Enhancing the resolution of brain MRIs using deep learning

Malo Gicquel

Master's thesis
2024:E67

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Computer Vision and Machine Learning

CENTRUM SCIENTIARUM MATHEMATICARUM

Master Thesis Report

Enhancing the resolution of brain MRIs using deep learning

Engineering mathematics
Ecole Centrale de Nantes - Lunds Tekniska Högskola

GICQUEL Malo - malogicquel@yahoo.fr

Supervisors :

FLOOD Gabrielle (LTH) - gabrielle.flood@math.lth.se
VOGEL Jacob (DeMON lab) - jacob.vogel@med.lu.se

2024

Acknowledgments

I wanted to sincerely thank my lab supervisor Jacob Vogel for enabling me to carry this project on, for all the help and for making sure to respond to all my needs. I am glad he made me discover this field and encouraged me to keeping working on this field during my PhD.

I am also very grateful to my LTH supervisor Gabrielle Flood for accepting to follow us on this project, making sure everything was going great and for helping me with all the technical matters.

Moreover, I would like to thank my lab mates Xiao Yu, Li Jun, Gabriele, Jonathan and Sara, for all the good times and the help. I also wanted to thank the rest of the members of BioFINDER, especially Macro, Linda and Nicola for helping me with the model, using Bianca and getting the data.

I also wanted to thank all the people at LTH, LU, BMC and my home school ECN who enabled me to carry on this project.

Abstract

Magnetic resonance imaging (MRI) is a medical imaging technique used to generate 3D cross-sections of body parts for medical diagnosis and monitoring. MRI scanners have a resolution that depends on the strength of their magnetic field, measured in Teslas (T). Most clinical scanners have a resolution of 1.5T and 3T resolution is used in wealthy hospitals and for research. A few research centers also have a 7T scanner, a resolution only ever used for research. Yet, 7T images offer a very detailed view of the body, which helps researching and diagnosing diseases.

Deep learning models can be used to create super resolution models that generate a synthetic 7T MRI scan from a 3T MRI scan. The generated images carry information of a higher resolution and display very tiny but important brain parts, otherwise invisible on the 3T. The creation of such a model relies on training it on pairs composed of a 3T and a 7T brain scan acquired from the same patient.

In this work, we present how we handled the preprocessing of our 7T and 3T images pairs, as well as the augmentation methods that can be used to enhance the diversity of the dataset. Then, we introduce our deep learning model, an attention residual U-net with an extra layer specific to super resolution. We also explain how to build a generative adversarial network using this model as a generator. We then discuss how we used the models on the data, the parameters of the models and how we can assess the quality of the results. Finally, we display the results of three models, one trained on a simple but small dataset, two trained on a lower quality but bigger dataset, one being a U-net and the other a GAN U-net. We discuss our achievements and the limitations that negatively affected the results, as well as what could be done to improve them.

Contents

Introduction

Presentation of the DeMON lab

BioFINDER is a cohort study based in Lund, aiming to discover key pathological mechanisms in Alzheimer's disease as well as other neurodegenerative diseases like Parkinson's disease.

Jacob Vogel's laboratory, the DeMON (Dementia Multi-Omics and Neuroimaging) lab, is a consulting data-analysis unit for BioFINDER. It researches on aging and neurodegenerative diseases, using the large data resources to model disease progression and discover contributions to disease pathogenesis.

Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is an in vivo non invasive medical imaging technique used to visualize detailed internal structures of the body. It relies on magnetic fields that interact differently with every tissue type. It generates high resolution images images that help the diagnosis of various medical conditions. It is particularly effective for imaging soft tissues, such as the brain, muscles and organs.

This method can create different image modalities by measuring different properties of the body part depending on what magnetic property it measures and on the modalities of the magnetic fields. The most used for brain MRI are T1-weighted (T1w) and T2-weighted (T2w). T2w is great at detecting fluids and fluids abnormalities. T1w provides images of anatomical structures with excellent contrast between different tissues. It is particularly good at visualizing the brain's anatomy. We will focus on this second modality.

These images can also differ by their resolution. The resolution is controlled by the strength of the magnets of the scanner, measured in Teslas (T). The most commonly clinically used scanners have a resolution of 1.5T. 3T scanners are also used in research, wealthy hospitals and specialized clinics. They provided high resolution images. 7T scanners are very rare and only ever used for research, yet they provide very valuable information about tiny structures. Higher resolution scanners exist, but are meant for researching MRI itself. Very low resolutions have also been used for portable scanners.

Going from 3T to 7T : Super resolution

7T scans offer a more detailed view of the brain, it allows a resolution of $0.65 \times 0.65 \times 0.65$ mm^3 against $1 \times 1 \times 1$ mm^3 for 3T scans. This enables us to see very tiny structures of the brain like laminar structures in the cerebral cortex and to see subcortical nuclei better, which might be very useful for dementia diagnosis. Moreover, 7T images have a much better tissue contrast, which can improve visual diagnosis or automatic tasks such as automatic segmentation.

The central idea of the project is to create a deep learning model that will learn how to go from 3T to 7T resolution. We say that we want to achieve super resolution (SR). Our assumption is that 3T images may contain information that might be used by a model to uncover more high-resolution anatomy, even if such anatomy may not be visible to the naked eye. With this goal come three questions :

- Can we visually enhance images in a way that could improve their utility in radiological diagnosis?
- Can we improve automatic tasks used for processing MR images, such as segmentation?

- Will we be able to image small and complex structures of the brain with greater resolution?

The aim of the project is to try to get as close as possible to answering these questions.

Brain anatomy

The brain is made of a few main parts shown in figure 1. Notice the cerebellum which we will mention multiple times throughout this work.

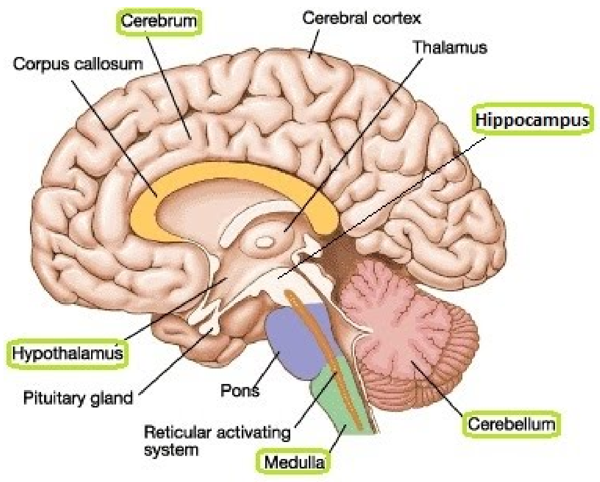

Figure 1: Anatomy of the brain. Source : "https://thealevelbiologist.co.uk/human-brain-structure-and-function/", free to use and share.

Related work

There are many related works that make us believe that this task is possible to perform. Firstly, the recent developments in deep learning have shown very impressive results on natural images, including super resolution. For instance Rombach et al. [46] show the results of their latent diffusion models. These works have also shown that deep learning models take a lot of RAM and can require to split the images into smaller blocks. Moreover, people have used these methods on medical images with a lot of success. It includes tasks relatively similar to ours, like super resolution of brain MRI from 1.5T to 3T resolution [56],[35] or super resolution from 3T to 7T but in a different modality [17].

Older techniques have also been tested on the task with a simple CNN [45], this was state of the art during my project. We used a more complex and state of the art model (U-net and generative adversarial network) with newer methods (residual blocks, attention mechanisms, conditioning, a perceptual loss, an edge sharpening layer). However, they used different inputs, as they use both the spatial and wavelet domains. We also used a bigger and more diverse dataset. Lastly, they decided to split the images into 2D slices (along the transversal dimension) to save RAM, which we also decided to do. Around the end of my thesis, a paper using recent techniques was released by Qiming Cui et al. [14]. It mentions good results and is now state of the art. Still, our work introduces new techniques like attention layers, an edge sharpening layer and a different architecture for the model (U-net against a V-net). Also, they work on 3D cubes of size $64 \times 64 \times 64$ instead of 2D slices.

Notations and abbreviations

- H : height
- C : number of channels
- D : depth
- W : width
- SR : super resolution
- GAN : generative adversarial network
- CA : cross attention
- SA : self attention
- Res : reisudal
- GMM : gaussian mixture model
- CSF : cerebro spinal fluid
- GM : grey matter
- WM : white matter

1 Data preprocessing

To build an efficient model, we need to make the data as easy to use and as diverse as possible. We thus want to remove data flaws and use a wide diversity of images in order to capture the heterogeneity of brain scans. Indeed, a brain lesion that the model has never seen during the training could be wrongly processed and be misleading or even missing on the high resolution generated image. This is especially important as we are working on medical images. Another major issue is that every scanner has its own settings. As a result, images from different scanners may vary slightly [39], which could worsen the model performance during inference. To remove the data flaws, we can use preprocessing tools. To artificially solve the diversity issue, we can augment the data.

1.1 Description of the datasets

Our data is composed of images saved as NIfTI-1 files and comes from two sources :

- **The UNC paired dataset :** this dataset contains 10 sets of T1w MRI brain scans on 3T and 7T resolution. It was made available by Xiaoyang Chen et al. [13], who defaced and aligned them. Since they are accessible on my computer and already aligned, this dataset was especially useful to run first experiments. The characteristics of the participants are shown in figure 2. All of them are cognitively normal. The aligned 3T and the 7T images are of size (256, 304, 308).

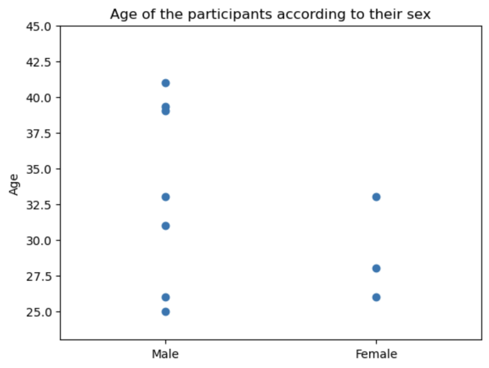

Figure 2: Age and sex of the 10 subjects of the UNC dataset, each subject is represented by a dot.

An example of scan from this dataset can be seen in figure 3.

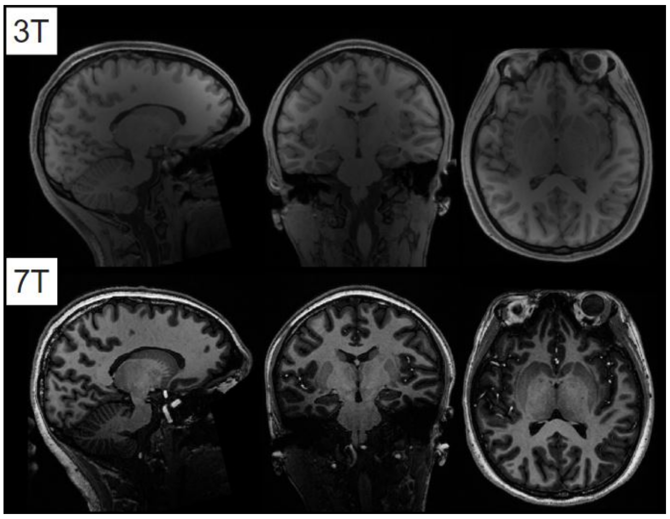

Figure 3: Example of 3 slices, one per dimension, of a scan in 3T and 7T resolution, from the UNC dataset.

- **The BioFINDER study :** this is a secured dataset provided by my laboratory. It contains a lot of scans and includes 172 unprocessed 7T T1w images that come from the same scanner, each paired with at least one 3T T1w scan [6]. The paired scans were taken at different points in time and were usually acquired with a gap of one year. We only used the closest 3T scan in time, but it could be interesting to use them all for a better generalization. The participants are older than in the first dataset and each subject has a clinical diagnosis made at a specialist memory clinic. The diagnosis can be : "Normal" - 74 patients -, "subjective cognitive decline" (SCD), which means the patients felt impaired cognitively but are not clinically diagnosed as such - 50 patients -, "mildly cognitively impaired" (MCI) - 46 patients - and "demented" - 2 patients -. The "MCI" and "dementia" diagnosis have been given according to the DSM-5 criteria [2]. We separated the diagnosis in two groups : the "normal" and "SCD" patients together and the "MCI" and "demented" patients together. We can see the age distribution according to the diagnosis and sex in figure 4.

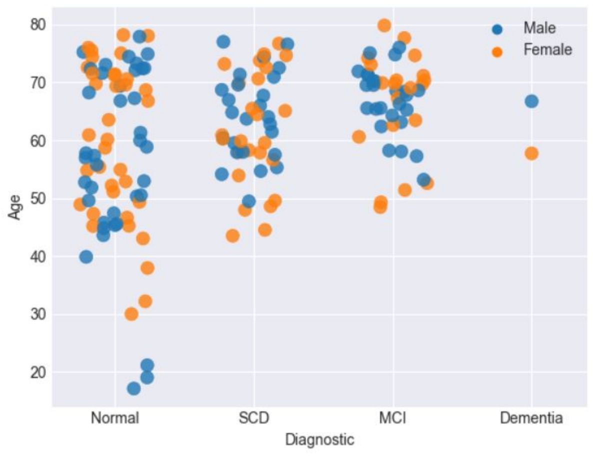

Figure 4: Age and sex of the participants of the BioFINDER 2 dataset, according to their diagnosis. Each dot represents a participant and has a small random offset along the x-axis for more visual clarity.

The BioFINDER images are lower quality than those of the UNC dataset. Some preprocessing (skull stripping and bias field correction, see the next section) has been performed on the BioFINDER 3T images, while the BioFINDER 7T images are still completely raw. The 7T images are of size (352,352,272) with a lot of background voxels. The images of the UNC dataset are raw, although they were aligned and defaced.

Examples of scans from this dataset can be seen in the next subsection (section 1.2).

1.2 Removing flaws

1.2.1 Bias field correction

A common issue for MRI scans is the presence of a bias field [29] : a very smooth low-frequency signal that corrupts the images. An example can be seen in image 5. The state of the art method to solve this issue is the n4 bias field correction algorithm [52], which is accessible on the software ANTs [53].

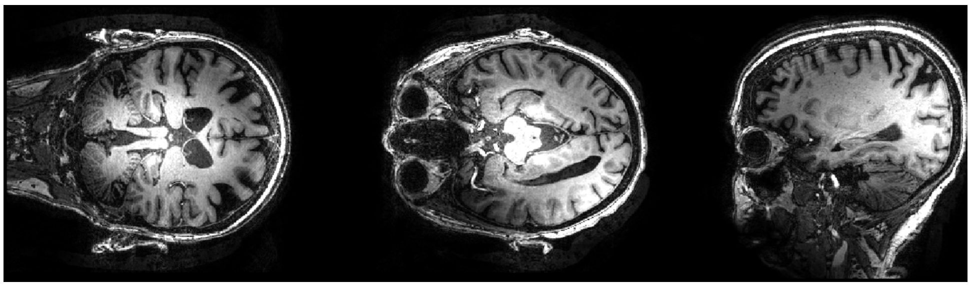

Figure 5: Example of 3 slices, one along each dimension, of a 7T scan, notice how the intensity of the white matter is much higher towards the left back and center parts of the head.

