University “Politehnica” Bucharest

Faculty of Electronics, Telecommunications and Information Technology

Deep Learning Strategies for Medical Image Reconstruction

Master Thesis

Study program: Multimedia Technologies in Biometrics and Information Security Applications

Author: Rebeca-Grațiela Predescu

Thesis advisors: Prof. Corneliu Burileanu,PhD

Bucharest
2020

Annex 2

University "Politehnica" of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
Master Program "Multimedia Technologies in Biometrics and Information Security Applications"

MASTER THESIS

of student PREDESCU A. Rebeca-Grațiela, 421-BIOSINF

1. Thesis title: Deep Learning Strategies for Medical Image Reconstruction

2. The student's original contribution (not including the documentation part) and design specifications:
The objective of this project is to increase the quality of reconstruction for Magnetic Resonance Images. In these images it is important to preserve some anatomical information. In this regard, a deep neural network architecture will be designed to complete the reconstruction task. A series of experiments will be performed to validate this approach. The task aims at reconstructing good quality images even when the number of the acquired data is small. The testing will be done on a public database.

3. Resources used for developing this project:
Programming language: Python; Open access medical imaging datasets

4. The project is based on knowledge mainly from the following 3 courses:
Image Processing and Analysis (PAIC), Human-Machine Visual Interface (IVOM), Data Analysis and Machine Learning

5. The Intellectual Property upon the project belongs to: UPB

6. Thesis registration date: 25.11.2019

Thesis Advisor,
Prof. dr. ing. Corneliu BURILEANU

Student,
Rebeca-Grațiela PREDESCU

Master program coordinator,
Prof. dr. ing. Dragoș BURILEANU

Dean,
Prof. dr. ing. Cristian NEGRESCU

STATEMENT OF ACADEMIC HONESTY

I hereby declare that the thesis **"Deep Learning Strategies for Medical Image Reconstruction"**, submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of **Master of Science** in the domain **Technology and Telecommunication Systems**, study program **Multimedia Technologies in Biometrics and Information Security Applications**, is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited "as is" or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations or measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations or measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest,
June 2020

Eng. Rebeca-Grațiela Predescu

Table of Contents

List of Figures

List of Figures

List of Tables

List of Abbreviations

A
ADMM: Alternating Direction Method of Multipliers
AE: Autoencoder

B
BM3D: Block Matching 3D

C
CNN: Convolutional Neural Network
CT: Computed Tomography

D
DNN: Deep Neural Network

F
FROC: Free-Response Receiver Operating Characteristic
FTB: Filtered Back-Projection

I
ISTA: Iterative Shrinkage-Thresholding Algorithm

J
JSR: Joint Spatial-Radon

M
MAE: Mean Absolute Error
MRI: Magnetic Resonance Imaging
MSE: Mean Square Error

N
NLL: Negative Logarithmic Likelihood

NMR: Nuclear Magnetic Resonance
NN: Neural Network

P
PANO: Patch-Based Nonlocal Operator
PDHG: Primal Dual Hybrid Gradient
PET: Positron Emission Tomography
PSNR: Peak Signal-to-Noise Ratio

R
RecPF: Reconstruction from Partial Fourier
ReLu: Rectified Linear Unit
ResNet: Residual Network

S
SGD: Stochastic Gradient Descent
SNR: Signal-to-Noise Ratio
SSIM: Structural Similarity Index
SVM: Support Vector-Machine

T
TV: Total Variation

Introduction

Motivation

In the recent years, a significant amount of research has been focused on increasing the quality of images. Medical images, in particular, could benefit enormously from this rise in interest, as they can assist medical doctors to give a better diagnostic to patients. The goal is to process the input image in such a way as to obtain a very high quality output. Indeed, one of the challenges the applications in the medical field rise is the intolerance to mistakes.

Although good results were obtained using mathematical implementations, the lack of flexibility poses major problems. Indeed, studies have shown the difficulty in mapping a large database using traditional approaches. Still, the robustness of such methods makes them desirable, at least in the preprocessing step.

The Artificial Intelligence techniques changed the approach regarding image processing. In present time, neural networks (NN) are used in object detection, face recognition, gesture recognition, image restoration with state-of-the art accuracies of over 95%. Since these results are promising, today's tendency is to use such solutions also for medical images. Despite that, almost all these high-performance classifiers are very resource-consuming, using very complex architectures as DNNs (Deep Neural Networks) or CNNs (Convolutional Neural Networks). This complexity makes them difficult to integrate in real-time applications.

An important aspect that should be considered when working with neural networks for developing applications on medical images, is the robustness to malicious data that can completely change the outcome. The challenge is to design a system robust to such situations.

Applications and Challenges

The design of a system that reconstructs medical images plays an important role in clinical trials. The goal is to have a system that works on any kind of medical image, such as X-ray, computed

tomography (CT), positron emission tomography (PET) or magnetic resonance imaging (MRI). This imposes some flexibility to the system, as it should give the same high quality output regardless of the input image.

Using deep learning models for such tasks could introduce some problems that should be taken into account, as they could limit the applicability of this solution. Some of the most important challenges that appear when working with neural networks are [1]:

- *The lack of labeled data* – This problem of data labeling is incredibly time-consuming and, unfortunately, it requires an expert. As opposed to other applications, where data could be easily labeled by a non-expert, in the case of medical images, correct labeling is crucial.
- *The number of observations* – The most pressing problem for this kind of applications is the lack of data. Due to privacy concerns on non-disclosure agreements, the obtaining of medical images could prove problematic.
- *Human expertise* – Any medical doctor takes into account a number of factors, not just the image. So, the decision should be made considering more information than just the one offered by an image. This expertise is usually acquired by doctors in time, and it would be difficult to include these aspects in the neural network's decision.

Despite these major drawbacks, deep learning models are extremely attractive to solve image reconstruction problems.

Thesis objectives and outline

This thesis aims to develop a system that, when fed with a medical image affected by noise, a good quality reconstruction will be returned. The task will be fulfilled by using a neural network and the results will be compared with state-of-the art techniques.

When developing the algorithm, a public dataset will be used. The chosen data should contain a large enough number of samples such that the system offers satisfactory results when new scans are desired to be reconstructed. Several tests will be carried out in order to verify the dependence of the reconstruction quality with the variation of training parameters.

The objective is to obtain high quality reconstructions of the input image. The quality of the output is judged both in terms of noise removal and preservation of structural integrity.

The rest of the paper is structured as follows: in Chapter 1, state-of-the-art solutions are presented, along with some theoretical background that will be necessary in the next sections. Chapter 2 briefly introduces magnetic resonance images, presenting the challenges imposed by working with them and a detailed exposition of the database employed in this thesis. Chapter 3 introduces some basic image segmentation techniques in the first part and in the second part, the algorithm used is described and the results are shown. The network design, details of implementation and experiments are presented in Chapter 4. In the end, conclusions are drawn, emphasizing the personal contributions and indicating possible future work directions.

Chapter 1

Theoretical Background

1.1 Image reconstruction – general aspects

Medical imaging plays a decisive role in medical doctors' lives by assisting them put the correct diagnosis. Image reconstruction is crucial in the medical field, as the acquired data may be lacking the quality necessary to draw conclusions regarding the patient's state. The objective of image reconstruction is to have high quality results with minimum cost and risk for the patient. Traditional methods make use of some mathematical models and their main advantage is robustness. A disadvantage they present is their reduced flexibility, this aspect making them not suitable for large datasets. Modern approaches take advantage of the increased computation power and relay on deep learning techniques. This way, the human assistance in the restoration process is reduced to a minimum. Deep learning models are efficient in extracting information from large datasets, making them attractive from this point of view. Their disadvantage is the lack of theoretical foundation.

Image reconstruction can be modeled as it follows:

$$\mathbf{f} = \mathbf{A}\mathbf{u} \oplus \boldsymbol{\eta} \tag{1.1}$$

where **A** is the operator modeling the physical system, **u** represents the desired image to be reconstructed, **f** is the measured data and **η** models the noise with known or unknown distribution. With $\oplus$ was denoted the addition operation in case of Gaussian distributed noise and some other nonlinear operator in case of Poisson, Rayleigh, Rician etc. noise. [1]

Depending on the application, **A** takes different forms. In the medical field for example, in case of magnetic resonance imaging (MRI) **A** is a sub-sampled Fourier transform [2]. For X-Ray based computed tomography (CT), **A** is a sub-sampled Radon transform [3].

It is difficult to solve the problem raised by 1.1 directly so, in general, the following solution is commonly accepted:

$$\min_{u \in D} \mathcal{L}(\boldsymbol{u}) = F(\boldsymbol{Au}, \boldsymbol{f}) + \boldsymbol{\lambda \Phi}(\boldsymbol{W}, \boldsymbol{u}) \tag{1.2}$$

having the solution u* = arg $\min_{u \in D} \mathcal{L}(u)$. In 2.2, F(.) denotes the fidelity term, that shows how well the approximated solution fits the measured data, **f**. It depends on the noise statistics, taking different forms [1]:

- Gaussian: $F(\mathbf{Au,f}) = \frac{1}{2}\|\mathbf{Au} - \mathbf{f}\|_2^2$,where $\|\mathbf{x}\|_2$ is the notation for Euclidean norm and gives the ordinary distance from **X** to the origin
- Poisson: $F(\mathbf{Au,f}) = \langle 1, \mathbf{Au} \rangle - \langle \mathbf{f}, \log(\mathbf{Au}) \rangle$, where $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i a_i b_i$
- Impulse: $F(\mathbf{Au,f}) = \|\mathbf{Au} - \mathbf{f}\|_1$,where $\|x\|_1 = \sum_{i=1}^{n} |x_i|$ is the Manhattan norm

Φ(W,u) is the regularization term that maps the prior knowledge. Parameter λ gives the contribution of the prior knowledge to the final result. The **W** term from the prior knowledge is a transform used to extract features from the image, such as a differential operator or a wavelet transform. Starting with 1990 such mathematical approaches were used for image reconstruction, their main advantage being the theoretical background and so the high interpretability [1]. Some of the most successful models based on mathematical characterization include total variation model [4], shock-filters [5], wavelet [6].

Deep learning models use nonlinear mapping to change the input data **f** to a high quality output **u**. The nonlinear mapping is defined as $\mathcal{N}(\cdot,\Theta) : \mathcal{F} \rightarrow \mathcal{U}$. **Θ** is trained on the dataset $\mathcal{F} \times \mathcal{U}$. Parameter **Θ** is obtained from the following relation:

$$\min_{\Theta} \mathcal{L}(\boldsymbol{f}, \boldsymbol{u}; \boldsymbol{\Theta}) = \frac{1}{\#(\mathcal{F} \times \mathcal{U})} \sum_{(\boldsymbol{f},\boldsymbol{u}) \in \mathcal{F} \times \mathcal{U}} \mathcal{C}(\mathcal{N}(\boldsymbol{f}, \boldsymbol{\Theta}), \mathbf{u}) + \mathcal{R}(\boldsymbol{\Theta}) \tag{1.3}$$

$\mathcal{C}(\cdot,\cdot)$ is an appropriately chosen metric to compute the difference between the approximated image, $\mathcal{N}(\mathbf{f}, \boldsymbol{\Theta})$ and the ground truth, **u**. $\#(\mathcal{F} \times \mathcal{U})$ is the cardinality of the data set and $\mathcal{R}(\cdot)$ is the regularization term, that is recommended to be introduced in order to prevent the overfitting. It can be chosen as the Manhattan or Euclidian norm [1].

With the introduction of various types of convolutional neural networks (CNN), more parameters can be taken into account when working with images. These parameters are trained on large datasets which is also the main advantages of such models, making them state-of-the-art in the present time [1].

1.2 Image reconstruction models

A problem that occurs with large datasets is the latency of response. This issue makes optimization a crucial step when designing a system. Some algorithms that proved efficient in this direction are presented in the following paragraphs.

1.2.1 The primal-dual algorithm

The optimization problem proposed in this algorithm is:

$$\min_u F(u) + \boldsymbol{\Phi}(\mathbf{W}u) \tag{1.4}$$

where $F(u)$ is the fidelity term and $\boldsymbol{\Phi}(\mathbf{W}u)$ is the regularization term, both introduced in the previous paragraphs. Assuming convexity for both these functions, the problem can be alternatively posed as:

$$\min_u \max_w F(u) + \langle Wu, w\rangle - \Phi^*(w) \tag{1.5}$$

Starting from (1.5), the primal-dual gradient is used.

$$w^{k+1} = (I + \partial\Phi^*)^{-1}(w^k + \alpha_k W u^k)$$

$$u^{k+1} = (I + \partial F)^{-1}(u^k - \beta_k W^T w^{k+1}) \tag{1.6}$$

α_k and β_k are tuning parameters [1].

1.2.2 Iterative shrinkage-thresholding algorithm (ISTA)

This algorithm proposes the following solution to the problem posed by (1.2):

$$\alpha^{k+1} = \mathcal{T}_{\lambda\tau_k}(\alpha^k - 2\tau_k \mathbf{W}(\mathbf{W}^T\alpha^k - f)) \tag{1.7}$$

In the above relation, $\mathbf{W}^T$ is the decoding operator, α^k is the code at step k which needs to be decoded, τ_k is the step size, $\mathcal{T}_\lambda(\cdot)$ is called the soft-thresholding operator and is defined as $\mathcal{T}_\lambda(x) = \text{sgn}(x)\max(|x| - \lambda, 0)$ [1].

1.2.3 Stochastic gradient descent (SGD)

SGD is an iterative method for optimization that replaces the gradient with its estimate. The general form is:

$$\min_\Theta F_N(\Theta) = \frac{1}{N}\sum_1^N f_i(\Theta) \tag{1.8}$$

Parameter Θ that minimizes $F_N(\Theta)$ needs to be estimated, $f_i(\Theta)$ is the i-th observation and N is the total number of observations in the dataset. As N increases, the evaluation for $F_N(\Theta)$ poses problems, as the time needed for it increases. As a consequence, the relation used to determine Θ is:

$$\Theta^{k+1} = \Theta^k - \alpha_k \frac{1}{|\mathcal{S}_k|}\sum_{i_k\in\mathcal{S}_k} \nabla f_{i_k}(\Theta^k) \tag{1.9}$$

where α_k is the learning rate, $\mathcal{S}_k$ is a random subset from the dataset. $\frac{1}{|\mathcal{S}_k|}\sum_{i_k\in\mathcal{S}_k}\nabla f_{i_k}(\Theta^k)$ is an approximation of the gradient [1].

1.2.4 Alternating direction method of multipliers (ADMM)

The optimization problem from (1.2) can be equivalently written as [1]:

$$\min_{\mathbf{u},\mathbf{d}} \mathcal{L}(\mathbf{u},\mathbf{d}) = \frac{1}{2}\|\mathbf{Au}-\mathbf{f}\|_2^2 + \lambda\|\mathbf{d}\|_1 \quad (1.10)$$

In the above relation, d = **W**u. The augmented Lagrangian function with the multiplier *b* is:

$$\mathcal{L}(u,d;b) = \frac{1}{2}\|\mathbf{Au}-\mathbf{f}\|_2^2 + \lambda\|\mathbf{d}\|_1 + \langle Wu-d,b\rangle + \frac{\mu}{2}\|\mathbf{Wu}-\mathbf{d}\|_2^2 \quad (1.11)$$

In (1.11), μ is the tuning parameter. Then, the ADMM takes the form:

$$u^{k+1} = (A^TA + \mu W^TW)^{-1}(A^Tf + \mu W^T(d^k - v^k))$$

$$d^{k+1} = \mathcal{T}_{\lambda/\mu}(Wu^{k+1} + v^k)$$

$$v^{k+1} = v^k + (Wu^{k+1} - d^{k+1}) \quad (1.12)$$

1.3 Deep Neural Networks (DNN) used for medical images

While the traditional methods have the advantage of a solid theoretical foundation, they are not very flexible, making them hard to describe large datasets. Deep learning models are much more flexible if designed correctly and thus can make use of larger datasets. These major advantages come, however, with major drawbacks regarding the model's interpretability. Still, deep neural networks are powerful tools, especially when working with complex data, as they extract features very efficiently. They can be used for medical image reconstruction [1].

1.3.1 General Aspects

Although not a new concept, neural networks were not very popular until recently, when the computation power has increased to such heights that the complexity of such a structure does not pose any problem.

The model for NNs was the human nervous system: the parallel was traced between biological neurons and cells interconnected in a network, in a similar way to synapses.

