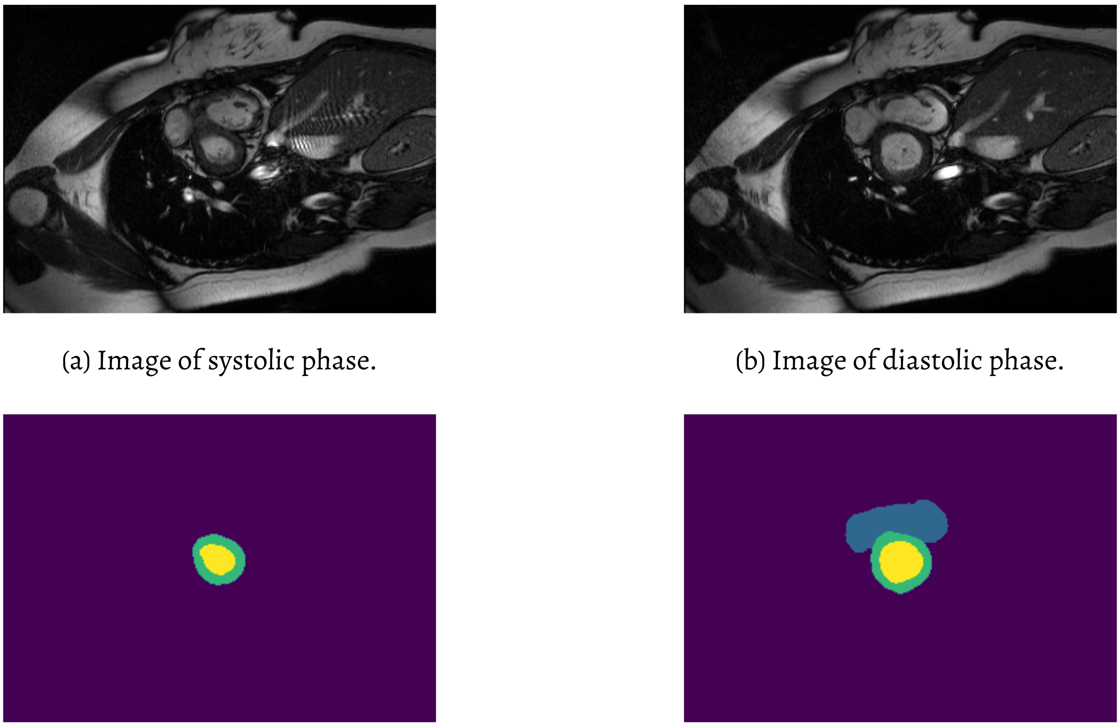
(d) Mask of the cardiac region generated from the sum of differences by thresholding and opening.

Figure 3.4: Cropping of an example image by locating the movement between frames for each slice. Note that the cropped image appears larger than it actually is compared to the original.

3.3 ACDC

The *ACDC* dataset [31] from the *Automated Cardiac Diagnosis Challenge (ACDC)* during the *MICCAI 2017* conference. It contains cardiac cine-MRI short-axis data from 150 subjects that were divided into 5 subgroups (4 pathological, 1 healthy). For each subject systolic (see Figure 3.5a) and diastolic frames (see Figure 3.5b) are provided with corresponding segmentations (see Figure 3.5c and 3.5d), which enables direct comparison via e.g. the Dice score. The segmentations contain only four values with 0, 1, 2 and 3 representing pixels located in the background, in the RV cavity, in the myocardium, and in the LV cavity. The frames themselves are 3D volumes with size 216 × 256 × 10, thus ten image slices can be extracted which each have size 216 × 256. The same holds true for the segmentation, which means that both the image data and segmentations can generated in the same manner.

For the training data, the original 4D data was used as we do not require the segmentations for the end-systolic and end-diastolic frames. As the 4D data has size 216×256×10×30 we can extract 30 frames for each of the ten slices. These can be sorted into 251376 image pairs for training. For the validation and test data the end-systolic and end-diastolic frames with their segmentations were used given us 641 image pairs each as we can only have these two frames to align.



(a) Image of systolic phase.

(b) Image of diastolic phase.

(c) Segmentation of systolic phase.

(d) Segmentation of diastolic phase.

Figure 3.5: Example images for systolic (a) and diastolic frames (b) with corresponding segmentations (c),(d) taken from the *ACDC* dataset [31].

4

Network Architectures

As a starting point *Fourier-Net* [7] and its successor *Fourier-Net+* [8] were used. These networks, which are explained in the following pages, enable fast and accurate registration while needing less resources compared to similar approaches. These attributes are very beneficial for a potential application like motion correction because the current networks, e.g. *LAPNet* [32], are usually supervised and require large computational resources.

4.1 Fourier-Net

Fourier-Net is a new unsupervised approach that aims to learn a low-dimensional representation of the displacement field in a band-limited Fourier domain instead of the full field in the spatial domain. This band-limited representation is then decoded by a model-driven decoder to the dense, full-resolution displacement field in the spatial domain. This allows for fewer parameters and computational operations, resulting in faster inference speeds [7]. The architecture is based on the U-Net [15], like most deep registration approaches, but replaces the expanding path with a parameter-free model-driven decoder as mentioned before. The encoder of *Fourier-Net* consists of a CNN, which takes two images (fixed and moving) as inputs. The output is a displacement field that is then converted from the spatial domain into the Fourier domain via an discrete Fourier transformation (DFT). From there, this band-limiting representation is padded with zeros to the full resolution of the original displacement field. The field is then recovered by using the inverse DFT (iDFT) to convert it back into the spatial domain. This displacement field is then used to warp the moving image into the fixed image. Additionally, squaring and scaling layers [33] can be added before warping the image in order to encourage a diffeomorphism in final deformation.

Encoder

The encoder of *Fourier-Net* consists of a CNN that generates the displacement field between the two inputs followed by a DFT layer that produces a band-limited representation of the full displacement field. The fully convolutional neural network (FCN) from *SYMNet* [19] was modified to function as the CNN of the encoder. Its (original) architec-

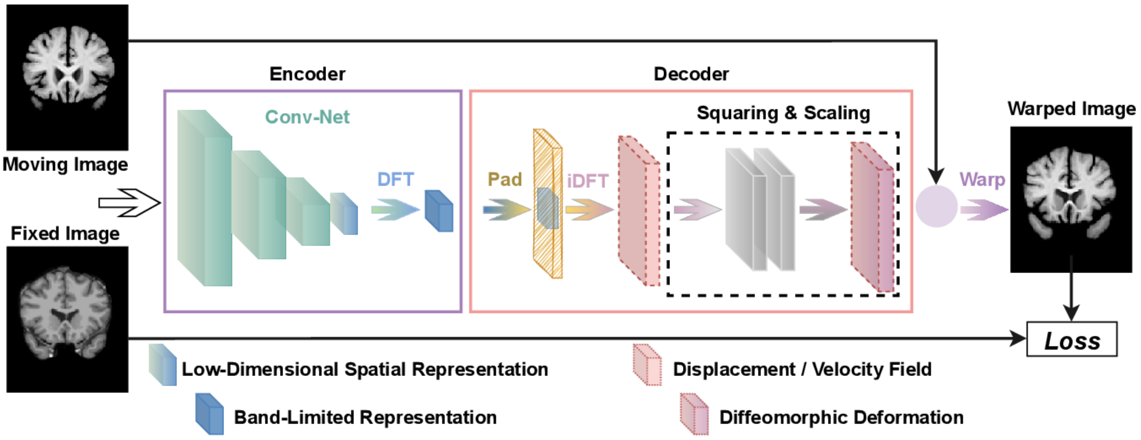

Encoder
Decoder
Conv-Net
DFT
Pad
iDFT
Squaring & Scaling
Moving Image
Fixed Image
Warp
Warped Image
Loss
Low-Dimensional Spatial Representation
Band-Limited Representation
Displacement / Velocity Field
Diffeomorphic Deformation

Figure 4.1: Architecture of *Fourier-Net* taken from [7].

ture (see Figure 4.2) is again based on the *U-Net* with a contracting and expanding path. The FCN concatenates the inputs images X and Y as a single 2-channel input and estimates two dense, non-linear displacement fields ϕ_{XY} and ϕ_{YX}, however we only need the displacement field for the moving image, denoted as $\mathbb{S}_{\phi}$, since we are not interested in transforming the fixed image. This is actually a low dimensional representation because *Fourier-Net* does not utilize the last two levels of the FCNs expanding path that would be needed to reconstruct the actual full-resolution displacement field ϕ.
For each level in the contracting path of the FCN, two successive convolution layers are applied, which contain one $3 \times 3 \times 3$ convolution layer with a stride of 1, followed by a $3 \times 3 \times 3$ convolution layer with a stride of 2 to further compute the high-level features between the input images as well as downsample the features by half until the lowest level of the network is reached. For each level in the expanding path of the FCN, the feature maps from the contracting path are concatenated through skip connections and apply $3 \times 3 \times 3$ convolution with a stride of 1 and $2 \times 2 \times 2$ deconvolution layer for upsampling the feature maps to twice of its size. At the end of the expanding path, two $5 \times 5 \times 5$ convolution layers with a stride of 1 are appended to the last convolution layer and generate the displacement fields θ_{XY} and θ_{YX} [19]. Each convolution layer in the FCN is followed by a rectified linear unit (ReLU) activation, except for the output convolution layer that does not have an activation function because *Fourier-Net* only uses the first two levels of the expanding path, thus leading the FCN to generate a low dimensional representation $\mathbb{S}_{\phi}$ of the full-resolution displacement field ϕ.
As discussed previously, the encoder aims to learn a displacement (or velocity) field in the band-limited Fourier domain. Intuitively, this may require convolutions to be able to handle complex-valued numbers, which can be done by using complex-valued CNNs [34], which are suitable when both input and output are complex values, however these complex-valued operations sacrifice computational efficiency. Other approaches like *DeepFlash* [35] tackle this problem by converting the input images to the Fourier domain and using two individual real-valued CNNs to learn the real and imaginary parts separately. This, however, increases training and inference cost. To bridge the domain gap between real-valued spatial images and complex-valued band-limited displacement fields without increasing