The skull stripped images of the 3T BioFINDER dataset have been bias corrected too. We show the raw 3T scan of the same patient in image 6 and the skull stripped bias corrected version in image 7. Notice how the bias in the raw 3T of BioFINDER is much less important than in the 7T scans.

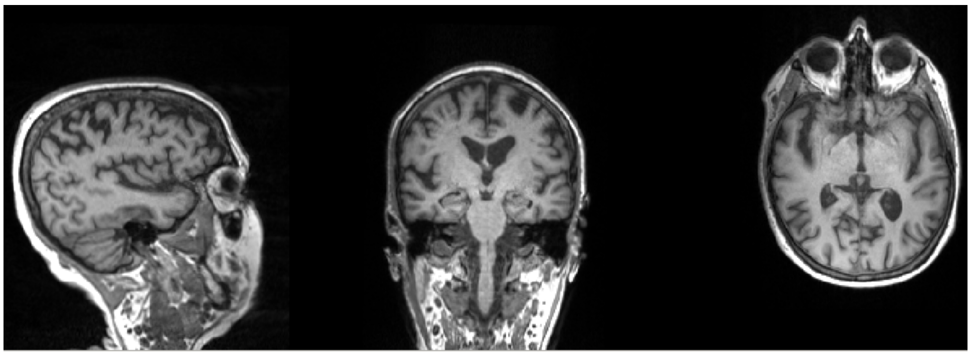

Figure 6: Example of 3 slices, one along each dimension, of a 3T scan of the same patient as figure 5. The intensity scales of this image and the previous image are not the same and they should not be compared.

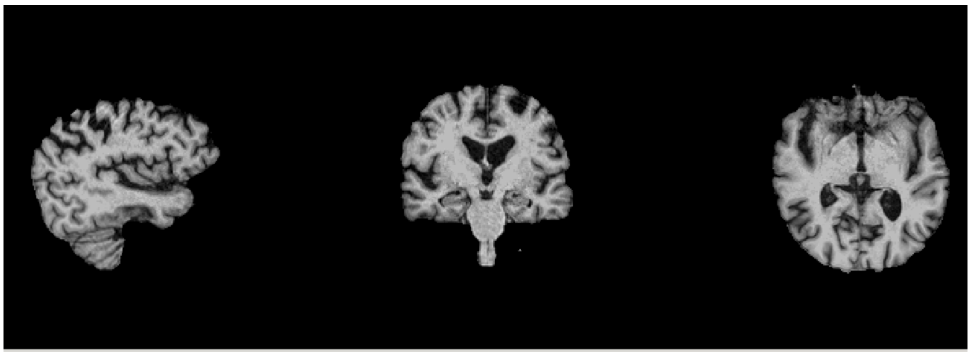

Figure 7: Example of 3 slices, one along each dimension, of a 3T brain of the same patient as in the images 5 and 6, here the intensity of the white matter is the same everywhere since they were bias field corrected.

It is important to remove this bias on both image resolutions. When it comes to the 3T images it will make the task easier for the model, as the intensity of a voxel will be better correlated with its tissue type (white matter, grey matter or other). For the 7T images it is paramount, indeed the modalities of the bias field are random, so the model cannot predict it and will average it out, making the evaluation of the performance of the model less accurate (see section 4.2).

The bias field is much harder to correct on the 7T images than on the 3T images. The first reason behind that is that the n4BiasFieldCorrection algorithm has been extensively used on 3T images and its default parameters are perfectly suited for this resolution. Meanwhile, few people mention the parameters suited for 7T scans. A second issue is that the background, skull and neck can make the algorithm perform badly. This can be solved using a mask indicating where to calculate the bias field. A mask is typically created using a threshold but this cannot be done here as the intensities near the edges of the brain are too close to those of the background (look at figure 5). To solve these issues I followed this procedure :

- Do a first bias field correction.
- Perform a skull stripping using MRI SynthStrip [25] keeping the cerebrospinal fluid (CSF). The resulting skull stripped image can then be used to create a brain mask.
- Do a second bias field correction with the brain mask.

I also did the last steps while tuning the parameters of the algorithm on a few images to optimize them. With this procedure, I got very good results, which can be seen in figure 8.

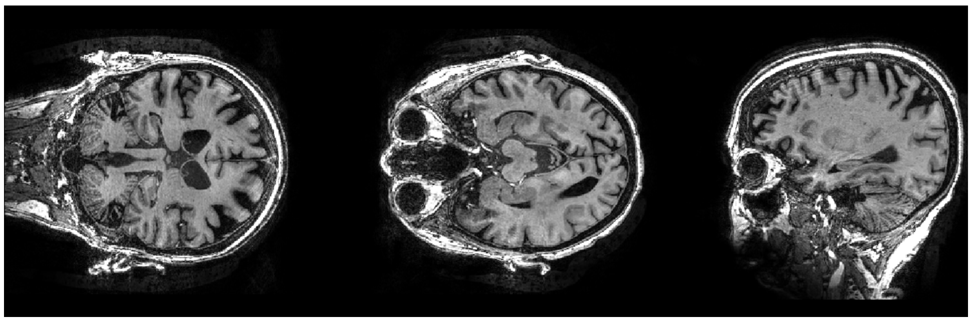

Figure 8: Example of 3 slices, one along each dimension, of a bias corrected 7T scan of the same patient as in figure 5. The intensity scales of this image is not the same as any other image.

Remarks :

- The first bias field correction is not so important, but it can improve the results of the skull stripping.
- The parameters I ended up using are :
  - Bias FWHM : 0.18
  - Wiener filter noise : 0.1
  - Convergence Threshold : 1e-7
  - Maximum number of iterations : 150x4 (150 iterations at four resolutions)
  - Shrink factor : 4

  The results are very slightly better when the shrink factor is smaller, but the running time is much higher (run time with shrink factor 4 : around 2 minutes, with shrink factor 2 : around 10 minutes).
- The skull stripping algorithm MRI SynthStrip [25] is a deep learning model that has not been trained on 7T images. Yet, it has been trained on 1.5T and 3T images in different modalities (T1w, T2w, infant T1w, flair, PDw, DWI, CT and PET) and using many augmentation techniques to make it work on a wide variety of brains. It thus also worked on our 7T scans. The results were not perfect, but very good.
- We have been told that another way to remove this bias field is to apply the algorithm multiple times in a row.

1.2.2 Corruptions and bad acquisitions

As said previously, MRI relies on magnetic fields whose interactions with the brain tissues are measured. The issue with this method is that the field can react strongly with undesired objects, such as a metal tooth for instance, which results in a corruption in the image. This is particularly the case with 7T images, as the magnetic fields are very strong. This cannot be easily solved and we decided to remove the corrupted image parts during training. In practice we did it by only including the 2D slices above the corruption (along the transversal dimension), as our model will be used on transversal 2D slices (see section 3.1). Examples of corruptions we removed, although we are not sure what exactly caused them, are shown in images 9 and 10.

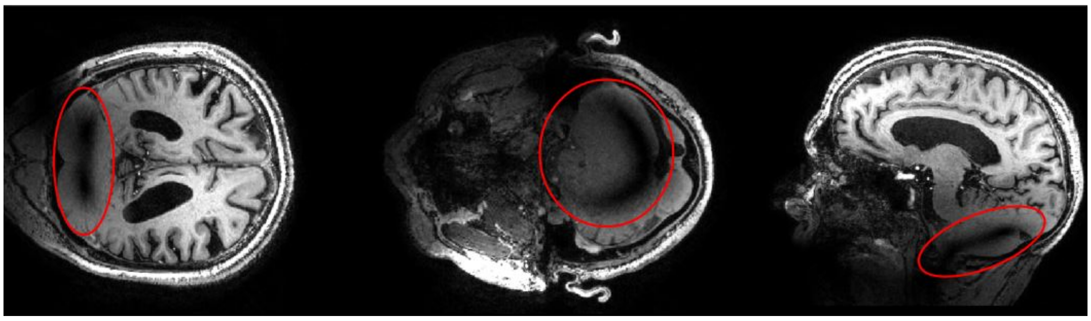

Figure 9: Example of 3 slices, one along each dimension, of a 7T scan, notice the huge black corruption in the red circle.

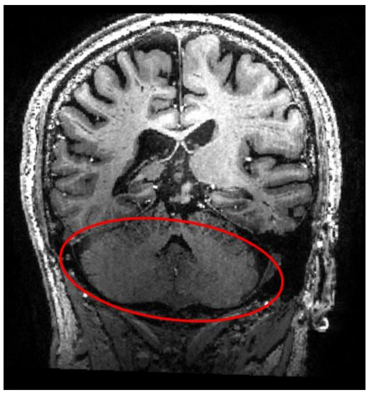

Figure 10: Example of a slice of a 7T scan with a grey corruption, the part of the cerebellum in the red circle is extremely smooth.

A second issue is that an acquisition takes a long time during which the patient can move, thus creating artifacts. The most common artifact is stripes (see the stripes in figure 11), that cannot be removed without a 4D acquisition, which we do not have. For most patients, the stripes are small and will probably be averaged out by the model as they are completely unpredictable. They will unfortunately make the evaluation of the performance of the model a bit worse. Some patients moved a lot, which not only resulted in big stripes but also a blurry image. We decided to remove these images as they could worsen the model, which resulted in 7 removed images. An example of a removed scan is shown in figure 11.

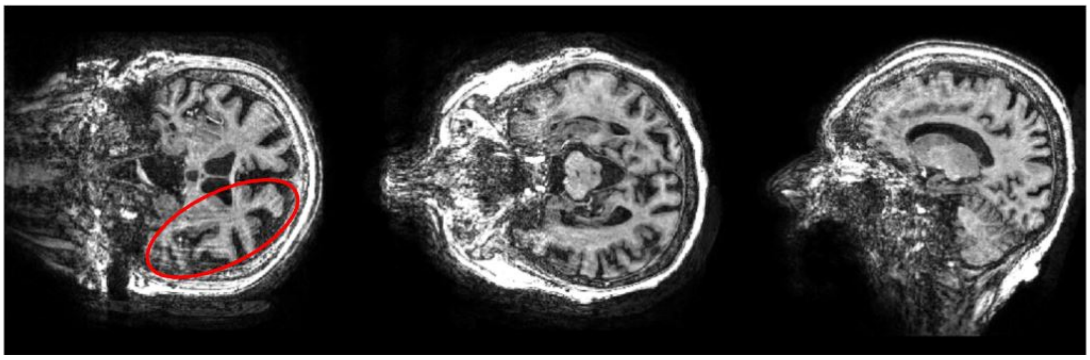

Figure 11: A 7T scan with too much movements, notice the big stripes in the red circle and how the front of the head is blurred out.

1.2.3 Coregistration

Another important task to perform is coregistration. Indeed, in the BioFINDER dataset, the brains are not aligned and the 3T image is smaller. This is especially problematic as the location of the 7T brain is independent from the location of the 3T image and thus unpredictable. The registration is also necessary when it comes to cutting the images in matching parts. The registration tools of ANTs are very efficient for this kind of task. We used the default tool which relies on the algorithm SYN [3]. This algorithm does an affine transform, which has 7 degrees of freedom (3 for a translation, 3 for the rotation and 1 for the scale) and a small deformation map (which has many degrees of freedom). To make sure this was as efficient as possible, I skull stripped the 7T bias corrected images without CSF using the same tool as before (SynthStrip [25]) and coregistered the 3T brains to their corresponding 7T brain. The registration worked perfectly. An example is shown in figure 12.

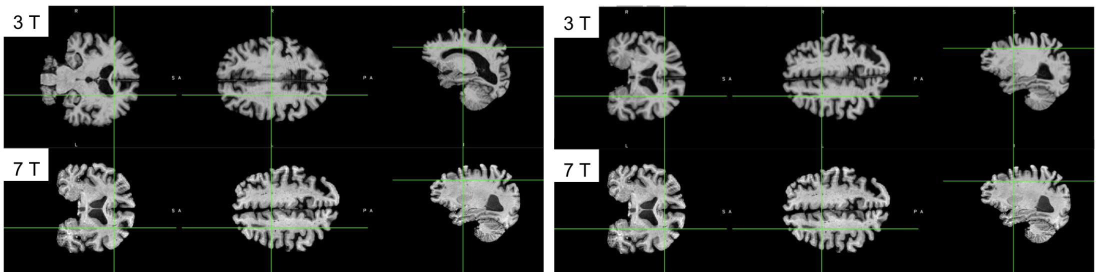

Figure 12: On the left : 3 unregistered slices of a skull stripped and bias corrected brain in 3T and 7T resolution. On the right : 3 registered slices of a skull stripped and bias corrected brain in 3T and 7T resolution, of the same patient. The green cross indicates where the slices are. Notice how the slices do not match on the left image and match on the right image.

The registration was a success. The transform used has been saved and can be used to register the full scans too.

1.2.4 Anonymization

The BioFINDER data is considered sensitive data, especially as a scan can be used to reconstruct the patient's face. The patient could also be found using their associated data (birth date, diagnosis and gender). To avoid privacy issues, we did all the preprocessing on secured servers and only extracted anonymized data, which is the skull stripped images without their associated index. We also kept the birth dates secured and rounded the ages to prevent anyone, including us, from easily matching patients and brains.

1.3 Solving the scanner diversity issue with augmentations

As said previously, the model should work for a wide variety of scanners and brains. This variety can be expanded using data from other labs, but it can also be artificially increased using different augmentations. Even though we explored the different solutions, we did not implement them as we were short in time and we were not always sure that they would improve the results. Considering that we have a large dataset, we should have a sufficiently large dataset to have a working model anyway. In this section we talk about the possible ways to expand our dataset.

1.3.1 Gaussian mixture model augmentation

Maria Ines Meyer et al. [39] proposed a gaussian mixture model (GMM) data augmentation to help deep learning models generalize to other 3T scanners. The idea is that there exist three main kinds of brain elements : white matter (WM), grey matter (GM) and cerebrospinal fluid (CSF). Each have their own intensity distribution determined by the scanner settings. Although these settings have other impacts over the image characteristics (e.g. noise, artifacts, distortions), the impact of this phenomenon is said to be significant on segmentation tasks and seemed interesting for our purposes too. This method is performed on the bias corrected skull stripped brain with the CSF. We first have to change the range of the data. So as to do it, we clip the voxels whose intensity belongs to the one percent lowest or one percent highest intensities of the image and calculate the minimum and maximum and perform the following transform, where I_c is the clipped image :

$$I' = \frac{I - \min(I_c)}{\max(I_c) - \min(I_c)}$$

These percentiles are removed because very intense parts of the image (the blood vessels) can vary a lot and are not that relevant to this task, see an illustration of this phenomenon in figure 13.