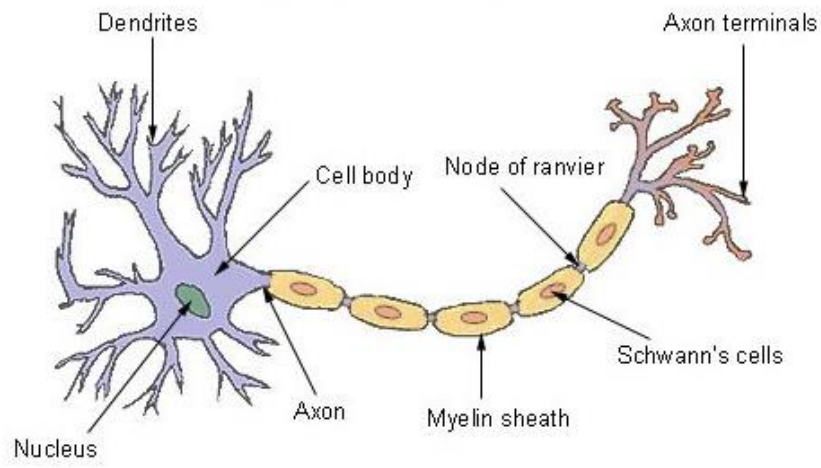

Figure 1.1 Neuron [7]

A neuron functions according to the following principle: the input is taken from multiple other neurons that are linked to its dendrites. After all processing is done, the output is sent through the axon terminals to other neurons. An artificial neuron functions on the same principle, taking the input information from all neurons that are connected to it, perform some operation and then pass the result to the neurons from the next layer that are connected to it.

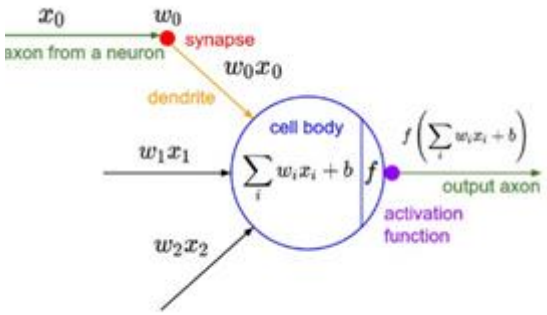

Figure 1.2 Mathematical model for the neuron [8]

In Figure 1.2 is presented the general structure of an artificial neuron. x_i represents the input data, having the corresponding weights $\mathbf{w_i}$. The bias is denoted with b, the transfer function is $\text{net} = \sum_i x_i w_i + b$, the activation function is out = f(net) and the error is computed using the relation: $\text{err} = \frac{1}{2}(\text{out} - \text{target})^2$.

There are three types of layers in a NN:

- *input layer* – represents the input in the network and is fed with the training data
- *hidden layers* – they are intermediate layers that take the information from the input layer and process it by creating relationships between data
- *output layer* – it's the final layer, where the decision is made

In Figure 1.3 the architecture of a neural network can be observed. In this case, each neuron is connected to all others, so the network is fully connected.

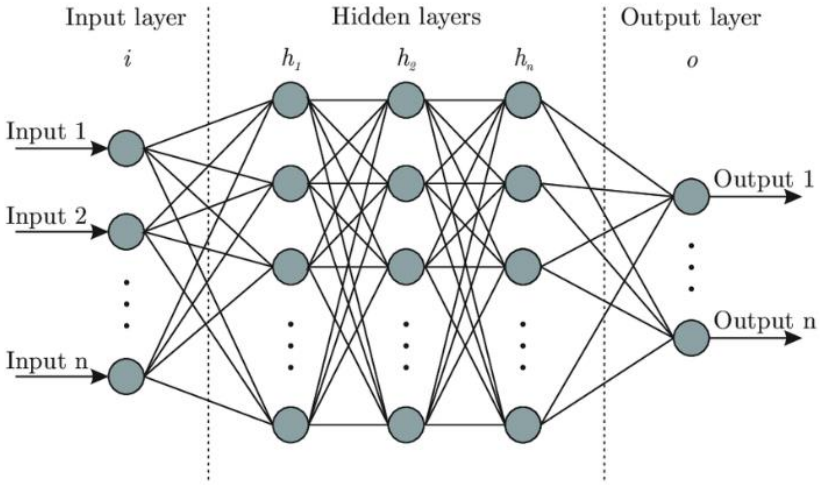

Figure 1.3 Neural Network Architecture [9]

1.3.1.1 Activation functions

- **Binary step**

It is one of the earliest activation functions introduced, used for binary classifications, as it has only two possible states for the neuron to be in: either activated or deactivated. The mathematical representation of the binary step is:

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{1.13}$$

The plot is presented in Figure 1.4:

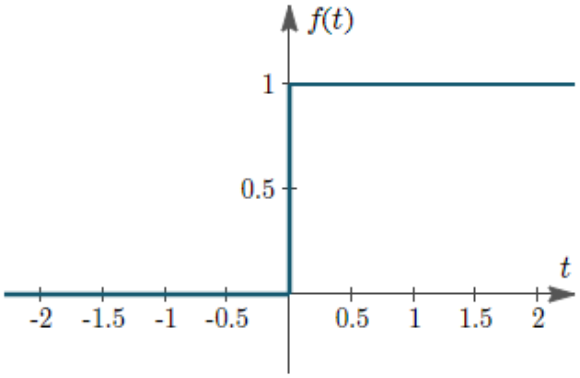

Figure 1.4 Binary step activation function [8]

- **Sigmoid function**

It is used for non-binary classification, when it is desired to prioritize the neurons depending on how much information they carry and have different outputs accordingly. The main advantage of the sigmoid function is that it is smooth, so differentiable. It takes values in the interval [0, 1] and is defined as it follows:

$$f(x) = \frac{1}{1+e^{-x}} \tag{1.14}$$

The graphical representation of the sigmoid function is offered below:

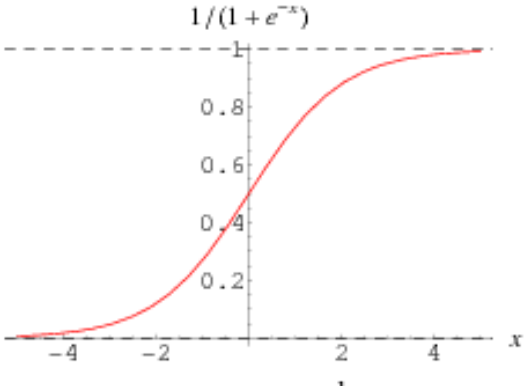

Figure 1.5 Sigmoid function [10]

- **Hyperbolic Tangent**

The hyperbolic tangent has a similar shape with the sigmoid's, but the results are in the interval [-1, 1]. Its introduction was necessary to solve backpropagation problmes that appear due to the values close to zero that the sigmoid outputs for strongly negative numbers. In this case, the parameters won't be updated and the network will not converge. The hyperbolic tangent offers negative results for negative inputs, so this situation is avoided. The definition of the hyperbolic tangent is:

$$f(x) = \frac{2}{1+ e^{2x}} - 1 \tag{1.15}$$

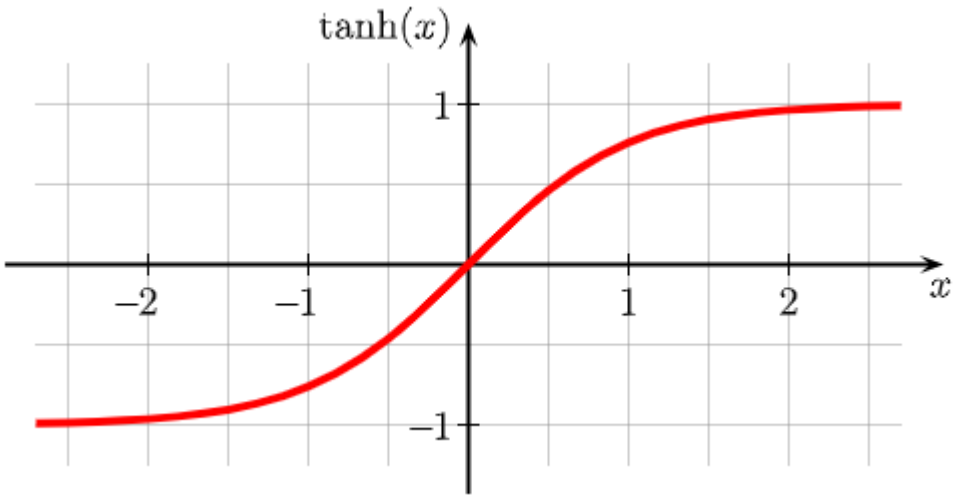

Figure 1.6 Hyperbolic tangent [8]

- **Rectified linear unit (ReLu)**

It is a popular activation function, as it involves only basic mathematical concepts. However, the function is not differentiable in 0, where a random value should be assigned. Much like the sigmoid function, for negative inputs the output will always be zero, so there could be problems at backpropagation.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{1.16}$$

The graphical representation of ReLU function is shown in Figure 1.7.

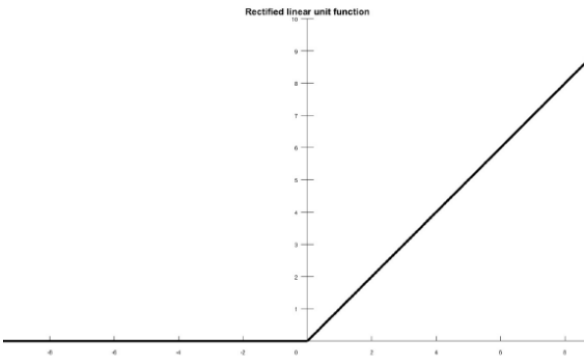

Figure 1.7 Rectified linear unit function [11]

- **Leaky ReLU**

To overcome the backpropagation problem that can occur if the output of ReLU is constant zero, leaky ReLU was introduced. The vanishing gradient is solved by applying leaky ReLU instead of the classical approach.

$$f(x) = \begin{cases} \alpha x, x \leq 0 \\ x, x > 0 \end{cases} \tag{1.17}$$

$$\frac{\delta f(x)}{\delta x} = \begin{cases} \alpha, & x \leq 0 \\ x, x > 0 \end{cases}$$

In relation (1.17), α is a constant. The representation of leaky ReLU is given in Figure 1.8:

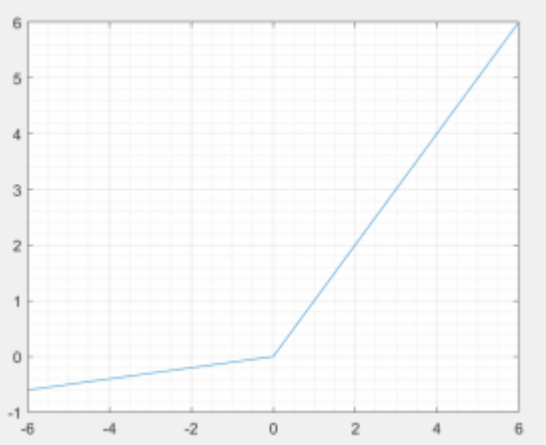

Figure 1.8 Leaky ReLU

1.3.1.2 Loss function

The training is done by minimizing a conveniently chosen function, called loss function. It represents the difference between the obtained output and the desired one. To have good accuracy, the differences of the two should be minimal, that is the loss function should be at its minimum.

- **Mean Square Error (MSE)** is one of the most common functions used as loss functions. Its minimization means finding the line that minimizes the distance from each point to the line, called *regression line*. It is defined as:

$$\text{Loss} = \frac{1}{N} \sum_{i=1}^{N} (\text{input}_i - \text{target}_i)^2 \tag{1.18}$$

- **Mean Absolute Error (MAE)** is similar to MSE, but an advantage it presents is the robustness to outliers. The definition of MAE is:

$$\text{Loss} = \frac{1}{N} \sum_{i=1}^{N} |\text{input}_i - \text{target}_i| \tag{1.19}$$

- **L2** is similar to MSE, the only difference being that the result is not divided to the number of observations:

$$\text{Loss} = \sum_{i=1}^{N}(\text{input}_i - \text{target}_i)^2 \quad (1.20)$$

- **L1** is similar to MAE, but for the division to the number of observations:

$$\text{Loss} = \sum_{i=1}^{N}|\text{input}_i - \text{target}_i| \quad (1.21)$$

- **Negative Logarithmic Likelihood (NLL)** is commonly used to express mathematically the confidence score:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}\log(\text{target}_i) \quad (1.22)$$

1.3.2 DNNs used in image reconstruction

Some of the most commonly used DNNs that were successfully used for medical image reconstruction are residual network, autoencoder and U-net.

1.3.2.1 Residual Network (ResNet)

Network's depth plays a major role in the quality of results, as proven by tests done on ImageNet dataset [12]. It was shown that better results are obtained when using deeper models. However, sometimes, as the network's depth increases, the results seem to degrade. This phenomenon appears as a cause of vanishing gradient during back propagation. Vanishing gradient emerges when the weights of the network stop updating. Adding more layers increases the effect, so the solution is to skip some layers, as not all systems are similar from the optimization point of view [13].

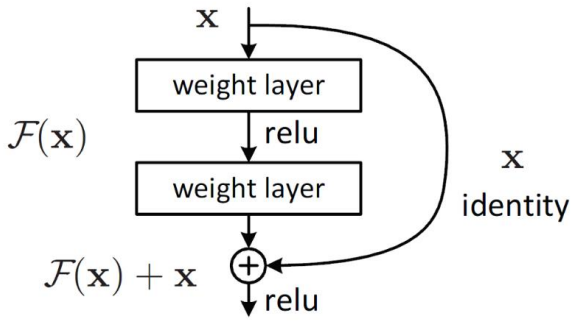

Figure 1.9 ResNet architecture [14]

In Figure 1.9, ***x*** represents the input feature, the *weight layer* could be convolutional, ReLu is the rectified linear unit, defined as $f(x) = \max(0, x)$. $\mathcal{F}(x)$ is the nonlinear residual block. So, the output of the block can be written as $x_{k+1} = x_k + \mathcal{F}_k(x_k)$, where with k was denoted the current layer. Shortcut connections are used to solve the vanishing gradient problem. The identity shortcuts have no parameters. Tests done by Kaiming He et al. in [13] proved that using such architecture the error rate decreases as compared to any other method. On ImageNet dataset, they obtained 3.57% error rate, while if using GoogLeNet, for example, the error was 6.66%. All results are public and can be verified on ImageNet Large Scale Visual Recognition Challenge website.

1.3.2.2 Autoencoder (AE)

Autoencoder is used to learn data representation in an unsupervised manner. The representation is nonlinear, being learned from the input dataset. Christopher Poultney et al. proposed in [15] a model based on three components:

- The encoder that consists of a set of fed-forward filters, used to compute a code from the input image
- The sparsifying logistics, consisting of a module that transforms the code vector into a sparse one, taking sub unitary values
- The decoder, that reconstructs the image from the sparse vector

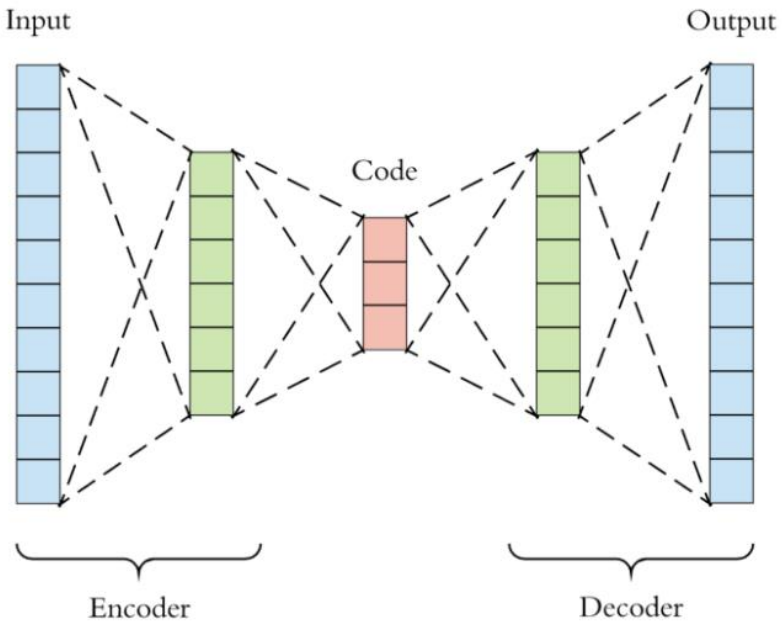

Figure 1.10 Autoencoder – general representation [16]

In [17], Hu Chen et al. proposed a combination of autoencoder and convolutional neural network, using convolutional layers instead of the fully-connected ones that were typically used for encoding and decoding. Their proposed method also includes residual learning with short-circuits that are used to improve the learning process. They managed to increase the peak signal-to-noise ratio (PSNR) at 39.159, as compared to 38.9907, as obtained using a 10 layer CNN. The testing was done on a real clinical database, authorized by Mayo Clinics.