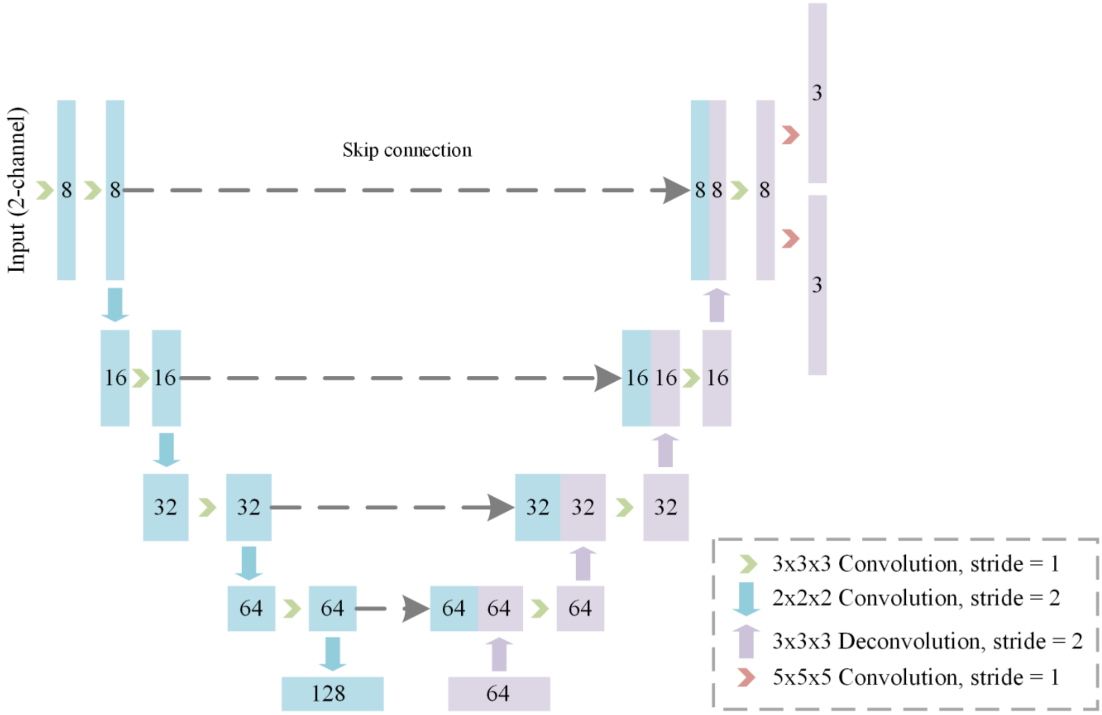

Input (2-channel)
8
8
Skip connection
8
8
8
3
3
16
16
16
16
16
32
32
32
32
32
64
64
64
64
64
128
64
3x3x3 Convolution, stride = 1
2x2x2 Convolution, stride = 2
3x3x3 Deconvolution, stride = 2
5x5x5 Convolution, stride = 1

Figure 4.2: Architecture of the FCN from *SYMNet* [19].

complexity, *Fourier-Net* uses a DFT layer after the FCN. This is a simple and effective way to produce complex-valued band-limited displacement fields without the network needing to be able to handle complex values itself. The DFT applied to the displacement field ϕ can be defined as follows:

$$[\mathcal{F}(\phi)]_{k,l} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} \phi_{n,m} \cdot \exp\left(i \cdot \left(\frac{2\pi k}{H} \cdot n + \frac{2\pi l}{W} \cdot m\right)\right), \tag{4.1}$$

where ϕ has size $H \times W$, $n \in [0, H-1]$ and $m \in [0, W-1]$ are the discrete indices in the spatial domain, and $k \in [0, H-1]$ and $l \in [0, W-1]$ are the discrete indices in the frequency domain with i being the imaginary unit. However, ϕ in this equation is actually the low dimensional representation of the displacement field output by the modified FCN, which can be formulated as follows:

$$\mathbb{S}_\phi = \text{FCN}(M, F; \Theta), \tag{4.2}$$

with M being the moving and F the fixed image, as well as Θ representing the parameters of the FCN. Thus, the whole encoder can be defined as:

$$\mathbb{B}_\phi = \mathcal{F}(\mathbb{S}_\phi) = \mathcal{F}(\text{FCN}(M, F; \Theta)), \tag{4.3}$$

with the DFT layer $\mathcal{F}$, full-resolution spatial displacement field ϕ and the complex band-limited displacement field $\mathbb{B}_\phi$. The low dimensional representation $\mathbb{S}_\phi$ actually contains all the information of the band-limited Fourier coefficients in $\mathbb{B}_\phi$. As such, *Fourier-Net* does not need to learn the coefficients of $\mathbb{B}_\phi$, but instead only the real-valued coefficients in $\mathbb{S}_\phi$, which is the low dimensional spatial representation of the full-resolution spatial displacement field ϕ, which is then reconstructed by the decoder.

Decoder

The decoder contains no learnable parameters, instead the usual expansive path is replaced with a zero-padding layer, an iDFT layer, and an optional squaring and scaling layer.
The output from the encoder is a band-limited representation $\mathbb{B}_\phi$ in the frequency domain of the low dimensional displacement field $\mathbb{S}_\phi$ in the spatial domain. To recover the full-resolution displacement field ϕ in the spatial domain, we first pad the patch $\mathbb{B}_\phi$, containing mostly low frequency signals, to the original image resolution with zeros. We then feed the zero-padded complex-valued coefficients, denoted as $\mathcal{F}(\phi)$, to an iDFT layer consisting of two steps: shifting the Fourier coefficients from centers to corners and then applying the iDFT to convert them into the spatial domain:

$$\phi_{n,m} = \frac{1}{HW} \sum_{k=0}^{H-1} \sum_{l=0}^{W-1} \mathcal{D}_{k,l}[\mathcal{F}(\phi)]_{k,l} \cdot \exp\left(i \cdot \left(\frac{2\pi n}{H} \cdot k + \frac{2\pi m}{W} \cdot l\right)\right). \tag{4.4}$$

The $H \times W$ sized sampling mask $\mathcal{D}$ is a low-pass filter that has zeros as entries if they are on the positions of high-frequency signals in ϕ and ones if they are on the low-frequency positions. Thus we can reconstruct the full spatial displacement field ϕ from $\mathbb{B}_\phi$ despite the latter being band-limited. Approaching the problem from the other side we can also think about working backwards from the final displacement. For this, after applying equation (4.1), the low-frequency signals are shifted to a center patch with size $\frac{H}{a} \times \frac{W}{b}$ with $a = 2 \cdot Z_a, b = 2 \cdot Z_b, Z_a, Z_b \in \mathbb{Z}^+$, which is then center-cropped to get $\mathbb{B}_\phi$. This crop of $\mathcal{F}(\phi)$ can be reconstructed using the iDFT from equation (4.4) with the cropping functioning as a kind of low-pass filtering:

$$[\mathbb{S}_\phi]_{n,m} = \frac{ab}{HW} \sum_{k=1}^{\frac{H}{a}-1} \sum_{l=1}^{\frac{W}{b}-1} [\mathbb{B}_\phi]_{k,l} \cdot \exp\left(i \cdot \left(\frac{2\pi an}{H} \cdot k + \frac{2\pi bm}{W} \cdot l\right)\right), \tag{4.5}$$

with $n \in [0, \frac{H}{a} - 1]$ and $m \in [0, \frac{W}{b} - 1]$ being the indices of the spatial domain, while $k \in [0, \frac{H}{a} - 1]$ and $l \in [0, \frac{W}{b} - 1]$ are the indices of the frequency domain with i being the imaginary unit. Thus $\mathbb{S}_\phi$ actually contains all the necessary information from ϕ, as long as they have the same low-frequency coefficients $\mathbb{B}_\phi$. This can be formulated as:

$$[\mathbb{S}_\phi]_{n,m} = ab \cdot \phi_{an,bm}, \tag{4.6}$$

because most entries of $\mathcal{F}(\phi)$ are zeros, and the remaining values are exactly the same as in $\mathbb{B}_\phi$, which means that $\mathbb{S}_\phi$ contains all the information ϕ can provide. For this, however, $\mathbb{B}_\phi$ needs to be padded to the original image size first in order to get the full-resolution displacement and not a low dimensional representation. This ultimately shows that there is a unique mapping between $\mathbb{S}_\phi$ and ϕ, which means that it is reasonable to use a network to learn $\mathbb{S}_\phi$ directly from image pairs and then reconstruct the displacement field in a very efficient manner [7]. The complete reconstruction process is visualized in Figure 4.3.

As both padding and iDFT layers are differentiable, *Fourier-Net* can be optimized via backpropagation. For *Diff-Fourier-Net* extra squaring and squaring layers [33] are needed in the decoder turning the displacement field into a stationary velocity field. Typically seven scaling and squaring layers are used to impose such a diffeomorphism [7, 33].

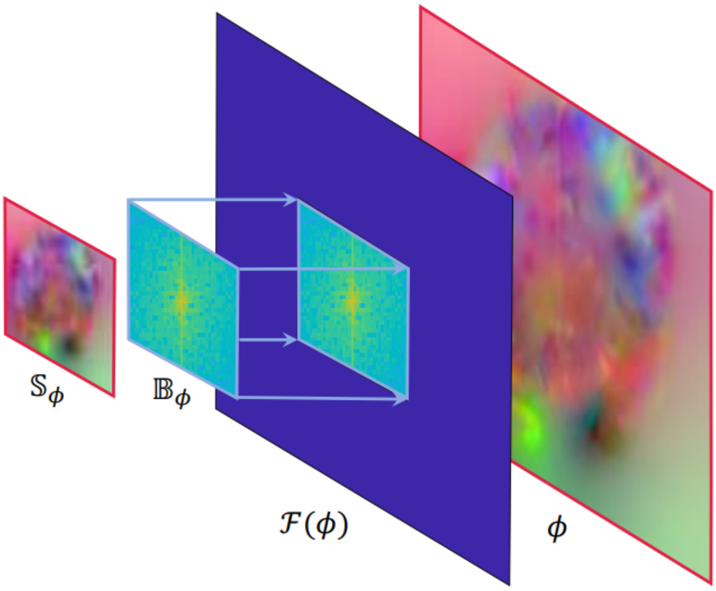


Figure 4.3: Reconstruction of the displacement field via the decoder taken from [8].

Diffeomorphic Transforms

Diffeomorphic deformations are differentiable and invertible, thus preserving topology, which is a desirable property for transformations. $\phi : \mathbb{R}^N \to \mathbb{R}^N$ represents the deformation that maps the coordinates from one image to coordinates in another image, as long as both images have the dimension N. When using a stationary velocity field representation like e.g. *DARTEL* [36], the deformation field is defined through the following ordinary differential equation (ODE) [33]:

$$\frac{\partial \phi^{(t)}}{\partial t} = v(\phi^{(t)}), \tag{4.7}$$

where $\phi^{(0)} = \mathrm{Id}$ is the identity transformation and t is time. The final registration field $\phi^{(1)}$ can be obtained by integrating the stationary velocity field v over $t = [0, 1]$. This is typically done by integration numerically using scaling and squaring [37]. The integration of a stationary ODE represents a one-parameter subgroup of diffeomorphisms. In group theory, v is a member of the Lie algebra and is exponentiated to produce $\exp(v) = \phi^{(1)}$, which is also a member of the Lie group. From the properties of one-parameter subgroups, for any scalars t and t', $\exp((t + t') \cdot v) = \exp(t \cdot v) \circ \exp(t' \cdot v)$, where $\circ$ is a composition map associated with the Lie group. Starting from $\phi^{(1/2^T)} = p + v(p)$ where

p is a map of spatial locations, we use the recurrence $\phi^{(1/2)^{t-1}} = \phi^{(1/2^t)} \circ \phi^{(1/2^t)}$ to obtain $\phi^1 = \phi^{(1/2)} \circ \phi^{(1/2)}$. T is chosen so that $v \approx 0$ [33].
As diffeomorphic deformations are defined as smooth and invertible deformations, the output of the iDFT layer in *Fourier-Net* can be regarded as a stationary velocity field v instead of a displacement field ϕ. In Figure 4.1 scaling and squaring layers are visualized. These apply the diffeomorphic transformation in three steps [37]:

1. Scaling: Divide the velocity field v by a factor 2^N, so that $\frac{v}{2^N}$ is close to zero (depending on the desired accuracy).
2. Exponentiation: Compute $\exp(\frac{v}{2^N}) = \phi^{(1)}(\frac{v}{2^N})$ with a first-order explicit numerical scheme.
3. Squaring: N recursive squarings of $\phi^{(1)}(\frac{1}{2^N}) = \exp(\frac{v}{2^N})$ to yield an accurate estimation of $\phi^{(1)}(1) = \phi^{(1)}(\frac{1}{2^N})^{2^N} = \exp(\frac{v}{2^N})^{2^N} = \exp(v)$.