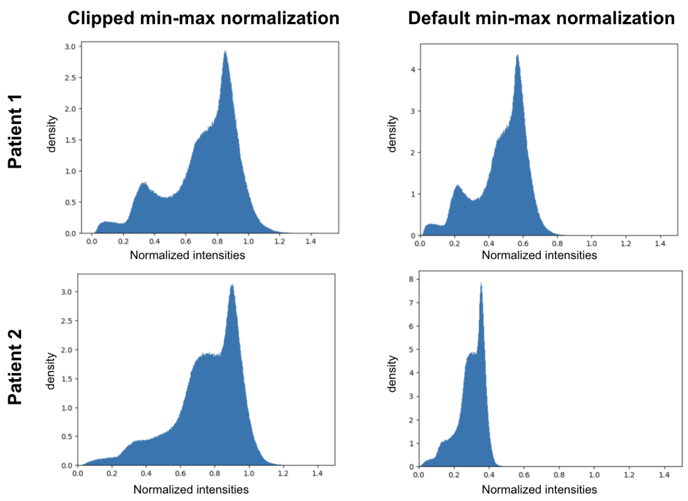

Figure 13: Intensity distribution of two 3T brains with CSF after two normalization strategies. Notice how the clipped min-max normalization gives more consistent results between subjects.

Then we can approximate the intensity distribution p as a sum of three weighted gaussians (each representing the intensity distribution of one brain element CSF, WM or GM) using sklearn's tool GaussianMixture [40] with 3 components. We note π_k the weight of the k^{th} gaussian and $f_{\mu,\sigma}$ the probability density function of the normal law of mean μ and of standard deviation σ.

$$\forall x \in \mathbb{R}, f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

We approximate p by :

$$p \simeq \sum_{k=1}^{3} \pi_k f_{\mu_k,\sigma_k}$$

We display an example of normalized intensity distribution along with the approximation and the three weighted components in figure 14.

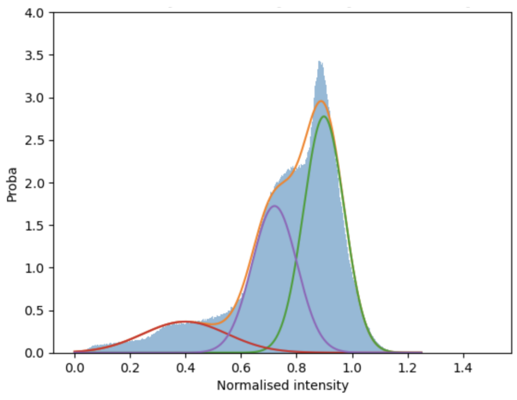

Figure 14: Distribution of the intensity of a brain with CSF of a UNC subject in blue, with the gaussian mixture model in orange in its 3 components in red, purple and green, which are respectively mostly associated with CSF, grey matter and white matter.

Unfortunately, two peaks are very close, the grey matter (purple component in figure 14) and the white matter (green component in figure 14) are not well separated by the gaussian components. This makes the gaussian components no so relevant which reduces the quality of the model.

According to Maria Ines Meyer et al. [39] the means and standard deviations of the three components are mostly scanner dependant. To artificially expand our dataset, we introduce small variations to the mean and std of each component k : q_{μ_k} and q_{σ_k}, such that our new means and stds are $\mu'_k = \mu_k + q_{\mu_k}$ and $\sigma'_k = \sigma_k + q_{\sigma_k}$. The range of these variations must be found by looking at images from many different scanners, which we could not do and would thus require us to contact the Maria Ines Meyer et al. . In order to keep the structural information, we want to modify the voxels while preserving their distance to the mean, defined for every voxel v by $d_{\mu,\sigma}(v) = \frac{v-\mu}{\sigma}$. In order to do that we apply the following transformation to a voxel v belonging to the k^{th} component :

$$v' = \mu'_k + d_{\mu_k,\sigma_k}\sigma'_k$$

If needed, the skull can then be added back and the augmentation is done. An example of such augmentation can be seen in figure 15, along with the distribution of the augmented brain shown in figure 16.

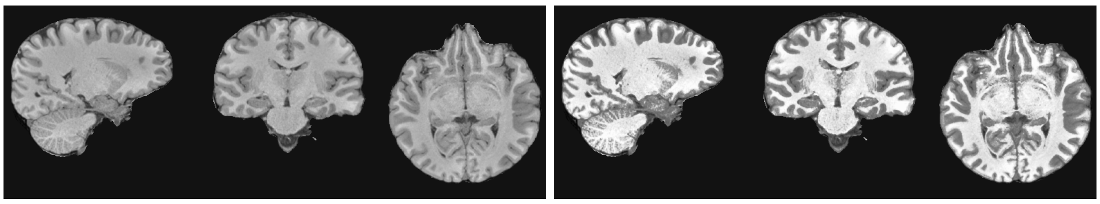

Figure 15: Three slices of a brain and on the right their augmented version.

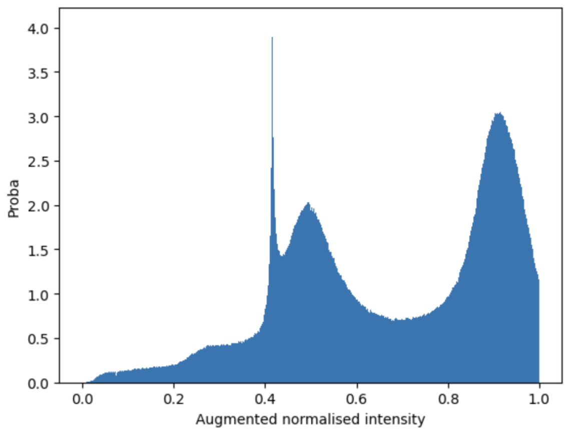

Figure 16: Intensity distribution of the augmented brain of figure 15. There is a small bug around intensity=0.4, which is easily solvable and is due to the fact that some voxels have intensities too far from the component they should belong to.

1.3.2 Other data augmentations

Many deep learning models rely on several augmentation techniques for brain MRIs [19], here are the ones that we thought were interesting :

- **Image distortions :** to account for the variety of brain shapes, we can apply grid distortions or elastic transforms to the data. This is represented in the figure 17 below :

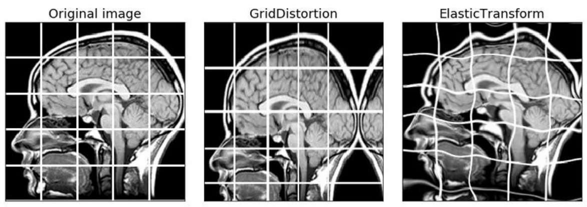

Figure 17: Alexander Buslaev et al. [9], example of grid distortion and elastic transform. Available via license CC BY

  The advantage is that we can apply the same transform to both of the paired images. Unfortunately, the results can be unrealistic which can make the results worse [20], which sounds especially bad for super resolution.

- **Adding noise :** 7T images have a better signal to noise ratio [50] than 3T images. It could be interesting to add noise to the images to help the model learn how to take it into account. This is known to be extremely important in super resolution tasks involving real life images [46].

- **Adding lesions :** some papers mention taking a lesion from a brain and artificially adding it to a healthy brain [48]. This task seems to be time consuming and requires a lot of care to be done successfully.

- **Linear shifts :** changing the location of the brain in the image. This did not seem so useful as we have many pairs.

- **Intensity modifications :** this consists in changing the intensity distribution of the image. The easiest way to do it is to multiply by a factor close to 1. The aforementioned gaussian mixture model data augmentation [39] is another example.

2 Deep learning super resolution models

Super resolution (SR) is a complex task, as such, it has heavily benefited from the development of deep learning. The first big development was super resolution convolutional neural networks (SR CNNs), as they are known to give good results while being very easy to train and do not require a huge dataset (10 pairs of images can be enough to get a working model).

The most popular super resolution method is perhaps the super resolution generative adversarial networks (SR GANs). It is an evolution of SR CNNs in the sense that it uses a CNN to generate a high resolution image (the generator) and a second CNN trained to estimate the probability that a given image (real or generated) is fake, we call it the discriminator or the critic. During training, both are updated, the discriminator learns how to tell a fake image from a real one, while the generator essentially learns how to fool the discriminator. This is known to improve the results significantly and has widely been used in medical super resolution tasks [4], [1], [51].

Finally, the recent developments of diffusion models have shown nice results in natural image super resolution and have very recently started to be used for several medical image processing tasks such as fake brain generation [42] and super resolution of brain T1w MRIs going from 1.5T to 3T [56]. Although there are still not so many such studies, as their development is still very recent and as these models are very complex. Moreover, some people have shown that diffusion model do not really perform better than GANs for super resolution of natural images [32]. One can also be concerned that such models would generate fake artifacts, like text to image models do by creating hands with 6 fingers for instance. Diffusion models are also known to be more complex and take a lot of compute time. For all these reasons, it did not seem reasonable to build and train a SR diffusion model.

2.1 Super Resolution using a CNN

The model we used is based on Jonathan Ho's code [24], he used this model as a noise estimator he used to build a diffusion model. It is a CNN built using classical layers, residual blocks and attention mechanisms.

2.1.1 The main layers

CNN models are based on four main layers :

- **Convolution :** 2D convolutional layer [16] with a kernel a size 3, 0 padding of 1 and stride 1 (=apply the convolutions on each pixel) to keep the image size.
- **Activation function :** the most common and simplest activation function is ReLU (rectified Linear Unit), it basically keeps the value as is if it is bigger than 0, otherwise it sets it to 0. It is very efficient to generate features and makes the model very easy to train.

  To add some complexity, LeakyReLU [59] can be used. It keeps the negative values using a small positive slope with a predetermined parameter noted α. This parameter can also be trained using Parametric ReLU (PReLU).

  More recently, people have started using more complex functions such as SiLU (Sigmoid Linear Unit, also called Swish) [23], which we used. It does not weight the values according to their sign like ReLU but more according to how big it is compared to the other values. It is equal to the identity function times the sigmoid function σ.

$$\begin{aligned}
&\textbf{ReLU} : x \mapsto \max(0, x)\\
&\textbf{LeakyReLU}(\alpha) : x \mapsto \max(0, x) + \alpha \min(0, x)\\
&\textbf{SiLU} : x \mapsto x\sigma(x) = \frac{x}{1 + e^{-x}}
\end{aligned}$$

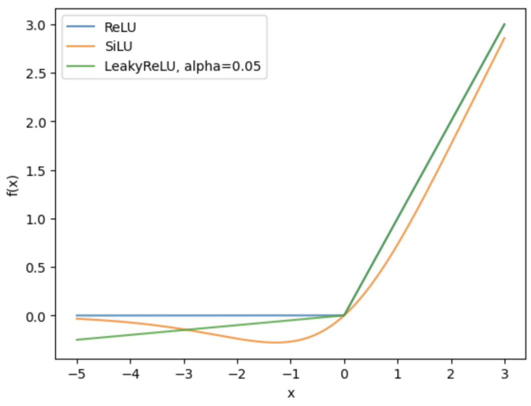

Figure 18: Graphs of three activation functions : ReLU, LeakyReLU and SiLU.

- **Normalization layer :** there exists different normalization layers : batch normalization, layer normalization, group normalization etc. These layers are used to solve the internal covariate shift [27], a problem that makes training harder and where the distribution of the features shifts because of the weight updates. To solve it, it creates groups of pixels/voxels, channels and mini-batch elements, which are then normalized to a trained mean and variance. This makes the distribution easier to use for the following layers during forward propagation. Figure 19 represents the different grouping strategies, by highlighting in blue an example of a normalization group.

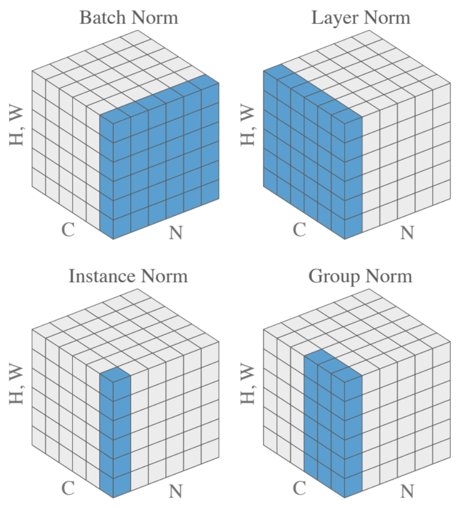

Figure 19: Groups of pixels among which the normalization happens, the “C” dimension represents the channel/layer dimension, the “N” dimension represents the mini batch dimension, while the “H.W” dimension represents the flattened image. Image Credits: Siyuan Qiao et al. [44], licensed under CC-BY.

Batch normalization normalizes along the mini-batch and image dimensions. This is not efficient when using very large images like we are, as the batch size is small. We can then use layer or group normalization, which normalize along the image dimension and one or more (respectively) groups of channels. We used group normalization (GN).

- **Downsampling :** The main flaw of convolutions is that they only process the image at one scale, by looking at the relationships between a pixel and its neighbors. To generate features at a wider scale, we can include a downsampling layer, that divides each dimension by two. This is typically done either using a max pooling (grouping the pixels by four and keeping the maximum only) or using a convolution with stride 2 (skips every other pixel/voxel). In our model, we used max pooling.

Other important layers :

- **Upsampling layer :** after a downsample, one can need to upsample the image to turn it back to its original size. The most common method is to do an interpolation. We used nearest neighbor interpolation.

- **AdaDM :** Adaptive Deviation Modulator (AdaDM) is a layer created by Jie Liu et al. [37]. It is specifically used for super resolution to compensate one flaw of the normalization layers : they make the edge information less distinguishable. This is a big issue as making edges sharper is an important part of super resolution. They say this is due to the fact that normalization layers make the feature standard deviation very small and proposed to increase it with their layer.

  Let x be the input and h the output after a few layers including a normalization layer. Then the AdaDM layer will output :
$$h \exp(w \log(\sigma(x)) + b)$$
  Where σ is the standard deviation of each mini-batch element and w, b are trained parameters.

2.1.2 Residual Blocks

These layers are often organised in **Residual Blocks**, the one we used is shown in figure 20.

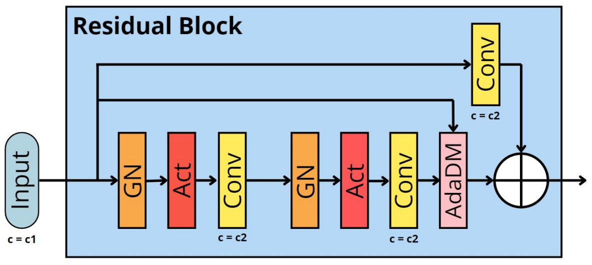

Figure 20: Drawing of the residual blocks used in Jonathan Ho's code [24].

