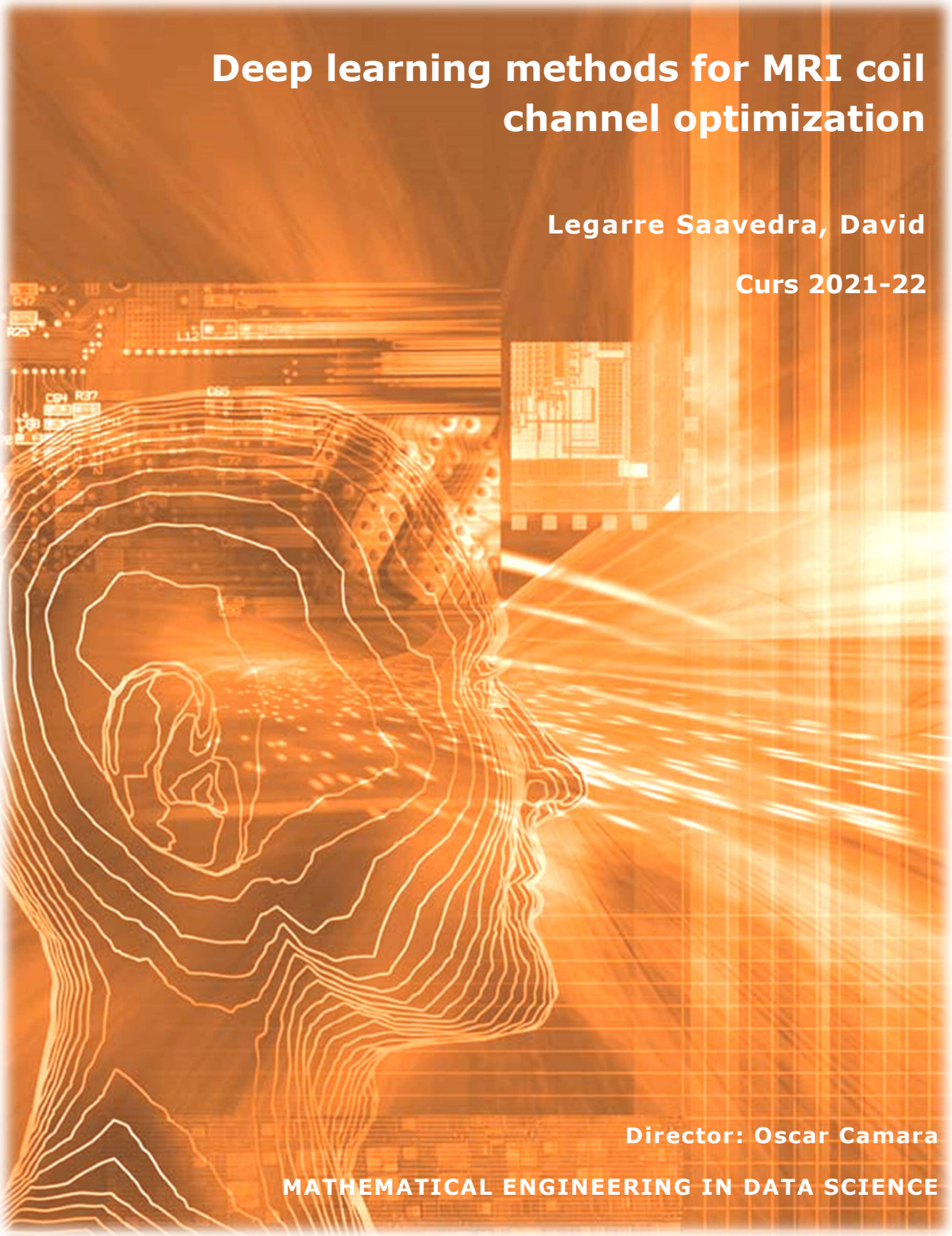
Deep learning methods for MRI coil channel optimization

Legarre Saavedra, David

Curs 2021-22



Director: Oscar Camara

MATHEMATICAL ENGINEERING IN DATA SCIENCE

Universitat Pompeu Fabra Barcelona
Escola d'Enginyeria

Treball de Fi de Grau

Acknowledgements

I would like to thank Gaspar Delso for guiding me through this project and for giving me the opportunity to work and learn. To the Hospital Clínic de Barcelona and its patients, for allowing us access to the data used in this project.

To my family, for giving me their support and the chance to work on projects like this, and to my friend Alfonso, for putting up with my writing mistakes.

Abstract

Magnetic resonance imaging (MRI) scanners use powerful magnets and make use of the human body reactions to such fields to produce images. Modern MRI uses multi-channel receiver coils to enable accelerated data acquisition. With the introduction of parallel image acquisition techniques, the time it takes for a scan of this type has been significantly reduced. Despite their many advantages, acceleration techniques cause a number of image artifacts, particularly in the presence of patient motion. In particular, we will aim at improving the diagnostic quality of clinical 3D myocardial delayed enhancement (MDE) data by optimizing the set of coil channels used in the reconstruction of the images. We will implement and use the ROVir algorithm to remove these artifacts from the image, and implement a DL approach to automatize its parameters.

Resumen

Las imágenes de resonancia magnética (IRM) utilizan potentes imanes y aprovechan las reacciones del cuerpo humano a dichos campos para producir imágenes. Los IRM modernos utilizan bobinas receptoras multicanal para permitir la adquisición acelerada de datos. Con la introducción de técnicas de adquisición de imágenes en paralelo, el tiempo necesario para realizar una exploración de este tipo se ha reducido considerablemente. A pesar de sus muchas ventajas, las técnicas de aceleración provocan una serie de artefactos en la imagen, sobre todo en presencia del movimiento del paciente. Este proyecto pretende mejorar la calidad diagnóstica de los datos clínicos en 3D de realce tardío del miocardio (RTM) optimizando el conjunto de canales de bobina utilizados en la reconstrucción de las imágenes. Para ello, implementaremos y utilizaremos el algoritmo ROVir para eliminar estos artefactos de la imagen, e implementaremos un enfoque de DL para automatizar los parámetros de este.

Resum

Les imatges de ressonància magnètica (IRM) utilitzen potents imants i aprofiten les reaccions del cos humà a aquests camps per a produir imatges. Els IRM moderns utilitzen bobines receptores multicanal per a permetre l'adquisició accelerada de dades. Amb la introducció de tècniques d'adquisició d'imatges en paral·lel, el temps necessari per a realitzar una exploració d'aquest tipus s'ha reduït considerablement. Malgrat els seus molts avantatges, les tècniques d'acceleració provoquen una sèrie d'artefactes en la imatge, sobretot en presència del moviment del pacient. Aquest projecte pretén millorar la qualitat diagnòstica diagnòstic de les dades clíniques en 3D de realç tardà del miocardi (RTM) optimitzant el conjunt de canals de bobina utilitzats en la reconstrucció de les imatges. Per a això, implementarem i utilitzarem l'algorisme ROVir per a eliminar aquests artefactes de la imatge, i implementarem un enfocament de DL per a automatitzar els paràmetres d'aquest.

Keywords: Deep Learning, MRI, Noise reduction, MDE, Medical Imaging

Contents

1. INTRODUCTION **1**
1.1. Context 1
1.2. Motivation 1
1.3. Parallel Imaging 1

2. BACKGROUND **5**
2.1. Introduction 5
2.2. Artifacts in PI 5
2.2.1. Aliasing artifacts 5
2.2.2. Acceleration artifacts 6
2.2.3. Noise 6
2.3. Proposed solution 7

3. METHODS **9**
3.1. Introduction 9
3.2. Dataset and data preparation 9
3.2.1. Dataset 9
3.2.2. Data preparation 10
3.3. ROVir implementation 12
3.3.1. Coil sensitivities 12
3.3.2. Masks 12
3.3.3. Find weights 13
3.3.4. Dropping coils and manual results 14

4. MODEL **17**
4.1. Introduction 17
4.2. Preprocessing data 17
4.2.1. Data augmentation 18
4.3. Model structure 20
4.4. Training 21
4.5. Automating ROVir 24

5. DISCUSSION **27**
5.1. Conclusion 27
5.2. Posible Expansions 27

List of Figures

1.1. Output image from a coil, receiving the signal from the "top right" corner. This image was extracted using the software ITK-SNAP [Yushkevich et al., 2006] 2
1.2. Samples of coils before reconstruction 2
1.3. Reconstructed image 3

2.1. An example of aliasing artifact (red) 5
2.2. An example of acceleration artifact (red) 6
2.3. An example of noisy acquisition 7