1.3.2.3 U-Net

The U-Net proved to be successful in image segmentation. In [18] such architecture was proposed for medical images. It consists of a contracting path, following the typical CNN architecture, and an expansive path. It consists of 3*3 convolutions, followed by ReLu and a 2*2 max pooling operation. The number of feature channels is doubled at every downsampling step. On the other hand, in the expansive path, the upsampling of the feature map is followed by 2*2 convolutions; this way the number of feature channels is reduced with a 2 factor. Then, the output is concatenated

with the corresponding feature map obtained in the contracting path. Their result is then passed through 3*3 convolution blocks, each followed by ReLu. At the final layer, a 1*1 convolution is used. The graphical representation of the U-Net proposed by Olaf Ronneberger et al. in [18] is presented in Figure 1.11. The number of channels is written at the top of the box, while the x-y dimensions are presented at the bottom. Blue boxes correspond to multi-channel feature map, white boxes to copied feature maps. It can be noted that skip connections were also used.

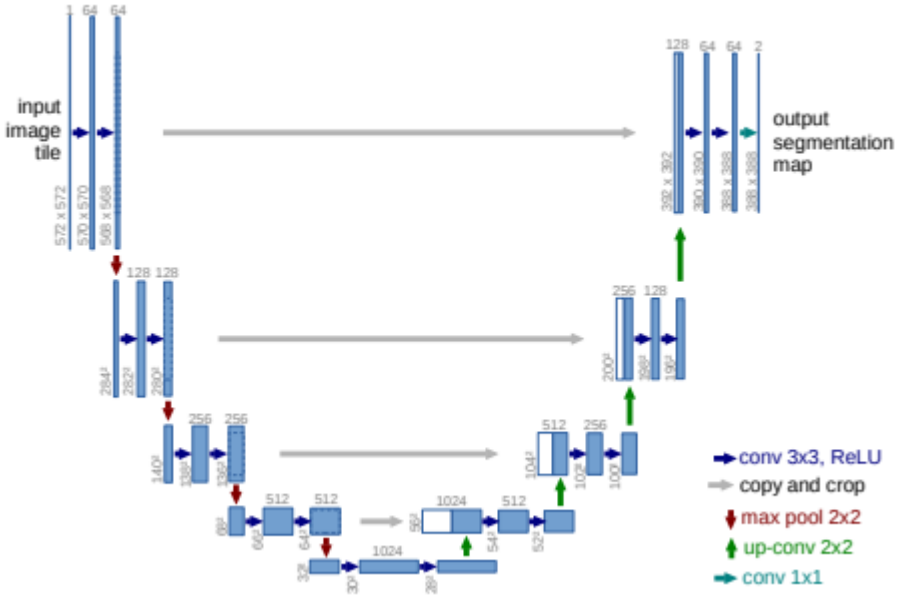

Figure 1.11 U-Net architecture [18]

1.4 Medical image reconstruction using DNNs

Taking as an example of medical image a computer tomography (CT) scan, studies have shown that classical image reconstruction models are efficient, but at the same time sensitive to noise. In the recent years, it was desired to introduce also deep models to help traditional reconstruction methods be more robust. This combination of classical modeling and deep modeling can be divided into two major categories: post-processing and raw-to-image [1].

1.4.1 Post-Processing

This method requires the estimation of the mapping between the input, low quality image and the high quality output. This kind of approach is preferable when the input and output do not differ in a very drastic measure. One of the problems this method presents is the presence of artifacts in the output image that are caused by the presence of noise. If these artifacts are complex enough, they cannot be removed by the deep network. So, this approach has limited performances and is suitable only some particular situations, when it gives high quality results [1].

In [17], it was proposed to use a residual encoding-decoding CNN to approximate the mapping between the input and output image. This approach proved efficient in removing the noise from the filtered back-projection (FTB) reconstructed CT scans.

Recently, as it was desired to reduce the radiation dose, the number of projections was decreased, this way resulting sparse view CT. As the measurements are incomplete this way, artifacts appear in the reconstructed image. A DNN can be used in such situations to learn the pattern of unwanted items and later remove them. In [19], a U-net was used with such a purpose, the output image being obtained from the subtraction of the artifacts that were learned by the U-net, from the input image.

1.4.2 Raw-to-Image

In this approach, it is desired to directly estimate the mapping between the reconstructed image and raw data. Using directly a DNN to solve the problem is not feasible because the data distributions of the two images of interest are very different. A traditional approach, based on differential equations, would not have this problem. In this case, usually a traditional technique is first used and then the results are passed to the DNN, in this way benefiting from the advantages of both [1].

1.4.2.1 ADMM-Net

In classical ADMM approach, it is difficult to determine the tuning parameters. This motivated the authors of [17] to introduce a deep model that solves this problem. In their proposal, the tuning parameters are learned from the training data. The regularization term, Φ is approximated with a piecewise linear function, whose parameters can also be learned. They tested the functionality on magnetic resonance (MR) images from a database provided by *CAF Project: Segmentation Challenge* and they obtained a 37.17 PSNR for images having 20% as a sampling factor. They report an outperformance of their solution over the state-of-the art by 0.71dB, at the same time the processing time decreasing with 2 magnitude orders [17].

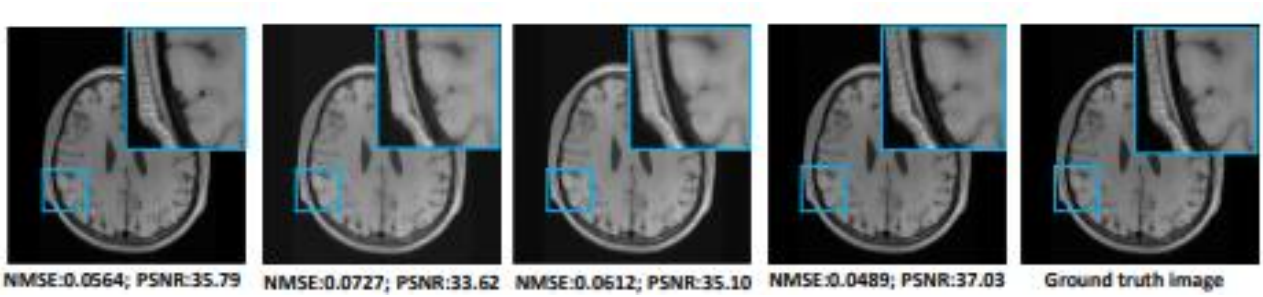

Figure 1.12 Reconstruction results [17]

In Figure 1.12 there are presented the reconstructed images using different solutions. From left to right, they are obtained using ADMM-Net, reconstruction from partial Fourier data (RecPF), patch-based nonlocal operator (PANO), block matching 3D (BM3D-MRI). The rightmost image represents the ground truth.

1.4.2.2 Primal-Dual network (PD-Net)

In [21], the authors used primal-dual hybrid gradient (PDHG) to design a new model for CT image reconstruction. Their idea was to approximate each operator from PDHG with a neural network, and using this approach they reported a 38.8 PSNR, as opposed to 19.75 obtained when using filtered back projection (FTB) and 28.06, when using total variation (TV). Using FBP and U-net for denoising increases the PSNR to 29.20 and when using residual denoising instead of U-net, to 32.38. Still, the results obtained by [17] are by far better than all other presented approaches.

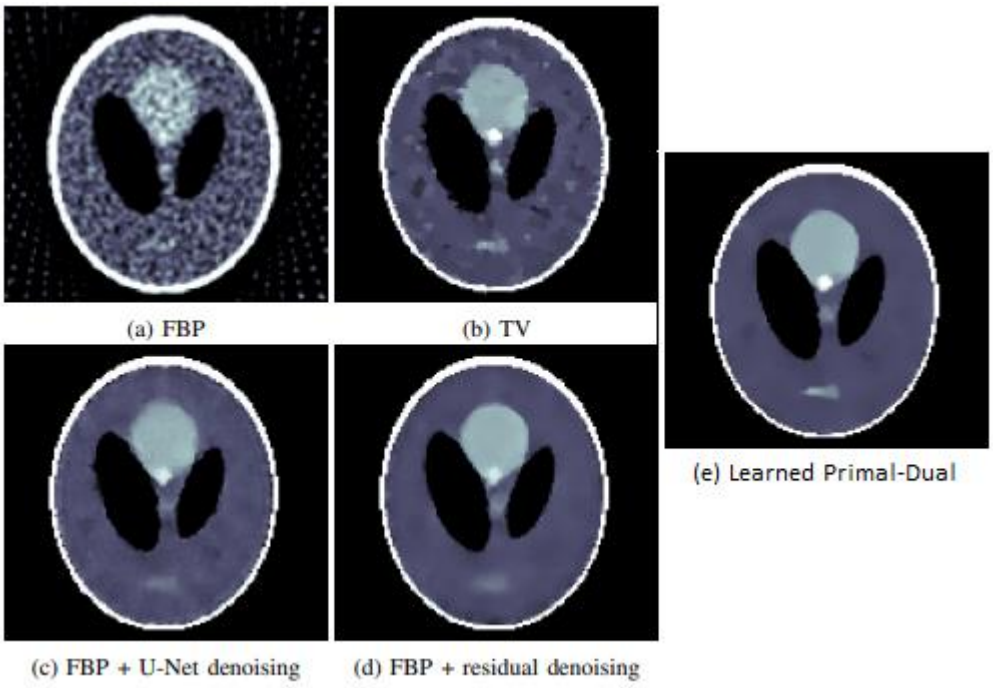

Figure 1.13 Results obtained using different methods [21]

1.4.2.3 Joint spatial-Radon (JSR-Net)

In [22] a JSR model was proposed, whose purpose was to reduce the artifacts that appeared in noisy images or when the database contained incomplete data. In this approach, the data fidelity term is defined as:

$$\mathcal{J}(u, f, Y) = \frac{1}{2}\|R_{\Gamma^c}(f - Y)\|^2 + \frac{\alpha}{2}\|R_{\Gamma}(Au - f)\|^2 + \frac{\beta}{2}\|R_{\Gamma^c}(Au - Y)\|^2 \quad (1.23)$$

The regularization term takes the following form:

$$\mathcal{R}(u, f) = \|\lambda_1 \cdot W_1 u\|_{1,2} + \|\lambda_2 \cdot W_2 u\|_{1,2} \quad (1.24)$$

In the above relations, R_Γ is a restriction operator that indicates the missing data from the region Γ. Γ^c is the complement of Γ, as it indicates the region of available data. The discrete form of Radon transform is mapped by A and Y is the measured projection data. $W_1$and W_2 are tight wavelet frame transforms and λ_1 and λ_2 denote the regularization parameters. The CT image is **u** and the restored one is **f** [1].

Tests done in [22] show that the relative error decreases as compared to models using total variation. The relative error for TV-based model is 3.8% while for both anisotropic and isotropic wavelet based models the results are better, being 3.5% and 2.7% respectively.

The authors of [23] proposed an improved version, where the tight frame transforms are learned from data. They used neural networks to approximate the proximity operators. Using their model on head CTs, they obtained the relative error 3.58%.

1.4.3 Tasks

There are two major directions when it comes to medical images. The first one refers to image reconstruction, such that the results are satisfactory when analyzing their quality. The second step consists of the analysis of the high-quality image, being this way able to draw some conclusions regarding the health of the patient. These two steps can be combined using a convolutional neural network (CNN). The simplest solution would be to connect the network doing the reconstruction to the one that analysis the image and do an end-to-end training [1].

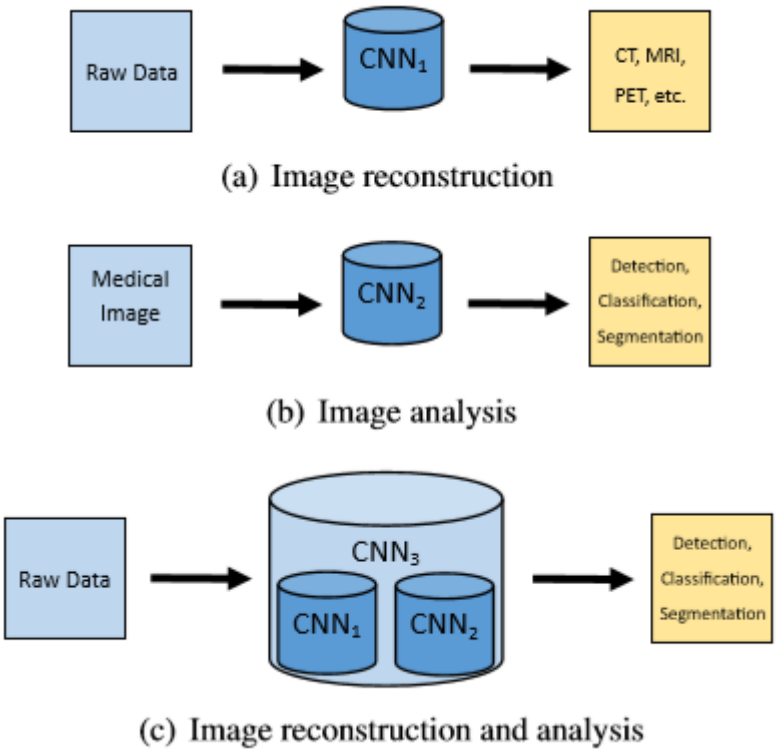

Figure 1.14 Workflow [1]

This idea was implemented in [24]. The authors first converted the raw data to images using a reconstruction sub network. The results were fed to a three-dimensional CNN that was in charge of detecting lung nodules in images. The two sub networks were trained individually, but the last step was to perform some tuning on the whole network, with the purpose of increasing the detection accuracy. Their tests were done on *The Lung Image Database Consortium* image collection. The metric of choice was the free-response receiver operating characteristic (FROC) and it is defined as the plot of sensitivity versus the average number of false-positives per image [25]. In case of images

that were not affected by noise, the FROC score for end-to-end training was 0.608 as compared to 0.560 that was obtained when the two networks were trained individually. When there was introduced a noise of level $5x10^{-4}$, the FROC score was 0.587 for end-to-end training and 0.512 for individual training, which suggest a more rapid deterioration in the second case.

Chapter 2

Magnetic Resonance Imaging

2.1 Principles

The nuclear magnetic resonance (NMR) was described by Purcell and Bloch in 1946 and it introduced lots of benefits especially in the organic chemistry area. The typical NMR spectrometer consists of a magnet, a radio-frequency oscillator and the corresponding receptor. The introduction of wire-bore superconducting magnets allowed a new approach in clinical applications [26].

The nucleus of any atom consists of protons and neutrons. Protons have a positive charge, while neutrons have no electric charge, this leading to a positive net charge for the whole nucleus. Some atomic nuclei present the "spin" property, depending on the number of protons. Spin angular momentum is the property presented by nuclei that allows the magnetic interactions to occur [27].

If an external magnetic field B_0 is applied, the nucleus will be aligned parallel or perpendicular to it. The nuclear spins will be either parallel to the external magnetic field and thus in a low-energy state, or perpendicular to it and in a high-energy state. To excite nuclei that are already placed in a magnetic field B_0, a radiofrequency magnetic field B_1 is applied in pulses. B_1 is perpendicular on B_0. This has as effect the transition from a low-energy state to the high-energy state and the other way around. These transitions induce a voltage that can be measured and amplified, known as the "free-induction decay". Placed in the same magnetic field, each nucleus will resonate at a characteristic frequency [26].

To obtain a magnetic resonance image, multiple radio-frequency pulses are applied, this way obtaining a time-domain signal. By applying the Fourier transform, biochemical information can be acquired. Gradients are applied in the three directions of the orthogonal coordinate system, this way the reconstruction of three-dimension images being possible [26].