These blocks are composed of the **residue** (the bottom path of figure 20), where the input is processed by going through a series of layers, here (group normalization, activation, convolution)x2+AdaDM. The input is then added to the residue.

The first convolution can modify the number of channels. If so, the input also goes through a convolution that does it too.

The reason why we add the input to the residue is to put a **skip connection**. These connections are here to address the degradation issue that appears when using a very deep model [22], where the information can have a hard time propagating during back propagation. This is solved by creating paths on which the information can skip layers, which is what is done in a residual block.

2.1.3 U-net model

The main flaw of convolutional residual blocks is that they only generate features at the lowest scale. As mentioned earlier, this can be solved using downsampling layers. In order to use them, we use a U-net architecture (figure 21).

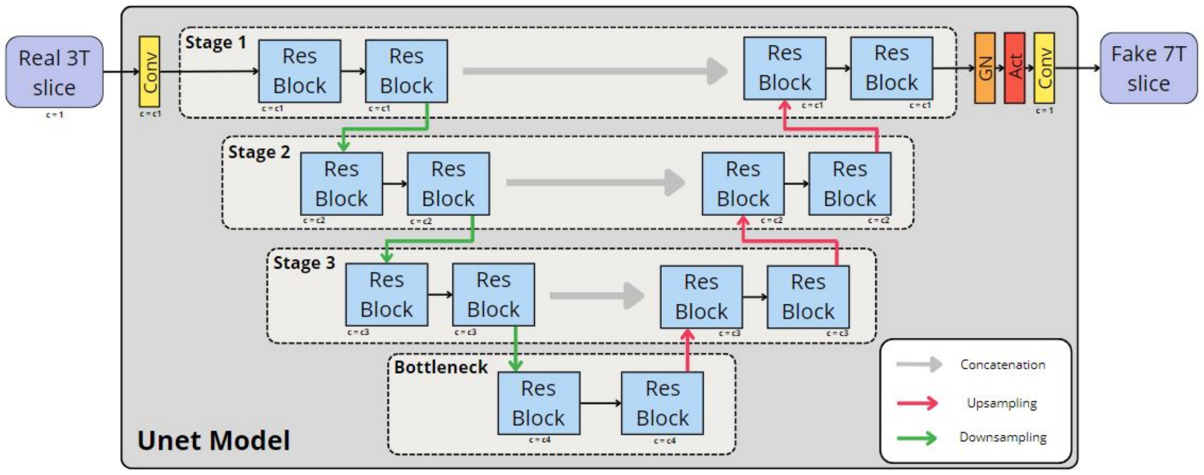

Figure 21: U-net architecture based on Residual (Res) Blocks shown in figure 20, that takes a 3T slice as an input and outputs the estimated 7T slice. c_i is the number of channels in the i^{th} stage.

After a first convolution that sets the number of channels to $n_{channels}$, features of the lowest scale are generated by n_{res} residual blocks. These features are saved for later and are also downsampled. The downsampled features then go through n_{res} residual blocks, which generate features at a wider scale. Then the process is repeated a certain number of times until we reach the bottleneck where the features are not downsampled, while the number of residual blocks is set to 2 (although I do not know the reason behind this choice made in Jonathan Ho's code). At this point we have generated features at different scales and we want to process them all to generate the final result.

To do so, we upsample the features and concatenate them with the features generated by the previous stage. The concatenation also acts as a skip connection, which improves the information propagation during back propagation. The result is then processed by n_{res} residual blocks. This is then repeated until the first stage is reached.

At this point, information at different scales has been generated and processed. To generate the final output, it goes through a final group of normalization+activation+convolution.

U-nets have been created for segmentation of medical images [47] and are now used for many other tasks such as generation and discrimination in GANs [49], noise prediction in diffusion models [24], super resolution [26] etc. Depending to the task, each scale of feature is more or less important. For segmentation, high level features are very important as segments can be very large. For super resolution the information at the lowest scales is probably the most interesting. We say it is a low level task.

Remarks :
1) The output has to be of equal size or smaller than the input, if it is not the case one can add an upsampling operation before the U-net. In our case, our images are of the same size thanks to the coregistration, that upsamples the 3T images during the process.

2) At each stage, the number of channels is a multiple of the initial number of channels $n_{channels}$, changed in the first residual block. The deeper the stage, the more channels there are. We can describe the number of channels as an increasing sequence of integers $(n_1, n_2, ..., n_k)$, where k is the number of stages and such that for the i^{th} stage, the number of channels is n_i times the initial number of channels. Since we are working on a low level task, we typically use $(1, 2, 2, ..., 2)$ and $k \in \{4, 5\}$.

2.1.4 Attention mechanisms

As said previously, U-nets are based on convolutions that create local features of the input images on different scales. Yet, some distant parts of the image can be related and it can thus be interesting to study more global information. This can be done using self attention mechanisms. It can also be interesting to take into account external variables, such as diagnosis, age, sex. This can be done using cross attention mechanisms.

Attention mechanisms in natural language processing

Attention mechanisms were popularized in 2017 by Ashish Vaswani et al. in the notorious "Attention is all you need" [55] paper, as a natural language processing deep learning tool. This mechanism is now widely used in language and image processing. It has been created to solve an important issue in language translation : a word can depend on another one that is very far apart. To solve this, we create a matrix describing how much two words are related, with a value between 0 and 1. We call it an attention matrix.

When using it for image processing, it indicates the relationships between two pixels of the image (self attention) or between a pixel and an external variable such as age, sex, diagnosis or features generated previously in the model (cross attention).

This is particularly interesting when it comes to a segmentation, as an attention map can represent long range interactions between the different segments. An example of this is highlighted in figure 22. An example of this has been done by Olivier Petit et al. [41], who used a U-net with self attention at the bottleneck and cross attention mechanisms instead of concatenation as skip connections.

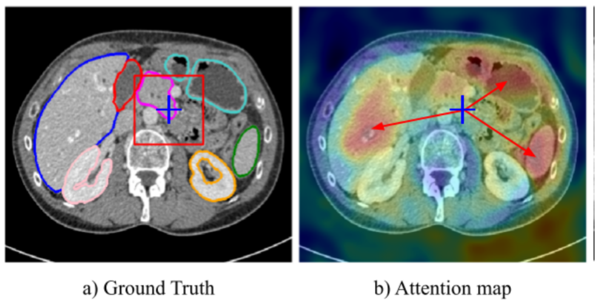

Figure 22: On the left : segmentation of the Thorax. On the right : example of attention map for the pixel under the blue cross, the more red the more intense the attention. Source : Olivier Petit et al. [41], licensed under CC-BY

This map shows the regions that are important for segmenting the pixel under the blue cross. We can see that the attention map leverages the long range interactions of the image and allows the model to "check" if the pixel is in each plausible segment.

Attention mechanisms have also shown nice results for natural and medical images super resolution [11],[58],[15].

Building an attention map

An attention matrix is usually built using the following mathematical formula :

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where Q, K, V are, respectively, the query, the key and the value matrices, while d_k is the integer such that $K \in \mathbb{R}^{n \times d_k}$. These matrices change according to the use case of the attention layer.

Actually, it is common to calculate more than one attention matrix and concatenate them, to leverage more information. This is done using multi head attention :

$$\begin{aligned} \text{Multihead}(Q, K, V) &= \text{Concat}(head_1, ..., head_n)W^0 \\ &\quad \text{where } head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \end{aligned}$$

The matrices $W_i^{\{Q,K,V\}}$ and W^0 are weight matrices that are trained with the rest of the model.

Given an input $X \in \mathbb{R}^{c \times h \times w}$, when performing self attention we have : $Q = K = V = \phi(X) + PE$, where $\phi : \mathbb{R}^{c \times h \times w} \mapsto \mathbb{R}^{c \times hw}$ is the "flatten along the last dimensions" operation and PE is the positional encoding. The reason why we add this matrix is to take into account the position of the words/pixels/voxels in the sentence/image. For cross attention, Q, K are changed to be the vectors containing the context.

Positional encoding

The positional encoding we use comes from "Attention is all you need" [55], where it is used on sentences. Since the order of the words matters but it is not taken into account in the attention mechanisms, we have to inject some information about it, in order for the models to learn how to take it into account.

In order to do it, they consider a matrix representing a sentence $s \in \mathbb{R}^{c \times d}$, where c represents the number of channels and d represents the number of words (for images d would be the length of the flattened image). They add it to $PE \in \mathbb{R}^{c \times d}$, the matrix of the cosinusoidal and sinusoidal positioning :

$$\begin{aligned} PE(2i, pos) &= \sin\left(pos/10000^{2i/c}\right) \\ PE(2i+1, pos) &= \cos\left(pos/10000^{2i/c}\right) \end{aligned}$$

This encoding has two main advantages, the first one is that it can adapt to any sentence/image size, even ones that are longer that than any seen during training. The second is that for any fixed offset k, $PE(\cdot, pos + k)$ is a linear function of $PE(\cdot, pos)$. For our task, the first point is not interesting.

Remark :
One issue with attention mechanisms is the amount of memory that they require, they thus cannot be applied in the highest stages of a U-net.

2.1.5 Attention U-net

There are many ways to implement cross and self attention layers in a U-net. Olivier Petit et al. [41] decided to modify a U-net to add a self attention layer in the middle block and cross attention instead of the concatenations as skip connections. In the code we used, created by Jonathan Ho [24], a self attention mechanism can also be included at the middle of the bottleneck and cross attention mechanisms can be included after each residual block of the desired stages.

In this code, the residual block from figure 20 is modified to include a positional encoding, see figure 23.

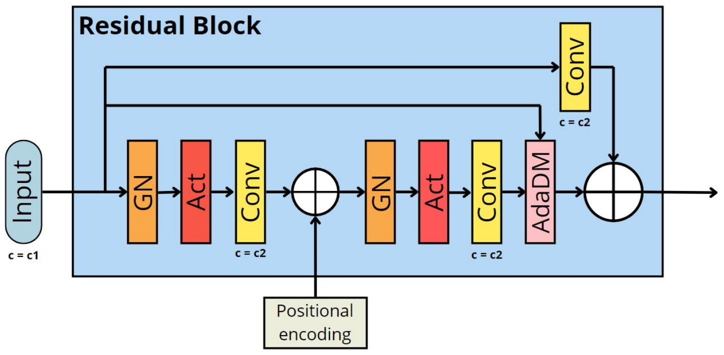

Figure 23: The residual block with positional encoding.

The code also provides three attention blocks, a very simple one, shown in figure 24, that we did not use and two big residual attention blocks, one self and one cross, respectively shown in figures 26 and 25.

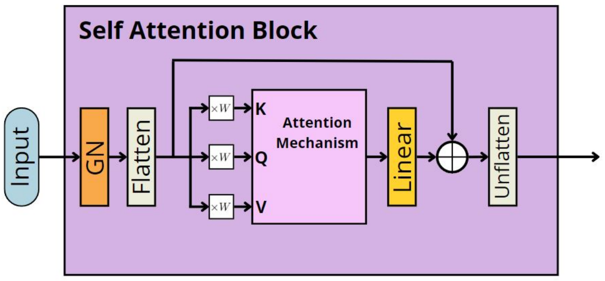

Figure 24: The simple attention block in Jonathan Ho's code.

In this self attention block, a group normalization is added to make the distribution more convenient. There is also a linear transformation and a residual addition at the end. We did not use this simple block, but used a more complex version also provided in the code, see figure 25 for the self attention block and figure 26 for the cross attention block. In our model, the cross attention is done between an input and the context, which is a vector containing information about the sex, age and diagnosis of the patient.

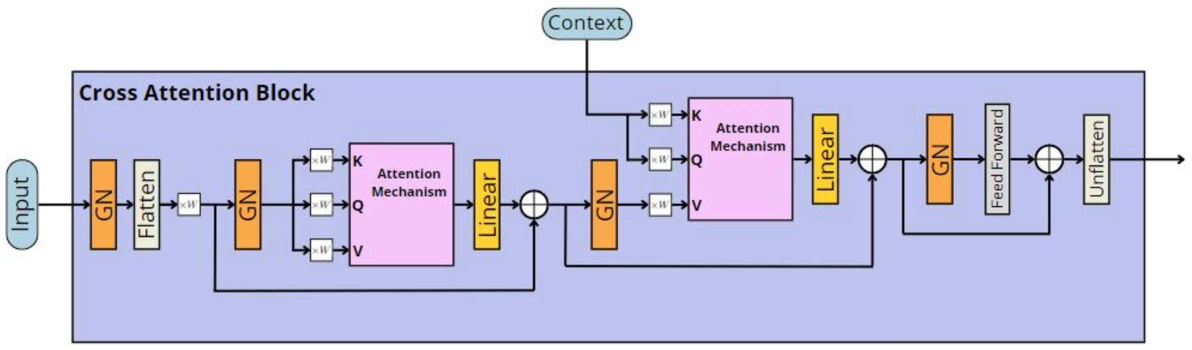

Figure 25: Cross attention block used in Jonathan Ho's code [24], that calculates an attention map between an input and a context.

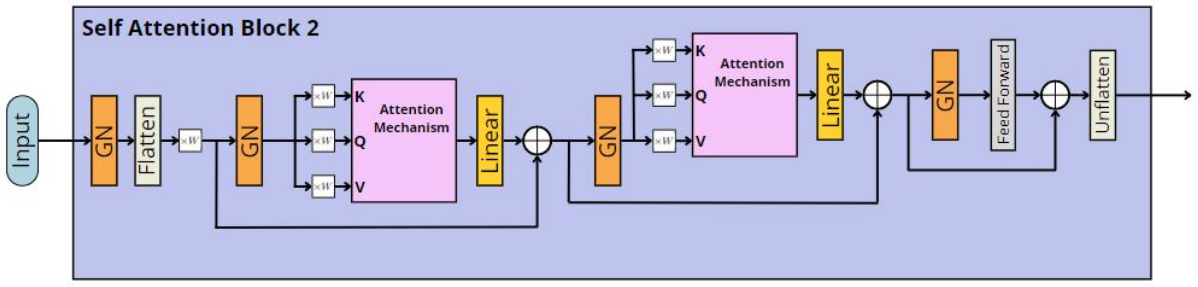

Figure 26: Second self attention block used in Jonathan Ho's code [24]

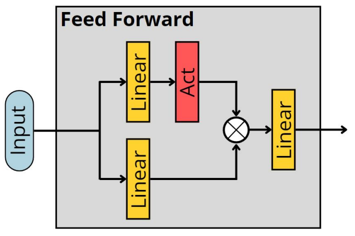

Figure 27: Feed forward block used in self and cross attention mechanisms of Jonathan Ho's code [24], shown in figure 26 and 25.

These blocks are composed of two consecutive attention blocks almost similar to the one shown in figure 24, although the second attention mechanism can be a cross attention. Then there is also a feed forward at the end, shown in figure 27.