3.1. A sample of a DICOM file 10
3.2. intensities histogram found with ITK-SNAP[Yushkevich et al., 2006] . . . 10
3.3. Comparison between not adapting (left), and adapting (right) the contrast 11
3.4. Histogram of an example image before normalization, visualization done with Seaborn [Waskom, 2021], and MatPlotlib [Hunter, 2007] python libraries. 11
3.5. Histogram showing the intensities of the example of Figure 3.4 after our normalization step. 12
3.6. An example of one coil from the real weighted sensitivities 15
3.7. A comparison of a single coil. The Gaussian blur was applied using the SciPy library [Virtanen et al., 2020] 15
3.8. An example of our regions, green being region A, and red being region B. 15
3.9. Comparing results with different F values 16
3.10. Comparison of intensities before and after applying the ROVir algorithm . 16
3.11. 16

4.1. Interface label-studio 17
4.2. Comparison of sections taken from the same patient. 18
4.3. Examples of bad data augmentation following SVHN 19
4.4. Examples of our data augmentation policy 19
4.5. Example of annotation drawn 19
4.6. Diagram showing the structure of our initial netowrk 20
4.7. With a learning rate of 0.01 21
4.8. With a learning rate of 0.001 22
4.9. We can see that in all the predictions the bounding box was the same . . . 23
4.10. Training with the overfitting prevention techniques implemented, with a learning rate of 0.004 23
4.11. Final predictions 24
4.12. Masks automatically generated 24

4.13. In our case, the best limit our method found was to drop at most 8 coils before losing to much information 25

Chapter 1

INTRODUCTION

1.1. Context

This project is done in collaboration with Hospital Clínic de Barcelona and General Electric Healthcare. The aim of this project is to find an automatic solution to the problem of artifacts produced in Magnetic Resonance Imaging (MRI) when using parallel imaging acceleration. This will make easier to interpret images from the MRI scan, thus, avoiding misdiagnosis in the case an artifact may be interpreted as an injury in the myocardium.

1.2. Motivation

The discovery of MRI scans has provided hospitals with a powerful tool to form pictures of the inside of the body to be used in diagnosis. MRI scanners work by using strong magnetic fields and radio waves instead of X-rays. To perform this scan, often, the patient is injected with a contrast agent that enhance the contrast of the obtained images. image. In this project the contrast agent was applied using a technique known as Myocardial Delayed Enhancement [Vogel-Claussen et al., 2006]. This technique consists on letting the contrast be redistributed by the body for a period of time, usually 15 minutes, instead of injecting the contrast agent and immediately proceeding with the scanning. This technique facilitates the identification of scars and damaged tissue in the heart.

A widespread technique to accelerate the acquisition of MR images is Parallel Magnetic Resonance Imaging, have allowed hospitals to be able to perform these kinds of procedures on a daily basis faster than with previous methods.

1.3. Parallel Imaging

As mentioned previously, MRI scans work by using a strong magnetic field to align the spins of the hydrogen nuclei in the body. Radiofrequency (RF) pulses are then used to excite these nuclei, which are then released to return to their equilibrium state. This produces a signal that can be then detected by coils in the MRI scanner. Typically, multiple coils are used to detect the MRI signal, and then these signals are combined into one complete image

In parallel imaging (PI) the redundancies between the signals acquired by the different coils are exploited to increase the resolution of the final reconstructed image, enabling the use of undersampled data and, therefore, shortening the acquisition time [Deshmane et al., 2012].

In our case, the samples we took are formed by the signal coming from 25 receiving coils.

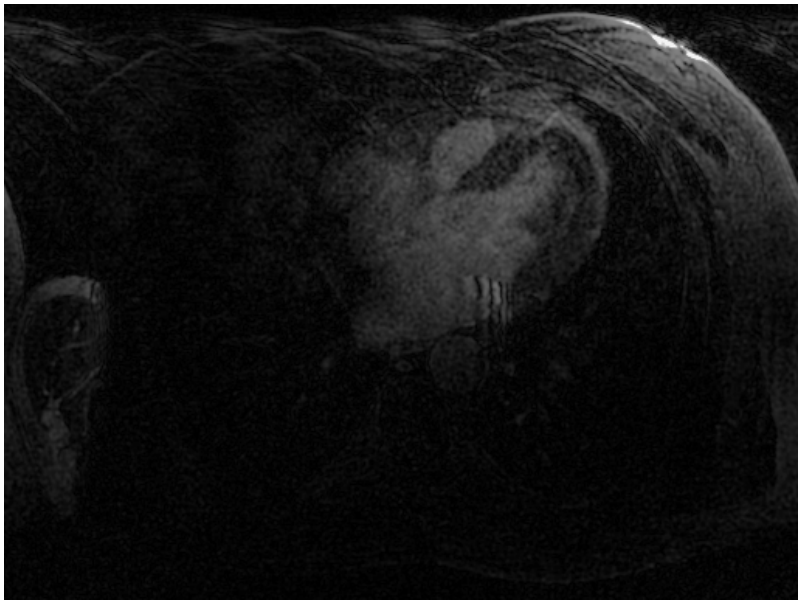


Figure 1.1: Output image from a coil, receiving the signal from the ”top right” corner. This image was extracted using the software ITK-SNAP [Yushkevich et al., 2006]

In figure 1.1 we can see an example of the output of a coil, in which we can see how the intensities of the image indicate the position of the receiving coil, since the closer the tissue is the stronger the signal. We can also see two artifacts. At the bottom left corner we can see an example of the ”wrap-around” artifact and in the center of the heart is a case of acceleration ghosting (the vertical lines that look ”unnatural”). These kind of artifacts and their origins will be explored later.

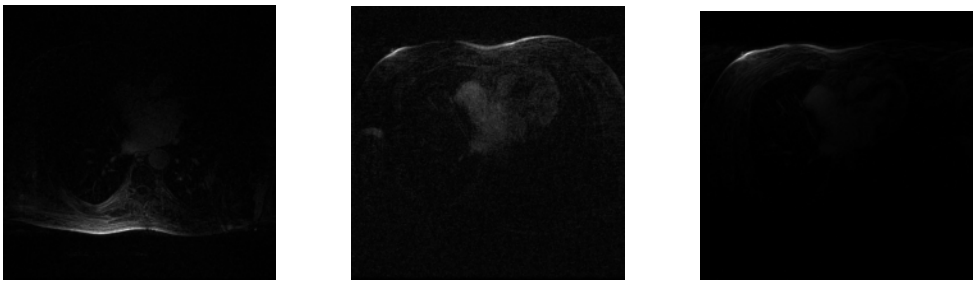


Figure 1.2: Samples of coils before reconstruction

These images are then combined using a root of a sum of squares, and we obtain:

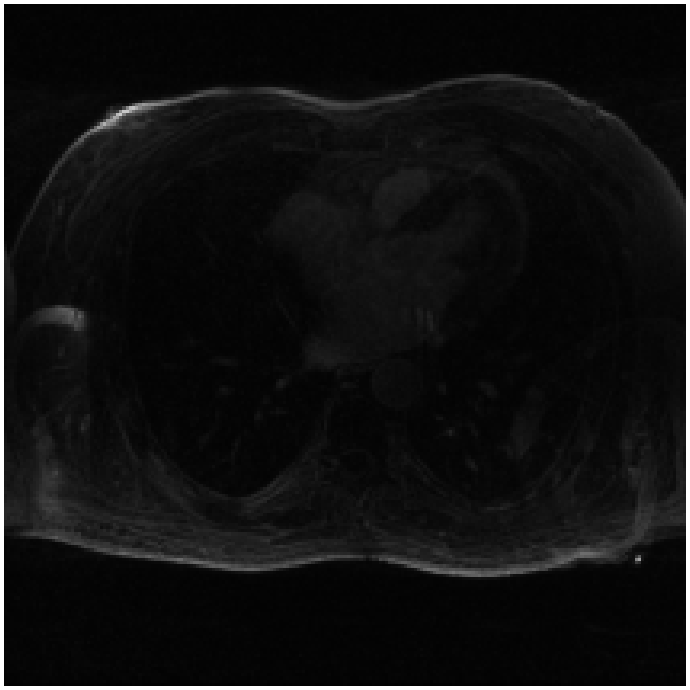


Figure 1.3: Reconstructed image

Chapter 2

BACKGROUND

2.1. Introduction

In this chapter we will explain the context, source, and type of the artifacts that may be produced in MRI scans during PI acquisition, and the solutions proposed to deal with them.

2.2. Artifacts in PI

There are a number of undesired efects that can affect the quality of cardia MR acquisitions. In our project, we will focus on two artifacts that often degrade 3D MDE datasets.

2.2.1. Aliasing artifacts

A typical artifact that can be seen on any kind of MR acquisition is the aliasing artifact.