Thus, the diffeomorphic deformation can be efficiently calculated. When used, the specific version is then called *Fourier-Net Diff* to differentiate it from the baseline version.

Spatial Transformer

The warping layer of *Fourier-Net* utilizes the *Spatial Transformer* [38], which allows for spatial image manipulation within the network. This is a differentiable and learnable module for neural networks which applies a spatial transformation to a feature map during a single forward pass. The spatial transformer mechanism is split into three parts as seen in Figure 4.4. First is the localization network, which takes the input and outputs the parameters for the transformation. These are then used to create a sample grid using the grid generator. Lastly, the sampler produces the output feature map based on the input at the grid points.
From the input feature map $U \in \mathbb{R}^{H \times W \times C}$ with width W, height H and channels C the localization network f_{loc} computes the parameters $\theta = f_{\text{loc}}(U)$ of the transformation $\mathcal{T}_\theta$ which is later applied to the feature map. Thus the size of θ varies depending on the transformation. The localization network function can both be implemented as a fully-connected network or as a CNN, but should include a final regression layer to produce the transformation parameters.
In order to warp the input feature map, each output pixel is computed by applying a sampling kernel centered at a particular location in the input feature map. The output pixels are defined to lie on a regular grid $G = G_i$ of pixels, forming an output feature map $V \in \mathbb{R}^{H' \times W' \times C}$, where H' and W' are the height and width of the grid with C again being the number of channels, which is the same for input and output.
In order to perform a spatial transformation of the input feature map U, the sampler must take the set of sampling points $\mathcal{T}_\theta(G)$, along and produce the sampled output feature map V. Each coordinate (x_i^s, y_i^s) in $\mathcal{T}_\theta(G)$ defines the spatial location in the input where a sampling kernel is applied to get the value at a particular pixel in the output:

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y), \tag{4.8}$$

with Φ_x and Φ_y being the parameters for a generic sampling kernel k that defines the image interpolation, U^c_{nm} is the value of the input feature maps at location (n, m) in the channel $c \in [1, ..., C]$ and V^c_i is the value for every pixel $i \in [1, ..., H'W']$ for the output feature map. Any sampling kernel can be used, as long as (sub-) gradients can be defined with respect to (x^s_i, y^s_i) to allow the loss gradients to flow back not only to the input feature map, but also to the sampling grid coordinates and therefore back to the transformation parameters θ and localization network, thus enabling back-propagation [38].

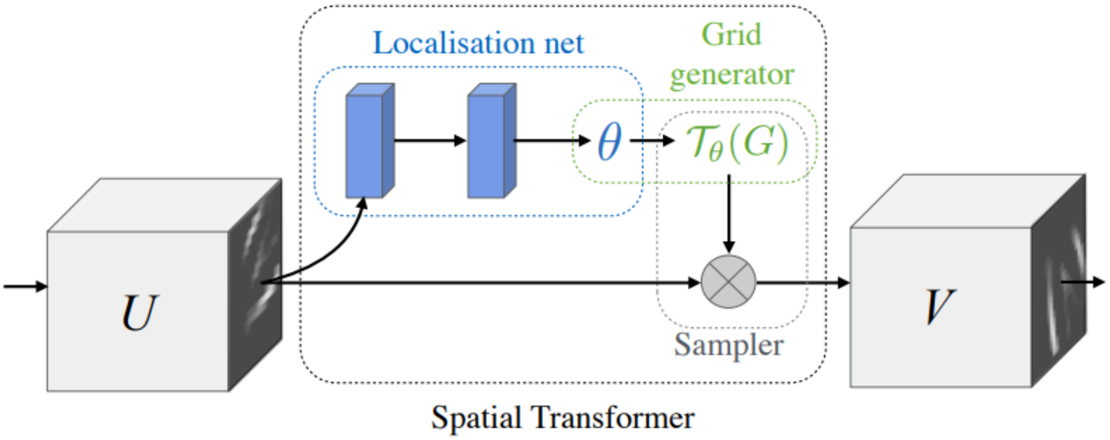


Figure 4.4: Architecture of the *Spatial Transformer* taken from [38].

Loss Function

The loss function consists of two parts to enable unsupervised learning, which are balanced using the scalar parameter λ. The first, $\mathcal{L}_1$, measures the similarity between the fixed image and the moving image after warping, while the second, $\mathcal{L}_2$, ensures a smooth displacement field. Thus, the unsupervised loss $\mathcal{L}$ can be calculated as follows:

$$\begin{aligned} \mathcal{L}(\Theta) &= \min\Big(\mathcal{L}_1(\phi(\Theta)) + \lambda \cdot \mathcal{L}_2(\phi(\Theta))\Big) \\ &= \min\Big(\mathcal{L}_1(v(\Theta)) + \lambda \cdot \mathcal{L}_2(v(\Theta))\Big), \end{aligned} \tag{4.9}$$

for both displacement fields ϕ and velocity fields v. The first part of the loss function consists of:

$$\mathcal{L}_1(\phi(\Theta)) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{Sim}(M_i \circ (\phi_i(\Theta) + \mathrm{Id}) - F_i), \tag{4.10}$$

where $\circ$ denotes the warping operation, N the number of training pairs with moving images M_i and fixed images F_i, Θ the network parameters, ϕ_i the displacement field, Id the identity grid. $\mathcal{L}_{Sim}$ determines the similarity between warped moving images and fixed images via MSE or NCC, and the second term of the unsupervised loss, $\mathcal{L}_2$, defines the smoothness regularization function that controls smoothness of the displacement fields:

$$\mathcal{L}_2(\phi(\Theta)) = \frac{1}{N}\sum_{i=1}^{N} ||\nabla\phi_i(\Theta)||_2^2, \tag{4.11}$$

with ∇ denoting the first order gradient and $||\cdot||_2^2$ denoting the squared L_2-Norm.
When using the squaring and scaling layers, thus making the deformation of the moving image diffeomorphic, the loss needs to be modified by replacing the displacement field θ with the velocity field v. Thus, both parts of the of the loss function need to be changed:

$$\mathcal{L}_1(v(\Theta)) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{Sim}(M_i \circ Exp(v_i(\Theta) - F_i), \tag{4.12}$$

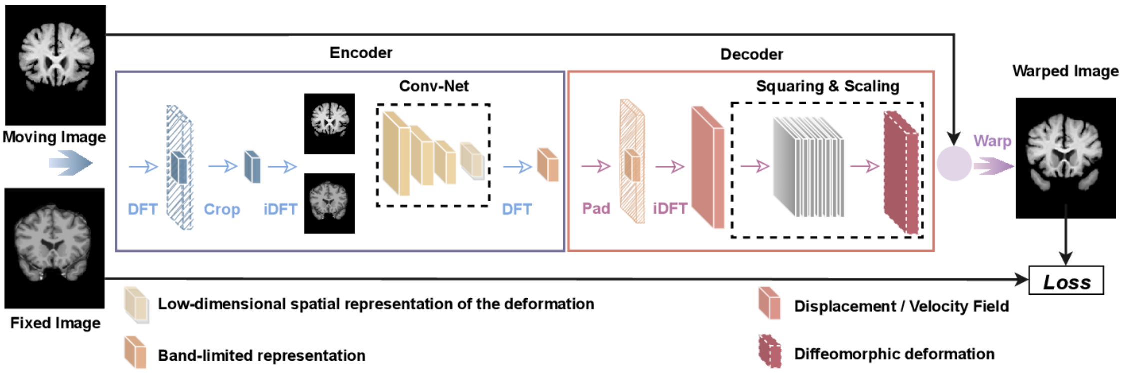
$$\mathcal{L}_2(v(\Theta)) = \frac{1}{N}\sum_{i=1}^{N} ||\nabla v_i(\Theta)||_2^2 \tag{4.13}$$

4.2 Fourier Net+

Fourier-Net+, as the name suggests, is an extension of Fourier-Net which takes the band-limited spatial representation of the images as input, instead of their original full-resolution counterparts. This leads to further reduction in the number of convolutional layers in the contracting path of the network, resulting in a decrease of parameters, memory usage, and computational operations. This makes *Fourier-Net+* even more efficient than its predecessor [8].
As seen in Figure 4.5, the network architecture is almost the same as for *Fourier-Net* (see Figure 4.1 for comparison). However, while the decoder, and thus the loss function, remain the same, the encoder is slightly altered to make the network even more efficient. For this, similarly to the decoder, a DFT is used, however this time the idea is applied to the input images. These are first transformed into the Fourier domain, then low-pass filtered by center-cropping and finally reconstructed from their band-limited representation back into the spatial domain via an iDFT. The two images, now compressed, are the input for the encoder of *Fourier-Net*, meaning the CNN and following DFT. However, due to the band-limiting before the CNN, the latter can be made much more light-weight, thus reducing computational cost. This is visualized in Figure 4.7. Thus, *Fourier-Net+* too is overall lighter than the baseline *Fourier-Net* in terms of the number of parameters and computations. However, such a light network may face limitations in accurately capturing complex deformations. To counter this potential weakness, the authors propose a cascaded version of *Fourier-Net+*, which uses multiple versions of *Fourier-Net+* cascaded one after the other to achieve a better overall displacement field [8]. A schematic for this can be seen in Figure 4.8.

Changes to the Encoder

In order to further reduce the amount of computational operations, *Fourier-Net+* discards the early layers of the encoder. Instead, a DFT $\mathcal{F}(I_M)$ followed by a center-crop to produce



Moving Image
Fixed Image
Encoder
Conv-Net
DFT
Crop
iDFT
DFT
Decoder
Squaring & Scaling
Pad
iDFT
Warp
Warped Image
Loss
Low-dimensional spatial representation of the deformation
Band-limited representation
Displacement / Velocity Field
Diffeomorphic deformation

Figure 4.5: Architecture of *Fourier-Net+* taken from [8].

the band-limited representation $\mathbb{B}_{I_M}$ in the frequency domain and iDFT are used to get the spatial patch $\mathbb{S}_{I_M}$, while the rest of the encoder from *Fourier-Net* stays the same. The input I_M is thus compressed to a lower resolution (i.e. band-limited) using the frequency space, which reduces the computational cost. This process is visualized in Figure 4.6. The encoder of *Fourier-Net+* has several convolutional layers less in the contracting path compared to *Fourier-Net*, which leads to a further accelerated registration process while reducing the memory footprint. These advantages are visualized in Figure 4.7 where the amount of different layers between a conventional *U-Net*, *Fourier-Net* (with the smaller decoder) and *Fourier-Net+* (with a smaller encoder and decoder) are shown.

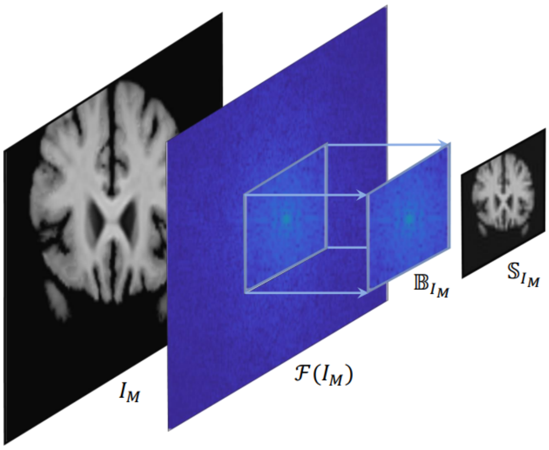


Figure 4.6: Compression in the frequency domain of the encoder used in *Fourier-Net+* taken from [8].