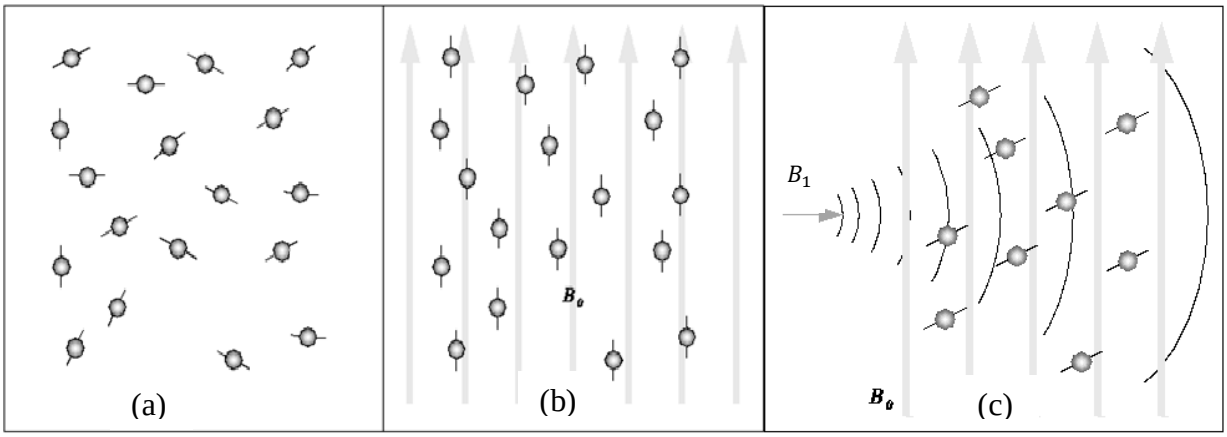

Figure 2.1 In the absence of a magnetic field B_0 , hydrogen nuclei are aligned as in (a). When a strong magnetic field is applied, they align with it as in (b). The radio-frequency pulse causes the tilt of the nuclei (c) [28]

The B_1field is applied by the transmitter magnetic coil which surrounds the region of interest. The receiver coil may be separate or included in the transmitter. To increase the signal-to-noise ratio (SNR), phased array coils are used for the transmitter, each connected to a different receiver, so the noise between coils is uncorrelated. Data is collected from the receivers and combined to form the resulting image [26].

Up until recently, the MRI scanners used magnets in the range 0.5 – 1.5 Tesla (T). Today, 3T magnets are widely available, some of the advantages introduced by the higher magnetic field being the increased SNR and the higher spectral, spatial and temporal resolution. The tradeoff is made in terms of magnetic field stability and magnetic susceptibility. The artifacts introduced by the magnetic susceptibility increase with the increase of the magnetic field strength; in the presence of an external magnetic field, the susceptibility shows the magnetization degree presented by any material as a response to the magnetic field [26].

When a nuclear spin returns in the equilibrium state, after the application of a magnetic field, it is said that the relaxation phenomenon occurs. The relaxation can be of two types: longitudinal or transversal, and, to differentiate between them, time constants are used. The first time constant, denoted by T_1, is also known as the "spin-lattice relaxation". T_1 represents the time interval that is needed for the system to return to the equilibrium by 63% after the excitation. Energy is dissipated in the surrounding nucleus environment during this longitudinal relaxation. The relaxation time differs depending on the tissue: for water and cerebrospinal fluid, T_1 typically takes values between 3 – 5 seconds, the result being that these areas appear dark in the image. Fat, for example, has much lower relaxation time, around 260 ms, and thus appears brighter [26]. Table 2.1 shows the representation differences that appear in an MRI scan, depending on the type of image.

When a pulse is applied, most nuclei align with the applied energy. Dephasing of the orientation occurs at relaxation as the energy is transferred between nuclei. This is the transverse relaxation and measures the exchange energy time of spins in the xy plane. The time constant is denoted by T_2 and is also known as "spin-spin relaxation" [26].

Appearance	T_1 image	T_2 image
White	• Fat • Protein rich fluid	• Water content
Gray	• Gray spinal matter	• White spinal matter
Dark	• Bone • Air • Water content	• Bone • Air • Fat

Table 2.1 Differences in representation of T_1 and T_2 images [29]

MRI scans can be obtained in three different anatomical views: sagittal, coronal and axial. Figure 2.2 exemplifies the possible options for brain MRI.

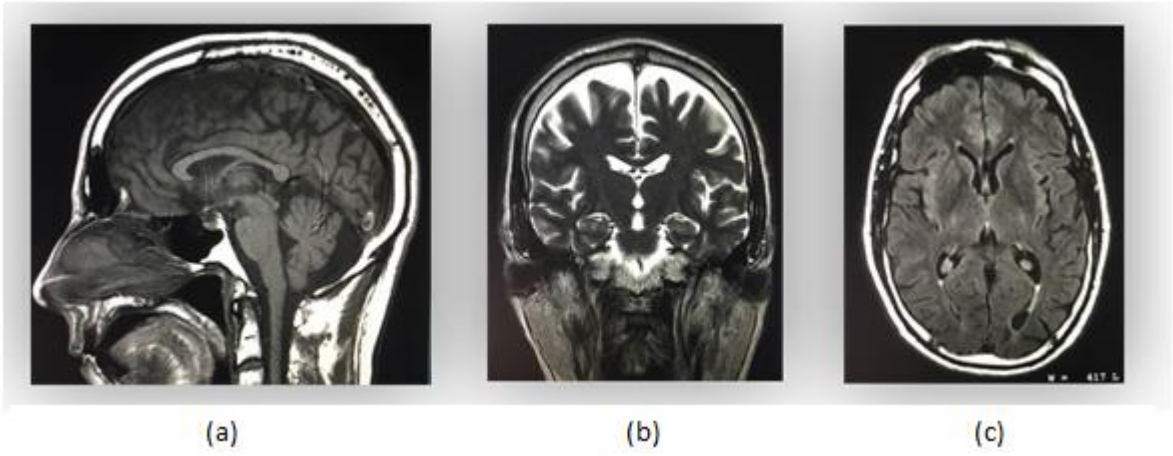

Figure 2.2 Main views obtained for brain MRI: (a) sagittal, (b) coronal, (c) axial [29]

Diffusion-weighted imaging is the technique that allows the measurement of a water molecule's movement. The water molecule movement is isotropic, that is random in all directions. In a structured environment, the molecules' movements are constrained: the gray and white matter in the brain restricts the movement, water molecules positioning parallel to the white matter in their majority. This motion in an anisotropic environment is described by the diffusion tensor. Three eigenvalues and their corresponding eigenvectors are used to define the ellipsoid, describing the magnitude and directions of the diffusion on three coordinates [26].

The cells in a human body are not aligned homogenously, which means a direction-dependent measure of water diffusion. To obtain an estimate of isotropic diffusion that is invariant at rotation, several measurements should be made. To obtain reliable data, the minimum number of diffusion directions should be around 20-30 [26].

Even if all nuclei are under the influence of the external magnetic field applied to the subject, the electrons in their immediate vicinity induce a local magnetic field which is dependent on the chemical structure. In the magnetic resonance spectrum one can identify different resonance

frequencies for different nuclei. Hydrogen and Phosphorus are the nuclei that present special interest in clinical applications [26].

A voxel within an organ is defined using gradients, its size being defined by the user. This data acquisition technique offers good signal to noise ratio. If spectra are acquired from a matrix of voxels, the process is called chemical shift imaging and it offers a better anatomical coverage [26].

The accuracy of data depends both on the hardware equipment, and also on the water suppression and voxel localization. If water is used as a reference for quantification, it should be taken into account the fact that its concentration may vary in different tissues, so the computations will be affected. An important aspect that should be considered when interpreting magnetic resonance data is that identical spectra analyzed by different investigators will produce different results for the metabolite [26].

Visible peaks on the cerebral MR spectrum are:

- **N-acetyl aspartate** is used in the study of multiple sclerosis, its presence indicating a neuronal dysfunction.
- **Choline** is associated with membrane activity. If the choline resonance is elevated, inflammatory diseases are present, while its decrease marks osmoregulatory in hepatic encephalopathy [26].
- **Creatine** can be taken as reference level, since it is constant, regardless of the brain health.
- **Myo-inositol** is involved in cerebral osmoregulatory process and an increased level marks microglial activation and astrogliosis [26].
- **Glutamate** is taken from the capillaries and combined with ammonia, resulting glutamine. Glutamine is converted by neurons in glutamate, which is a neurotransmitter [26].

The metabolite peaks are obtained using special software solutions which analyze the Fourier transform of the signal. If a prior-knowledge approach is used, the accuracy increases, as the user-dependent input is much reduced. If the analysis is done also in time domain, the artifacts introduced by a wide band signal could easily be eliminated [26].

2.2 Usage and Challenges of MRI Scanning

The main usage of MRI scans is to visualize soft tissues within the body, so they play a special role in tumor detection and bleeding localization, making them vital in early diagnosis. The images obtained with an MRI scanner are of high quality and detail. The most notable advantage MRI scanning presents is the absence of radiation exposure for the patient. Another important aspect the MRI scan possesses is the ability to see organs that are typically obstructed by bones. However, to achieve good quality images, an increased time interval is necessary, this aspect being a drawback MRI presents in trauma applications. Another disadvantage it presents is the cost, this type of scans being the most expensive imaging techniques [29].

MRI scanning has no adverse effects and it doesn’t radiate the patients. However, it is not recommended to patients having cardiac peacemakers or heart monitors [29]. Although there are no

studies showing it can affect the fetus, pregnant women are recommended to avoid it unless otherwise instructed.

Table 2.2 outlines the advantages of MRI scanning, by comparing them with other types of medical images, such as CT and X-ray.

Factor	CT	X-ray	MRI
Basic principle	X-ray projections of cross section body	X-ray radiation	energy transition of protons
Duration	3-7 min	2-3 min	30-45 min
Radiation source	X-ray (Ionizing)	X-ray	magnetic field
Radiation	10 mSv	0.15 mSv	None
Soft tissue details	poor	poor	good
Bone details	good	good	poor
Cost	medium	low	high
Type of noise distribution	Gaussian, quantum noise	Gaussian, Poisson	Gaussian, Rician, Rayleigh
Application	anatomical, functional	anatomical	anatomical, functional, molecular

Table 2.2 Comparison between CT, X-ray and MRI scans [29] [31]

The quality of MRI scans is directly influenced by the ability of patients to remain as still as possible during the procedure. Since the duration of a complete scan lasts around 30 minutes, this is a factor that should be considered, especially when dealing with infants or patients suffering from anxiety. Implants and other metallic objects will have the same influence on the overall resulted image as the movement of the patient, so it is recommended to remove all unnecessary items. For chest and abdomen MRIs, breathing will cause image distortions. Another factor that should be taken into account is the weight of the patient, since very large individuals may not enter the machine [30].

MRI scans are affected by Rician, Gaussian and Rayleigh noise. Intensive studies have been carried out for the elimination of Rician noise; however, not so many articles discuss the decrease of Gaussian or Rayleigh noise in MRIs. When the SNR value is greater than 2, the Rician distribution can be approximated by a Gaussian distribution and if the SNR is 0, it converges to a Rayleigh one. The noise sources are: electronic interferences in the circuits and the measurement chain in the process. The noise will reduce the image quality, sometimes increasing the difficulty of putting an early diagnosis [31].

2.2.1 Noise in MRI

Discarding the phase information is the preferred method to eliminate the artifacts phase introduces. This way, magnitude images are obtained, which are assumed to be affected by Gaussian noise. By

applying the Fourier transform to the data, the characteristics of the noise are left unaltered. The real and imaginary images are obtained, having uncorrelated noise in the corresponding voxels [32].

The real and imaginary images are used to compute the magnitude image, pixel by pixel. Magnitude computation is non-linear, meaning that after this operation, the noise will not be Gaussian anymore [32].

If by A is denoted the pixel intensity in the clean image and by M the measured one, the probability distribution for M is given by [32]:

$$p_M(M) = \frac{M}{\sigma^2} e^{-\frac{(M^2+A^2)}{2\sigma^2}} I_0\left(\frac{A \cdot M}{\sigma^2}\right) \quad (2.1)$$

By I_0 was denoted the modified zero-order Bessel function and σ is the standard deviation of the Gaussian noise. The formula given by (2.1) is the Rician distribution, Figure 2.3 mapping it for different SNRs, with $\sigma = 1$.

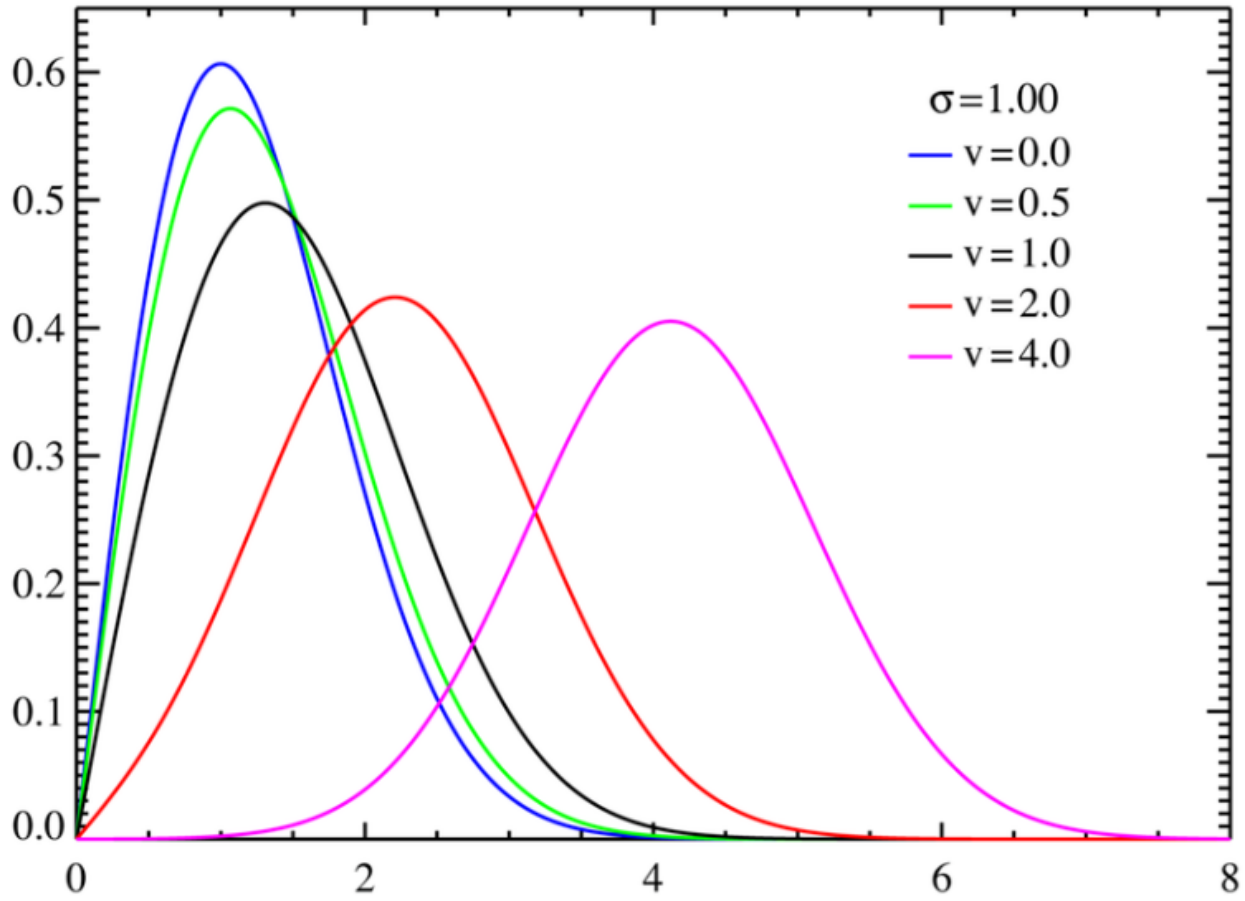

Figure 2.3 Rician distribution of *M* for different SNRs [33]

A special case of the Rician distribution is obtained for the regions where there is only noise present, meaning that $A = 0$. This special case is known as the Rayleigh distribution and formula (2.1) reduces to:

$$p_M(M) = \frac{M}{\sigma^2} e^{-\frac{M^2}{2\sigma^2}} \quad (2.2)$$

When the SNR value is increased, (2.1) can be approximated with:

$$p_M(M) \approx \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(M-\sqrt{A^2+\sigma^2})^2}{2\sigma^2}} \quad (2.3)$$

As it can be observed also from the graphical representation, for values of the signal to noise ratio larger than 2 dB, Rician distribution can be well approximated by a Gaussian distribution, having the variance σ^2 and the mean $\sqrt{A^2 + \sigma^2}$. The errors introduced by the approximation decrease as the SNR increases [32].