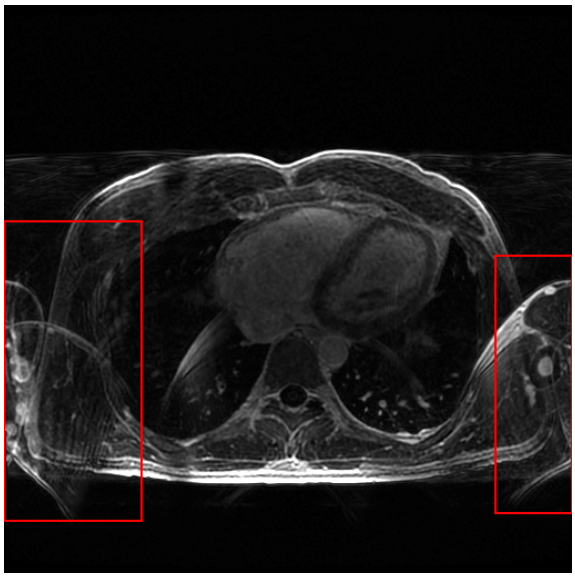


Figure 2.1: An example of aliasing artifact (red)

In Figure 2.1 we can see that the arms of the patient have been folded over, they are not in the side it would correspond them to. This is due to the fact that the field-of-view (FOV) defined when acquiring the image is smaller than the size of the patient. So, any object that it is outside this FOV is acquired on the other side of the image.

2.2.2. Acceleration artifacts

When introducing acceleration in MR image acquisition, a new artifact can appear the acceleration artifacts. One of the most common ones is the ”SENSE ghost” [Goldfarb, 2004] the object being scanned.

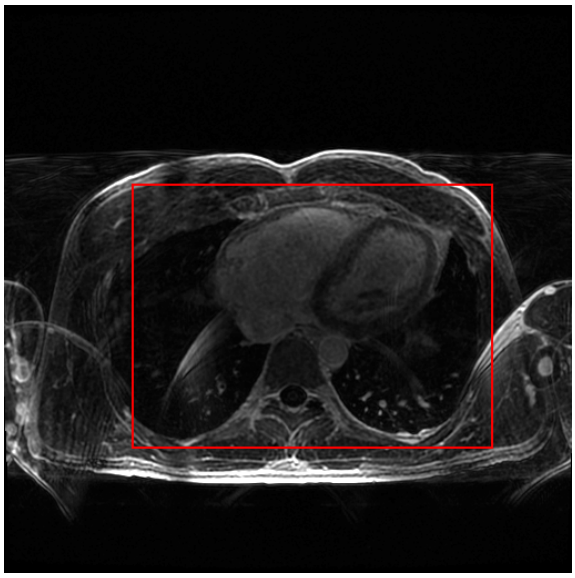


Figure 2.2: An example of acceleration artifact (red)

In Figure 2.2 we can see an example of this “SENSE ghost” these kind of artifact are particularly dangerous for the diagnosis since it can be mistaken for a pathology that is not there.

The name comes from the SENSE algorithm [Pruessmann et al., 1999] used to accelerate image reconstruction in parallel imaging, one of the most common used ones. This algorithm is based on 4 steps: First generate the coil sensisitivy maps, acquire reduced k-space data, reconstruct the partial images from each coil, and finally combine this partial images.

2.2.3. Noise

Not an artifact per se, but certainly an ongoing concern when acceleration methods are used is image noise. This is due to the fact that the main purpose of PI is to reduce the time it takes to construct an image. In order to increase these speeds algorithms such as SENSE, acquire less k-space lines, thus reducing redundancy and therefore the signal-to-noise ratio (SNR).

In Figure 2.3 we can see how the image looks grainy. Unlike other kinds of noise found in other imaging fields, MR noise is not uniformly distributed, it depends on material being detected in the MRI scan and the position on the scan.

Figure 2.3: An example of noisy acquisition

2.3. Proposed solution

As a solution to minimize these artifacts, we propose to use the algorithm described in [Kim et al., 2021] to find the Region-optimized virtual (ROVir) coils. We will adapt its implementation to be able to work with DL methods to automatize its parameters for a use without need for human intervention.

The idea of the algorithm is to give more importance, more weight, to the coils that give the most information coming from our Region of Interest (ROI), in the case of this project the ROI would be the heart, and dampening the weight of the coils that give low to no signal from our ROI which are most likely the producers of the artifacts. Then, by performing a linear combination of the coils with their respective weights we then create new coils, called ”virtual” coils, and use an optimal subset of these as our new coils for the image reconstruction.

Chapter 3

METHODS

3.1. Introduction

In this chapter we will introduce the methodology used in this project, how the data was prepared to be used and the implementation of the proposed solution in the previous chapter.[1]

3.2. Dataset and data preparation

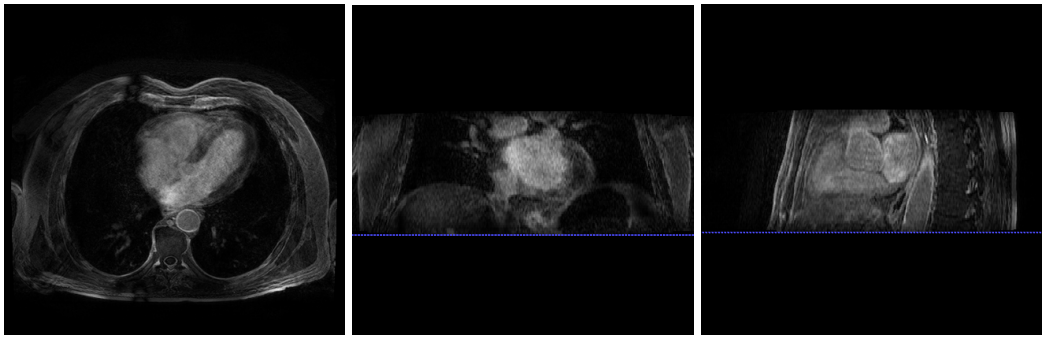
3.2.1. Dataset

The dataset we used is formed by scans obtained with GE Signa Architect 3.0T the MRI scanner at "Hospital Clínic de Barcelona". The data consist of two types, NIFTI, and DICOM filetypes.

DICOM (Digital Imaging and Communications in Medicine) datafiles are a standard datatype for storage of medical images. Their advantage is that this kind of data cannot only store images but also extra data. In the case of our dataset, in the metadata, is stored information such as: the institution where it comes from, the type of image (MR scan in this case), and patient information (sex, age, weight, etc.). In this kind of format we store the 3D scan of the chest area. These are divided in images of a resolution of 512×512, then, depending on the size of the scan, we can have different depths (these range from 100 to 200, approx.).

We also have NIFTI (from Neuroimaging Informatics Technonology Initiative), another standard datatype used in the medicine field. In this kind of data is where we store the signal coming from each of the coils used by the machine at the hospital. This is the main filetype we will be working on when testing and using the ROVir algorithm. Again, the data coming from filetype has a resolution of 512×512. But this time, each image is the signal coming from one out of the 25 coils in the machine. We can see an example of this kind of image in Figure 1.1.

[1] The code implementation of this project can be found in the github repository `https://github.com/DavidLegarreSaavedra1/Deep-learning-methods-for-MRI-coil-channel-optimization.git`



Axial Coronal Axial

Figure 3.1: A sample of a DICOM file

In order to form an image we can work on, first we need to combine the contributinos of the individual coils. There are many methods to do this, we chose to apply a root of sum of squares. A procedure where we take the signal coming from each coil, calculate its pixel-wise square, add all this up, and calculate the square root. This method is preferable due to its easy implementation and reasonable results.

3.2.2. Data preparation

The first thing we needed to tackle was the fact that the contrast of the image is unbalanced. Due to the nature of MRI, tissues return different intensities each time an image is acquired. In the human chest we find different kinds of tissue: air (the air inside the lungs) has an intensity of around 0, then our tissues of interest (blood, cardiac muscle, and other kind of soft tissue) has an intensity of around 400, and finally fat has a really strong response, that varies with the proximity to the coils and the local field strength. Although the sequence used in the study was designed to filter out most of the signal coming from fat tissue, there's still some residual intensities that act as outliers and unbalance our image.

So, we needed some way to normalize this issue accross our image database. In the ITK-SNAP software there's a tool that calculates the best automatic contrast for the image.

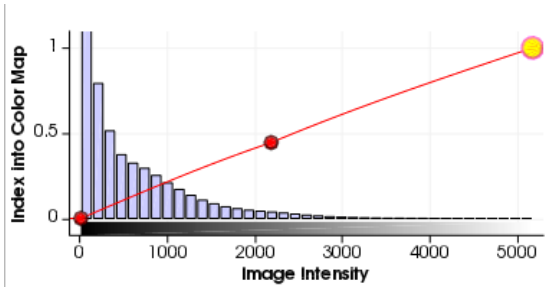


Figure 3.2: intensities histogram found with ITK-SNAP[Yushkevich et al., 2006]

The problem with this solution is that it's not automatic, we can't use ITK-SNAP for image processing. We need a solution that will take every input image and return a contrast-corrected visual image, that will also help us when we begin training our model.