Effects of Cascading

As seen in the previous section, *Fourier-Net+* is lighter than *Fourier-Net* due to the band-limited representation of both images and deformations lowering the number of parameters and computations. This, however, can lead to limitations when trying to accurately capture complex deformations. To this end, a cascaded version of *Fourier-Net+*

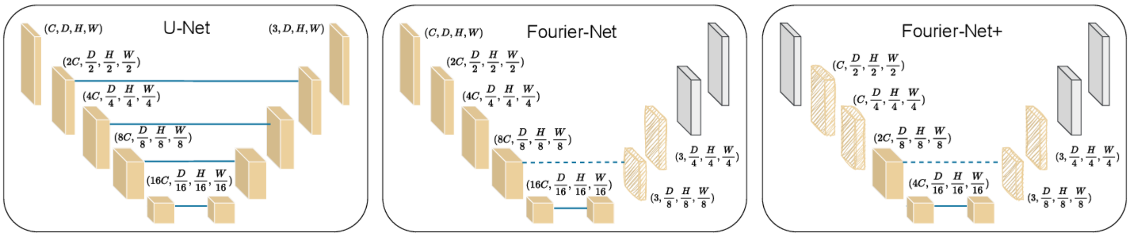


Figure 4.7: Architecture of the CNN for a typical U-Net, *Fourier-Net* and *Fourier-Net+* taken from [8].

called $K\times$*Fourier-Net+* (see Figure 4.8), with K denoting the amount of cascades, can be used where the the warped image of one cascade is used as the moving image of the next. It is important to note that the weights are not shared between cascades. The squaring and scaling layers, in case a diffeomorphic deformation is wanted, are applied after the last cascade. The same is true for the calculation of the loss function.

In order to more accurately describe this version of *Fourier-Net+* one can look at the in- and outputs of the different cascades. The first cascade of $K\times$*Fourier-Net+* has the moving image I_M and fixed image I_F as inputs. While the latter always stays the same for all cascades the warped image $I_M^{w(1)}$ from the first cascade is used as input for the second cascade instead of the original moving image. Thus, in general $I_M^{w(k-1)}$ and I_F are the inputs for a cascade $k \in [1, K]$ with output $\delta\phi^{(k)}$. Furthermore, $I_M^{w(k)}$ can be defined as:

$$I_M^{w(k)} = ((((I_M \circ \delta\phi^{(1)}) \circ \delta\phi^{(2)}) \circ \dots) \circ \delta\phi^{(k-1)}) \circ \delta\phi^{(k)} = I_M \circ \phi^{(k)}, \tag{4.14}$$

with $\phi^{(k)}$ being the displacement field computed by composing the outputs of the first cascade up to the k-th cascade:

$$\phi^{(k)} = \delta\phi^{(1)} \circ \delta\phi^{(2)} \circ \dots \circ \delta\phi^{(k-1)} \circ \delta\phi^{(k)}. \tag{4.15}$$

Thus, the output displacement field of the K-th cascade $\phi^{(K)}$ is the final displacement which is then used to warp I_M, the original moving image, in order to compute the loss [8].

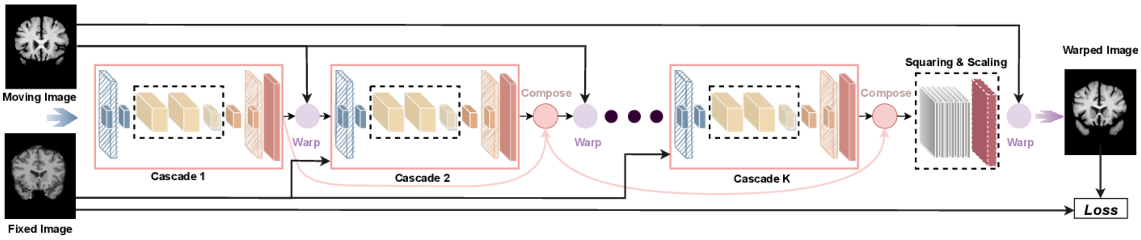


Figure 4.8: Cascaded version of *Fourier-Net+* taken from [8].

5

Experiments

In the following chapter the different experiments that were conducted are described and explained. First, experiments were conducted on the *CMRxRecon* dataset to find the optimal parameters for our model. As this dataset lacks ground truth segmentations, the *ACDC* dataset was used to extend these tests.

5.1 Parameter Studies on the CMRxRecon Dataset

As a first step, frame-to-frame registration on the *CMRxRecon* dataset was tested. The goal of these initial experiments was to examine whether the network could learn to align the cardiac image pairs despite the movement between the frames. To start, different parameters were tested in order to find the optimal model parameters for the *CMRxRecon* dataset.
As discussed in section 3.2, the CMR images were cropped to extract the moving cardiac region. This was done for both the fully sampled and subsampled (Acc4, Acc8, Acc10) data. Image pairs for each slice from each patient of the training set were generated to train the network. In order to evaluate the training progress the mean MSE and mean SSIM metrics calculated the validation data at each checkpoint (usually every 1000 iterations). The best model parameters, as determined by the metrics, were saved in a separate folder. To further evaluate the training success the unseen test data containing new patients was used. To benchmark the networks training success the MSE and SSIM were again used, but the mean inference time and the percentage of negative Jacobian determinant of deformation (% $|J_\phi| \leq 0$) were also calculated. The procedure, however, was slightly altered for the subsampled data as only the displacement field of the model was used to align the corresponding fully sampled image pairs. The evaluation metrics were then calculated on these images for better comparability.

Fourier-Net versus Fourier-Net+

First, the baseline network *Fourier-Net* was compared to its newer version *4xFourier-Net+* once with both using the diffeomorphic version and once without the transform. Thus the effect of the diffeomorphic deformation can be compared to the normal transformation on *Fourier-Net* and *4xFourier-Net+*. All of the experiments were conducted on the fully sampled data as the subsampling should not effect the tested properties. The other parameters were also kept constant with a learning rate of 0.0001, MSE-loss and $\lambda = 0.01$. FT crop in *4xFourier-Net+* was set to 24 × 24.

Starting Channel Size

Next, the impact of the starting channels was examined. For this, *4xFourier-Net+* was trained with MSE-loss, a learning rate of 0.0001, $\lambda = 0.01$, FT crop of 24 × 24 and no diffeomorphic transform. The experiments were again only conducted on the fully sampled data. Three different channel sizes were used for training *Fourier-Net+*: 8, 16 and 32. Due to the early stopping the networks were trained for 17, 17 and 10 epochs respectively.

Quantitative Comparison with NiftyReg

To establish a baseline, the similarity metrics were calculated between the unaligned test data. *4xFourier-Net+* was then trained with the MSE-loss, 8 channels, $\lambda = 0.01$ and a learning rate of 0.0001. *NiftyReg*, a traditional registration algorithm that iteratively optimizes the displacement [39], was used for comparison. *NiftyReg* provides the warped image for the fully sampled data directly, but for the subsampled data the control point grid produced by *NiftyReg* was used to resample the corresponding fully sampled data for comparability. Thus only the fully sampled test data is needed for an unregistrated baseline as the subsampled data is only used to produce the displacement which is then used to warp the fully sampled data.

5.2 Parameter Tests on the ACDC Dataset

As the evaluation on the *CMRxRecon* dataset is limited to image similarity measures, further testing was done on the *ACDC* dataset which contains cardiac data with segmentations. These can be used to better evaluate the registration performance by using e.g. the Dice score.

Fourier-Net versus Fourier-Net+

First, *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net* are compared for both normal and diffeomorphic versions. Additionally, network versions creating dense instead of band-limited displacements were used a baseline for best achievable performance. The registration performance was evaluated on the test set using the percentage of Dice scores calculated with and without the background label, percentage of SSIM, MSE multiplied by 10^{-3} (denoted by m for milli) and the percentage of negative Jacobian determinant of deformation. The mean inference time on GPU together with memory consumption for each model is also added to the comparison with the latter containing number of trainable model parameters, mult-add operations (in millions or billion as denoted by M or G) and the total memory in Megabyte (MB). The latter is only given for the normal versions as the diffeomorphic version require the same amount of memory and the baseline dense versions. The first network was trained for a maximum of 100 epochs with early stopping activating after 3 epochs without improvement of the validation Dice score. The other variants were than trained with the same number of epochs, which worked out to be only 6 epochs due to the large training set in the experiments. All networks were trained on the fully sampled *ACDC* data with MSE as the unsupervised similarity loss, a channel size of 8, learning rate of 0.0001 and $\lambda = 0.01$. *Fourier-Net+* and *4xFourier-Net* used a FT crop size of 48×48 for these experiments. All of these parameters were the same for the diffeomorphic and dense versions of the networks. The results and subsequent discussion can be found in section 6.2.

Starting Channel Size

Next, the impact of the starting channels on the *ACDC* data was examined. For this, *Fourier-Net+* and *4xFourier-Net+* was trained with MSE-loss, a learning rate of 0.0001, $\lambda = 0.01$, FT crop of 24×24 and no diffeomorphic transform. The experiments were again only conducted on the fully sampled data. Four different channel sizes were used for training: 8, 16, 32 and 64. The first network variant was trained with early stopping, which ended at 6 epochs. For better comparability, the other network variants were also trained with the same number of epochs. The registration performance was evaluated on the test set using the percentage of the Dice score calculated once on all labels, once without the background using only the cardiac labels, as well as the percentage of SSIM, MSE multiplied by 10^-3 and the percentage of negative Jacobian determinant of deformation. Additionally the number of network parameters, number of mult-add operations (in millions or billion as denoted by M or G) and memory consumption in Megabyte (MB) are listed for all network variants to better compare the relationship between registration performance and memory consumption to potentially find a optimal configuration for both performance and efficiency. The mean inference time was measured on GPU in seconds. The results and subsequent discussion can be found in section 6.2.

Fourier-Transform Crop Size

The FT crop size used for compressing the images is a parameter specific to *Fourier-Net+* and *4xFourier-Net+*. Four different sizes of the FT crop were analyzed: 80×168, 40×84, 48×48 and 24×24. A larger crop would obviously compress the images less, thus leading to less efficiency, however, a very small crop, while very efficient, might lose some of the image details leading to a decrease in performance. The experiments were again only conducted on the fully sampled data as one would expect similar results on subsampled data. The network variants were trained with MSE as the unsupervised similarity loss, a channel size of 8, learning rate of 0.0001 and $\lambda = 0.01$. Again, the networks were trained for only 5 epochs, as this is were the first model ended training due to early stopping. The registration performance was evaluated on the test set using the Dice score calculated once on all labels, once without the background, as well as the SSIM, MSE and the percentage of negative Jacobian determinant of deformation. Additionally, the number of network parameters, number of mult-add operations in millions (M) and memory consumption in Megabyte (MB) are listed for all network variants to better compare the relationship between registration performance and memory consumption to potentially find a optimal configuration for both performance and efficiency. The mean inference time was measured on GPU in seconds. The results and subsequent discussion can be found in section 6.2.