The phase image is obtained by computing the arctangent of the ratio of corresponding pixels from the imaginary and real images. This operation is non-linear, so the noise characteristic is not preserved. However, for SNR values high enough, the noise distribution in the phase image can be considered Gaussian with zero mean. It was empirically determined that starting with a SNR value of 3 dB, the Gaussian approximation is realistic [32].

2.3 Dataset

For the development of this thesis, all tests were done on a public dataset, fastMRI. The data was obtained through collaboration between *Facebook AI Research* and *NYU Langone Health*. Their goal was to obtain MR images 10 times faster than the classical approach, improving this way the user experience.

The dataset contains raw and DICOM data, which have been manually inspected, such that to eliminate any inconsistencies. All images containing unexpected information were disregarded [34].

This dataset is contains MRIs for both knee and brain:

- **Knee MRI** contains "coronal proton density-weighted with and without fat suppression axial proton density-weighted with fat suppression, sagittal proton density, and sagittal T2-weighted with fat suppression" [34]. These MRIs were obtained using 3 or 1.5 Tesla magnets. There are 1398 raw data scans in total.
- **Brain MRIs** are obtained using also 3 or 1.5 Tesla magnets. This dataset contains axial T1 weighted images, T2 weighted and FLAIR scans. The total number of scans reaches 7002 [34].

Four types of data can be found in each of these categories [35]:

- Raw multi-coil k-space data: in this category, unprocessed measurements are found.
- Emulated single-coil k-space data: it is derived from the multi-coil space data, such that to emulate a single coil. The introduction of emulated data was necessary for situations when single-coil reconstruction algorithms needed to be tested.

- Ground-truth images: this category is used as reference and it contains images reconstructed from fully-sampled multi-coil data.
- DICOM images: this category was introduced to map different machines and settings for them, such that to have a larger variety of data. They are obtained using several scanners, different acquisition techniques and reconstruction methods. Obviously, the quality differs from image to image.

In this thesis only the brain MRIs were taken into consideration. From all types of brain data from fastMRI dataset, the DICOM images were considered, the rationale behind this choice being the variety of scanners used, making this way the approach machine-independent. Data was collected in five different laboratories, using 3T and 1.5T magnets. Only axial 2D-images were taken for this dataset. All images were reconstructed in DICOM format and then each was visually checked by certified MR technologists. This last step was necessary in order to ensure the anonymity of the patients, as well as to eliminate distorted data. The resulted images were cropped to further eliminate inconsistencies across the dataset. All these images contain a broad set of pathologies; none of them contains tumors [35].

Chapter 3

Image Segmentation

3.1 Principles

Image segmentation is one of the first steps taken towards image processing. It is an operation that, if done properly, will ease the following processing steps. Image segmentation is a process by which all pixels are classified as belonging or not to an object. The image is divided into a number of regions such that the pixels inside the region exhibit high similarity and the pixels from different regions present high contrast. Over the years, multiple approaches have been developed: threshold based, clustering, graph based, neural network approach and others, the most efficient at the present time being the clustering technique [36].

- ***Thresholding segmentation*** is the simplest developed technique. It is used to differentiate between background and foreground. This method makes use of the intensity histogram and aims to determine the threshold value that best divide the classes. Due to its simplicity it is very fast and efficient from a computational point of view, but, on the other hand, it is sensitive to noise and intensity inhomogeneity [36].

- ***Region growing segmentation*** extracts a region of pixels having similar intensities. A seed point is defined and then all neighboring pixels which have the same intensity as the seed are tagged as belonging to the same region. This method is recursively applied until no other pixel is added to the region [36].

- ***Region splitting segmentation*** considers a big region and, instead of adding to it as in the case of region growing approach, pixels which do not fulfill the condition are removed from the region. This way the image is divided into unconnected regions [36].

- ***Clustering*** is an unsupervised image classification method. It is used to classify the image into a number of clusters which can be user-defined or found by another algorithm. The main advantage is the lack of training. The pixels which fulfill some conditions are grouped together to form a cluster. A disadvantage of this technique is the need for good initial values, which plays an important role in the quality of results [36].

- ***Edge detection*** is the method that tries to find pixels from different regions by applying a gradient to the image, to detect rapid changes in intensity. The pixels from each region are grouped together and mark as belonging to the same object [36].

- ***Classification*** methods use labeled data to derive characteristics from images and then partition the space into regions with similar features [36].

3.1.1 Clustering techniques

Clustering can be defined as the partition of given data into subsets having some common properties, without any prior knowledge about the data. The similarity between two images is hard to define, yet is obvious for the human brain. To quantify this similarity between descriptors, distances are computed [37].

Being given two feature vectors $X_i = [x_{i,1}, x_{i,2}, \dots, x_{i,n}]$ and $X_j = [x_{j,1}, x_{j,2}, \dots, x_{j,n}]$, some of the most common metrics used to compute the similarity between them are [37]:

- Minkovski distance:

$$d_{Mink}(X_i, X_j) = \sqrt[r]{\sum_{k=1}^{n} |x_{i,k} - x_{j,k}|^2} \tag{3.1}$$

  By $|\cdot|$ was denoted the modulus and $x_{i,k}$ with k=1, … ,n is the attribute of the X_i instance. If $r = 1$, a particular case is obtained, the distance being called Manhattan. If $r = 2$, the Euclidian distance formula is obtained.

- Canberra distance:

$$d_{Canb}(X_i, X_j) = \sum_{k=1}^{n} \frac{|x_{i,k} - x_{j,k}|}{|x_{i,k}| + |x_{j,k}|} \tag{3.2}$$

- Distance between histogram values:

$$d_{inter}(X_i, X_j) = \sum_{k=1}^{n} \min\{x_{i,k}, x_{j,k}\} \tag{3.3}$$

  In this instance, $x_{i,k}$ represents the values from the histogram, *n* is the total number of bins and the $\min\{\cdot\}$ operator returns the minimum value.

- Distance between histograms:

$$d_{hist}(X_i, X_j) = \sqrt{(X_i - X_j)^T \cdot A \cdot (X_i - X_j)} \quad (3.4)$$

Here, X_i represents a histogram, ^{T} is the transposition operation and $\mathbf{A} = [a_{l,k}]$ is a square matrix showing the correlation between bins *l* and *k*.

More complex formulae can be used to compute the distance between descriptors, the choice depending on the input data and the desired application. At some higher level, structural or even semantic similarity can be searched. In this case, a formal representation of prior knowledge should be made; also, relationships between concepts are needed in order to facilitate the structural representation of data.

In the following subchapters, some of the most common clustering methods will be briefly presented, highlighting the advantages and disadvantages of each of them.

3.1.1.1 K-means

K-means is one of the most popular clustering algorithms and it is used to partition the input data into k clusters. Each such cluster is characterized by a centroid, defined as the point for which the sum of distances between it and any data belonging to the cluster is minimized. The algorithm is iterative, repeating until no instance changes the cluster it belongs to.

Having an image of resolution x*y, by p(x,y) are defined the pixels in this image. k denotes the number of clusters, c the cluster centers and Γ a partitioning matrix:

$$\Gamma = [\gamma_{l,i}], \quad \gamma_{l,i} = \begin{cases} 1, & X_i \in c_l \\ 0, & \text{otherwise} \end{cases} \quad (3.5)$$

The k-means algorithm steps are as follows [36][37]:

p1. The value for k is chosen

p2. The centroids for each cluster are initialized

p3. For each pixel from the image a distance (e.g. Euclidian distance) is computed with respect to all centroids

p4. The pixels are assigned to the nearest centroid

p5. The partitioning matrix is computed, Γ

p6. After all pixels have been assigned, the centroid is updated as the mean of all pixels belonging to that class

p7. The process is repeated until no pixel changes its class, that is until Γ from two successive operations does not change

Among the advantages presented by k-means algorithm, the most important are its implementation simplicity and efficiency, having a reduced complexity [37].

From the drawbacks this algorithm presents, the most notable one is the dependence of results from the proper choice of number of classes and initial values for the centroids. It also is sensitive to atypical data and it's not suitable for all distribution types of data, causing problems especially for non-convex shapes [37]. Several optimization methods have been proposed in order to eliminate these disadvantages. In [38] the authors proposed the use of the flower pollination algorithm in order to eliminate the trapping of centroids in local minima.

Outliers will have a high contribution to the centroid choice and in this case, some information might be classified as belonging to a wrong class. A simple solution is that instead of taking the mean when updating the centroid position to consider as centroid the data nearest to the mean. This way, the outliers' effect is eliminated [37].

For clusters with non-convex shape, the problem is solved by transforming the data using a kernel, such that in the new representation space data is linearly separable. Kernels that can be used for this operation are the Gaussian kernel or the polynomial kernel [37].

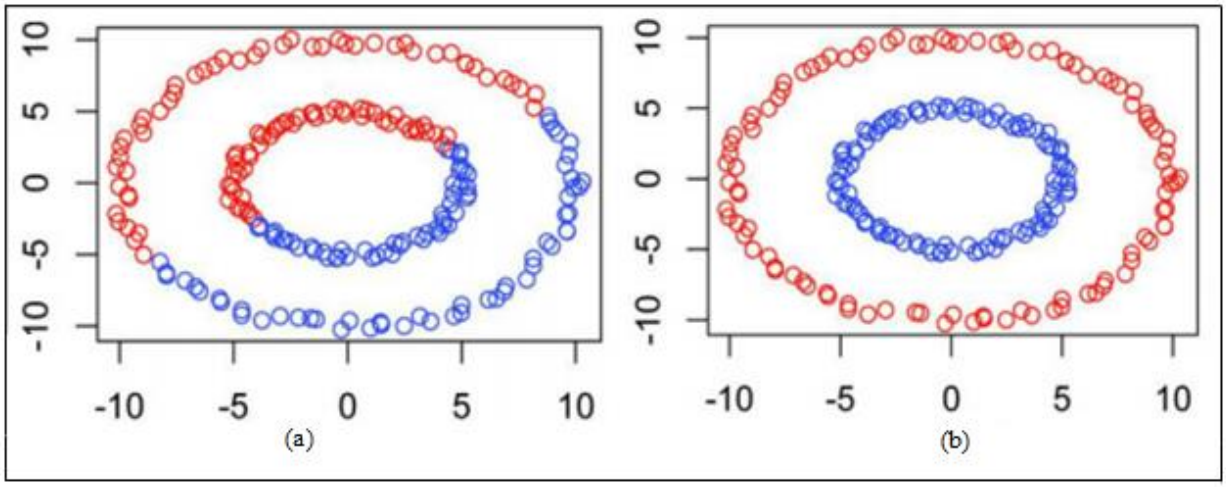

Figure 3.1 Non-convex clusters (a) before applying kernel function (b) after the application of a kernel function [37]

3.1.1.2 Gaussian Mixture Model

This approach is using models to classify data. Each class is considered to have a distribution that is mapped as a sum of Gaussian functions. The parameters of Gaussians are optimized such that to best fit the data [37].

The n-dimension Gaussian probability density function is defined as:

$$N(X;\ \mu, \Sigma): \quad f(x_1, \dots, x_k) = \frac{1}{\sqrt{(2\pi)^k \det(\Sigma)}} e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1} (X-\mu)} \tag{3.6}$$

In the (3.6) formula, $X = [x_1, \dots, x_k]$ represents a random variable having k dimensions, that is characterized by $\mu = [\mu_1, \dots, \mu_k]$, the vector containing the mean values for each dimension, and Σ, the covariance matrix. With ^{T} is denoted the transposition operation, $^{-1}$ is used to mark the invers of a matrix and the operator $\det(\cdot)$ returns the determinant of a matrix [37].

To apply this method, it is assumed that a source generates normal distributed data, of mean μ_i and covariance matrix Σ_i. The classifier determines the optimal parameters that best match the input data repartition in the feature space, after that, some optimization algorithm is applied [37].

The main advantage of this method is that a model of the data is offered and it can be used to generate new data. It is of relatively low complexity and the principles can be applied to other types of distributions. The disadvantages it has are connected to the optimization algorithm and to the fact that the number of classes should be determined a priori [37].

3.1.1.3 Fuzzy c-means algorithm

In this approach, each pixel from the image belongs to more than one class. A membership matrix is defined; its purpose is to show the percent by which a pixel belongs to a class. Data is divided into clusters by applying a specified condition. A cost function is defined, the algorithm stops when a minimum value for this function is reached [38].

The first step in this algorithm is to initialize the number of clusters and the membership matrix. After that, the center for each of the clusters is found and the cost function is computed. The obtained value is compared with a previously set threshold. If the condition is not fulfilled, the membership matrix is updated, otherwise the data is clustered and the algorithm reaches its end. This method was improved by introducing weights such that the members who have higher probabilities of belonging to a cluster have a more significant contribution when determining the center of the cluster [38].

3.1.1.4 Subtractive Algorithm

This clustering technique is making use of the density of surrounding points when computing the centroids. Having defined $X = [x_1, \dots, x_n]$, the subtractive method gives weights, called *potentials*, to each point in accordance with their probability of being centroids for each cluster, according to the formula [38]:

$$p_n = \sum_{i=1}^{n} e^{\frac{-4||x_n - x_1||^2}{r_a^2}} \tag{3.7}$$

In the formula, r_a is the radius of a hyper sphere that defines the neighborhoods, so it should be a positive constant. The Euclidian distance is defined using $||\cdot||$. After all these potentials are found, the point having the maximum value is chosen as centroid. If x_i having the potential p_i is the new cluster center, all other potential values should be updated as follows [38]:

$$p_n = p_n - p_i e^{\frac{-4||x_n - x_i||^2}{r_b^2}} \tag{3.8}$$

The newly introduced parameter, r_b, is the radius of the penalty hyper sphere. The points in the immediate vicinity of the centroid will lose more potential, this way ensuring they are not chosen as other cluster's centers. The whole process is repeated until the number of desired clusters is reached. This method can be used to determine the initial centroids in other clustering techniques [38].

3.1.2 Classification techniques

Classification techniques involve supervised classification, meaning that data is clustered based on previously learned examples [37].

3.1.2.1 k-NN

The input data is classified based on a majority vote. The number of neighbors, *k*, to be taken into account should be chosen in the beginning. The training part of the algorithm involves storing the labeled data. For classification, the distance between the data to be classified and all training data is computed. After that, the first *k* neighbors are retained and the new instance is considered to belong to the class having majority votes. In the voting process only the *k* neighbors resulting in the previous step are taken into consideration. This process repeats until all input data is classified [37].

When choosing the number of voting instances, several aspects should be taken into account. First of all, a small *k* will make the class choice sensitive to noise. On the other hand, if the value for *k* is too high, data from a considerable distance may influence the vote. As an improvement, the vote may be weighted with a value inversely proportional to the distance between the input data and the *k* voting neighbors [37].

The advantage of such an approach, besides its simplicity, is the fact that it can be applied to any data, regardless of their distribution. It was proven that with the increase of the training data, its efficiency increases [37].

A major drawback this approach presents is the selection of *k*, since the classification result is highly dependent on it. Another disadvantage is represented by the elevated mathematical complexity [37].

3.1.2.2 Support Vector-Machine (SVM)

The input data is divided into two classes by a hyperplane, whose equation is optimized such that the distance between the two classes is maximized [37].

Being given the training data, the vector normal to the hyperplane and the shift from the origin should be found. Data nearest to the hyper plane is called *support vectors*, and the distance between the support vectors and the class separating plane defines a margin. So, the optimal hyper plane to differentiate between classes will maximize this margin. If the classes are not linearly separable, data is represented in a different space, using a suitably-chosen kernel [37].

For multi-class classification, the number of SVM classifiers used is equal to the number of desired classes. The approach is one-vs-all, that is data belongs either to the *j*-th class or to any other one, without caring which one. Another possible solution is to employ a number of SVM classifiers equal to the number of combinations between any two classes. The classifiers are trained using only the data from the two corresponding classes. After the test data passes through the system, a majority vote is cast and the class having most votes will be considered [37].

The authors in [39] successfully used the SVM classifier to segment tumors in MRI scans. A Gaussian kernel was used to map data into a different representation space, where separation is possible. In their article, the SVM was used to classify regions from the brain as being tumors or healthy cells.

3.2 Setup

The chosen dataset, fastMRI, contains brain scans affected by different maladies, but without any tumors. In the early stages of the project, it was discussed to use proprietary software from *Laboratoire d'Imagerie Biomédicale Multimodale Paris Saclay* to add some tumors on the clean scans and then focus on reconstructing images of cancer patients. In order to position the tumor realistically, segmentation of the brain was needed.