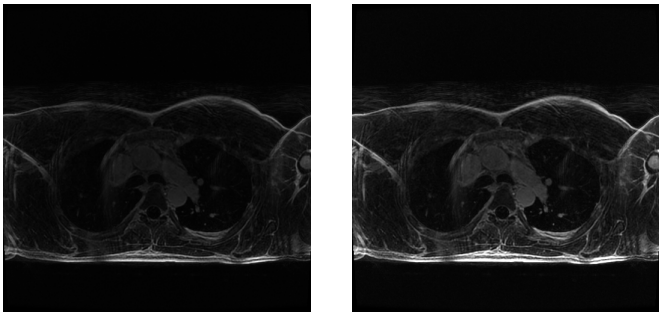


Figure 3.3: Comparison between not adapting (left), and adapting (right) the contrast

As we can see in Figure 3.2 most of the intensities are located on the low-end of the histogram, there are many factors influencing the distribution of this plot. The amount of contrast material, the time since the material was injected etc., all influence in the accumulation of the intensities and the number of peaks that we will see. An initial approach would be to divide by the maximum intensities, this would not work, since the points with maximum intensity are generally outliers, subcutaneous adipose tissue directly in fron of a receiver coil. As we can see on Figure 3.3 in the left image

we can observe how most of the signal in the image is coming from the bottom part, this is due to the fat in that area that returns a strong signal.

First, we needed to visualize the histogram of the intensities of our initial image.

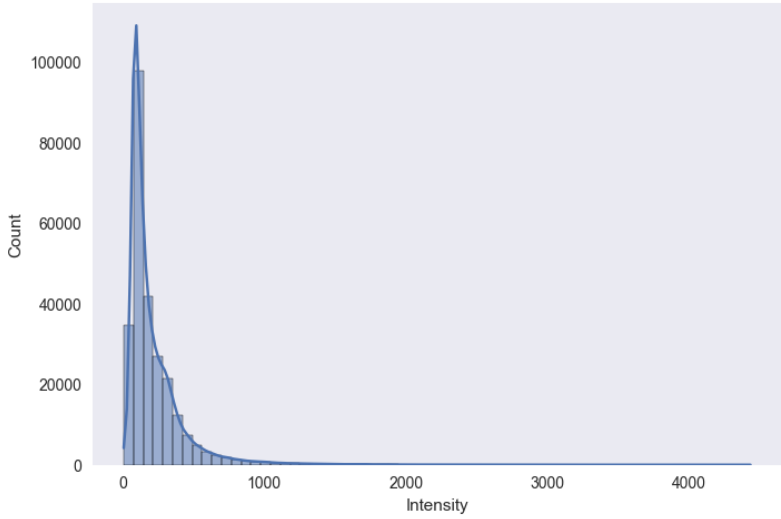


Figure 3.4: Histogram of an example image before normalization, visualization done with Seaborn [Waskom, 2021], and MatPlotlib [Hunter, 2007] python libraries.

As we can see, the histogram is similar. We cannot expect that most of our images will follow a similar distribution of intensities. This is why we must implement a method that will dynamically normalize our intensities. The method we implemented was: by taking the values calculated with the histogram take a quantile (after trial and error, we found that the best quantile was the 0.99 quantile, approx 1000) of the bins. This way we can clamp the extreme values to the top 1% of the intensities of the image. Since our ROI is the heart, which showed to have an intensity of around 500, we can just assign intensities much larger than this quantile to the same value so our model can better differentiate the

values. After this, in order to improve training we normalize the values of the image between 0 and 1.

This will improve the training process since it will give a fairer chance to all images, since if we make every image to have a similar intensity distribution, we will learn better feature extractors that should work with more images.

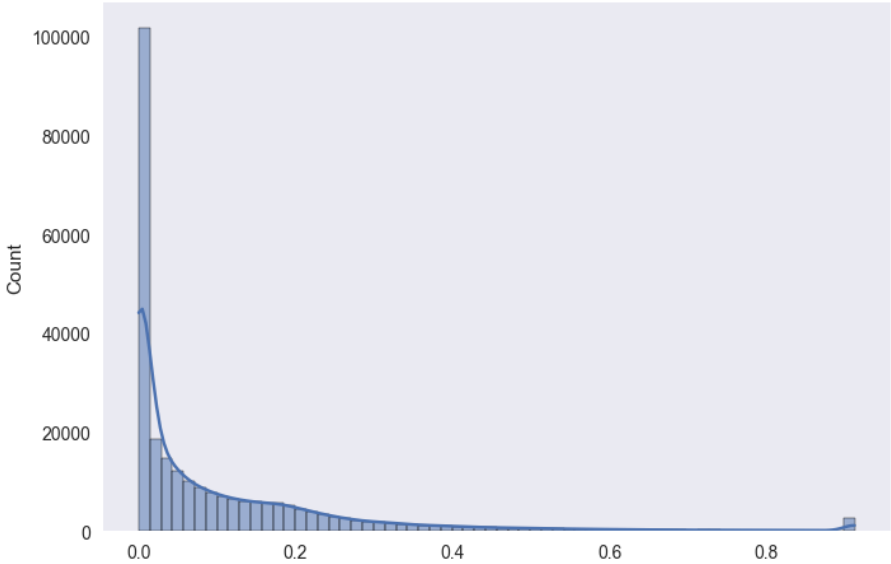


Figure 3.5: Histogram showing the intensities of the example of Figure 3.4 after our normalization step.

3.3. ROVir implementation

3.3.1. Coil sensitivities

Coil sensitivities are calibration values acquired during a MRI scan. These values show the sensibility of the input received depending on the distance from the coil. The brighter the image is, the closer it is to the original coil.

As we can see in Figure 3.6, the resolution from the real sensitivity mapping is quite low. In our experiments, obtaining the weighted sensitivities was difficult, as these are an intermediate result of the acquisition and reconstruction process so we had to work with alternatives.

In order to work datasets for which we don't have access to the internal sensitivity data, we can simulate our own sensitivities. For that, we need to perform a Gaussian Blur over all coils. This is done in order to simulate the weighted sensitivities from each coil in the original machine.

3.3.2. Masks

Then, we need to define two regions: Region 'A' will work as our ROI and 'B' as the region that we want to reduce the signal from. Initially this will be done manually but as we will se later, we will implement a DL network to automatically form the regions for us.

Then with the coordinates of these areas, we will define two new matrices of 0s with the dimensions of our sensitivity map, and selecting the range of the area we will set these new matrices to the original values in the image.

Then, once these areas are defined, we need to build the matrices that will be used to calculate the weights later. In order to make the process easier, we will begin by flattening the matrices. And, for each coil, we will multiply and sum with all the other coils, let N_c be the number of coils, each matrix A and B will have the shape $N_c \times N_c$ and in each coordinate we will store the square and sum. These can be interpreted as the inter-coil signal correlation.

From the original ROVir paper [Kim et al., 2021], the algorithm consists of 3 steps:

First form two matrices **A** and **B** as:

$$A = \int_{\Omega} g(x)^H g(x) dx \tag{3.1}$$

$$B = \int_{\Gamma} g(x)^H g(x) dx \tag{3.2}$$

Where Ω is the spatial ROI that we want to maximize, and Γ is the spatial region containing the unwanted information. The definition of $g(x)$ is explored more profoundly in [Kim et al., 2021, equation 8].

3.3.3. Find weights

In the original paper [Kim et al., 2021], the 'SIR' metric is defined as the ratio between the signals coming from the region of interest and the region that we want to ignore.

$$SIR(w_j) \triangleq \frac{w_j^H A w_j}{w_j^H B w_j} \tag{3.3}$$

Where w_j are the weights that will be used latter to form the virtual coils. So, the next step is to find those weights w_j, that , will maximize the SIR.

To calculate these weights the following equation is used:

$$Aw_i = \lambda_i B w_i = B^{-1} A w_i = \lambda_i w_i \tag{3.4}$$

We then get that these weights can be found calculating the eigenvalues and eigenvectors that satisfy said equation. Those eigenvalues can then be calculated by trying to find the eigenvalues in the matrix $B^{-1}A$.

Once these eigenvectors and eigenvalues have been found (we calculated them using the NumPy library [Harris et al., 2020]), we will need to sort them from highest to lowest by their eigenvalues. These found eigenvectors will be our weights that we will use to create our new 'virtual' coils.

The eigenvalues obtained will give us a sense of which weights maximize the SIR. So, if we order these from higest to lowest, the associated eigenvector of the first eigenvalue will be weight that gives us the biggest SIR.

Finally with these new weights we can create our new virtual coils by:

$$v_j(k) = \sum_{l=1}^{N_c} w_{lj} d_l(k) \tag{3.5}$$

where $v_j(k)$ is the k-space data for the *j*th virtual coil and $d_l(k)$ is the k-space data for our original coils, its more in depth definition can be found in [Kim et al., 2021, equation 1].

3.3.4. Dropping coils and manual results

Then, given a value F that will represent the number of coils, we will choose how many to drop, since the last coils will be the ones that give us the worst signal ratio.

Finding the value F that will give the best result is mostly a trial and error procedure. In the next chapter we will show the methods we used to automatize this step.