These layers can then be added to the model. In our code, we indicate at which stages we put a cross attention (CA) layer, which are then put after each residual (res) block of the stage. There is also a self attention block (SA) between the two residual blocks of the bottleneck. The resulting U-net is shown in figure 28.

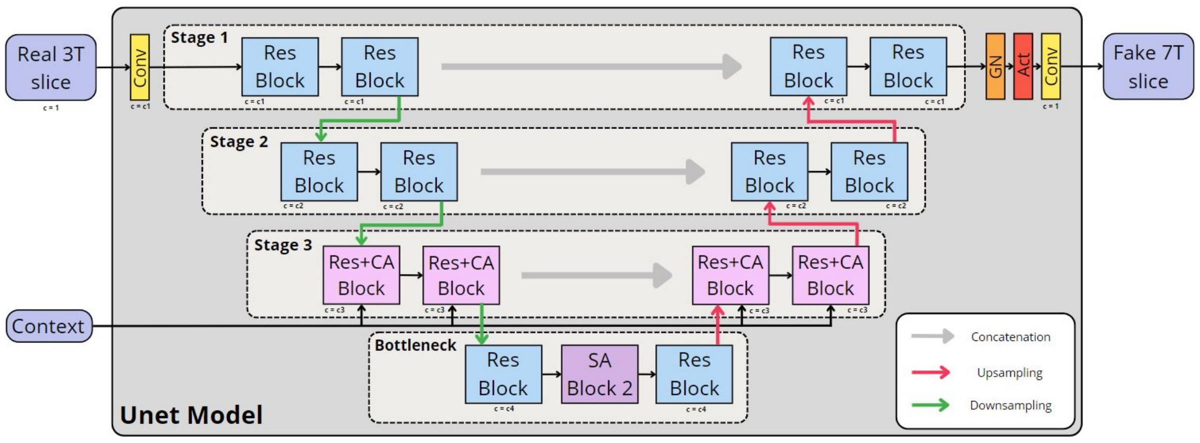

Figure 28: Attention U-net with cross attention layers at the third stage.

2.1.6 Losses used

We note I, I' two images of the same size and $\mathcal{V}$ the set of all the indexes. Most image deep learning models use a L2 loss :

$$\mathcal{L}_2(I, I') = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} (I(v) - I'(v))^2.$$

For most tasks this is very interesting as big errors will be severely punished. Yet, this loss performs badly for super resolution models as it tends to create very smooth results and a L1 loss is thus preferred :

$$\mathcal{L}_1(I, I') = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} |I(v) - I'(v)|.$$

It is also advised to use a perceptual loss [60] which are used because Lp losses are based on pixel to pixel differences, which is limited for many computer vision tasks, as our perception takes into account the quality of the image on multiple scales, from the small details to the more global appearance. This is especially important for super resolution as one of the main points is to make the images look better. The idea is to compare features on different scales of the generated image and the ground truth. This is usually done using a CNN, pretrained for other tasks like segmentation. Let L be the set of all the layers of the perceptual model and n_{c_i} be the number of channels in the i^{th} channel. Let $\phi_l(I)$ be the features of I generated by the layer l of the pretrained CNN :

$$\mathcal{L}_{perc}(I, I') = \frac{1}{|\mathcal{V}| \times |L|} \sum_{v \in \mathcal{V}, l \in L} \frac{1}{n_{c_i}} \sum_{c=1}^{n_{c_i}} (\phi_l(I)(c, v) - \phi_l(I')(c, v))^2.$$

We then add a L1 loss and a perceptual loss weighted by a parameter λ_{perc}, which represents how important the perceptual loss should be compared to the L1 loss. We train our model using the resulting loss :

$$\mathcal{L} = \mathcal{L}_1 + \lambda_{perc} \mathcal{L}_{perc}.$$

Other brain MRI super resolution models have been trained using such losses, [42] and [56] mention using it for MRI brain generation and super resolution respectively. They do not mention which pretrained model they used, but we know they used the python library LPIPS [60]. This library allows us to use a few perceptual losses, including three meant for medical images.

They are based on the backbone of segmentation CNNs. Two of them "medicalnet_resnet50_23datasets" and "medicalnet_resnet50_23datasets" were trained on 3D MRI scans of different parts of the body, including the brain [12]. The difference is the number of residual layers used, respectively 10 and 50. The third is "radimagenet_resnet50" ("radimagnet" for short), it has been trained on 2D slices of different parts of the body and scanner modalities, including brain MRI [38].

2.2 Generative adversarial networks

To make the learning even more efficient, we can add a loss not pretrained but trained at the same time as our model, using a second CNN model that learns to differentiate real images from generated ones. This process is called Generative adversarial networks (GANs). In this context, the U-net is called the generator while the trained loss is the discriminator.

GANs have been used for many tasks [28], [49] including super resolution of natural images [33], [8], [57], [21] and medical images [4], [1], [51].

The generator learns the distribution of the data, while the discriminator estimates the probability that an image is fake. They are trained at the same time, the discriminator receives a fake and a real image and is trained according to how well it distinguished the real one from the fake. In the mean time, the generator is trained according to how well it did at generating the image (quantified by the L1 and perceptual losses) and at fooling the discriminator (quantified by the probability that the generated image is fake), see figure 29. At the end the discriminator has learned to distinguished a real and generated image and the generator has learned how to fool it. Actually, to add more complexity and to leverage more information, we can use a patch GAN [34]. A patch GAN outputs the probability of an image patch to be fake. The output of the discriminator is thus a matrix with the probability that each patch is fake. The probabilities are then averaged to calculate the loss. Figure 30 shows the patch discriminator we used.

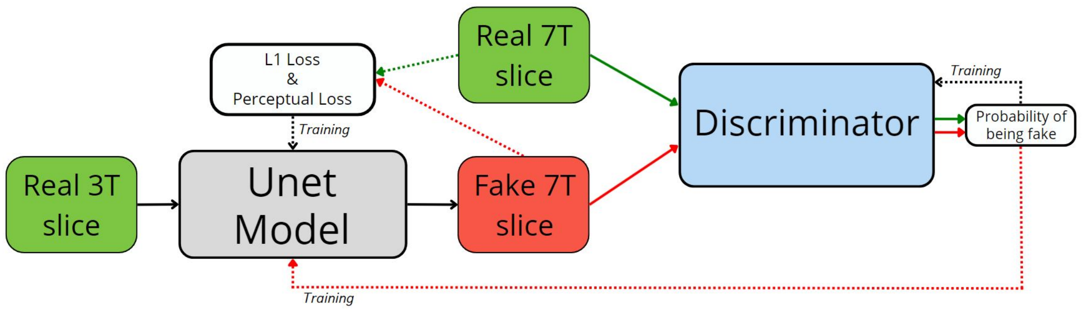

Figure 29: Drawing of our GAN and the losses used. The red arrows are associated with the fake 7T slice, the green arrows with the real 7T slice. The dashed arrows show the information transmission for the backward propagation.

The loss we use to train the discriminator is then, with D the discriminator that outputs the mean probability (over all the patches) of a slice to be true, y the real slice and $\tilde{y}$ the generated slice :

$$\mathcal{L}_{GAN}(y, \tilde{y}) = \log(D(y)) + \log(1 - D(\tilde{y})),$$

and the loss used to train the generator is, with λ_{GAN} the weight of the GAN loss :

$$\mathcal{L}(y, \tilde{y}) = \mathcal{L}_1(y, \tilde{y}) + \lambda_{perc}\mathcal{L}_{perc}(y, \tilde{y}) + \lambda_{GAN} \log(D(\tilde{y})).$$

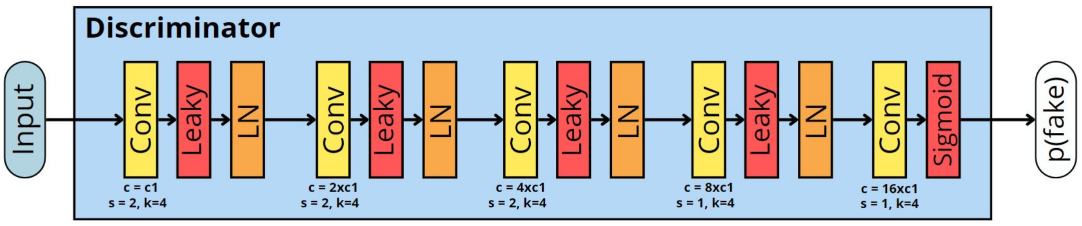

Figure 30: Drawing of our discriminator, k is the kernel size and s is the stride, s=2 implies that the image is downsampled by a factor two. The leakyReLU has a parameter of 0.2. LN stands for layer normalization. c1 is the initial number of channels. The number of patches is $H/2^n \times W/2^n$, where n is the number of convolutions minus one, H is the height of the input and W its width.

Although this type of models have shown very impressive results, they are notorious for being hard to train, due to the fact that we train two different agents at the same time. The most common encountered issues are :

- **Mode collapse :** this happens when the generator learns to output a very efficient solution and thus fails to represent the entire data distribution. For super resolution, the generator can always output the same kind of images regardless of the input. This can be solved using a Wasserstein GAN with Gradient Penalty (WGAN GP) [21], which penalizes on the norm gradient of the discriminator with regards to the input.
- **Diminishing gradient :** the discriminator becomes too efficient so the gradient of the generator vanishes and thus stops learning.
- **Oscillations :** the parameters can oscillate and never converge.
- **Wrong balance between the models :** if one of the models is trained too fast, it can overfit with regards to the other one. This can be fixed by training the discriminator multiple times in a row.

3 Implementation

Now that the dataset and the model are ready, we have to consider how we implement them to actually perform the super resolution. To do so, we have to look at how to feed the data to the model, how we assess the quality of the model and what hyperparameters to choose.

3.1 Handling the data

Even after the processing steps described in section 1, there are a few issues we have to solve before feeding the data to our algorithm.

- Should the image be skull stripped?
- What should the range of the image intensities be?
- How to have a reasonable memory usage?
- How to handle to aforementioned corruptions (section 1.2.2)?
- On what external variables do we condition the model?

3.1.1 Skull stripping

Initially, we thought about using the model on the entire image to reduce the number of preprocessing steps needed during inference. Yet the skull and neck parts of the scans are low quality and will thus be hard to predict. Since they are not so interesting anyway, we decided to work on the skull stripped images. An example of the bad results on the skull obtained by a model can be found in section 4.2.

3.1.2 Normalizing

It is generally advised to normalize the input data, using either min-max normalization or mean-std normalization. This is especially useful in our case, as the perceptual loss we use (radimagenet from lpips [38]) has been trained on data between 0 and 1 and because the range of the data varies a lot after the bias field correction, which can be corrected using a clipped min-max normalization (see subsection 1.3.1, figure 13).

3.1.3 Reducing the memory usage

Large CNN models take a lot of RAM and even good GPUs cannot handle training a model on our full 3D images. Therefore, we decided to use a 2D CNN by working on the 2D slices of the image. This has a major issue, as we lose the information contained by the neighboring slices. This issue can be reduced by inputting more than one slice at a time, by including neighboring slices.

Moreover, the skull stripped brains are surrounded by a lot of background and we can crop the images. Initially, our images were of size (352,352,272) and we removed all the slices along the three dimensions, in which for every brain, no voxel was different from zero. As a result, we had images of size (288,244,222). This cropping transform can easily be reversed using a padding.

3.1.4 Image corruptions

Since we are working on 2D slices instead of the entire images, we can handle the cerebellum issue fairly easily, by slicing from bottom to top (along the transversal dimension, the second dimension) to then remove the slices at the bottom until we are above the corruption. About 44 scans had to get some part of their cerebellum removed, which is around 1/4 of the dataset. This is a lot, but not so terrible either. We also removed the remaining empty slices along this dimension for every patient, to make the training faster. The slices are then of size (288,222), although we padded them to a size of (288,224) to make them divisible by 2^5 and thus compatible with at most five downsamplings.

3.1.5 External variables

Along with the images, BioFINDER provided a file with information about the patients and the scan dates. As a result, I could get their age during the scan, their sex and diagnosis. As said before, we group the patients by ill (demented or MCI) or healthy (cognitively normal or SCD). We also conditioned by the slice location (relatively to the middle slice), in order for the model to know approximately where the slice is located. Jueqi et al. [56] mention conditioning on the age, sex, brain volume and ventricular volume, although they must do it as they use a pretrained diffusion model built to generate fake brain according to these variables.

After all these last steps, the data has been saved using numpy.save, to not perform these steps every time. Since the matrices that contain all the data are huge, we decided to save them in float16 data type. The associated errors are negligible, as the raw images only have a few hundreds to a few thousands of unique values. Pytorch is not really compatible with float16 tensors, we thus convert the images to float32 after loading them.

3.2 Model implementation

The U-net model we used comes from Walter Hugo Lopez Pinaya website [43], in which he adapted Jonathan Ho's code [24] but in 3D instead of 2D. Since we wanted to use a 3D model at first, we started using and modifying this code and then I adapted it to 2D. It contains the codes needed for a latent diffusion model, including the code for a U-net and the code attention layers, which we took. We modified it to make a 2D version of it and added the adaDM layer [37]. To implement the WGAN-GP, we used the code of Erik Linder-Norén [36]. All these models are based on the python package MONAI [10], which includes the code to use the perceptual losses of LPIPS [60].

3.3 Testing the quality of the results

One of the issues with super resolution is that it can be hard to assess how good the model performed and to compare it with other models. To do so, there exists three main solutions :

- **Perceptual metrics :** a simple way to compare our results to the ground truth is to do mathematical assessments. The most popular metrics for super resolution are SSIM and PSNR. Let I be the ground truth 3D image, I' be the prediction and $\mathcal{V}$ the set of the image indexes. We define :

$$R(I) = \max(I) - \min(I), \text{ the intensity range of I,}$$

$$MSE(I, I') = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} (I(v) - I'(v))^2, \text{ the mean square error,}$$

$$PSNR(I, I') = 10 \log_{10} \left(\frac{R(I)^2}{MSE(I, I')} \right).$$

We also define :

$$k_1 = 0.01, k_2 = 0.03, c_1 = (k_1 R(I))^2, c_2 = (k_2 R(I))^2,$$

and :

$$SSIM(I,I') = \frac{(2\mathbb{E}(I)\mathbb{E}(I') + c_1)(2Cov(I,I') + c_2)}{(\mathbb{E}(I)^2 + \mathbb{E}(I')^2 + c_1)(\mathbb{V}(I) + \mathbb{V}(I') + c_2)}.$$

The PSNR metric is the log of the L2 error of the min-max normalized intensities. This is flawed as some voxels (representing blood vessels) are very intense, making the range very fluctuating and not so relevant. We thus decided to replace the maximum by the maximum of the clipped images in the formula for R. The SSIM compares the first and second moments of the prediction and the ground truth. The variables c_1, c_2 are used to stabilize the results and not divide by a value close or equal to 0.