The number of classes is known, since the requirement was to differentiate between white matter, gray matter, cerebrospinal fluid, skull (if visible) and background. The proposed clustering algorithm was k-means, the value of *k* being fixed to five.

The steps from subchapter 3.1.1.1 were followed for the classification. First, the DICOM image is loaded and the pixel array is saved in a two dimensional array. The centroids are initially chosen randomly. All pixels from the image are assigned to the centroid closest to them; the distance considered was the Euclidian distance. After this, the centroids are updated by computing the mean of all values assigned to it. The process is repeated until convergence or, as proven by experiments, a certain number of iterations passed. Since the centroids are randomly initialized, the convergence may never occur and in this case, a new initialization is required. Convergence is reached when the centroids do not change their position or the change is insignificant [40]. As a last step, median filtering was applied to eliminate salt-and-pepper noise.

By analyzing the data, an important aspect could be observed: there are scans where the skull or cerebrospinal fluid is not visible. Since for those scans the number of classes should be smaller, it was thought to not force the algorithm to always return five different classes. Instead, if for two consecutive runs a smaller number of clusters resulted, the image was supposed to lack one of the classes defined in the hypothesis and the segmentation result was accepted as it was.

The workflow is presented in Figure 3.2.

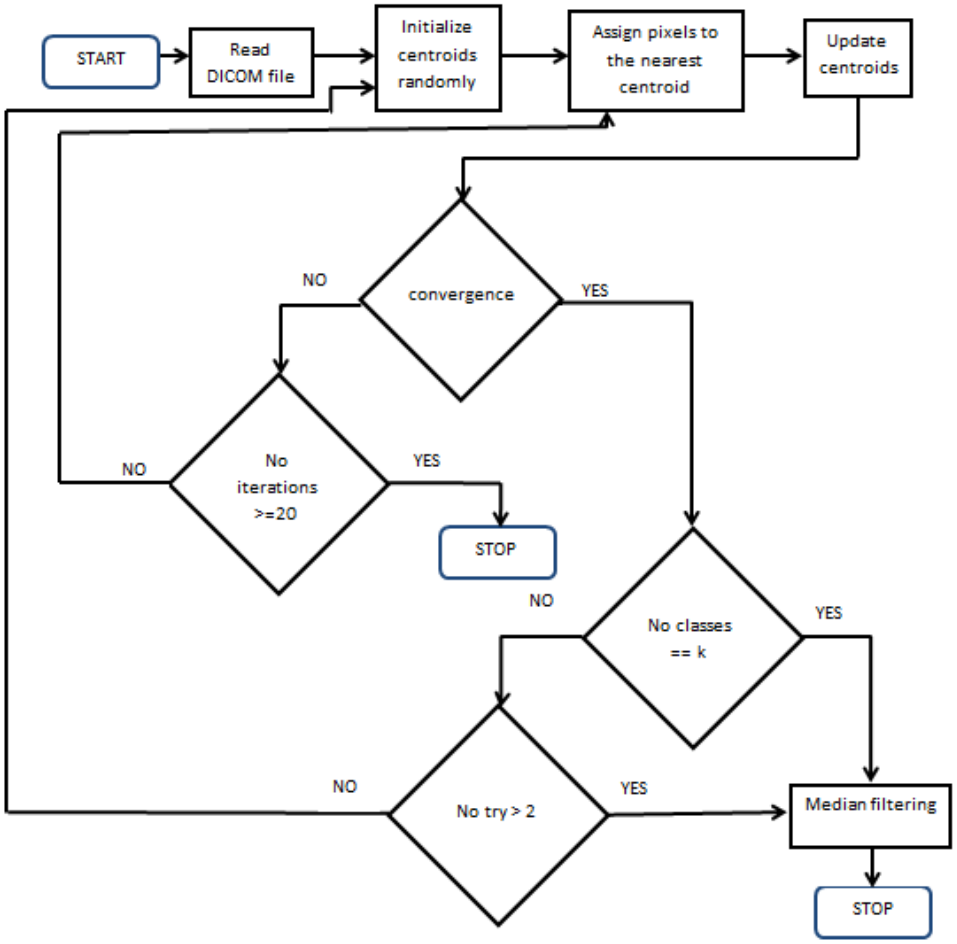

Figure 3.2 Workflow

3.3 Results

The algorithm was tested on a subset from the database, containing 335 images. Each image was segmented and the results saved. The reason for working on a smaller set of images was that results should be validated by MR specialists. Also, data annotation should be done by trained personnel.

Since this database is not segmented and no similar projects were conducted on it, the quality of results cannot be mathematically proven, as for any metrics that can be computed, ground truth information is needed. However, the SSIM between the segmented image and the original one was computed and the value obtained was 0.62, this way proving that some improvements should be considered.

Considering that any operation on an image may introduce distortions that translate into noise, a median kernel of dimension 3*3 was used. This way, impulse noise was removed from the segmented scans. The low dimension of the kernel is motivated by the desire to keep as much as possible the information unaltered. A median filter having a higher dimension would have been faster, but some details might have been lost. After applying the filter, the PSNR values were computed for the segmented image reported to the initial one. The lowest value obtained was

57.188 dB and the best one 67.009 dB. The high value of this parameter indicates good quality images, that can be further used without concerns regarding the noise.

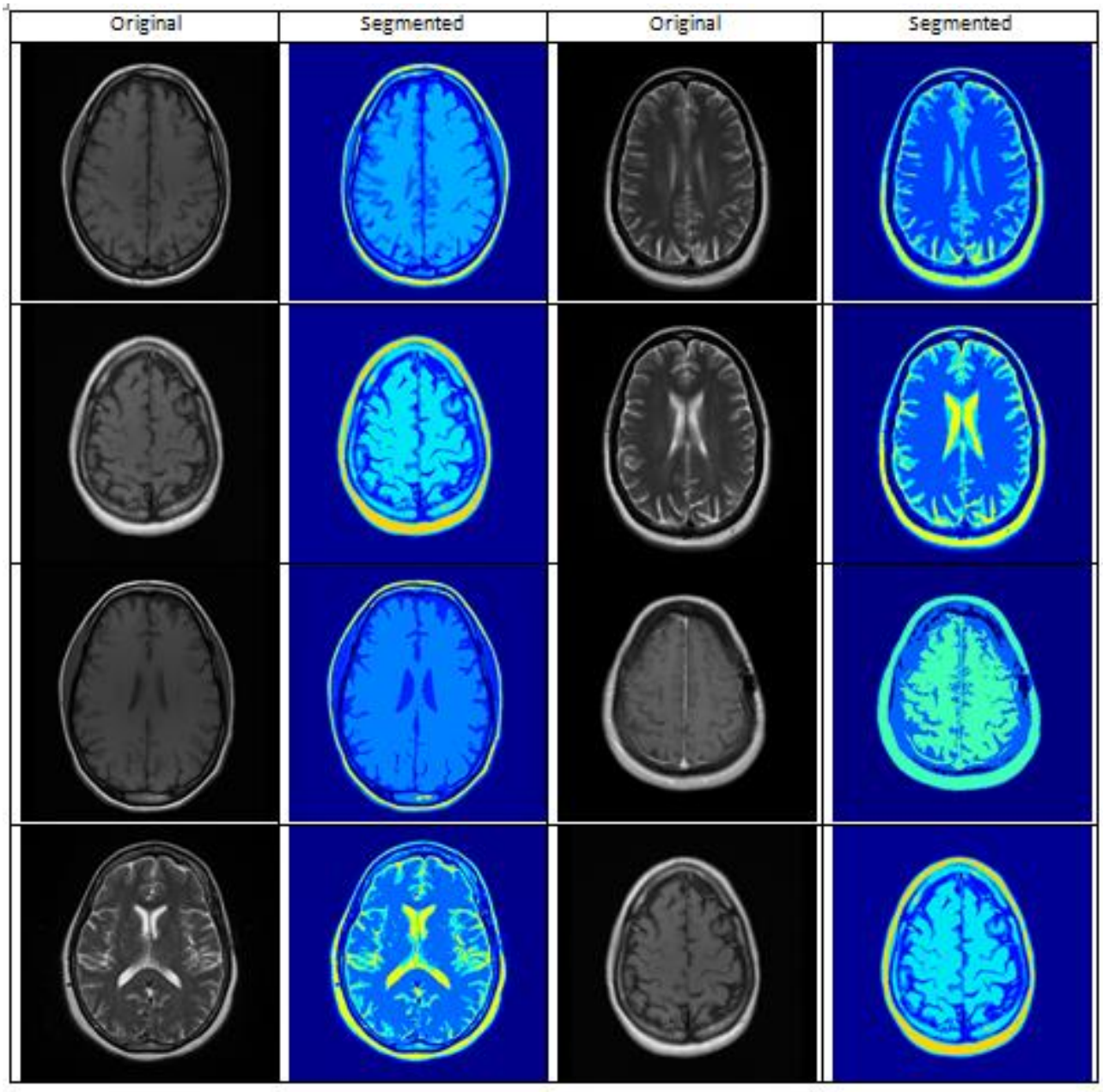

Figure 3.3 Segmentation results

MRI segmentation plays an important role in brain analysis, as it enables visualizing the anatomical structure. More than that, if segmentation is applied to scans taken at different time intervals, possible brain changes may become easily observed. The most notable aspect, however, that motivates the development of good segmentation algorithms is the desire to have a good delimitation for pathological regions, being this way of crucial importance in image-guided interventions [41].

Chapter 4

MRI Reconstruction

4.1 Data Acquisition

Since the MR scans offer good image quality, it is safe for the patient and non-invasive, its popularity has grown tremendously over the years. However, its major drawback is the long acquisition time, making it unpractical for some categories of people like young children, people suffering from claustrophobia and other such affections, or for trauma patients, where time is of the essence. In order to make it accessible to more people, it is desired to reduce the time needed to obtain good quality images. The reduced acquisition time introduces artifacts, so image reconstruction techniques should be used to obtain the final scan as close as possible to the original image, taken if the whole period of time necessary to obtain state of the art quality with today's technology is considered [42].

For the development of this thesis, fastMRI dataset was considered. The reason behind this choice is motivated by the desire to have a sufficient number of images of high quality, taken with different scanners. As briefly mentioned in Chapter 2, the DICOM images were obtained using five different scanners, making them ideal for the proposed task. In this thesis only the brain scans were considered, both T1 and T2 weighted. The database contains axial slices of the brain, at different positions in the 3-dimension scan.

All processes were run on a GPU cluster consisting of two Cuda Tesla M2070 devices and having 32GB RAM.

4.1.1 Training data

The data selected from the whole fastMRI database was divided into training and testing data as follows: 80% for training and 20% for testing the algorithm. Multiple tests were carried out, varying the training parameters, but also the database's dimension, as it is desired to have good results even for a small number of images in the database.

Since the acquired images are of high quality, they were considered the ground truth. Noisy images were not provided, so the need to choose the type of noise to apply to them emerged. After extensive research, it was decided to introduce Gaussian noise in the images. As discussed in Chapter 2, MR images contain noise that is Rician distributed. However, as the signal to noise ratio increases, this distribution can be well approximated by a Gaussian one. Intensive research has been conducted on eliminating Rician noise form MRIs, but only few articles concentrate on Gaussian or Rayleigh noise in this type of scans. So, in this thesis, the accent was placed on reconstructing MRI scans from images affected by Gaussian noise. To obtain the complete database necessary to perform the training task, the images were corrupted by adding standard normal noise. The Gaussian distribution which characterizes the introduced noise is described by the parameters $\mu = 0$ and $\sigma = 1$. The amplitude of the noise was also varied in order to test the effects a lower initial SNR has on the reconstruction result.

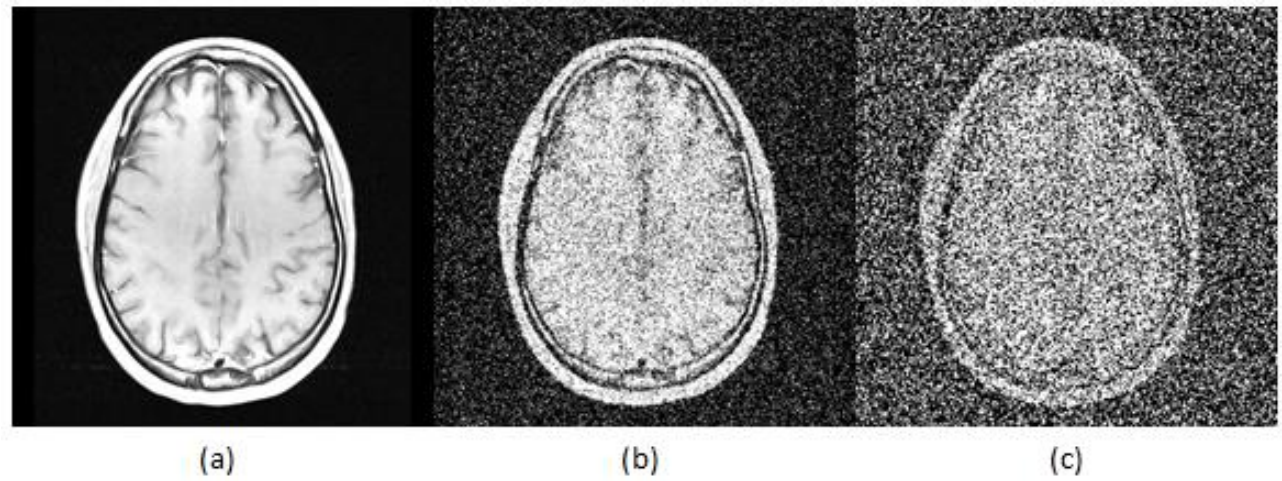

Figure 4.1 MRI scan (a) clean (b) Gaussian noise, noise factor = 0.1 (c) Gaussian noise, noise factor = 0.9

Another test that was carried out was the reconstruction of scans affected by Rayleigh distributed noise. As mentioned before, Rayleigh distribution is a particular case of the Rice one that occurs when only noise is present in the scan. Figure 4.2 shows an example of MRI scan before and after applying Rayleigh distributed noise over it.

Test dataset contains 20% of the total data. The same type of noise at the same amplitude as the ones for the train data was used to distort the test images. In the ideal case, a network trained to reconstruct images from their noisy representations should be able to remove different levels of noise, but practical applications demonstrate that this task is of elevated complexity if good results are desired.

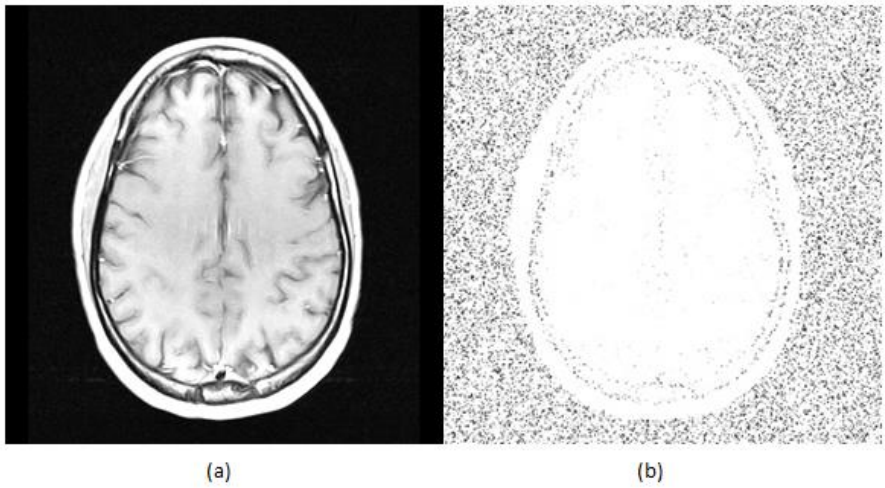

Figure 4.2 MRI scan (a) clean and (b) affected by Rayleigh noise

4.2 Network

To complete the task, a convolutional autoencoder was used. An autoencoder is a feed-forward network that tries to copy the input at the output. The input data is compressed and then reconstructed at the output. The reduced dimensions decrease the runtime and the memory used, making autoencoders attractive in applications involving large datasets [43].

The autoencoder consists of two parts:

- Encoder: compresses the input information and is represented in the following by $\mathbf{h} = f(\mathbf{x})$
- Decoder: reconstructs the input and is represented by $\mathbf{r} = g(\mathbf{h})$

The whole autoencoder is described by the relation $\mathbf{r} = g(f(\mathbf{x}))$. By **x** is denoted the input in the network and by **r** the reconstructed image. Function f is used to map the input into a latent representation **h** and is called encoder, while function g is the decoder and it is using the information offered by **h** to create a reconstruction of the input data, denoted by **r** [43].