In Figure 3.9 we can see how, as F gets larger, the region 'A', as seen in Figure 3.8, stays constant as the region 'B' dampens. This makes sense, since we are completely removing part of the original information we had from the patient and taking a new weighted signal as our new image. Due to the nature of our data sample, we have found that completely removing the acceleration artifact has proven challenging and it is not always possible. When looking at the way the algorithm is designed, in the case that the coils that give the most signal from our ROI also give us the most signal from the region we are not interested in, then this makes difficult removing said artifact.

One way we decided to visualize how good our method worked in overcoming the artifact is by manually taking note of where the artifact is approximately, and compare the intensity plots of both images before and after.

As we can see, after the ROVir algorithm, most of the intensities have been attenuated, and the intensity around the main acceleration artifact that was blocking the heart has been smoothed.

The main reason this occurs is that our sample had a problem that ROVir is not capable to solve

As we can see in Figure 3.11, most coils that have signal from the heart aslo have it coming from the arm, and this makes it so that ROVir cannot reduce and amplify the same coil in different areas.

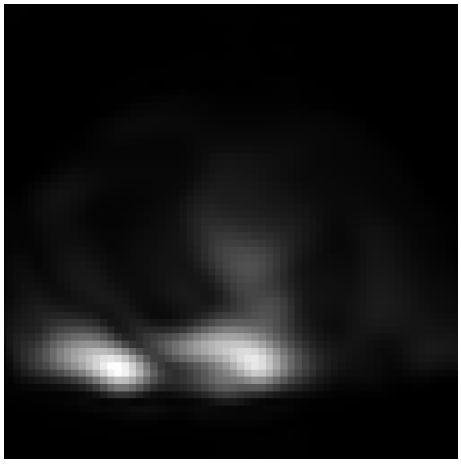
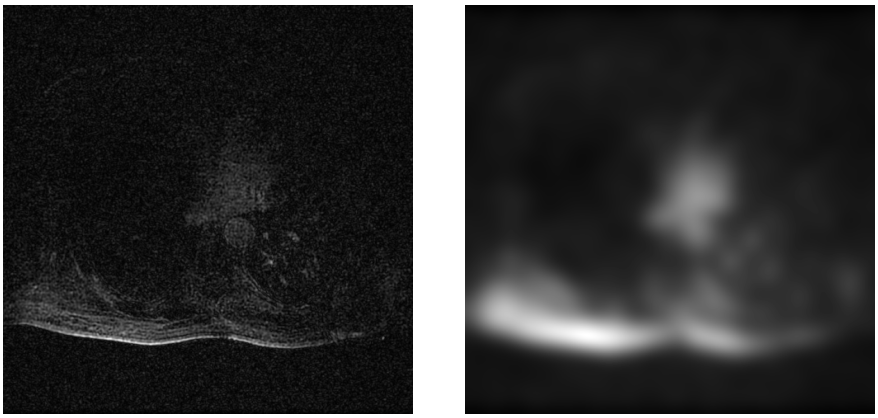


Figure 3.6: An example of one coil from the real weighted sensitivities



(a) Unfiltered coils (b) Filtered coils

Figure 3.7: A comparison of a single coil. The Gaussian blur was applied using the SciPy library [Virtanen et al., 2020]

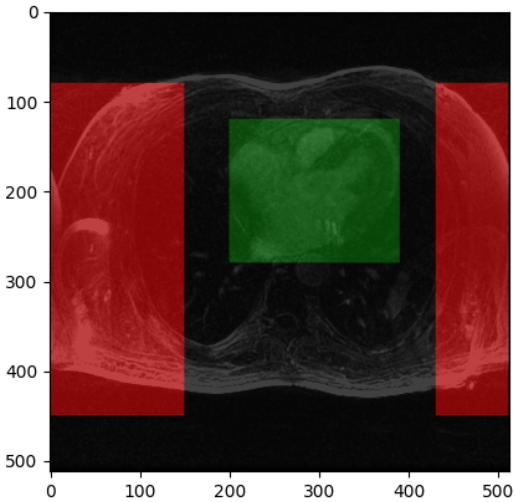
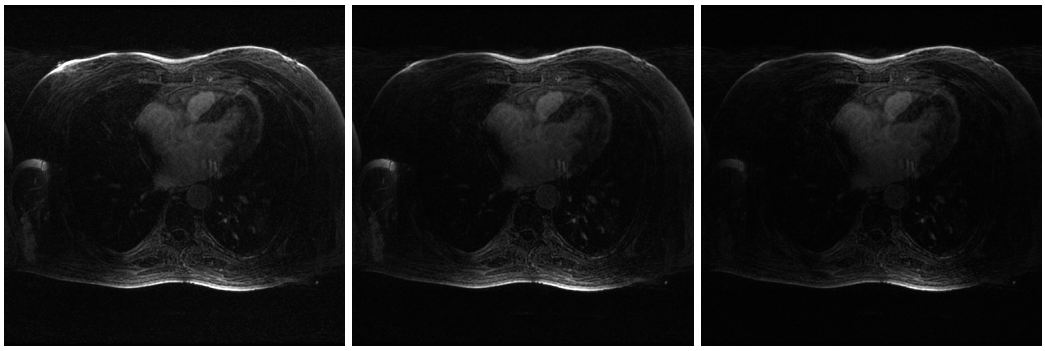


Figure 3.8: An example of our regions, green being region A, and red being region B.



(a) Dropping the 5 worst coils (b) Dropping the 10 worst coils (c) Dropping the 15 worst coils

Figure 3.9: Comparing results with different F values

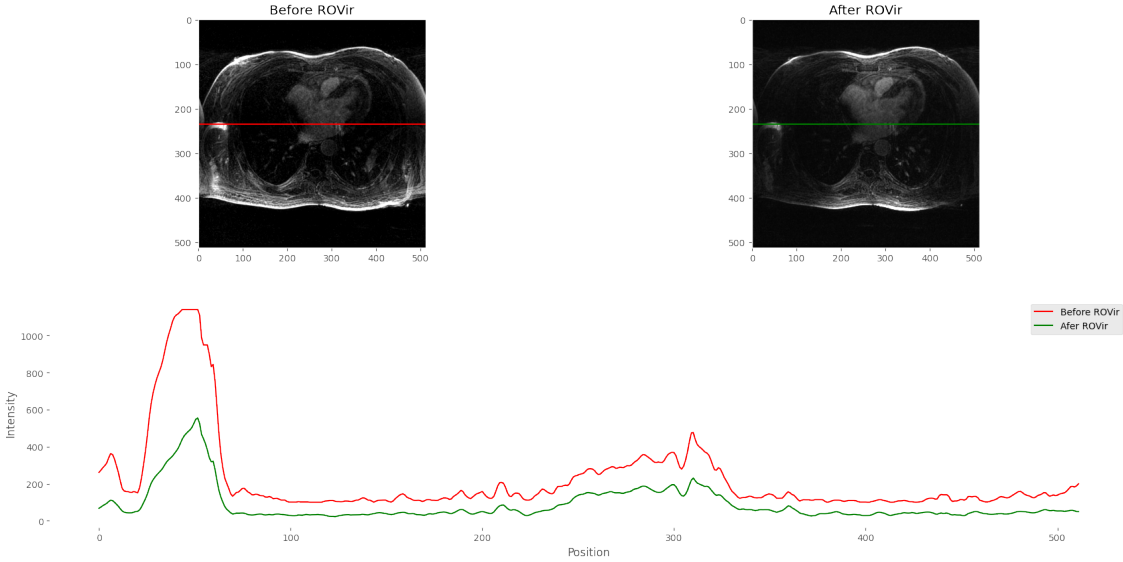
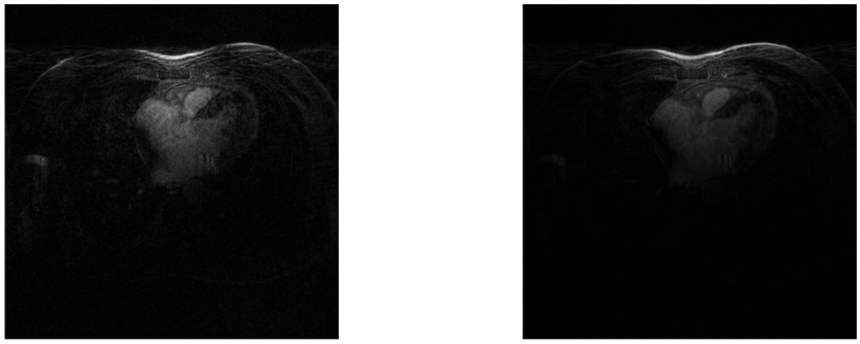


Figure 3.10: Comparison of intensities before and after applying the ROVir algorithm



(a) Coil number 12 (b) Coil number 17

Figure 3.11

Chapter 4

MODEL

4.1. Introduction

In this chapter we will introduce the model used, the training process of said model, some results, and the solutions we implemented to reduce human interaction in the ROVir process.

The reason we are using a Deep Learning (DL) approach in this project is so that we can have a tool that will automatically detect and localize the heart in our given reconstructed image, and define this new region as our region 'A' for it to be used by ROVir.