A major flaw of these metrics is that they depend on the amount of background, which is very easy to predict. We decided to calculate it on both the full image, the image without the background and the image without the background with no corruption.

- **Visual assessment :** Since the goal of super resolution is to make images look better, a great way to assess such a model is to look at it ourselves or to ask people their opinion. We could ask physicians to look at ground truth 7T scans and scans generated by different models and to tell them apart. The issues with asking people is the time it takes and the fact that physicians are not used to looking at 7T images.

- **Performing another automatic task :** thanks to the better tissue contrast and the higher number of details, 7T scans should perform better than 3T scans. Qiming Cui et al. [14] mention using SynthSeg [7], a deep learning segmentation tool, to compare the results given by the 7T image and the associated generated 7T image. We define V the volume of a set and $s(I)$ the set that represents a specific brain segment of the image I. We can then compare the results of a segmentation on both images with the DICE metric for the segment considered :

$$\text{DICE}(I,I') = \frac{2V(s(I) \cap s(I'))}{V(s(I)) + V(s(I'))}$$

  This is done on all of the 31 segments generated by SynthSeg (except the background), which generates a brain image composed of integers indicating to which segment each voxel belongs to. We removed 4 of the segments as they belong to the cerebellum, which is often corrupted.

Remark :
The three aforementioned metrics should be as high as possible. $SSIM$ and $DICE$ are between 0 and 1 while $PSNR$ is between 0 and $+\infty$.

3.4 Model hyperparameters

3.4.1 U-net hyperparameters

Here are the hyperparameters of the U-net :

- **Initial number of channels $n_{channels}$:** we need to indicate the number of channels in the model after the first convolution. The number of channels at each convolution is a multiple of this parameter. It is a positive integer, usually a power of 2.

- **Number of groups n_{groups} :** group normalization requires a parameter that tells how many groups should the layers be divided in during the normalization. It is a positive integer that divides the initial number of channels. If it equal to 1, then this is equivalent to layer normalization.

- **Channel multiplication :** at the i^{th} stage of the U-net, there are $n_{channels} * n_i, n_i \geq n_{i-1} \in \mathbb{N}$ channels. Thus, we have a parameter $(n_1, n_2, ..n_k)$ that tells how the number of channels evolves, where k is the number of stages (it is also the number of downsamplings). For super resolution, it is usually (1,2,2,2) or (1,2,2,2,2), as the first stages are the most important.
- **Number of res blocks n_{res}:** number of res blocks put in a row. It is a small integer, usually around 3.
- **Dropout parameter:** a dropout an operation that deactivates a certain percentage of neurons in the model during training, that helps reducing overfitting. The parameter is the percentage of neurons deactivated and is between 0 and 1. Yet, for low-level tasks such as super resolution, this is said to have a bad effect [31], so we set it to 0.
- **Optimizer parameters :** our optimizer, "adam" has 3 main parameters : the learning rate (lr), a small positive real number which controls the pace at which the weights are updated, β_1 and β_2 which respectively control the weights of the past first and second moments of the gradient in the adam optimizer [30]. They are between 0 and 1 and for a super resolution CNN, it is common to take $\beta_1 = 0.9$ and $\beta_2 = 0.999$, which we used. The learning rate is very important, if it is too high the model will diverge and if it is too low, it will converge too slowly. After a few steps or epochs, the learning rate will be too high and prevent the model from learning more, this is solved by reducing the learning rate after a while. We did this either by multiplying the learning rate by a real number smaller than one (like 0.8) every epoch or every so many steps (like 200*) ("decay" schedule), or by using the function "ReduceLROnPlateau" of Pytorch ("reducePlateau" schedule), which divides the learning rate by 10 if the loss did not decrease after so many steps, usually 5 steps.
- **Number of attention heads :** in each attention mechanism, it is possible to calculate multiple attention maps and concatenate them. However, attention mechanisms are very costly in RAM, so we set it to 1.
- **Cross attention locations :** in the model, we have to indicate at which resolution the cross attention between some feature layer and external variables happens. Since it takes a lot of RAM, we do not use it before the third stage.
- **Number of input slices $n_{inputslices}$:** this parameter tells the number of slices inputted in the model, the middle slice being the one associated to the output slice. More slices means more information, but the information of slices distant to the middle slice is less relevant. We usually took 3 or 5.
- **Perceptual loss weight :** this is the parameter in front of the perceptual loss, used to tell how important it is compared to the L1 loss. If it is too high, the L1 loss will not decrease much and if it is too low, the perceptual loss will not decrease enough either. Usually set to 10^{-3}, we increased it to 5×10^{-3} as the perceptual loss was not decreasing so much.

*the number of steps depends on the batch size but at every epoch there are around 26,000 training slices in the BioFINDER dataset and 2000 for the UNC dataset. For the UNC dataset, there are not so many steps, so we updated the learning rate after every epoch. For the BioFINDER dataset and a batch size of 16, which I used a lot, I used they decay every 200 or 300 steps.

3.4.2 Hyperparameters of the GAN

There are other parameters specific to the GAN :

- **Discriminator training ratio n_{critic}:** this is a positive integer that tells how many times the discriminator should be trained every time the generator is trained. It is common to set it to 5 to have an efficient discriminator.
- **Number of layers n_{layers}:** the number of layer convolution+activation+normalization. It is also the number of downsamplings, which also sets the size of the patches as the output is of size $\frac{H}{2^{n_{layers}}} \times \frac{W}{2^{n_{layers}}}$ with H the height and W the width of the images.
- **Initial number of channels $n_{channels,GAN}$:** same parameter as in the generator. The number of channels is multiplied by two at each convolutional layer.
- **Discriminator optimizer parameters :** we also use a adam optimizer. Following the results of the experiments of [5], we chose $\beta_{GAN} = (0, 0.9)$. We used a learning rate of 10^{-4}, which we chose based on a few tests.
- **GAN loss weight :** the weight of the discriminator loss when training the generator. Usually around 10^{-2}.
- **Gradient penalty weight :** the weight of the gradient penalty. The common value is 10, which we decided to use.

3.4.3 Choosing the hyperparameters :

To chose the parameters, besides taking into account advice found online and in literature, we can do a grid search or a Bayesian optimization. We wanted to do a multi objective Bayesian optimization to optimize around 5 parameters but the biggest models were extremely long to train and we did not have enough compute resource allocation to perform this optimization anyway.

4 Experiments

4.1 Resources

To run experiments, we had access to three compute sources :

- A computer at LTH with 4 Nvidia Titan X 12GB GPUs.
- A secured cluster : Bianca. It has 10 nodes with 2 Nvidia A100 40GB GPUs each. All the data is stored on it and must be anaonymized to be extracted.
- A non-secured cluster : Berzelius. It has 34 nodes each with 8 Nvidia A100 80GB GPUs. I had a limited amount of compute hours on this cluster.

I have a remote access to these resources through secured shell (ssh) protocol and I could transfer data using secure file transfer protocol (sftp). These computers run on linux, which I had to learn to use. Running a job on the two clusters is done using Simple Linux Utility for Resource Management (SLURM), which is an open-source cluster management and job scheduling system. To use SLURM, I also had to learn how to use shell scripts.

For security reasons, Bianca does not have access to the internet and there are restrictions on the administrative rights. This made setting up the preprocessing tools and python environments much more difficult, especially as I was the first person of the lab to run GPU jobs.

4.2 First experiments

Our first experiment has been done by training the U-net model on the unprocessed UNC dataset using the LTH computer. We did it to validate the code and to help me understand the data as well as the model better. It is based on these results that we chose to implement many ideas such as the bias field correction, the skull stripping and the clipped min-max normalization. I used the 10 pairs from the UNC dataset, I kept 2 for later validation, 7 for training and 1 for testing. The results were visually good and the results for the test subject are shown in figure 31.

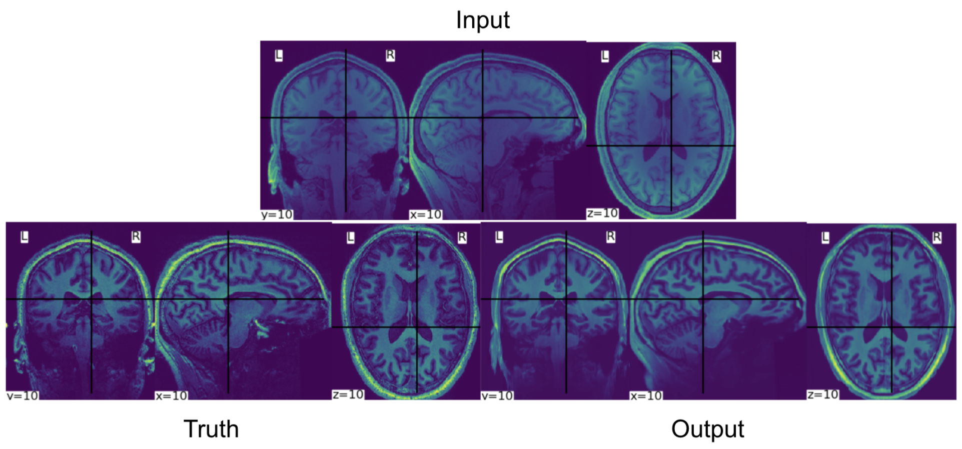

Figure 31: Three slices, one in each dimension, of the test patient, on 3T and 7T resolution with the generated 7T resolution.

We noticed a few things :

- **The model requires a lot of RAM :** training a U-net on 3D images with 26 million voxels (size (256,320,320)) is not feasible even with huge GPUs. This convinced us to train the model on 2D slices of size (320,320). Even after that, we had to use a low batch size and a small model to not go over 12GB of RAM. The RAM is better on the clusters.

- **Poor results on the skull :** as one can see on figure 32, the model does a poor job at predicting the skull. This is not surprising as during the scan acquisition, the machine spends much more time on the brain than on the skull, as it matters much less. As a result, the skull part is very noisy and of a very low quality.

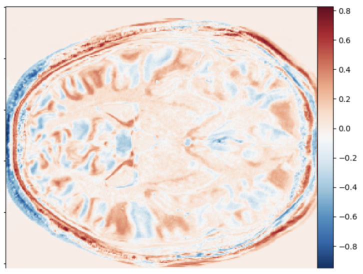

Figure 32: Difference map $I - I'$, where I is a ground truth slice and I' is the associated generated slice of the test subject. Notice that the absolute difference is very high on most of the skull.

- **The bias field is a big problem :** The location and intensity of the bias field are a bit random and it does not depend on the patient's brain, the model cannot predict it. The model thus attenuates it by averaging it, which is good for us but the loss functions punish the model for it, which makes the training harder. The attenuation can be seen in figure 32, where the bias field is less important on the output than on the ground truth. The bias fields on the 3T scans are not as terrible because the model can learn how to remove it, but it is better if the model does not have to.
- **The model rapidly overfits :** the test loss rapidly converges, while the training loss keeps going down. Training the model on the BioFINDER dataset will solve this issue.
- **Normalization :** we realized that it was better to clip the first and last percentiles when doing the min-max normalization. We did not use the mean-std normalization as it was not compatible with the perceptual loss, trained with inputs between 0 and 1.

4.3 Experiments on Bianca and Berzelius

Actually, we did not have access to Berzelius until the end of the project. So we started all of our tests on Bianca.

We worked on the processed data of both datasets, using the U-net model. We built two models, one with the UNC dataset and one with the BioFINDER dataset. We did not mix them as they were not acquired by the same 7T scanner, which creates unpredictable differences.

On Berzelius, we have very big GPUs but we had a small allocation. We worked on the processed anonymised data to build a GAN. Since we did not have much allocation, we could not do so many tests and we believe that we could achieve better results.

4.4 Results of the experiments

We present the results for three models, the U-nets trained on Bianca on both datasets and the GAN trained on Berzelius on the BioFINDER (BF) dataset. The parameters used are shown in the appendix.

We measured the efficiency of our model with 3 metrics : DICE, SSIM and PSNR. We applied the DICE on the whole brain, removing the background, CSF and four segments belonging to the cerebellum to avoid the corruptions. This results in 27 segment. We applied the two other metrics on : the entire image and only the brain with and without the corruptions. Since the corruption part does not apply to the UNC dataset, removing the corruption changes nothing. The amount of background on both dataset is different, so it is not relevant to compare the metrics with background.

These metrics have been used on the two validation subjects of the UNC dataset and the 29 validation subjects of the BioFINDER dataset. We display the averages in table 1. It also includes the results of the comparison between the normalized 3T and 7T images, to give a baseline.

Table 1: Table of average results for the three models and the seven metrics.

Model	PSNR ↑	SSIM ↑	brain PSNR ↑	brain SSIM ↑	brain PSNR ↑ no corrupt.	brain SSIM ↑ no corrupt.	Mean DICE ↑
BF 3T	25.4	0.955	16.8	0.554	16.8	0.559	0.879
U-net BF	26.2	0.966	17.3	0.559	17.4	0.564	0.874
GAN BF	26.1	0.957	17.1	0.558	17.2	0.564	0.882
UNC 3T	16.6	0.870	9.58	0.384	9.58	0.384	0.859
U-net UNC	24.6	0.936	17.5	0.583	17.5	0.583	0.878

We also show the boxplot of these metrics in figures 33,34 and 35. Note that we only have two patients in the UNC validation dataset, so two PSNR and SSIM values each time. There are 31 DICE values per patient and two validation patients on the UNC dataset (we did not remove the cerebellum). We removed the UNC 3T to 7T from the PSNR and SSIM results boxplots, as they were too different.

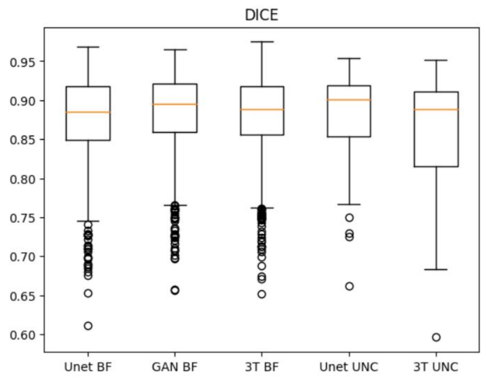

Figure 33: Boxplot of the DICE results for different models and datasets.

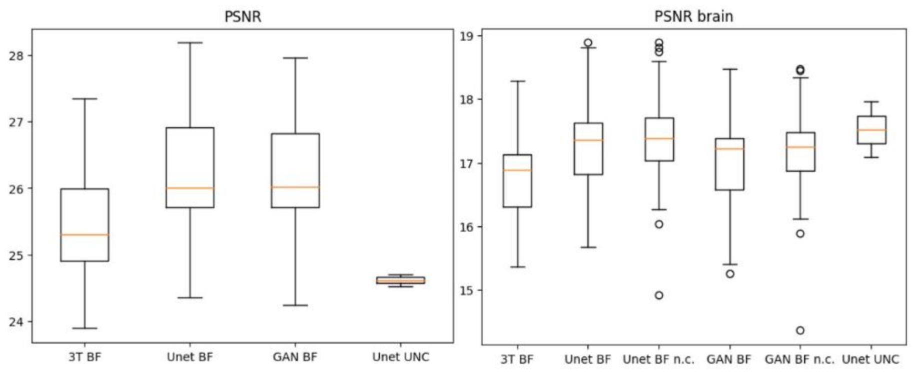

Figure 34: Boxplot of the PSNR results for different models and datasets.