Comparison with VoxelMorph

After finding parameters which optimize registration performance while minimizing memory consumption, a comparison to another commonly used registration network should be made. For this, the famous *VoxelMorph* [6] was used. It was trained with ... and provides a dense displacement field. For comparison *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* were trained for 6 epochs with MSE as the unsupervised similarity loss, a channel size of 16, learning rate of 0.0001 and $\lambda = 0.01$ as well as a FT crop size of 48×48, which provide a good balance of performance and memory imprint for the networks. To evaluate registration performance, the Dice score was again computed for all labels and without the background as well as the similarity metrics SSIM, MSE and the percentage of negative Jacobian determinant of deformation. All networks were trained on the fully sampled *ACDC* data. The inference times of all networks were computed on the GPU and the number of network parameters, number of mult-add operations in billions (G) and memory consumption in Megabyte (MB) are also given to further compare the efficiency of the networks.

Dense Displacement on Accelerated Data

The difference between a dense displacement field and a band-limited one was already explored in section 5.2. This, however, only includes fully sampled data, not accelerated data. We will explore potential performance changes on the latter in this section with the hypothesis that the network variants with the dense displacement will perform worse than the variants with the band-limited displacement field. This expectation stems from the fact, that frequencies outside the center region of the k-space are dropped for the accelerated data thus reducing the advantage of the dense displacement in comparison to the band-limited (i.e. center-cropped) displacement.
The tests again included *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+*, which were all trained for 6 epochs with MSE as the unsupervised similarity loss, a channel size of 16, learning rate of 0.0001 and $\lambda = 0.01$. The registration performance was evaluated on the test set using the Dice score calculated once on all labels, once without the background, as well as the SSIM, MSE and the percentage of negative Jacobian determinant of deformation. Additionally, the number of network parameters, number of mult-add operations in millions (M) and memory consumption in Megabyte (MB) are given.

Comparison on Subsampled Data

After comparing *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields as well as a comparison with *VoxelMorph*, which also has a

dense displacement field, on fully sampled data to access the difference in registration performance and memory consumption an extension to subsampled data needs to be made. For this *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* were compared to a baseline consisting of the unaligned test image pairs as well as *NiftyReg*, *VoxelMorph* and *LAPNet*. Thus, observations of the performance on accelerated data can be made as well as comparisons to the aligned baseline (which all methods should hopefully beat), a traditional registration algorithm, a dense registration network working in the image domain and a another dense network which works in the k-space domain.
Fourier-Net, *Fourier-Net+* and *4xFourier-Net+* were trained for 6 epochs with MSE as the unsupervised similarity loss, a channel size of 16, learning rate of 0.0001 and $\lambda = 0.01$ as well as a FT crop size of 48×48 for all subsampling data. VoxelMorph was trained with ... Three different accelerations were tested: *Acc4* (R=4), *Acc8* (R=8) and *Acc10* (R=10). For reference, the data for the fully sampled data (R=0) is also provided. The registration performance of all methods was evaluated on the test set using the Dice score calculated once on all labels, once without the background, as well as the SSIM and MSE as image metrics. All times are computed on CPU for better comparability as *NiftyReg* currently only works on CPU as the GPU capable versions were deprecated and the other networks would gain an unfair advantage when ran on GPU. The baseline obviously has no time associated as it only denotes the metrics on the test data before registration.

5.3 Data Consistency Experiments

As seen in the previous sections, the performance for all methods decreases drastically for highly subsampled/accelerated data due to reconstruction artifacts in the image domain. However, since we are interested in maintaining performance on subsampled data, it might prove useful to use k-space information for the loss calculation and thus for the network training as it should not contain artifacts compared to the image domain. Thus, it would act as a kind of data consistency contraint during training. In order to achieve this, the similarity loss was calculated not on the warped and fixed images, but instead on their corresponding k-spaces. This would hopefully still provide the network with useful information during training that is free of any aliasing artifacts, which might decrease performance on subsampled data.
After trying the MSE as the similarity loss, it quickly became apparent that this yielded unrealistic and wildly deformed displacements which could no longer be reigned in by the smoothness constraint of the loss function. Thus, normalized cross-correlation (NCC) was used for the calculation of the similarity loss with $\lambda = 1$ to place a greater emphasize on the smoothness of the displacement.

5.4 Integration into a Motion-Corrected Reconstruction Pipeline

Describe qualitative test to see whether the network can be used to improve a reconstruction pipeline for cardiac data.

General Reconstruction Pipeline

Overview of the reconstruction pipeline using *Fourier-Net+*.

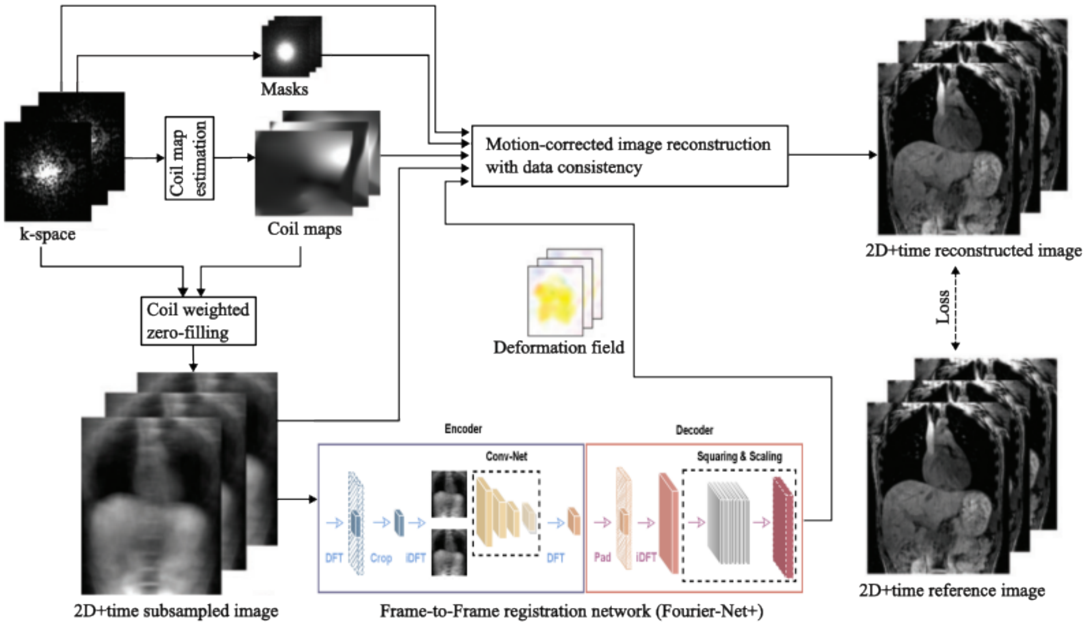


Figure 5.1: Overview of the general reconstruction pipeline using *Fourier-Net+* as a registration network.

6

Results and Discussion

In this chapter, the results for the experiment described in the previous chapter will be examined. Possible implications will be discussed to determine the effectiveness of this new method.

6.1 Parameter Studies on the CMRxRecon Dataset

Different parameters were tested in order to find the optimal model parameters for usage on the *CMRxRecon* dataset. After some tests on the fully sampled data, a comparison with the traditional iterative algorithm *NiftyReg* was conducted on both the fully sampled and subsampled data.

Fourier-Net versus Fourier-Net+

First, the baseline network *Fourier-Net* was compared to its more efficient successor *4xFourier-Net+* with both using the diffeomorphic version as we can then compare the effect of the diffeomorphic deformation compared to the normal transformation on *4xFourier-Net+*. However, this effect should be similar for *Fourier-Net*. All of the experiments were conducted on the fully sampled data as the subsampling should not effect the tested properties. The baseline MSE (note that all MSE values are multiplied by $10^{-}3$ as denoted by the m) for unregistrated test images is 0.535 and the SSIM percentage (% SSIM) is 91.71. The results for *Fourier-Net* and *4xFourier-Net+* can be seen in Table 6.1.
The *Fourier-Net* model without the diffeomorphic transform was trained for 17 epochs, while the diffeomorphic version trained for 100 epochs. Despite this, the latter is only marginally better in terms of MSE and SSIM. The percentage of negative Jacobian determinants was actually better for the normal transformation compared to the diffeomorphic one. The normal *Fourier-Net+* model trained for 37 epochs, while the diffeomorphic version trained for 100 epochs. *Fourier-Net+* performs slightly worse than *Fourier-Net*, however it is more efficient.

Table 6.1: Results for *Fourier-Net* and *Fourier-Net+* on the fully sampled *CMRxRecon* test data with a diffeomorphic transform.

Model	MSE (m)	% SSIM	% $\lvert J_\phi \rvert \leq 0$	Time [s]
Fourier-Net	0.319 ± 0.352	93.29 ± 3.85	0.0022 ± 0.0183	0.0025
4xFourier-Net+	0.352 ± 0.449	93.08 ± 4.09	0.0003 ± 0.0057	0.0123
Diff-Fourier-Net	0.318 ± 0.388	93.39 ± 3.82	0.0273 ± 0.0897	0.0028
Diff-4xFourier-Net+	0.352 ± 0.448	93.08 ± 4.11	0.0029 ± 0.0214	0.0103

Starting Channel Size

Next, the impact of the starting channels was examined. For this, *4xFourier-Net+* was trained with three different channel sizes: 8, 16, 32 and 64. Due to the early stopping the *4xFourier-Net+* variants were trained for x, 17 and 10 epochs respectively. The results can be seen in Table 6.2 with the inference time for each version being calculated on the GPU. Despite training for the least amount of epochs channel size 32 performs the best for *4xFourier-Net+*.

Table 6.2: Results for three different starting channel sizes of *Fourier-Net+* on the fully sampled *CMRxRecon* test data.

Starting Channels	MSE (m)	% SSIM	% $\lvert J_\phi \rvert \leq 0$	Time [s]
8	0.352 ± 0.449	93.08 ± 4.09	0.0003 ± 0.0057	0.0123
16	0.347 ± 0.439	93.12 ± 4.04	0.0003 ± 0.0059	0.0113
32	0.344 ± 0.429	93.13 ± 4.01	0.0008 ± 0.0098	0.0106

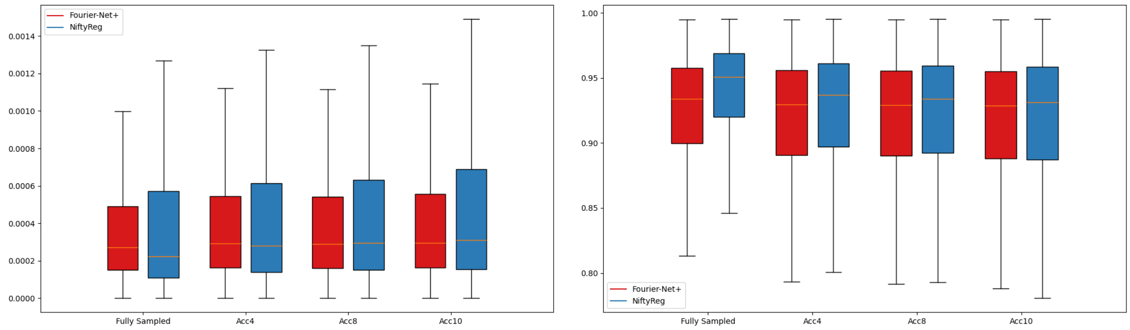
Quantitative Comparison with NiftyReg

For this first quantitative test similarity metrics such as MSE and SSIM were used to judge image similarity as an indicator for proper image alignment. The results can be seen in Table 6.3 with the SSIM given in percent % and the MSE multiplied by $10^{-}3$ for better readability (this is denoted by m for milli although the MSE has no unit strictly speaking). They are further visualized in Figure 6.4a for the MSE and Figure 6.4b for the SSIM results.