4.2. Preprocessing data

Whenever we are training a DL model, the first thing we must do is to prepare the data we will use.

Expanding on our introduction to our data preparation, shown in section 3.2.2, we needed to build a dataset from which the model would extract the samples to train and validate during the training process. For this Purpose we used the tool Label-Studio [Tkachenko et al., 2022], a free to use python-based app that allows us to build datasets and define the regions that our model will later predict.

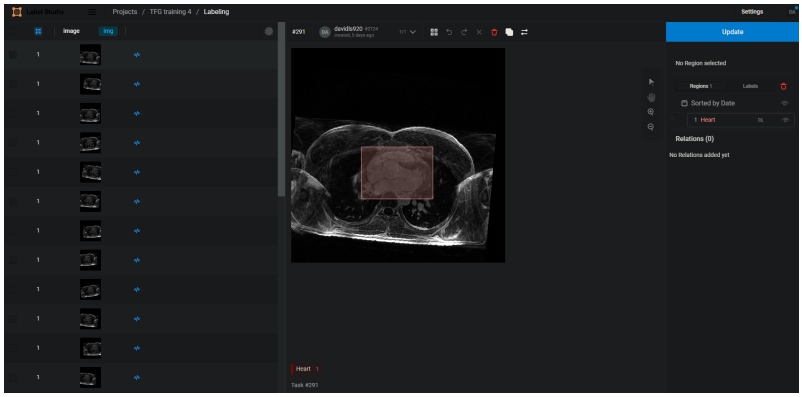


Figure 4.1: Interface label-studio

Starting with predefined templates for object localization that Label-Studio offers, we

add our data to the project, and start to paint the regions. These regions are rectangular areas defined by 4 values following the COCO [Lin et al., 2014] format for training images. The first two values indicate x and y positions of the top left corner of the region, and then we have the width and height of this region. To make easier the process of predicting and drawing the regions, we decided to normalize the coordinates of said region. Then, we set up our bounding box to predict as an array of $[x, y, x + \text{width}, y + \text{height}]$, so now we have a rectangle defined by two points, the top left and bottom right corners, making it easier to extrapolate the other two points later.

4.2.1. Data augmentation

In the study we had access to the complete scans of 5 patients. Initially we tried to use all slices from each patient, but we quickly realised that most of them were really similar. Then, from each patient we extracted 7 to 10 images, now we need our model to be able to detect and localize all these slices. To this we also had the coil information of one slice used in the ROVir implementation (see Figure 3.9). Once combined, this is another image for our dataset.

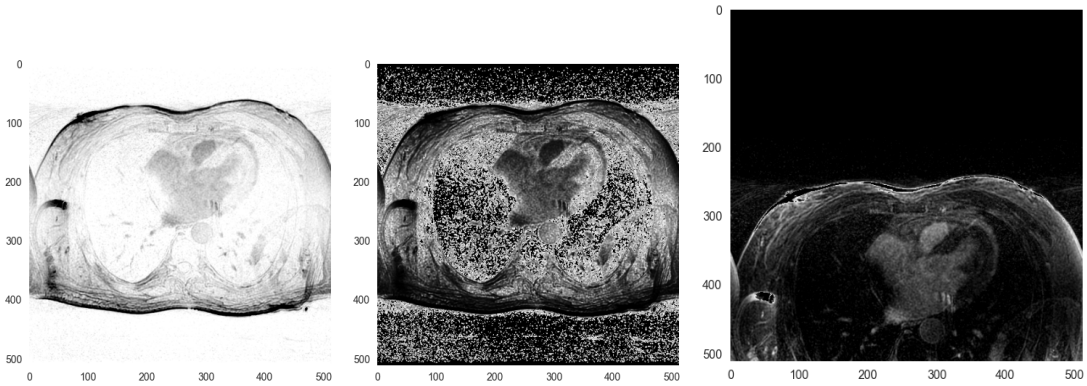
(a) Top section (b) Middle section (c) Lower section

Figure 4.2: Comparison of sections taken from the same patient.

As we can see in Figure 4.2 after the lower section we start seeing other organs that may prove challenging for our model to separate from the heart when predicting its bounding box.

Quickly, we realize that this will not be a dataset large enough to train a DL model successfully. So we needed to implement data augmentation techniques [Van Dyk and Meng, 2001] to increase the size of our dataset using our original one. For the implementation of the DL model we used the python library PyTorch [Paszke et al., 2019]. We approached our data augmentation technique with an automatic data augmentation policy following the type of data in the SVHN [Netzer et al., 2011], initially this gave us a good data augmentation but a bit extreme in some cases: Following the data augmentation philosophy of this dataset, we finally decided to opt for smaller, random transformations that include rotating the image -10 and 10 degrees, translating a 20% in the x axis and a 10% in the y axis, and setting the image into a scale of 90% to 110% the original image size.

After this process, we were able to create a dataset of 320 images total. Once the dataset images were obtained ,we had to draw the regions one by one in Label-Studio as explained in section 4.2.



(a) Inverted images (b) High noise and contrast images (c) positions unlikely to be found in a real scan

Figure 4.3: Examples of bad data augmentation following SVHN

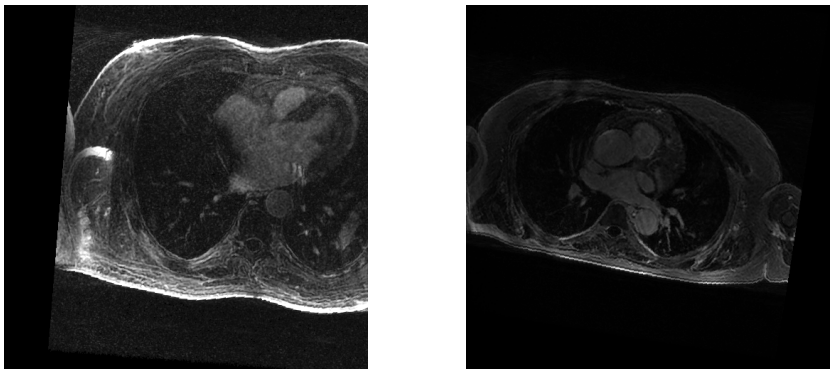


Figure 4.4: Examples of our data augmentation policy

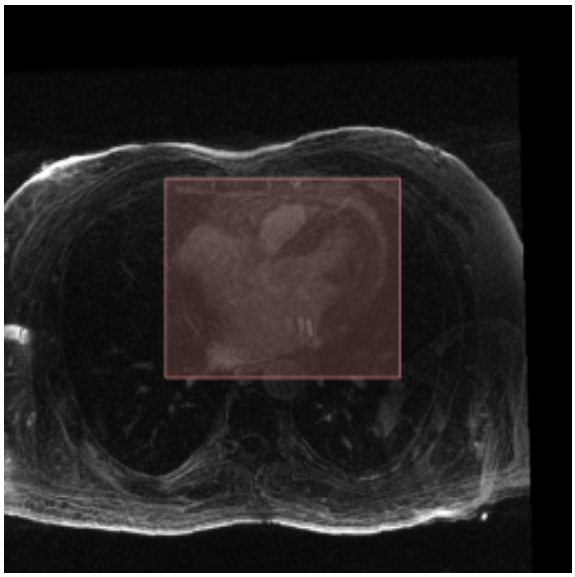


Figure 4.5: Example of annotation drawn

4.3. Model structure

Whenever approaching a computer vision problem, it is good to start with models used previously in the literature that solve a similar problem to the one you need. Then using those models as a baseline for our own model and trained them with our own dataset. This process is called “transfer learning”.

When doing research on the most used architectures for object localization, it is a good start to look at the top performers in the IMAGENET [Deng et al., 2009] competition that takes place every year and groups everywhere compete to get the highest accuracy in object segmentation. One of the top performers has been the RCNN [Ren et al., 2015] model architectures, but when trying to implement these, we quickly realised that this type of architectures was too much powerful for the necessities of the hospital.

Also, these models were originally designed to not only predict the location of an object, but to also classify it, since we are only trying to locate a heart, and we expect to only be one, and always there will be one in most of our scans. We only needed a model that would predict the region where the heart was located. Hospitals, for now, don’t have access to fast and big GPU centers that can take care of all the calls that would be needed to be made in a day. So, we need a smaller, and quicker kind of model that would predict the localization of the heart.

When trying to segment medical images, the model architecture most usually used in segmentation problems is called U-Net [Ronneberger et al., 2015]. Although this kind of model is widely used nowadays in Biomedical Imaging segmentation [Oktay et al., 2018, Siddique et al., 2021], this will prove to make our implementation of ROVir more difficult.

Thus, we needed a network that was small enough to perform fast with a limited hardware. Some studies even show the advantages of using custom neural network for the task at hand, rather than applying transfer learning from a model trained with other datasets [Amit et al., 2017]. For doing this, our initial approach was to build the simplest model and start adding blocks until we get good enough results. The advantage of our application, is that we want our model to predict a region for ROVir, and in our tests ROVir showed similar results with an extremely precise region or a more ‘loose’ one.