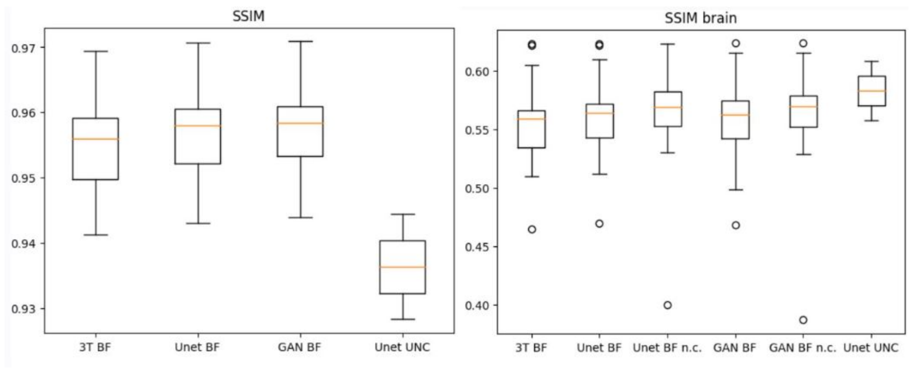

Figure 35: Boxplot of the SSIM results for different models and datasets.

The results are about the same between the U-net and GAN on the BioFINDER dataset. A few results are very poor (lower dots in “SSIM brain” and “PSNR brain”), this is probably due to the fact that the 7T acquisition of the associated brain is bad. The results on the UNC dataset are lower on the default PSNR and SSIM because there is less background, but it becomes better when the background is removed. Our hypothesis is that although the BioFINDER dataset is bigger, it is of a lower quality than the UNC dataset as the scan acquisition is less advanced and it has more diversity as it includes older and unhealthy brains. It is difficult to interpret these values, but we can see that they are slightly better than the 3T baseline for the BioFINDER data and a lot better for the UNC dataset. The DICE scores seem to be good, as they compare to Qiming Cui et al. ’s results [14].

Remark :
The calculations of the PSNR and SSIM depend on the range of intensities of the ground truth image, which we changed to the range of the clipped intensity of the image, as the range was is very unstable. This choice reduces the values of our metrics, but makes them more reliable.

We also display a visualisation of the results for the BioFINDER dataset (figures 36 and 37) and the UNC dataset (figure 38).

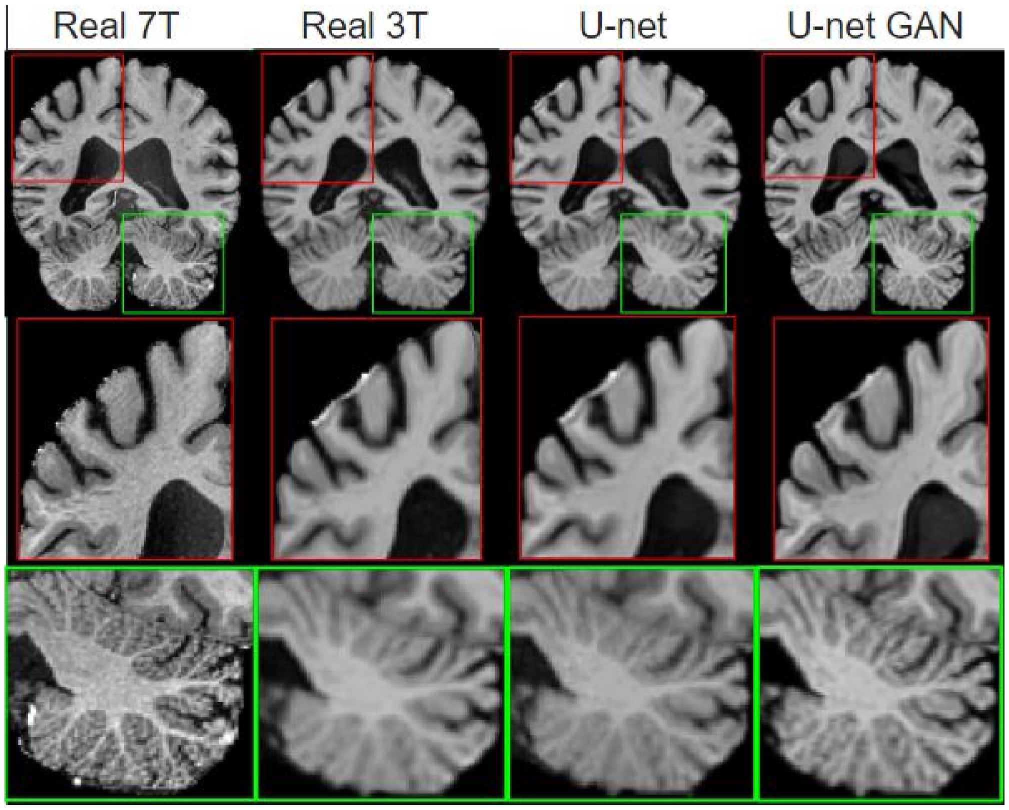

Figure 36: Example of results of an unhealthy patient of the BioFINDER dataset, with a close-up on the cortex and the cerebellum.

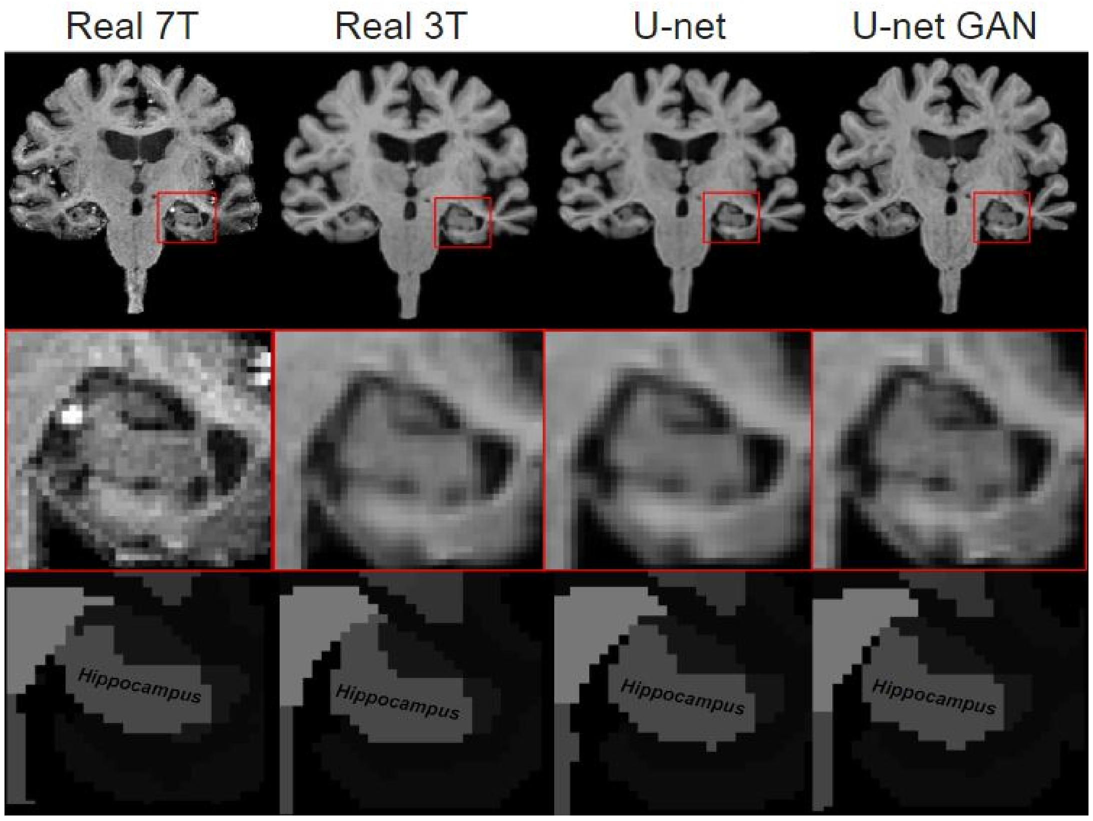

Figure 37: Example of results of an unhealthy patient of the BioFINDER dataset, with a close-up on a small brain part : the hippocampus. The bottom line also includes the segmentation of the middle line.

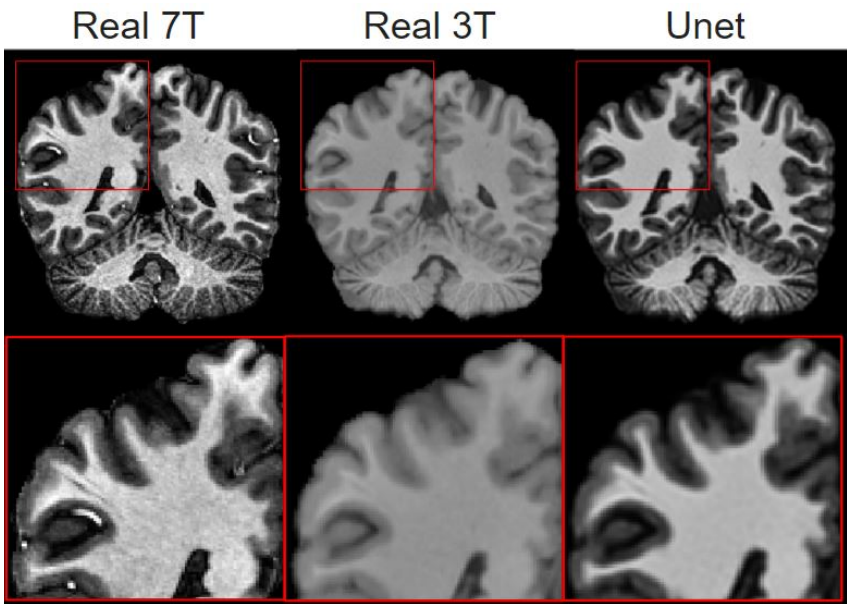

Figure 38: Example of results of the UNC dataset.

We see that the U-net on the BioFINDER dataset outputs an image very similar to the input. The GAN outputs a result that is less blurry than the 3T, which is promising. These results are not very good.

The results on the UNC dataset are visually better. The output is less noisy than the ground truth, which was expected and is good. The results are a bit more blurry than the ground truth, but much better than the 3T input. We can also see that the 3T scan looks more similar to the 7T in the BioFINDER example (figure 36) than in the UNC example (figure 38). This dataset difference is also what explains the low PSNR and SSIM results in the baseline shown in table 1.

Remark :
One of the reasons the results for the BioFINDER are not as appealing as those for the UNC dataset might be due to the fact that the images are not as consistent. This can be seen in figure 37, where the hippocampus looks very noisy in the 7T scan and where some very intense features (probably blood vessels) are only in the 7T scan. The lack of consistency can also be seen in the intensity distributions. Figures 39 and 40 show the intensity distribution of four T1w scans in 3T and 7T resolutions of four patients, respectively from the UNC and BioFINDER datasets.

This inconsistency is probably due to the larger brain diversity and a poor choice of scanner parameters. Indeed, we have been told that a 7T scanner requires an entire team dedicated to its settings to give good results, which is not the case for the scanner we used.

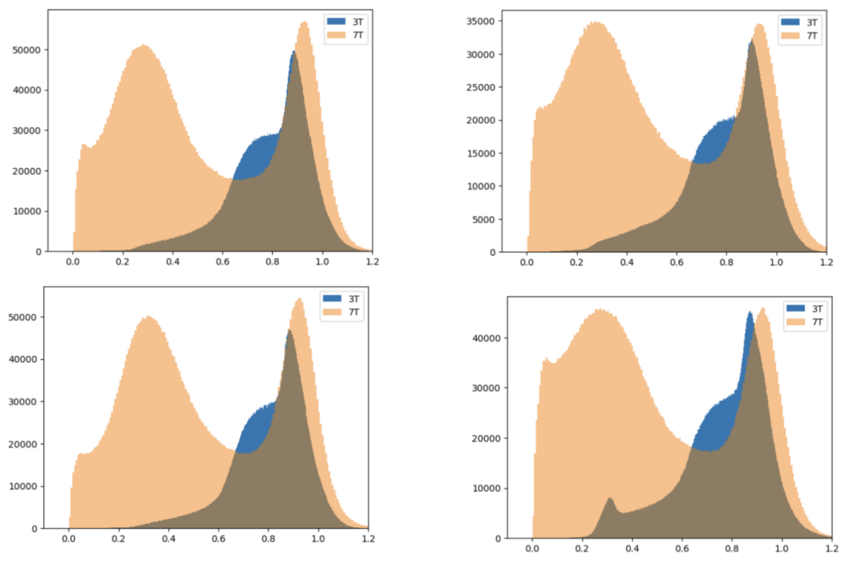

Figure 39: Intensity distributions for both resolutions of four UNC patients

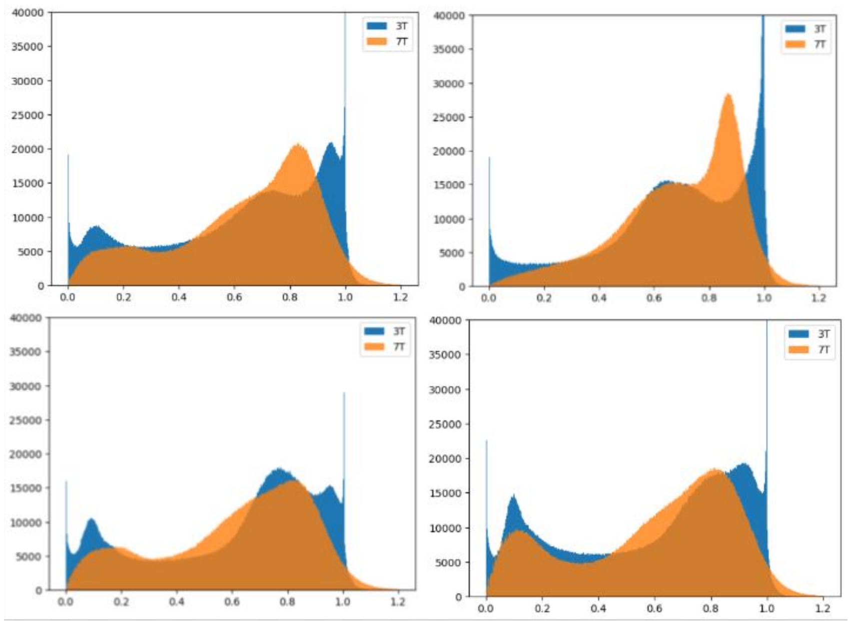

Figure 40: Intensity distributions for both resolutions of four BioFINDER patients.