In order for the autoencoder to learn, **h** should have smaller dimensions than **x**. This way, the autoencoder is forced to learn the most important features from the available data. If this would not be the case, the output would be very similar to the input, as the network extracts little useful information from the data [44]. An autoencoder having this property is called undercomplete [43].

A denoising autoencoder is fed with corrupted data, forcing the network to learn features from the image rather than the image itself. In order to learn about the input data, it will try to remove the noise so that it will be able to correctly reconstruct the image [43].

An autoencoder containing also a convolutional layer is called convolutional autoencoder. In deep learning, the desired operation is commonly the cross-correlation, as it involves fewer operations [43]. This type of autoencoder proves advantageous in image processing since the convolution operation exploits the image structure [45].

Figure 4.3 presents the structure of the autoencoder.

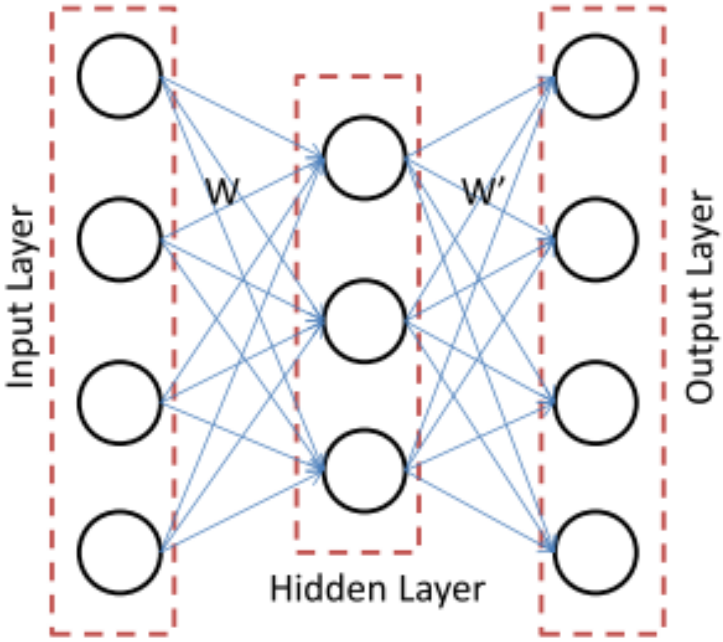

Figure 4.3 Autoencoder structure [46]

Having the training input **x**, the autoencoder maps it to a latent representation by applying a non-linear function to it, s:

$$\boldsymbol{h} = s(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{4.1}$$

The latent representation obtained after the application of s to the input **x** is then used to obtain the reconstruction **r**:

$$\boldsymbol{r} = s(\boldsymbol{W}'\boldsymbol{h} + \boldsymbol{b}') \tag{4.2}$$

The reconstruction has the same shape as the input. By $\boldsymbol{W}, \boldsymbol{W}', \boldsymbol{b}\ and\ \boldsymbol{b}'$ were denoted the model parameters [45].

4.3 Design and implementation

A good implementation of a network depends on the optimal choice of some design parameters. In this thesis, the variation of image quality with the variation of such parameters is studied.

Stochastic gradient descent is the optimization algorithm generally used in the training phase. Adam optimization algorithm is an extension of the stochastic gradient descent. The mean and the uncentered variance of the gradient are used in computing each weight from the network, by adapting the learning rate [47].

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{4.3}$$

$$\gamma_t = \beta_2 \gamma_{t-1} + (1-\beta_2) g_b^2$$

$$E[m_t] = E[g_t] \tag{4.4}$$

$$E[\gamma_t] = E[g_t^2]$$

In formulae (4.3), m and γ represent the moving averages, β_1 and $\beta_1$are the hyper parameters of the algorithm and g is the gradient on the current batch. Because m and γ are estimates for the first, respectively second moments, the relations in (4.4) should be fulfilled. As t increases, the effect the earliest estimates have becomes less and less important. An equivalent, more compact writing of (4.3) is:

$$m_t = (1 - \beta_1) \sum_{i=0}^{t} \beta_1^{t-i} g_i \tag{4.5}$$

$$\gamma_t = (1 - \beta_2) \sum_{i=0}^{t} \beta_2^{t-i} g_i^2$$

By imposing the condition from (4.4), to obtain the correct estimator, bias correction should be done. So, the formulae for the estimator are:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \tag{4.6}$$

$$\widehat{\gamma_t} = \frac{\gamma_t}{1 - \beta_2^t}$$

The values obtained are used to update the weights, **w**. In this operation, the step size η should be also taken into account [47]:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{\widehat{m_t}}{\sqrt{\widehat{\gamma_t}} + \epsilon} \tag{4.7}$$

A *sample* is an instance of the data that contains both the input and the output that should be compared to the input in order to obtain the error [48].

The *batch size* is the hyperparameter that shows how many samples should be considered before making an update to the parameters. The *number of epochs* is a hyperparameter whose task is to mark the number of times the entire input dataset passed through the algorithm [48]. An *iteration* is defined as the number of batches that form an epoch.

If the batch size is low, the update operation will be more frequent, increasing this way the complexity. On the other hand, if the batch size is too big, the updates will not occur with a frequency that allows the algorithm to actually learn something. A compromise between the two should be made [43].

The learning rate should also be carefully chosen. If this parameter is too small, the optimizer might remain stuck in a local minima. If the value considered for it is, on the contrary, too big, the global minima might be missed [43].

In this thesis, the optimizer used was Adam. For its hyperparameters it was decided to take the learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e-7$. The batch size parameter was varied;

the values taken for it were 2, 4, 16 and 32. The number of epochs considered was 50,100, 200 and 500.

The autoencoder developed consists of the following layers: convolutional layer, using a kernel with dimension 3*3 and stride 2. After that, leaky ReLU was used, with parameter alpha = 0.2. The results were normalized and scaled. In the end, the dimensions are flattened and a wholly connected layer is added. For the decoder part, deconvolution layer with a kernel of dimension 3*3 and stride 2 is applied, then leaky ReLU with alpha = 0.2 and normalization. Another deconvolution layer is added, to bring the image at the original depth. In case of gray scale scans, as in this application, this layer is not necessary, but it was still introduced in case colored maps should used. In the end, the activation function used is the sigmoid [49].

4.4 Experimental results

4.4.1 Performance metrics

In order to measure the performance of the algorithm, the performance metrics used are: PSNR and structural similarity index (SSIM).

PSNR is a quality measurement between two images: as its value increases, the quality of the reconstruction increases. In order to compute it, the MSE is needed. The mean square error represents the cumulative error between the reconstructed image and the original. Being given two images I_1 and I_2 having the dimensions (M,N):

$$\mathrm{MSE} = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M \cdot N} \tag{4.8}$$

$$PSNR = 10log_{10}\left(\frac{R^2}{MSE}\right) \tag{4.9}$$

In the PSNR formula, by R was denoted the maximum variation of the input data.

SSIM is a measure of similarity between images and to obtain its value, the luminance, contrast and structural terms should be computed:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4.10}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4.11}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{4.12}$$

In the expressions above, $\mu_x, \mu_y, \sigma_x, \sigma_y$ and σ_{xy} represent the local means, variances and covariance for the two images and C_1, C_2 and C_3 are regularization constants. Exponents are given for the luminance, contrast and structural terms and so the SSIM is obtained using the formula:

$$\mathrm{SSIM}(x,y) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma} \tag{4.13}$$

4.4.2 Results

The quantitative results prove a good quality of the reconstructed images. This quality is dependent on the chosen hyperparameters and the noise factor used to distort the images. As mentioned before, multiple tests were carried in order to verify the dependence of reconstruction on various parameters.

Initially, a subset from the database was considered, containing both T_1 and T_2 weighted scans. The subset consisted of 339 scans, divided into training and testing images.

For the first test, the scans were not distorted at all. The chosen batch size was 32 and the number of epochs was 100. The PSNR value obtained was 21.57 and the SSIM was 0.726, so a serious degradation is suggested. As it can be observed in Figure 4.4, presenting the original and the reconstructed scans, details are lost in the reconstruction process.

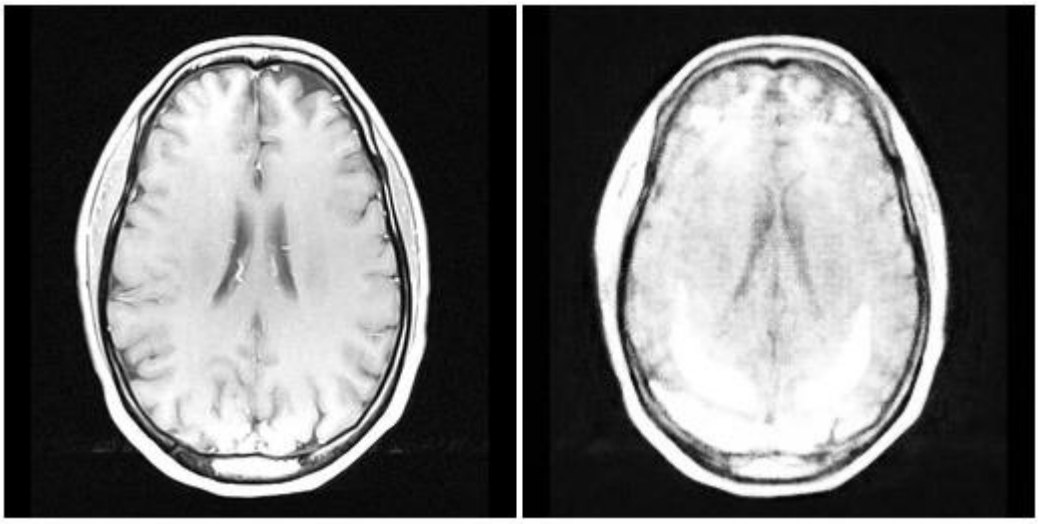

Figure 4.4 Reconstruction for scans unaffected by noise

All images in the subset were distorted using Gaussian noise of mean zero, variance equal to one and a noise factor of 0.1. Keeping the number of epochs to 100, the batch size was varied, taking the values 2 and 32. Results are reported in Table 4.1.

Batch Size	PSNR [dB]	SSIM
2	24.988	0.770
32	22.644	0.773

Table 4.1 Quality variation with the variation of the batch size

As it can be seen in Figure 4.5, the reconstructed image preserves the structural integrity of the original one. However, the noise is not completely removed. If the batch size is increased to 32, the quality of the reconstruction decreases.

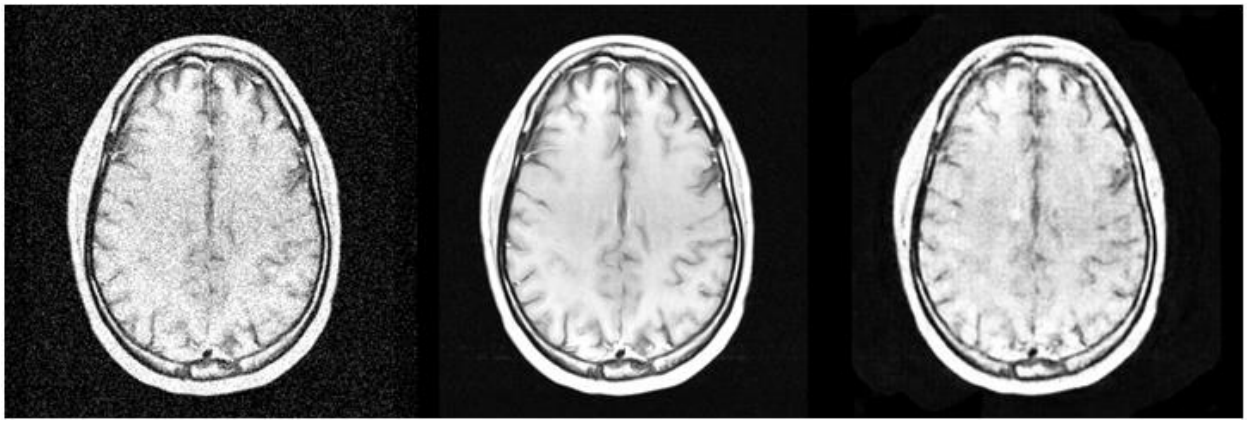

Figure 4.5 Reconstruction results for batch size = 2, 100 epochs

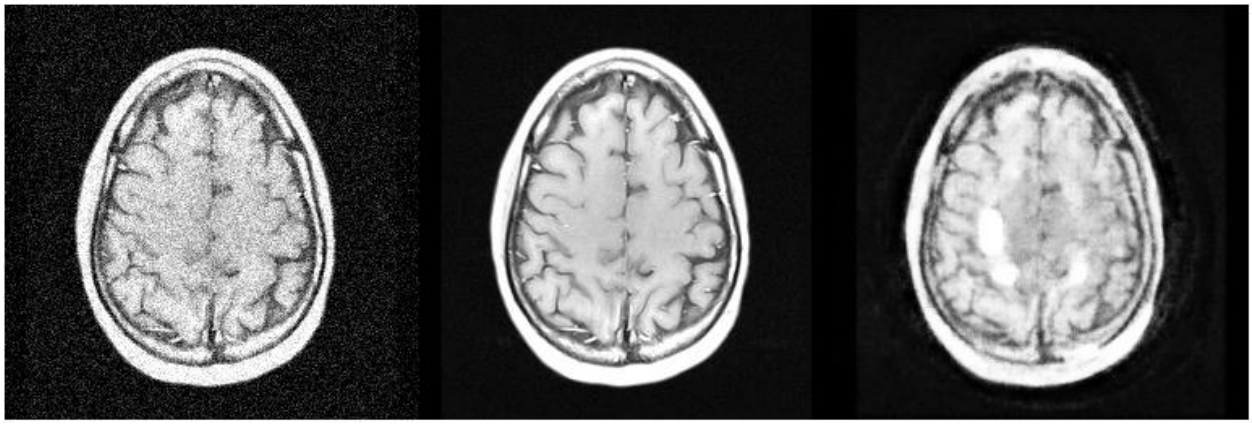

Figure 4.6 Reconstruction results for batch size = 32, 100 epochs

For the next test, the noise factor was increased to 0.9. The same noise distribution was kept, the number of epochs considered was 100 and the batch size was 32. In this case, PSNR = 18.957 dB and SSIM = 0.694.

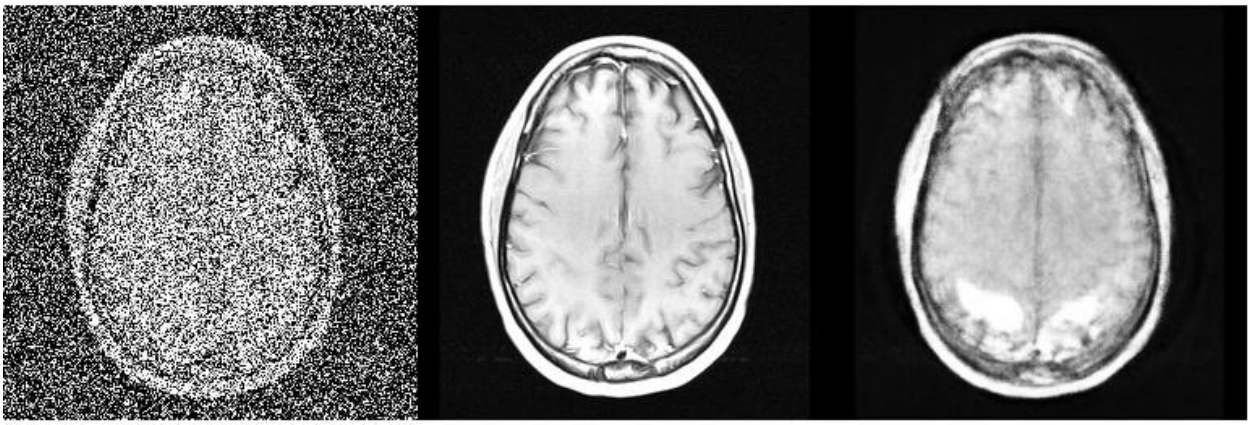

Figure 4.7 Reconstruction results for batch size = 32, 100 epochs, noise factor=0.9

As proven by Figure 4.7, the results obtained if the input image is distorted by such a powerful noise are of poor quality. The PSNR for the distorted image is 6.650 dB and the SSIM between the noisy input and the reference image is 0.060. However, even if the quality greatly improved in comparison with the input, most details are not preserved in the reconstruction, making the output unusable in real life applications.