The design for our initial model followed:

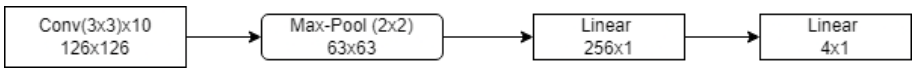


Figure 4.6: Diagram showing the structure of our initial netowrk

With this initial model we start the training process. As we trained this model, we added improvements upon it that will be discussed in the following section.

4.4. Training

For now, to train our neural network we just need that the width and height of the input image be same, so that the input is square and then we can optimize the size of an image by resizing it before forwarding through the model. During training we separated our dataset in two, 90% of the data was assigned to our training set, and the rest was assigned to a validation set. This was done in order to see if the model was suffering or not from overfitting.

A big initial mistake we made during training was not taking into account the amount of memory and how quickly the number of parameters to train can increase with the dimensions of the input image. We started training with the original image size 512×512 since our model was quite small, we managed to train a few epochs with our resources but we started having memory problems as soon as we increased the size of our model. To avoid this, in the initialization of our network we ask for the image size of the input image to train, so that we can check for the best image size during training. We finally decided for an image size of 96×96, a good balance between image information loss and memory saving.

Our biggest fear during training was the lack of generalization of the model. Although the heart will be in a similar place most of the times be, predicting a region that would be smaller or cut part of the region of the heart would make our final image worse, since it would be possible for parts of the heart to be dumped and then we would lose image signal.

For our training process, since it was a computer vision problem, we decided to go for a small batch size initially of 16 according to [Masters and Luschi, 2018] and not a very large number epochs so that we could explore fast how the model was doing.

A problem we constantly faced during training was that the model would always predict the same 4 coordinates since that was the one that minimized the loss the most. So, choosing the right loss function was crucial for our purposes. Since this is a regression, our initial guess for a good loss function was to use the Mean Squared Error (MSE) loss. For the optimization at each epoch we initially used the Adam [Kingma and Ba, 2014] optimizer already implemented with PyTorch.

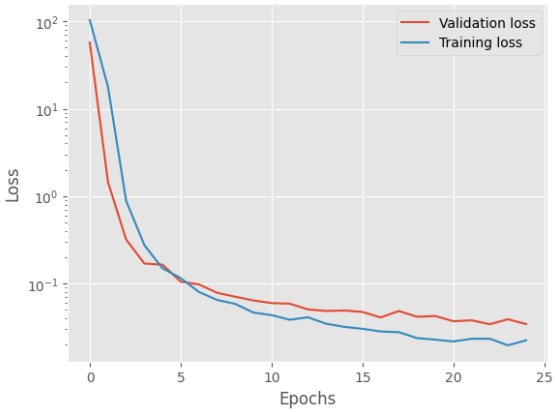
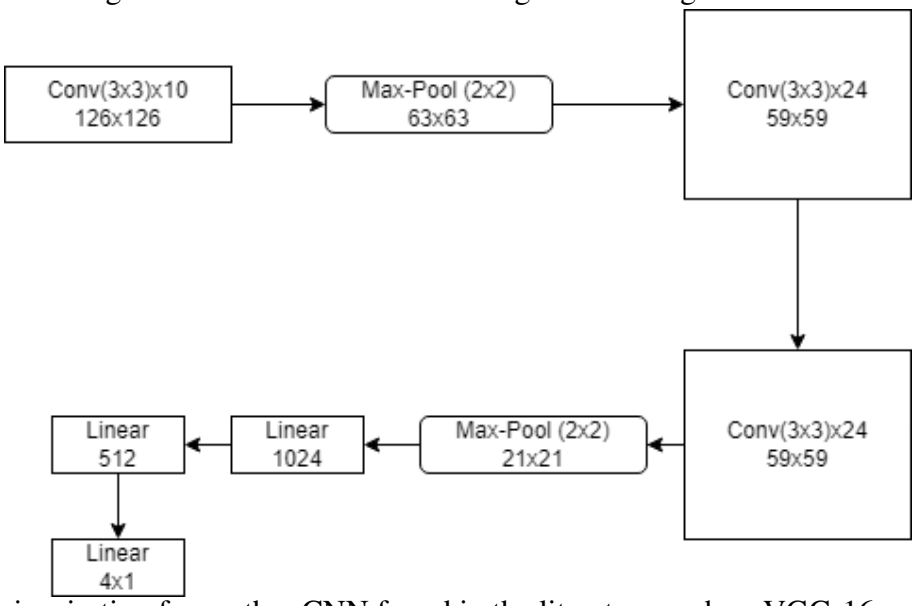


Figure 4.7: With a learning rate of 0.01

MSE is more sensitive to outliers and less sensitivie to close data points. To fix this we changed the MSE loss function for the L1 loss function. This function calculates instead

the average error between the ground truth and the prediction.

We decided also that increasing the number of layers would improve our predictions. We started increasing the size of the model until we got something that satisfied us:



Drawing inspiration from other CNN found in the literature such as VGG-16 [Simonyan and Zisserman, 2014], we enlarged our model. By adding a block of two big filters of depth 128 and a kernel size of 3, still, since a bigger kernel size would imply slower processing and more data to store. After that, a pooling layer connected to the linear output divided into three fully connected layers.

With these fixes now we got the following loss curve after training:

Figure 4.8: With a learning rate of 0.001

In this figure, we can already see a big difference between training loss and validation loss. After the 15th, epoch we start to see how the training loss reduces but the validation stays the same or even increases around the last epoch.

Figure 4.9: We can see that in all the predictions the bounding box was the same

This may be due to the fact that Adam found the best solution to our problem at around the 10th epoch and was stuck in the local minima for the rest of the training. The model found the perfect 4 points that would give it always the best results, and this turns into the same 4 regions being outputted no matter the input we tried.

One solution to avoid overfitting is early stopping. This solution consists into implementing solutions that would stop the training before the validation loss starts to worsen. For this implementation, we will keep track of the best validation loss up until that point. If the new calculated validation loss is lower than the previous best, we update the best validation loss and save the state of the model up to that point, otherwise, we continue training without saving the model. Then, by a value that we will call 'patience' defined before training, if after the last time the best validation loss was found, the number of epochs is superior to our 'patience' we stop the training since it is, at least, not learning anymore and there is no need to keep training, wasting resources in the process.

Also, as another way to avoid overfitting, we introduced Batch Normalization (BN) layers, and dropout in the final fully connected layers of the model. A study comparing the effects of BN and dropout in DL models [Garbin et al., 2020] showed that adding BN layers to a CNN model improved the accuracy of their models, but adding dropout depended on the model and the application. This time, training over 25 epochs and with a batch size of 4 we can see an improvement in the results:

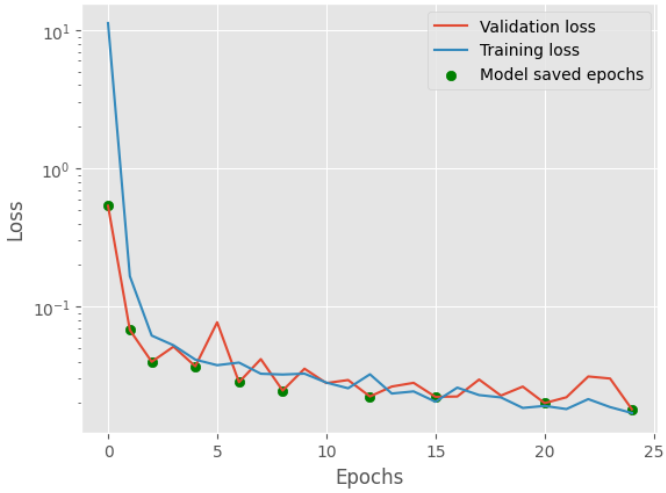


Figure 4.10: Training with the overfitting prevention techniques implemented, with a learning rate of 0.004

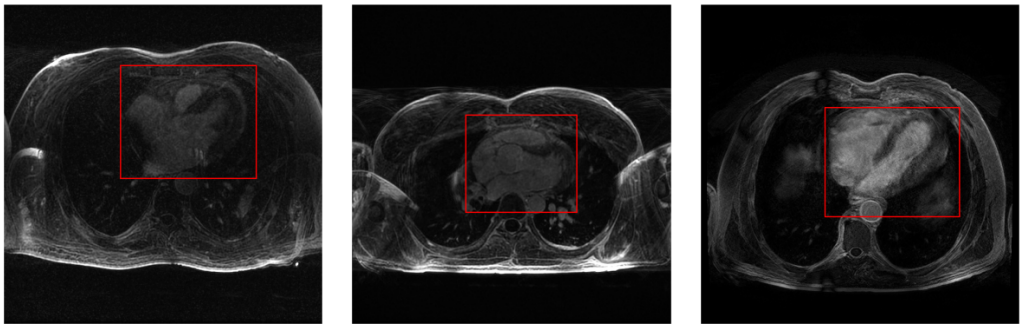


Figure 4.11: Final predictions

As we can see in Figure 4.10 the green dots show the epochs at which the model found the best validation loss up to that point to save the model. Thanks to the addition of BL and dropout in the network, early stopping was not necessary in the end, but it helped training other designs for the model until we got our final one.