Fourier-Net+ needed about 0.0104 seconds (s) per frame pair for registration with $\lvert J \rvert_{<0}\% < 0.001$ for all levels of subsampling, while *NiftyReg* took up to 15 s for aligning an image pair. The results for *Fourier-Net+* look good on their own as the metrics are stable even for the subsampled data compared to the fully sampled data. However, its baseline performance is slightly worse than *NiftyReg* except for the Acc10 (R=10) data, where both perform worse than the unalinged baseline of the test data, however *4xFourier-Net+* is slightly better than *NiftyReg*. Overall, the network is much faster than the iterative registration algorithm as expected, but the latter performs better in terms of similarity metrics.

Table 6.3: Benchmark results for *Fourier-Net+* and *NiftiReg* on the *CMRxRecon* test data.

Subsampling	Method	MSE (m)	% SSIM
R=0	4xFourier-Net+	0.435 ± 0.554	92.35 ± 4.59
	NiftyReg	0.514 ± 0.771	93.97 ± 3.91
R=4	4xFourier-Net+	0.482 ± 0.626	91.81 ± 5.01
	NiftyReg	0.551 ± 0.798	92.38 ± 4.94
R=8	4xFourier-Net+	0.479 ± 0.620	91.74 ± 5.12
	NiftyReg	0.565 ± 0.803	92.06 ± 5.09
R=10	4xFourier-Net+	0.491 ± 0.640	91.59 ± 5.23
	NiftyReg	0.926 ± 2.867	90.66 ± 10.32



(a) Boxplot of MSE results.

(b) Boxplot of SSIM results.

Figure 6.4: Boxplots for MSE and SSIM results of the test set for fully sampled and subsampled (Acc4, Acc8, Acc10) data for both *Fourier-Net+* and *NiftyReg*.

6.2 Further Tests on the ACDC Dataset

Due to the limitations of the *CMRxRecon* dataset in terms of segmentations, the registration performance is further evaluated on the *ACDC* dataset. Some of the parameter studies done on *CMRxRecon* were repeated, others were added to further test the networks ability.

Fourier-Net versus Fourier-Net+

Again, *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net* are compared in terms of registration performance and inference time (on GPU) together with their diffeomorphic versions, however, versions with dense instead of band-limited displacements are also added as a baseline for peak performance as these utilize all of the k-space information. The memory consumption is also added to the comparison with the number of parameters, mult-add operations and the total memory. The latter is only given for the dense and normal versions as the diffeomorphic version required roughly the same amount of memory as the normal ones, while the dense versions have a slightly larger encoder. Note that all network variants were trained for 6 epochs. The results can be seen in Table 6.5.

Table 6.5: Results for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields as well as diffeomorphic transforms on the fully sampled *ACDC* test data.

Metrics	Dense Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	78.22 ± 13.84	78.43 ± 13.96	79.34 ± 14.03
% Dice*	71.03 ± 18.87	70.50 ± 19.17	72.02 ± 18.94
% SSIM	91.92 ± 3.32	89.09 ± 4.10	89.65 ± 3.85
MSE (m)	0.09 ± 0.06	0.17 ± 0.13	0.15 ± 0.12
% $\lvert J_\phi\rvert \leq 0$	0.53 ± 0.52	0.32 ± 0.55	0.04 ± 0.09
Time [s]	0.0070	0.0107	0.0219
	Diff-Fourier-Net	Diff-Fourier-Net+	Diff-4xFourier-Net+
% Dice	78.56 ± 14.13	77.88 ± 13.81	79.49 ± 14.26
% Dice*	71.31 ± 19.13	70.19 ± 18.82	72.12 ± 19.33
% SSIM	91.79 ± 3.38	89.08 ± 4.10	89.62 ± 3.98
MSE (m)	0.09 ± 0.06	0.16 ± 0.13	0.15 ± 0.12
% $\lvert J_\phi\rvert \leq 0$	0.03 ± 0.06	0.00 ± 0.00	0.00 ± 0.00
Time [s]	0.0082	0.0075	0.0203
Parameters	645,216	380,470	1,521,880
Mult-Adds (G)	1.12	0.04	0.18
Memory [MB]	97.63	5.81	21.90
Metrics	Band-limited Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	77.95 ± 14.71	75.35 ± 14.28	76.59 ± 14.84
% Dice*	69.96 ± 21.55	64.74 ± 21.12	66.82 ± 21.55
% SSIM	91.35 ± 3.51	88.42 ± 3.94	88.59 ± 4.23
MSE (m)	1.00 ± 0.71	2.06 ± 1.54	1.92 ± 1.46
% $\lvert J_\phi\rvert \leq 0$	0.36 ± 0.38	0.13 ± 0.35	0.03 ± 0.12
Time [s]	0.0145	0.0145	0.0335
	Diff-Fourier-Net	Diff-Fourier-Net+	Diff-4xFourier-Net+
% Dice	77.93 ± 14.93	75.17 ± 13.43	78.20 ± 14.01
% Dice*	69.91 ± 21.99	65.67 ± 18.61	69.61 ± 19.11
% SSIM	91.24 ± 3.56	87.81 ± 3.94	88.81 ± 4.13
MSE (m)	0.98 ± 0.67	2.29 ± 1.61	1.85 ± 1.41
% $\lvert J_\phi\rvert \leq 0$	0.01 ± 0.03	0.00 ± 0.00	0.00 ± 0.00
Time [s]	0.0150	0.0207	0.0188
Parameters	434,519	75,429	301,716
Mult-Adds (M)	595.04	10.98	43.91
Memory [MB]	44.69	2.25	7.66

Starting Channel Size

Next, the impact of the starting channels was examined. For this, *Fourier-Net+* and *4xFourier-Net+* were trained with four different channel sizes: 8, 16, 32 and 64. All network variants were trained for 6 epochs. The results can be seen in Table 6.6. The memory consumption as well as inference time on the GPU were calculated to extend the evaluation given by the other metrics.

Table 6.6: Results for different starting channel sizes of *Fourier-Net+* and *4xFourier-Net+* on the fully sampled *ACDC* test data.

Metrics	Starting Channels - Fourier-Net+			
	8	16	32	64
% Dice	75.50 ± 13.79	76.88 ± 13.86	77.66 ± 13.60	77.81 ± 13.76
% Dice*	65.70 ± 18.96	67.86 ± 19.06	68.95 ± 18.65	69.05 ± 18.80
% SSIM	88.05 ± 3.89	88.67 ± 3.88	88.83 ± 3.83	89.02 ± 3.67
MSE (m)	0.22 ± 0.16	0.20 ± 0.15	0.19 ± 0.15	0.19 ± 0.15
% $\|J_\phi\| \leq 0$	0.07 ± 0.25	0.04 ± 0.14	0.01 ± 0.04	0.00 ± 0.03
Time [s]	0.0072	0.0072	0.0080	0.0079
Parameters	75,429	300,477	1,199,469	4,793,037
Mult-Adds (M)	10.98	42.89	169.54	674.10
Memory [MB]	2.25	4.64	11.22	31.57
Metrics	Starting Channels - 4xFourier-Net+			
	8	16	32	64
% Dice	77.54 ± 13.73	77.90 ± 13.92	78.33 ± 14.14	78.74 ± 13.91
% Dice*	68.52 ± 18.62	69.21 ± 19.02	69.99 ± 19.21	70.41 ± 19.02
% SSIM	88.70 ± 4.17	88.89 ± 4.05	89.08 ± 4.01	89.29 ± 3.74
MSE (m)	0.19 ± 0.14	0.19 ± 0.14	0.18 ± 0.14	0.18 ± 0.14
% $\|J_\phi\| \leq 0$	0.02 ± 0.07	0.01 ± 0.04	0.01 ± 0.05	0.00 ± 0.00
Time [s]	0.0301	0.0244	0.0297	0.0277
Parameters	301,716	1,201,908	4,797,876	19,172,148
Mult-Adds (G)	0.04	0.17	0.68	2.70
Memory [MB]	7.66	17.23	43.56	124.94

Fourier-Transform Crop Size

As a parameter specific to *Fourier-Net+* and *4xFourier-Net+*, the impact of the FT crop size used for compressing the images was analyzed. Four different sizes of the FT crop were used for training: 80 × 168, 40 × 84, 48 × 48 and 24 × 24. The experiments were again only conducted on the fully sampled data and each network trained for 5 epochs. The results can be seen in Table 6.7. Memory obviously decreases with smaller crop sizes for the same number of network parameters due to the smaller image size. However, the performance also decreases with crop size as seen with the Dice score, both with and without background labels. Jacoby determinants are small for all models and the inference times are very fast. It is notable that the SSIM and MSE metrics to not capture the deterioration of registration performance as well as the Dice score as the latter focuses more on the moving cardiac structure in the frames. To conclude, a balance between memory consumption and registration performance has to be found for *4xFourier-Net+* as both apparently cannot be maximized at the same time.

Table 6.7: Results for four different FT crop sizes for *Fourier-Net+* and *4xFourier-Net+* examined on the fully sampled *ACDC* test data.

Metrics	FT crop size - Fourier-Net+			
	80 × 168	40 × 84	48 × 48	24 × 24
% Dice	78.24 ± 14.43	76.61 ± 13.90	75.27 ± 13.49	73.77 ± 14.42
% Dice*	70.66 ± 19.70	67.60 ± 19.24	65.48 ± 18.76	63.37 ± 20.07
% SSIM	89.81 ± 4.09	88.58 ± 3.87	87.98 ± 3.91	87.00 ± 3.99
MSE (m)	0.15 ± 0.12	0.20 ± 0.15	0.22 ± 0.16	0.27 ± 0.19
% $\lvert J_\phi \rvert \leq 0$	0.22 ± 0.46	0.05 ± 0.15	0.09 ± 0.25	0.00 ± 0.02
Time [s]	0.0158	0.0077	0.0079	0.0081
Parameters	75,429	75,429	75,429	75,429
Mult-Adds (M)	64.03	16.37	10.98	2.74
Memory [MB]	9.51	2.97	2.25	1.12

Metrics	FT crop size - 4xFourier-Net+			
	80 × 168	40 × 84	48 × 48	24 × 24
% Dice	78.12 ± 15.01	77.49 ± 14.67	74.95 ± 14.15	72.58 ± 14.67
% Dice*	70.08 ± 21.92	68.03 ± 21.57	64.54 ± 20.75	61.19 ± 21.63
% SSIM	89.91 ± 4.01	89.03 ± 3.98	87.98 ± 3.94	87.02 ± 3.95
MSE (m)	1.45 ± 1.11	1.81 ± 1.37	2.22 ± 1.61	2.64 ± 1.84
% $\lvert J_\phi \rvert \leq 0$	0.12 ± 0.23	0.03 ± 0.15	0.06 ± 0.23	0.02 ± 0.15
Time [s]	0.0390	0.0301	0.0277	0.0289
Parameters	301,716	301,716	301,716	301,716
Mult-Adds (M)	256.14	65.50	43.91	10.98
Memory [MB]	36.70	10.54	7.66	3.15

Comparison with VoxelMorph

Compare *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+*, both of the latter with a FT crop size of 48 × 48 and all networks with channel size 16, to *VoxelMorph*. For this different similarity metrics as well as the Dice score for all labels (DICE) and Dice score without the background (DICE*) were used on the fully sampled data. Inference time and memory consumption are also listed for all models as these should be consistent with the subsampled data.