5 Conclusion

5.1 Discussion

In this thesis, I was able to make a lot of progress and build a good proof of concept. I preprocessed the data, removed the bias field and the skull. I also registered the 3T and 7T scans. The processing is not perfect as the skull stripping still left some artifacts and the data still has flaws (noise, stripes, ...), but it is very good and hard to improve. I was able to implement the model by using and modifying code found online and to gather a lot of information on how to use the model. To do all of this, I had to learn a lot of techniques and to use a lot of softwares (freesurfer, linux, shell script, fsl and SLURM) and python packages (MONAI, ANTs, LPIPS). We also showed how the usual metrics used to assess the quality of a model give limited information on our dataset, because they depend on an unstable intensity range and on the amount of background voxels. The final results are not so good yet, some more work is required to keep improving them, but we managed to create a great proof of concept.

Another step to consider once a model is working and ready is to care about the clinical applications. As of now, medical image super resolution is not much used as the models need to be very accurate and unbiased to not skew clinical decisions [54]. Yet, these models could significantly improve diagnostic accuracy and precision and could very well be widely used in a near future.

5.2 Things to explore

To further improve the model and its performance, there are a few things we can do :

- **Augmentations :** as mentioned earlier, it could be nice to augment the data. We have a lot of scans so most augmentation techniques would not be so useful, but increasing the scanner diversity could be useful. We could also ask other labs for their pairs of scans. For most patients, there are multiple 3T scans for one 7T scan. We could use them, although we would have to care about the time difference between the scans.
- **Scanner harmonization :** Qiming Cui et al. [14] mention training their model on images from two datasets. To do so, they harmonize the datasets using RAVEL [18]. We could do it to use our two datasets on the same model.
- **Diffusion :** as said earlier, we could try building and training a diffusion model. Nevertheless, it does not seem so promising as the amount of studies is limited and they have not proven a better efficiency than GANs for natural images super resolution [32].
- **More complex input :** in our model, we inputted one slice with its neighbors. We could also input the slice with its wavelet representation [45]. It is also possible to use a 3D model that takes blocks instead of slices, like Qiming Cui et al. [14].
- **Improving the discriminator :** the discriminator I used is pretty simple, we could improve it by adding residual blocks for instance.
- **Last normalization layer :** at the end of my experiments, I thought that the last normalization should be remove because of the fact that this layers makes the edges more blurry [37].
- **Histogram equalization :** we showed that the intensity histograms of the BioFINDER dataset are much less consistent than those of the UNC dataset. We could solve it by making the 7T histogram match the one of the associated 3T image. Although this could add new issues.

- **Doing more GAN tests :** due to a limited compute ressource allocation Berzelius, I was not able to run many GAN tests and optimize the parameters.

Other than improving our model, it would have been nice to spend time testing the model more. To do this, we could take someone else model like those of Qiming Cui et al. [14] and Liangqiong Qu et al. [45], train them on our dataset and assess their performance.

References

[1] B. Ankitha, Ch. Srikanth, D. Venkatesh, D. Badrinath, G. Aditya, and S. Akila Agnes. Enhancing the resolution of brain mri images using generative adversarial networks (gans). In *2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*, pages 19–25, 2023.

[2] I. Arvidsson, O. Strandberg, S. Palmqvist, et al. Comparing a pre-defined versus deep learning approach for extracting brain atrophy patterns to predict cognitive decline due to alzheimer's disease in patients with mild cognitive symptoms. *Alzheimer's Research Therapy*, 16(61), 2024.

[3] B.B. Avants, C.L. Epstein, M. Grossman, and J.C. Gee. Symmetric diffeomorphic image registration with cross-correlation: Evaluating automated labeling of elderly and neurodegenerative brain. *Medical Image Analysis*, 12(1):26–41, 2008. Special Issue on The Third International Workshop on Biomedical Image Registration – WBIR 2006.

[4] Ashwin Balasubramanian, Haripriya Dhanasekaran, Booma Raghu, and Kunaraj Kumarasamy. Mri super-resolution using generative adversarial network and discrete wavelet transform. In *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, pages 1314–1318, 2022.

[5] Arkaprabha Basu, Kushal Bose, Sankha Subhra Mullick, Anish Chakrabarty, and Swagatam Das. Fortifying fully convolutional generative adversarial networks for image super-resolution using divergence measures, 2024.

[6] David Berron, Jacob W Vogel, Philip S Insel, Joana B Pereira, Long Xie, Laura E M Wisse, Paul A Yushkevich, Sebastian Palmqvist, Niklas Mattsson-Carlgren, Erik Stomrud, Ruben Smith, Olof Strandberg, and Oskar Hansson. Early stages of tau pathology and its associations with functional connectivity, atrophy and memory. *Brain*, 144(9):2771–2783, 03 2021.

[7] Benjamin Billot, Douglas N. Greve, Oula Puonti, Axel Thielscher, Koen Van Leemput, Bruce Fischl, Adrian V. Dalca, and Juan Eugenio Iglesias. Synthseg: Segmentation of brain mri scans of any contrast and resolution without retraining. *Medical Image Analysis*, 86:102789, 2023.

[8] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2019.

[9] Alexander Buslaev, Vladimir Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11:125, 02 2020.

[10] M. Jorge Cardoso, Wenqi Li, Richard Brown, Nic Ma, Eric Kerfoot, Yiheng Wang, Benjamin Murrey, Andriy Myronenko, Can Zhao, Dong Yang, Vishwesh Nath, Yufan He, Ziyue Xu, Ali Hatamizadeh, Andriy Myronenko, Wentao Zhu, Yun Liu, Mingxin Zheng, Yucheng Tang, Isaac Yang, Michael Zephyr, Behrooz Hashemian, Sachidanand Alle, Mohammad Zalbagi Darestani, Charlie Budd, Marc Modat, Tom Vercauteren, Guotai Wang, Yiwen Li, Yipeng Hu, Yunguan Fu, Benjamin Gorman, Hans Johnson, Brad Genereaux, Barbaros S. Erdal, Vikash Gupta, Andres Diaz-Pinto, Andre Dourson, Lena Maier-Hein, Paul F. Jaeger, Michael Baumgartner, Jayashree Kalpathy-Cramer, Mona Flores, Justin Kirby, Lee A. D. Cooper, Holger R. Roth, Daguang Xu, David Bericat, Ralf Floca, S. Kevin Zhou, Haris Shuaib, Keyvan Farahani, Klaus H. Maier-Hein, Stephen Aylward, Prerna Dogra, Sebastien Ourselin, and Andrew Feng. Monai: An open-source framework for deep learning in healthcare, 2022.

[11] Haoyu Chen, Jinjin Gu, and Zhi Zhang. Attention in attention network for image super-resolution, 2021.

[12] Sihong Chen, Kai Ma, and Yefeng Zheng. Med3d: Transfer learning for 3d medical image analysis, 2019.

[13] Xiaoyang Chen, Liangqiong Qu, Yifang Xie, Sahar Ahmad, and Pew-Thian Yap. A paired dataset of t1- and t2-weighted mri at 3 tesla and 7 tesla. *Scientific Data*, 10, 07 2023.

[14] Qiming Cui, Duygu Tosun, Pratik Mukherjee, and Reza Abbasi-Asl. 7t mri synthesization from 3t acquisitions, 2024.

[15] Weizhi Du and Harvery Tian. Transformer and gan based super-resolution reconstruction network for medical images, 2022.

[16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018.

[17] Zach Eidex, Jing Wang, Mojtaba Safari, Eric Elder, Jacob Wynne, Tonghe Wang, Hui-Kuo Shu, Hui Mao, and Xiaofeng Yang. High-resolution 3t to 7t mri synthesis with a hybrid cnn-transformer model, 2023.

[18] Jean-Philippe Fortin, Elizabeth M. Sweeney, John Muschelli, Ciprian M. Crainiceanu, and Russell T. Shinohara. Removing inter-subject technical variability in magnetic resonance imaging studies. *NeuroImage*, 132:198–212, 2016.

[19] Fabio Garcea, Alessio Serra, Fabrizio Lamberti, and Lia Morra. Data augmentation for medical imaging: A systematic literature review. *Computers in Biology and Medicine*, 152:106391, 2023.

[20] Evgin Goceri. Medical image data augmentation: Techniques, comparisons and interpretations. *Artificial Intelligence Review*, pages 1–45, March 2023.

[21] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[23] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[25] Andrew Hoopes, Jocelyn S. Mora, Adrian V. Dalca, Bruce Fischl, and Malte Hoffmann. Synthstrip: skull-stripping for any brain image. *NeuroImage*, 260:119474, 2022.

[26] Xiaodan Hu, Mohamed A. Naiel, Alexander Wong, Mark Lamm, and Paul Fieguth. Runet: A robust unet architecture for image super-resolution. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 505–507, 2019.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[28] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

[29] Jaber Juntu, Jan Sijbers, Dirk Van Dyck, and Jan Gielen. Bias field correction for mri images. In Marek Kurzyński, Edward Puchała, Michał Woźniak, and Andrzej żołnierek, editors, *Computer Recognition Systems*, pages 543–551, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[31] Xiangtao Kong, Xina Liu, Jinjin Gu, Yu Qiao, and Chao Dong. Reflash dropout in image super-resolution, 2022.

[32] Denis Kuznedelev, Valerii Startsev, Daniil Shlenskii, and Sergey Kastryulin. Does diffusion beat gan in image super resolution?, 2024.

[33] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017.

[34] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks, 2016.

[35] Binhua Liao, Yani Chen, Zhewei Wang, Charles D. Smith, and Jundong Liu. A comparative study on 1.5t-3t mri conversion through deep neural network models, 2022.

[36] Erik Linder-Norén. Code for a wgan-gp.

[37] Jie Liu, Jie Tang, and Gangshan Wu. Adadm: Enabling normalization for image super-resolution, 2021.

[38] Xueyan Mei, Zelong Liu, Philip M. Robson, Brett Marinelli, Mingqian Huang, Amish Doshi, Adam Jacobi, Chendi Cao, Katherine E. Link, Thomas Yang, Ying Wang, Hayit Greenspan, Timothy Deyer, Zahi A. Fayad, and Yang Yang. Radimagenet: An open radiologic deep learning research dataset for effective transfer learning. *Radiology: Artificial Intelligence*, 0(ja):e210315, 0.

[39] Maria Ines Meyer, Ezequiel de la Rosa, Nuno Pedrosa de Barros, Roberto Paolella, Koen Van Leemput, and Diana M. Sima. A contrast augmentation approach to improve multi-scanner generalization in mri. *Frontiers in Neuroscience*, 15, 2021.

[40] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[41] Olivier Petit, Nicolas Thome, Clément Rambour, and Luc Soler. U-net transformer: Self and cross attention for medical image segmentation, 2021.

[42] Walter H. L. Pinaya, Petru-Daniel Tudosiu, Jessica Dafflon, Pedro F da Costa, Virginia Fernandez, Parashkev Nachev, Sebastien Ourselin, and M. Jorge Cardoso. Brain imaging generation with latent diffusion models, 2022.

[43] Walter Hugo Lopez Pinaya. Code for 3d diffusion model, attention modules and residual u-net.

[44] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization, 2020.

[45] Liangqiong Qu, Yongqin Zhang, Shuai Wang, Pew-Thian Yap, and Dinggang Shen. Synthesized 7t mri from 3t mri via deep learning in spatial and wavelet domains. *Medical Image Analysis*, 62:101663, 2020.

[46] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[47] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[48] Mostafa Salem, Sergi Valverde, Mariano Cabezas, Deborah Pareto, Arnau Oliver, Joaquim Salvi, Àlex Rovira, and Xavier Lladó. Multiple sclerosis lesion synthesis in mri using an encoder-decoder u-net. *IEEE Access*, 7:25171–25184, 2019.

[49] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. A u-net based discriminator for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[50] B.R. Steensma, M. Luttje, I.J. Voogt, et al. Comparing signal-to-noise ratio for prostate imaging at 7t and 3t. *Journal of Magnetic Resonance Imaging*, 49(5):1446–1455, 2019.

[51] Li Sun, Junxiang Chen, Yanwu Xu, Mingming Gong, Ke Yu, and Kayhan Batmanghelich. Hierarchical amortized gan for 3d high resolution medical image synthesis. *IEEE Journal of Biomedical and Health Informatics*, 26(8):3966–3975, August 2022.

[52] Nicholas J. Tustison, Brian B. Avants, Philip A. Cook, Yuanjie Zheng, Alexander Egan, Paul A. Yushkevich, and James C. Gee. N4itk: Improved n3 bias correction. *IEEE Transactions on Medical Imaging*, 29(6):1310–1320, 2010.

[53] Nicholas J. Tustison, Philip A. Cook, Alexander J. Holbrook, et al. The antsx ecosystem for quantitative biological and medical imaging. *Scientific Reports*, 11:9068, 2021.

[54] Sabina Umirzakova, Shabir Ahmad, Latif U. Khan, and Taegkeun Whangbo. Medical image super-resolution for smart healthcare applications: A comprehensive survey. *Information Fusion*, 103:102075, 2024.

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[56] Jueqi Wang, Jacob Levman, Walter Hugo Lopez Pinaya, Petru-Daniel Tudosiu, M. Jorge Cardoso, and Razvan Marinescu. Inversesr: 3d brain mri super-resolution using a latent diffusion model, 2023.

[57] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

[58] Yan Wang, Yusen Li, Gang Wang, and Xiaoguang Liu. Multi-scale attention network for single image super-resolution, 2024.

[59] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.

[60] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

Appendix

Parameters of the final models

See section 3.4 for the definition of the parameters and section 4.4 for the analysis of the models.

Parameters	U-net UNC	U-net	GAN
n_{epochs}	15	3	10
$n_{channels}$	128	64	256
channel multiplication	(1,2,2,2)	(1,2,2,2,2)	(1,2,2,2,2)
n_{groups}	32	16	64
n_{res}	3	3	4
CA stages	(3,4,5)	(3,4,5)	None
n_{batch}	16	16	48
$n_{inputslices}$	3	3	3
perc weight	5.10^{-3}	10^{-3}	10^{-2}
lr	5.10^{-5}	10^{-4}	10^{-4}
lr schedule	reducePlateau	decay $\times 0.5/200 steps$*	decay $\times 0.8/epoch$**

Table 2: Parameters used for every U-net model

Parameters	Value
n_{critic}	5
lr	2.10^{-5}
Gan loss weight	0.2
$n_{channels}$	256
n_{layers}	4

Table 3: Parameters used for the discriminator of the GAN.

*learning rate multiplied by 0.5 every 200 steps.

**learning rate multiplied by 0.8 at every epoch.

Master's Theses in Mathematical Sciences 2024:E67
ISSN 1404-6342

LUTFMA-3558-2024

Computer Vision and Machine Learning
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/