4.5. Automating ROVir

Finally, now that we have a model that can predict the region where the heart of the patient is in the combined image of the coils. We must implement the methods to automate the process.

The algorithm has stayed almost unchanged, except for the region selection and number of coils to drop. To define region 'A' we will use the prediction of our model, since it gives us the points of this same region. For the region 'B' this would prove more difficult, we decided to take the first and the last 20% of the width and use the whole height as the region 'B'

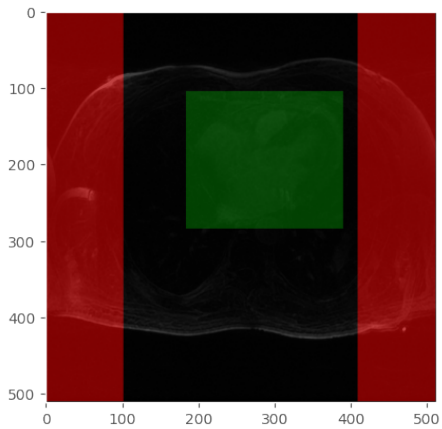


Figure 4.12: Masks automatically generated

Then, for the number of filters, we took the average intensity and of the heart region, and kept trying increasing the number of coils to drop until this average drops below a threshold (after trial and error the best threshold seems to be 70% of the original intensity) this way we balance the amount of information lost with reduction of artifacts.

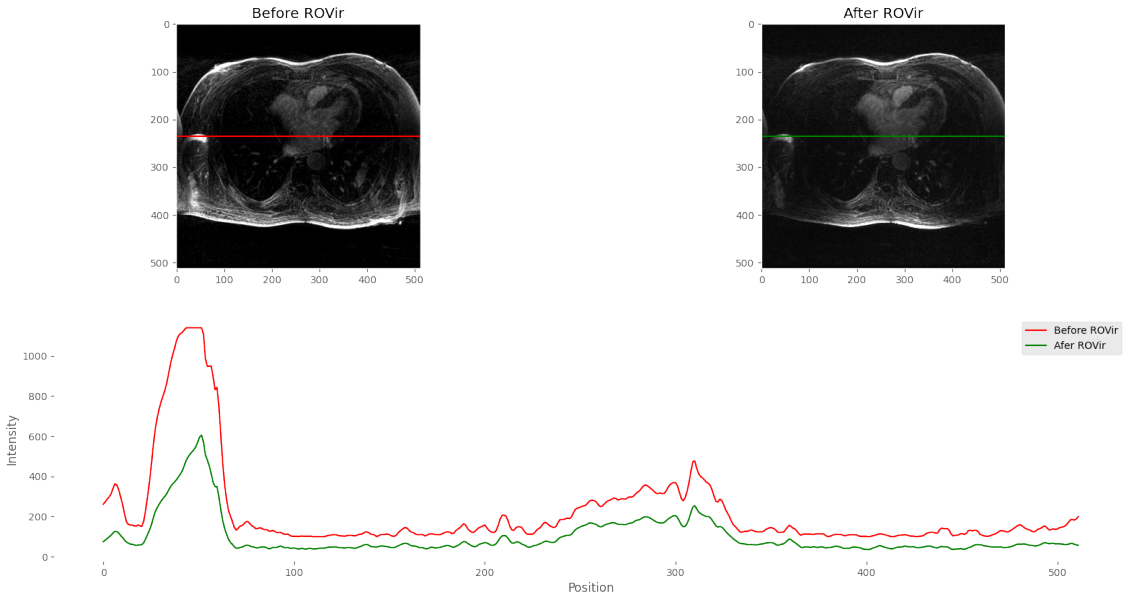


Figure 4.13: In our case, the best limit our method found was to drop at most 8 coils before losing to much information

As we can see in Figure 4.13, we were able to completely remove one wrap-around artifact on the right side, and significantly reduce the other on the left side. We were also able to mitigate a bit the acceleration artifact that is blocking the view of part of the heart, even though we reviewed that the coils used and the number of coils to drop were correct, we were still unable to remove it.

If this set of coils contained more coils that received signal only from the heart and none from the origins of the artifact. Both acceleration and aliasing artifacts would have been further reduced, if not, removed completely.

Chapter 5

DISCUSSION

5.1. Conclusion

In this project, we were able to introduce a DL method to automatize the process of reducing the artifacts produced in the acquisition of images from an MR scan.

Due to the intrinsic challenges of working with clinical data, our dataset was limited. Further exploration will be needed to see how the tool evolves with additional data. To confirm that our automatization of this artifact reduction procedure has been successful, eliminating completely the necessity for human intervention during the process and giving a noiseless output, validation by experienced cardiologists of the resulting images will be required.

We believe the presented work is a valuable contribution to myocardial delayed enhancement imaging, since this eliminates possible human error and solves the issue of finding the best number of coils to drop. Saving time, since, in the case of a failed image, there may be more issues such as contrast material being lost.

5.2. Posible Expansions

Further in the future we believe there are many points where this project could be improved upon. With more data, the model could be improved at generalization, since the data used to train the model included artifacts. This may influence in predictions where the artifacts are not present or are more severe than in the cases shown in this project. Also, the data used for training didn't contain any pathologies, in those cases the model would not be so optimized.

With our actual implementation generalizing this solution not only to the heart region, but to other organs may be possible as well. This could be implemented by either amplifying the model presented in this project to classify and locate both heart and a new organ, or building a new model entirely for each new ROI that we needed to use.

As an improvement to our own minimal number of coils to drop, we propose to create a new model that would identify the interesting parts usually seen in the scans seen in this project (myocardium, blood pool, etc.)

Bibliography

[Amit et al., 2017] Amit, G., Ben-Ari, R., Hadad, O., Monovich, E., Granot, N., and Hashoul, S. (2017). Classification of breast MRI lesions using small-size training sets: comparison of deep learning approaches. In III, S. G. A. and Petrick, N. A., editors, *Medical Imaging 2017: Computer-Aided Diagnosis*, volume 10134, pages 374 – 379. International Society for Optics and Photonics, SPIE.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[Deshmane et al., 2012] Deshmane, A., Gulani, V., Griswold, M. A., and Seiberlich, N. (2012). Parallel mr imaging. *Journal of Magnetic Resonance Imaging*, 36(1):55–72.

[Garbin et al., 2020] Garbin, C., Zhu, X., and Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, 79(19):12777–12815.

[Goldfarb, 2004] Goldfarb, J. W. (2004). The sense ghost: Field-of-view restrictions for sense imaging. *Journal of Magnetic Resonance Imaging*, 20(6):1046–1051.

[Harris et al., 2020] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

[Hunter, 2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

[Kim et al., 2021] Kim, D., Cauley, S. F., Nayak, K. S., Leahy, R. M., and Haldar, J. P. (2021). Region-optimized virtual (ROVir) coils: Localization and/or suppression of spatial regions using sensor-domain beamforming. *Magnetic Resonance in Medicine*, 86(1):197–212.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

[Masters and Luschi, 2018] Masters, D. and Luschi, C. (2018). Revisiting small batch training for deep neural networks.

[Netzer et al., 2011] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

[Oktay et al., 2018] Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., et al. (2018). Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Pruessmann et al., 1999] Pruessmann, K. P., Weiger, M., Scheidegger, M. B., and Boesiger, P. (1999). Sense: Sensitivity encoding for fast mri. *Magnetic Resonance in Medicine*, 42(5):952–962.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[Siddique et al., 2021] Siddique, N., Paheding, S., Elkin, C. P., and Devabhaktuni, V. (2021). U-net and its variants for medical image segmentation: A review of theory and applications. *IEEE Access*.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Tkachenko et al., 2022] Tkachenko, M., Malyuk, M., Holmanyuk, A., and Liubimov, N. (2020-2022). Label Studio: Data labeling software. Open source software available from https://github.com/heartexlabs/label-studio.

[Van Dyk and Meng, 2001] Van Dyk, D. A. and Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50.

[Virtanen et al., 2020] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J.,

Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

[Vogel-Claussen et al., 2006] Vogel-Claussen, J., Rochitte, C. E., Wu, K. C., Kamel, I. R., Foo, T. K., Lima, J. A., and Bluemke, D. A. (2006). Delayed enhancement mr imaging: Utility in myocardial assessment. *RadioGraphics*, 26(3):795–810.

[Waskom, 2021] Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.

[Yushkevich et al., 2006] Yushkevich, P. A., Piven, J., Cody Hazlett, H., Gimpel Smith, R., Ho, S., Gee, J. C., and Gerig, G. (2006). User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128.