Table 6.8: Comparison of *Fourier-Net*, *Fourier-Net+*, *4xFourier-Net+* and *VoxelMorph* with similarity metrics and memory consumption on the fully sampled *ACDC* test data.

	Fourier-Net	Fourier-Net+	4xFourier-Net+	VoxelMorph
% Dice	78.31 ± 13.96	76.88 ± 13.86	77.90 ± 13.92	75.07 ± 14.29
% Dice*	71.17 ± 18.99	67.86 ± 19.06	69.21 ± 19.02	66.54 ± 21.22
% SSIM	91.53 ± 3.49	88.67 ± 3.88	88.89 ± 4.05	92.79 ± 3.41
MSE (m)	0.09 ± 0.07	0.20 ± 0.15	0.19 ± 0.14	0.74 ± 0.53
% $\lvert J_\phi \rvert \leq 0$	0.44 ± 0.45	0.04 ± 0.14	0.01 ± 0.04	0.*xx* ± 0.*xx*
Time [s]	0.0099	0.0071	0.0244	0.0161
Parameters	1,735,447	300,477	1,201,908	84,322
Mult-Adds (G)	2.35	0.04289	0.17157	0.00157
Memory [MB]	90.08	4.64	17.23	39.04

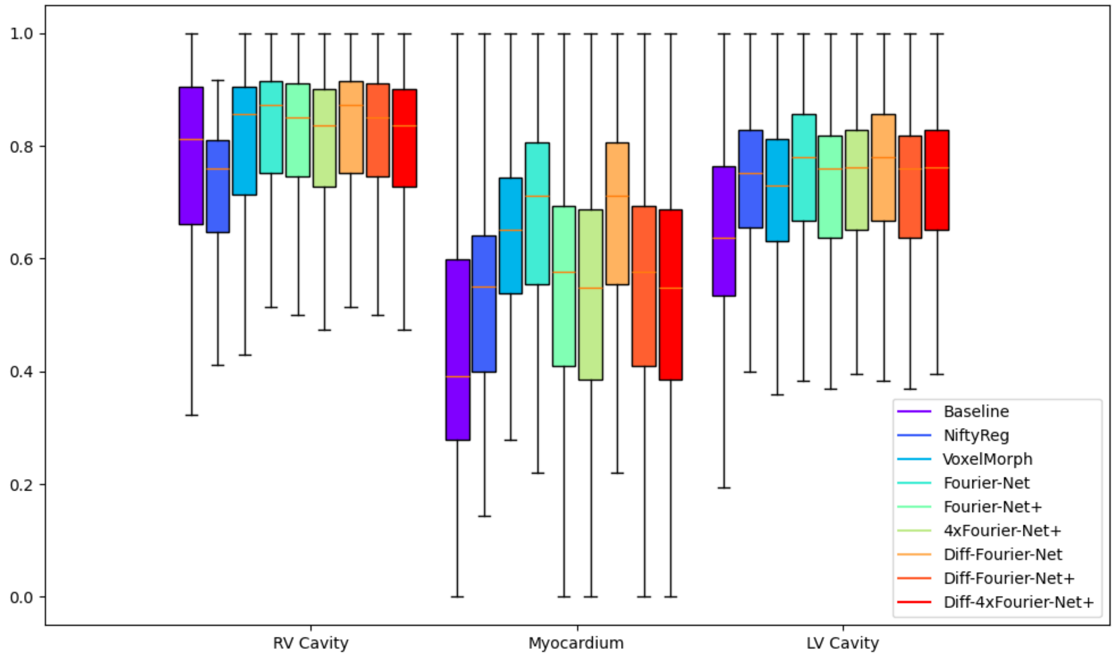


Figure 6.9: Boxplot of the Dice scores for all models sorted by label (excluding the background) on the fully sampled *ACDC* test data.

Dense Displacement on Accelerated Data

The difference between a dense displacement field and a band-limited one was already explored in section 6.2. However, the results in Table 6.5 only include fully sampled data, not accelerated data. The tests again included *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+*. Results for Acc4 data can be seen in Table 6.10, for Acc8 data in Table 6.11 and for Acc10 in Table 6.12. Out hypothesis that the dense displacements would perform worse compared to the band-limited ones on the accelerated data was…

Table 6.10: Results for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields on the Acc4 *ACDC* test data.

Metrics	Dense Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	67.84 ± 14.07	73.84 ± 15.18	72.83 ± 15.64
% Dice*	56.14 ± 18.45	64.31 ± 21.00	63.19 ± 21.59
% SSIM	71.91 ± 4.85	77.58 ± 10.64	77.32 ± 10.53
MSE (m)	0.28 ± 0.12	0.20 ± 0.14	0.21 ± 0.15
% $\vert J_\phi\vert \leq 0$	0.37 ± 0.37	0.15 ± 0.26	0.15 ± 0.27
Time [s]	0.0082	0.0092	0.0337
Metrics	Band-limited Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	72.30 ± 14.04	70.18 ± 17.43	70.36 ± 16.63
% Dice*	61.87 ± 19.32	59.36 ± 23.90	59.47 ± 22.80
% SSIM	78.77 ± 7.22	75.78 ± 10.63	75.91 ± 10.72
MSE (m)	0.16 ± 0.10	0.28 ± 0.19	0.27 ± 0.19
% $\vert J_\phi\vert \leq 0$	0.46 ± 0.54	0.00 ± 0.00	0.00 ± 0.00
Time [s]	0.0068	0.0079	0.0183

Comparison on Subsampled Data

Compare *Fourier-Net+* to a baseline of the unaligned test image pairs as well as *NiftyReg*, *VoxelMorph* and *LAPNet*. All times are computed on CPU due to *NiftyReg* only working on the CPU as the GPU capable versions were deprecated.

Table 6.11: Results for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields on the Acc8 *ACDC* test data.

Metrics	Dense Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	69.12 ± 14.05	–	–
% Dice*	57.99 ± 18.66	–	–
% SSIM	77.27 ± 3.74	–	–
MSE (m)	0.21 ± 0.10	–	–
% $\|J_\phi\| \leq 0$	0.47 ± 0.30	–	–
Time [s]	0.0117		
Metrics	**Band-limited Displacement**		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	66.07 ± 13.72	71.13 ± 15.37	73.13 ± 15.26
% Dice*	53.76 ± 17.03	60.10 ± 21.34	62.61 ± 21.39
% SSIM	73.34 ± 3.83	84.02 ± 4.36	84.61 ± 4.44
MSE (m)	0.26 ± 0.09	0.21 ± 0.16	0.19 ± 0.15
% $\|J_\phi\| \leq 0$	0.78 ± 0.37	0.00 ± 0.00	0.01 ± 0.04
Time [s]	0.0138	0.0133	0.0418

Table 6.12: Results for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields on the Acc10 *ACDC* test data.

Metrics	Dense Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	–	–	–
% Dice*	–	–	–
% SSIM	–	–	–
MSE (m)	–	–	–
% $\|J_\phi\| \leq 0$	–	–	–
Time [s]			
Metrics	**Band-limited Displacement**		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	–	–	–
% Dice*	–	–	–
% SSIM	–	–	–
MSE (m)	–	–	–
% $\|J_\phi\| \leq 0$	–	–	–
Time [s]			

Table 6.13: Benchmark results for *Fourier-Net+*, *NiftiReg* and *VoxelMorph* on the *ACDC* test data.

	Method	% DICE	% DICE*	% SSIM	MSE (m)	Time [s]
R=0	Baseline	70.85 ± 18.27	60.35 ± 25.24	86.39 ± 4.08	0.33 ± 0.23	-
	NiftyReg	70.74 ± 13.77	57.56 ± 18.86	91.26 ± 3.18	1.94 ± 1.61	20
	VoxelMorph	75.07 ± 14.29	66.54 ± 21.22	92.79 ± 3.41	0.74 ± 0.53	0.1845
	Fourier-Net	78.31 ± 13.96	71.17 ± 18.99	91.53 ± 3.49	0.09 ± 0.07	0.1918
	Fourier-Net+	76.88 ± 13.86	67.86 ± 19.06	88.67 ± 3.88	0.20 ± 0.15	0.0893
	4xFourier-Net+	77.90 ± 13.92	69.21 ± 19.02	88.89 ± 4.05	0.19 ± 0.14	0.3262
R=4	Baseline	70.85 ± 18.27	60.35 ± 25.24	76.80 ± 11.02	0.28 ± 0.19	-
	NiftyReg	69.89 ± 13.73	56.47 ± 17.86	86.03 ± 6.41	0.15 ± 0.14	80.08
	VoxelMorph	–	–	–	–	0.1858
	Fourier-Net	72.30 ± 14.04	61.87 ± 19.32	78.77 ± 7.22	0.16 ± 0.10	0.1890
	Fourier-Net+	70.18 ± 17.43	59.36 ± 23.90	75.78 ± 10.63	0.28 ± 0.19	0.0935
	4xFourier-Net+	70.36 ± 16.63	59.47 ± 22.80	75.91 ± 10.72	0.27 ± 0.19	0.3462
R=8	Baseline	70.85 ± 18.27	60.35 ± 25.24	85.35 ± 4.43	0.22 ± 0.17	-
	NiftyReg	70.04 ± 13.42	56.37 ± 17.63	91.07 ± 2.80	0.12 ± 0.13	88.36
	VoxelMorph	–	–	–	–	0.1725
	Fourier-Net	66.07 ± 13.72	53.76 ± 17.03	73.34 ± 3.83	0.26 ± 0.09	0.2109
	Fourier-Net+	71.13 ± 15.37	60.10 ± 21.34	84.02 ± 4.36	0.21 ± 0.16	0.1487
	4xFourier-Net+	73.13 ± 15.26	62.61 ± 21.39	84.61 ± 4.44	0.19 ± 0.15	0.5373
R=10	Baseline	70.85 ± 18.27	60.35 ± 25.24	89.17 ± 3.58	0.20 ± 0.17	-
	NiftyReg	70.40 ± 13.34	56.61 ± 17.71	93.47 ± 2.37	0.11 ± 0.13	47.44
	VoxelMorph	–	–	–	–	
	Fourier-Net	–	–	–	–	
	Fourier-Net+	–	–	–	–	
	4xFourier-Net+	–	–	–	–	

7

Conclusion

Summery of all stuff…

Bibliography

[1] Chen, X., Diaz-Pinto, A., Ravikumar, N., and Frangi, A. Deep learning in medical image registration. In: *Progress in Biomedical Engineering*, Dec. 2020. ISSN: 2516-1091. DOI: 10.1088/2516-1091/abd37c.

[2] Haskins, G., Kruger, U., and Yan, P. Deep learning in medical image registration: a survey. In: *Machine Vision and Applications* 31(1–2), Jan. 2020. ISSN: 1432-1769. DOI: 10.1007/s00138-020-01060-x.

[3] Fu, Y., Lei, Y., Wang, T., Curran, W. J., Liu, T., and Yang, X. Deep learning in medical image registration: a review. In: *Physics in Medicine & Biology* 65(20):20TR01, Oct. 2020. ISSN: 1361-6560. DOI: 10.1088/1361-6560/ab843e.

[4] Zou, J., Gao, B., Song, Y., and Qin, J. A review of deep learning-based deformable medical image registration. In: *Frontiers in Oncology* 12, Dec. 2022. ISSN: 2234-943X. DOI: 10.3389/fonc.2022.1047215.

[5] Chen, J., Liu, Y., Wei, S., Bian, Z., Subramanian, S., Carass, A., Prince, J. L., and Du, Y. *A survey on deep learning in medical image registration: new technologies, uncertainty, evaluation metrics, and beyond*. 2023. DOI: 10.48550/ARXIV.2307.15615.

[6] Balakrishnan, G., Zhao, A., Sabuncu, M. R., Guttag, J., and Dalca, A. V. VoxelMorph: A Learning Framework for Deformable Medical Image Registration. In: *IEEE Transactions on Medical Imaging* 38(8):1788–1800, Aug. 2019. ISSN: 1558-254X. DOI: 10.1109/tmi.2019.2897538.

[7] Jia, X., Bartlett, J., Chen, W., Song, S., Zhang, T., Cheng, X., Lu, W., Qiu, Z., and Duan, J. Fourier-Net: Fast Image Registration with Band-Limited Deformation. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 1. 2023, pp. 1015–1023.

[8] Jia, X., Thorley, A., Gomez, A., Lu, W., Kotecha, D., and Duan, J. *Fourier-Net+: Leveraging Band-Limited Representation for Efficient 3D Medical Image Registration*. 2023. DOI: 10.48550/ARXIV.2307.02997.

[9] Tam, A. L., Lim, H. J., Wistuba, I. I., Tamrazi, A., Kuo, M. D., Ziv, E., Wong, S., Shih, A. J., Webster, R. J., Fischer, G. S., et al. Image-Guided Biopsy in the Era of Personalized Cancer Care: Proceedings from the Society of Interventional Radiology Research Consensus Panel. In: *Journal of Vascular and Interventional Radiology* 27(1):8–19, Jan. 2016. ISSN: 1051-0443. DOI: 10.1016/j.jvir.2015.10.019.

[10] Chen, A. M., Hsu, S., Lamb, J., Yang, Y., Agazaryan, N., Steinberg, M. L., Low, D. A., and Cao, M. MRI-guided radiotherapy for head and neck cancer: initial clinical experience. In: *Clinical and Translational Oncology* 20(2):160–168, June 2017. ISSN: 1699-3055. DOI: 10.1007/s12094-017-1704-4.

[11] Rigaud, B., Simon, A., Castelli, J., Lafond, C., Acosta, O., Haigron, P., Cazoulat, G., and Crevoisier, R. de Deformable image registration for radiation therapy: princi-

ple, methods, applications and evaluation. In: *Acta Oncologica* 58(9):1225–1237, June 2019. ISSN: 1651-226X. DOI: 10.1080/0284186x.2019.1620331.

[12] Yang, D., Li, H., Low, D. A., Deasy, J. O., and Naqa, I. E. A fast inverse consistent deformable image registration method based on symmetric optical flow computation. In: *Physics in Medicine and Biology* 53(21):6143–6165, Oct. 2008. ISSN: 1361-6560. DOI: 10.1088/0031-9155/53/21/017.

[13] Vercauteren, T., Pennec, X., Perchant, A., and Ayache, N. Diffeomorphic demons: Efficient non-parametric image registration. In: *NeuroImage* 45(1):S61–S72, Mar. 2009. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2008.10.040.

[14] Kingma, D. P. and Ba, J. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980.

[15] Ronneberger, O., Fischer, P., and Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 2015, pp. 234–241. ISBN: 9783319245744. DOI: 10.1007/978-3-319-24574-4_28.

[16] Fan, J., Cao, X., Yap, P.-T., and Shen, D. BIRNet: Brain image registration using dual-supervised fully convolutional networks. In: *Medical Image Analysis* 54:193–206, May 2019. ISSN: 1361-8415. DOI: 10.1016/j.media.2019.03.006.

[17] Zhang, J. *Inverse-Consistent Deep Networks for Unsupervised Deformable Image Registration*. 2018. DOI: 10.48550/ARXIV.1809.03443.

[18] Chen, J., Frey, E. C., He, Y., Segars, W. P., Li, Y., and Du, Y. TransMorph: Transformer for unsupervised medical image registration. In: *Medical Image Analysis* 82:102615, Nov. 2022. ISSN: 1361-8415. DOI: 10.1016/j.media.2022.102615.

[19] Mok, T. C. and Chung, A. C. Fast Symmetric Diffeomorphic Image Registration with Convolutional Neural Networks. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.00470.

[20] Odille, F., Vuissoz, P., Marie, P., and Felblinger, J. Generalized Reconstruction by Inversion of Coupled Systems (GRICS) applied to free-breathing MRI. In: *Magnetic Resonance in Medicine* 60(1):146–157, June 2008. ISSN: 1522-2594. DOI: 10.1002/mrm.21623.

[21] Küstner, T., Pan, J., Gilliam, C., Qi, H., Cruz, G., Hammernik, K., Blu, T., Rueckert, D., Botnar, R., Prieto, C., et al. Self-Supervised Motion-Corrected Image Reconstruction Network for 4D Magnetic Resonance Imaging of the Body Trunk. In: *APSIPA Transactions on Signal and Information Processing* 11(1), 2022. ISSN: 2048-7703. DOI: 10.1561/116.00000039.

[22] Xuan, K., Sun, S., Xue, Z., Wang, Q., and Liao, S. Learning MRI k-Space Subsampling Pattern Using Progressive Weight Pruning. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 178–187. ISBN: 9783030597139. DOI: 10.1007/978-3-030-59713-9_18.

[23] Yiasemis, G., Zhang, C., Sánchez, C. I., Sonke, J.-J., and Teuwen, J. Deep MRI reconstruction with radial subsampling. In: *Medical Imaging 2022: Physics of Medical Imaging*. Ed. by W. Zhao and L. Yu. SPIE, Apr. 2022. DOI: 10.1117/12.2609876.

[24] Yiasemis, G., Sánchez, C. I., Sonke, J.-J., and Teuwen, J. *On Retrospective k-space Subsampling schemes For Deep MRI Reconstruction*. 2023. DOI: 10.48550/ARXIV.2301.08365.

[25] Marcus, D. S., Wang, T. H., Parker, J., Csernansky, J. G., Morris, J. C., and Buckner, R. L. Open Access Series of Imaging Studies (OASIS): Cross-sectional MRI Data in Young, Middle Aged, Nondemented, and Demented Older Adults. In: *Journal of Cognitive Neuroscience* 19(9):1498–1507, Sept. 2007. ISSN: 1530-8898. DOI: 10.1162/jocn.2007.19.9.1498.

[26] Hoopes, A., Hoffmann, M., Fischl, B., Guttag, J., and Dalca, A. V. HyperMorph: Amortized Hyperparameter Learning for Image Registration. In: *Information Processing in Medical Imaging*. Springer International Publishing, 2021, pp. 3–17. ISBN: 9783030781910. DOI: 10.1007/978-3-030-78191-0_1.

[27] Hering, A., Hansen, L., Mok, T. C. W., Chung, A. C. S., Siebert, H., Hager, S., Lange, A., Kuckertz, S., Heldmann, S., Shao, W., et al. Learn2Reg: Comprehensive Multi-Task Medical Image Registration Challenge, Dataset and Evaluation in the Era of Deep Learning. In: *IEEE Transactions on Medical Imaging* 42(3):697–712, Mar. 2023. ISSN: 1558-254X. DOI: 10.1109/tmi.2022.3213983.

[28] Wang, C., Lyu, J., Wang, S., Qin, C., Guo, K., Zhang, X., Yu, X., Li, Y., Wang, F., Jin, J., et al. *CMRxRecon: An open cardiac MRI dataset for the competition of accelerated image reconstruction*. 2023. DOI: 10.48550/ARXIV.2309.10836.

[29] Pruessmann, K. P., Weiger, M., Scheidegger, M. B., and Boesiger, P. SENSE: Sensitivity encoding for fast MRI. In: *Magnetic Resonance in Medicine* 42(5):952–962, Nov. 1999. ISSN: 1522-2594. DOI: 10.1002/(sici)1522-2594(199911)42:5<952::aid-mrm16>3.0.co;2-s.

[30] Hennig, J. K-space sampling strategies. In: *European Radiology* 9(6):1020–1031, July 1999. ISSN: 1432-1084. DOI: 10.1007/s003300050788.

[31] Bernard, O., Lalande, A., Zotti, C., Cervenansky, F., Yang, X., Heng, P.-A., Cetin, I., Lekadir, K., Camara, O., Gonzalez Ballester, M. A., et al. Deep Learning Techniques for Automatic MRI Cardiac Multi-Structures Segmentation and Diagnosis: Is the Problem Solved? In: *IEEE Transactions on Medical Imaging* 37(11):2514–2525, Nov. 2018. ISSN: 1558-254X. DOI: 10.1109/tmi.2018.2837502.

[32] Küstner, T., Pan, J., Qi, H., Cruz, G., Gilliam, C., Blu, T., Yang, B., Gatidis, S., Botnar, R., and Prieto, C. LAPNet: Non-Rigid Registration Derived in k-Space for Magnetic Resonance Imaging. In: *IEEE Transactions on Medical Imaging* 40(12):3686–3697, Dec. 2021. ISSN: 1558-254X. DOI: 10.1109/tmi.2021.3096131.

[33] Dalca, A. V., Balakrishnan, G., Guttag, J., and Sabuncu, M. R. Unsupervised Learning for Fast Probabilistic Diffeomorphic Registration. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2018, pp. 729–738. ISBN: 9783030009281. DOI: 10.1007/978-3-030-00928-1_82.

[34] Trabelsi, C., Bilaniuk, O., Zhang, Y., Serdyuk, D., Subramanian, S., Santos, J. F., Mehri, S., Rostamzadeh, N., Bengio, Y., and Pal, C. J. *Deep Complex Networks*. 2017. DOI: 10.48550/ARXIV.1705.09792.

[35] Wang, J. and Zhang, M. *DeepFLASH: An Efficient Network for Learning-based Medical Image Registration*. 2020. DOI: 10.48550/ARXIV.2004.02097.

[36] Ashburner, J. A fast diffeomorphic image registration algorithm. In: *NeuroImage* 38(1):95–113, Oct. 2007. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2007.07.007.

[37] Arsigny, V., Commowick, O., Pennec, X., and Ayache, N. A Log-Euclidean Framework for Statistics on Diffeomorphisms. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 924–931. ISBN: 9783540447085. DOI: 10.1007/11866565_113.

[38] Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. Spatial Transformer Networks. In: *ArXiv* abs/1506.02025, 2015. URL: https://api.semanticscholar.org/CorpusID:6099034.

[39] Modat, M., Ridgway, G. R., Taylor, Z. A., Lehmann, M., Barnes, J., Hawkes, D. J., Fox, N. C., and Ourselin, S. Fast free-form deformation using graphics processing units. In: *Computer Methods and Programs in Biomedicine* 98(3):278–284, June 2010. ISSN: 0169-2607. DOI: 10.1016/j.cmpb.2009.09.002.