To further test the quality of results, the noise factor was decreased at 0.5. Also, the batch size was decreased to 4. In this case, the PSNR was 24.719 dB and the SSIM was 0.799. As it can be seen in Figure 4.8, structural details are preserved. However, some noise is still present in the reconstructed scan, suggesting that for an input distorted by a significant noise the quality of results will slightly decrease.

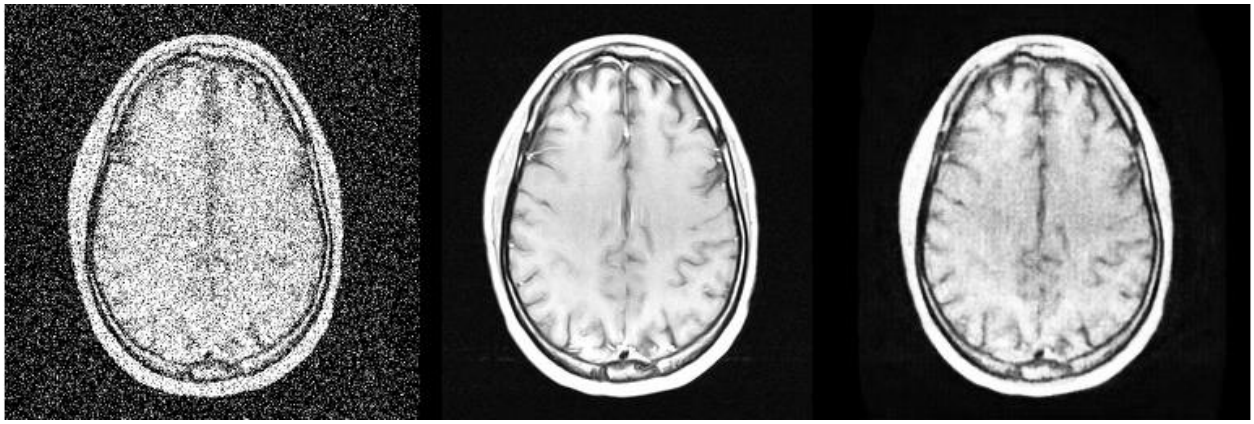

Figure 4.8 Reconstruction results for batch size = 4, 100 epochs, noise factor=0.5

In order to verify the effect that the increase of the number of epochs will have on the results, the batch size was maintained at 4 and the noise factor at 0.5. The number of epochs was increased to 200 and the results obtained are: PSNR = 25.653 dB and SSIM = 0.827.

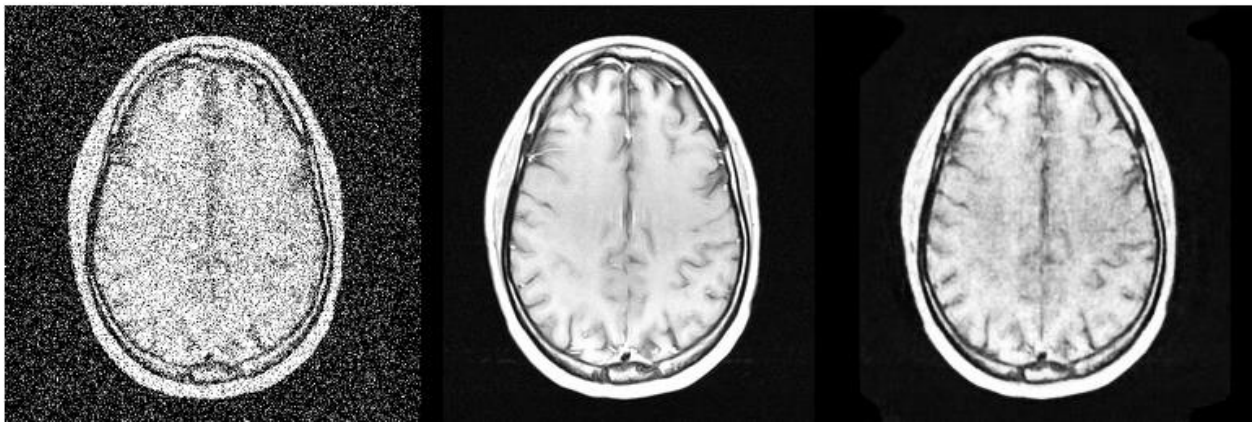

Figure 4.9 Reconstruction results for batch size = 4, 200 epochs, noise factor=0.5

The same dataset was distorted using a Rayleigh distribution. The number of epochs was kept to 200 and the batch size at 4. The SSIM between the original image and the reference one was 0.196. After passing the scan through the network, the SSIM was 0.698, while the PSNR remained quite low, at only 18.856 dB.

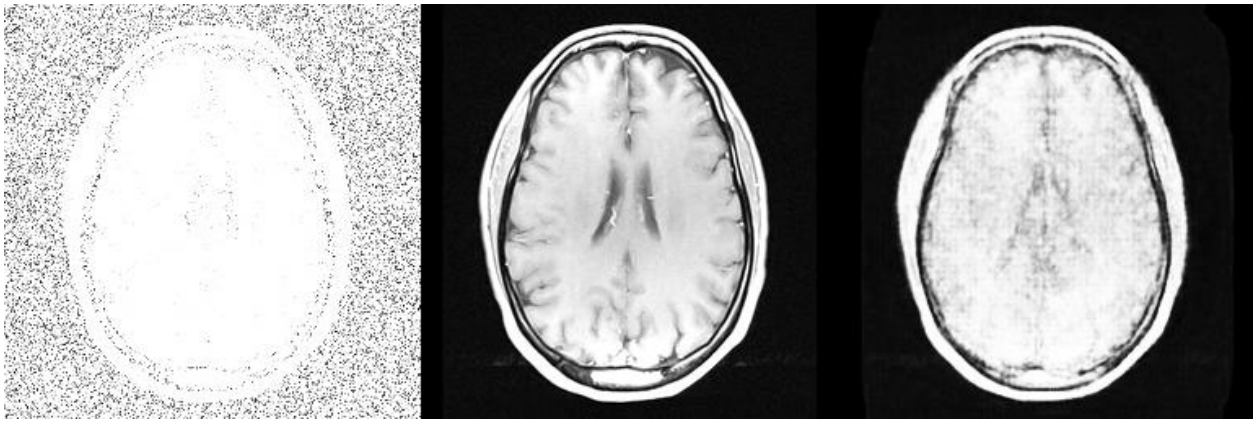

Figure 4.10 Reconstruction results for scans affected by Rayleigh noise, batch size = 4, 200 epochs

The very poor quality of the input increases the difficulty for the task of reconstruction. Little detail is reconstructed, proving that this approach is not suitable if the task involved Rayleigh noise.

To test the results for the whole database, the network was trained for 100 epochs, having the batch size 4. The input scans were distorted using Gaussian noise of mean 0 and variance 1, having the noise factor 0.3. In this case, the PSNR obtained was 31.818 dB and the SSIM was 0.820.

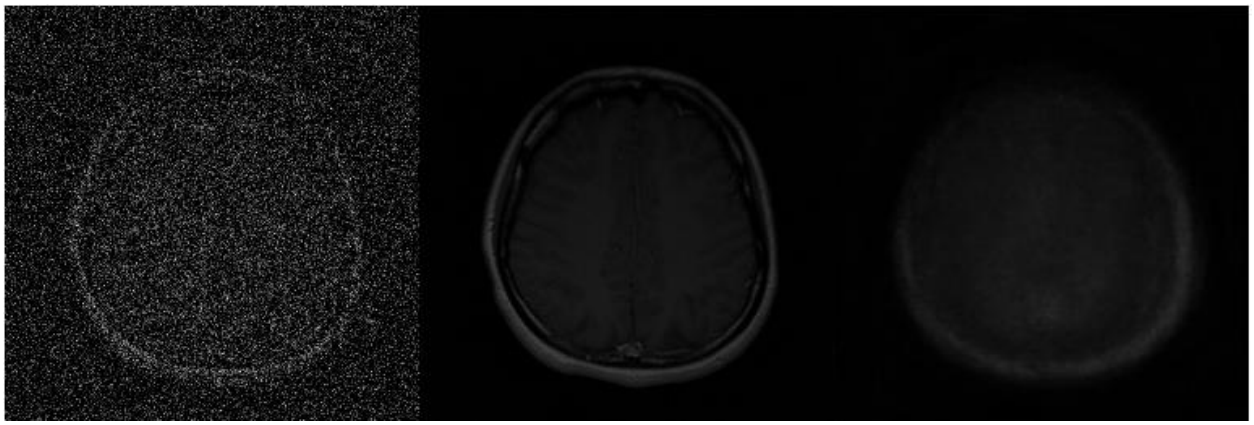

Figure 4.11 Reconstruction results for batch size = 4, 100 epochs, noise factor 0.3

Even though the PSNR obtained in this case competes with state-of-the-art results, the structural similarity could be improved. For this test, both T_1 and T_2 weighted scans were used. In an effort to increase the similarity between the reconstructed and reference scan, the training was repeated with the same parameters, but this time considering only the T_1 weighted scans. In this case, the PSNR obtained was 31.788 dB and the SSIM slightly increased as compared to the previous case, reaching a value of 0.843.

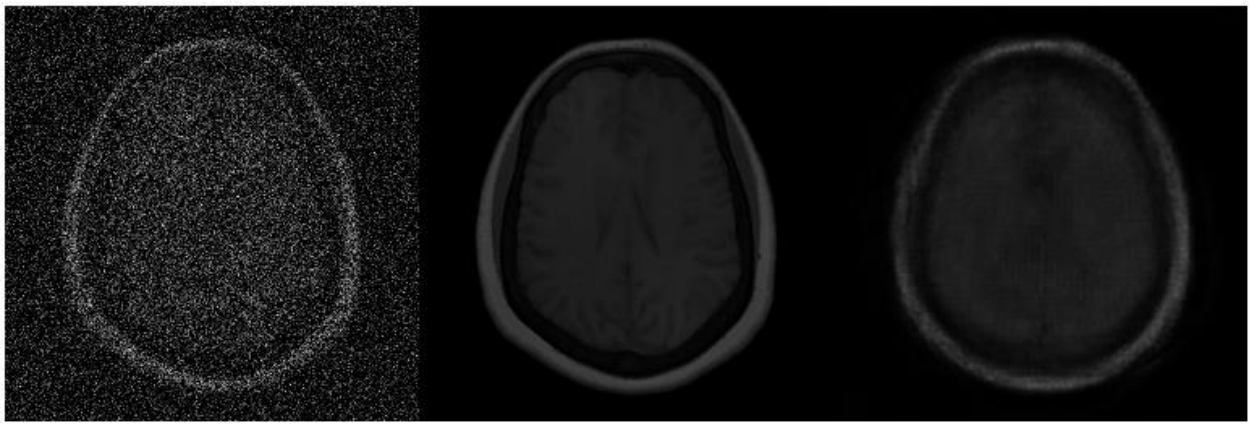

Figure 4.12 Reconstruction results for batch size = 4, 100 epochs, noise factor 0.3, only T_1 weighted scans

The following graph represents the learning curves for this last case.

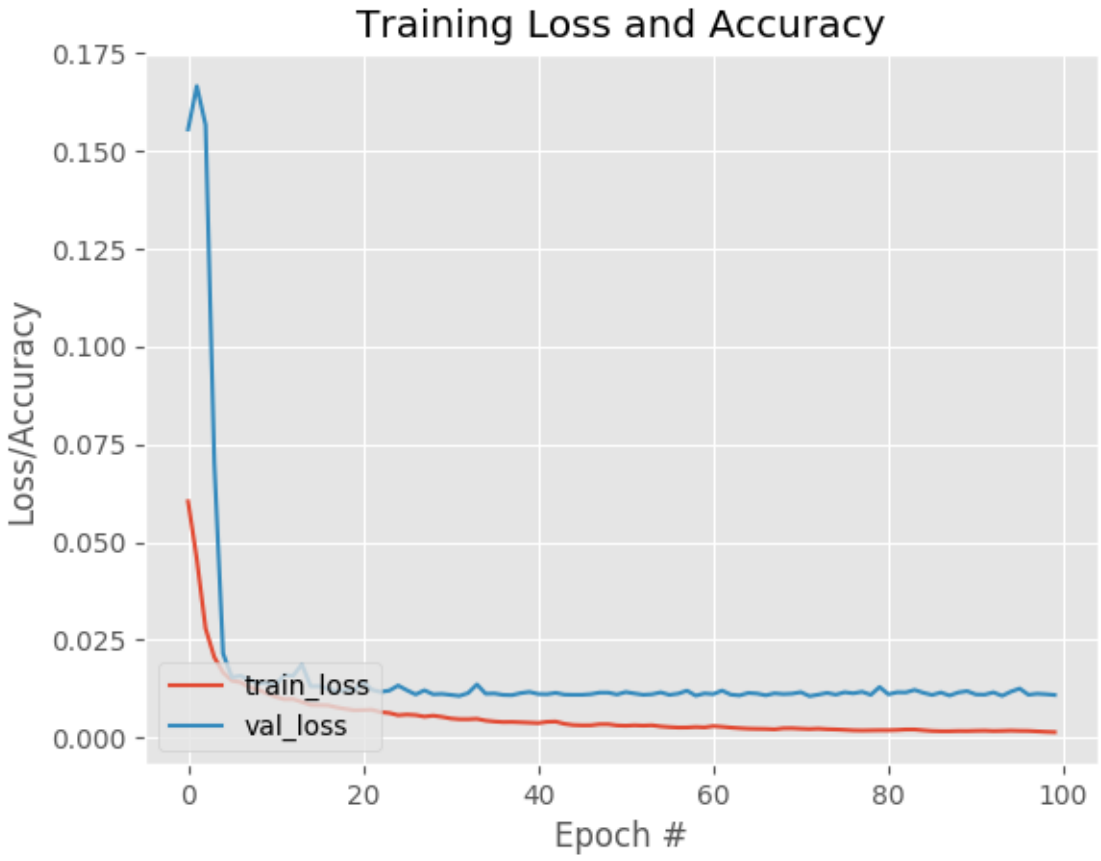

Figure 4.13 Learning curvess

For the subset containing only T1 weighted scans, affected by Gaussian noise of mean 0, variance 1 and noise factor 0.3, another test was conducted. Keeping the batch size at 16, the training was

repeated for 50, 100 and then for 500 epochs, to see the effect it has on the overall result. The obtained values are reported in Table 4.2.

epochs	PSNR [dB]	SSIM
50	33.239	0.775
100	33.951	0.835
500	29.454	0.802

Table 4.2 Quality variation with the number of epochs

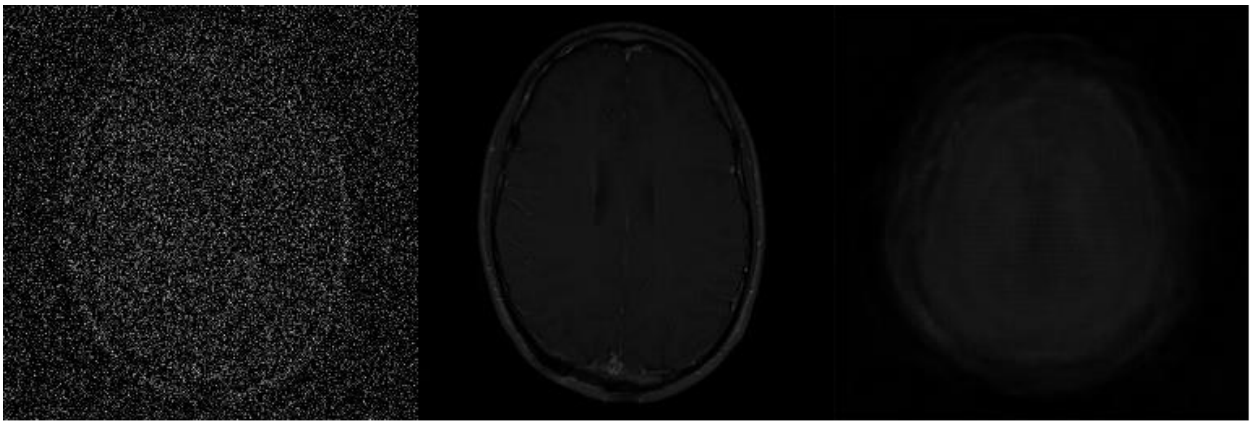

Figure 4.14 Reconstruction results for batch size = 16, 50 epochs, noise factor 0.3, only T_1 weighted scans

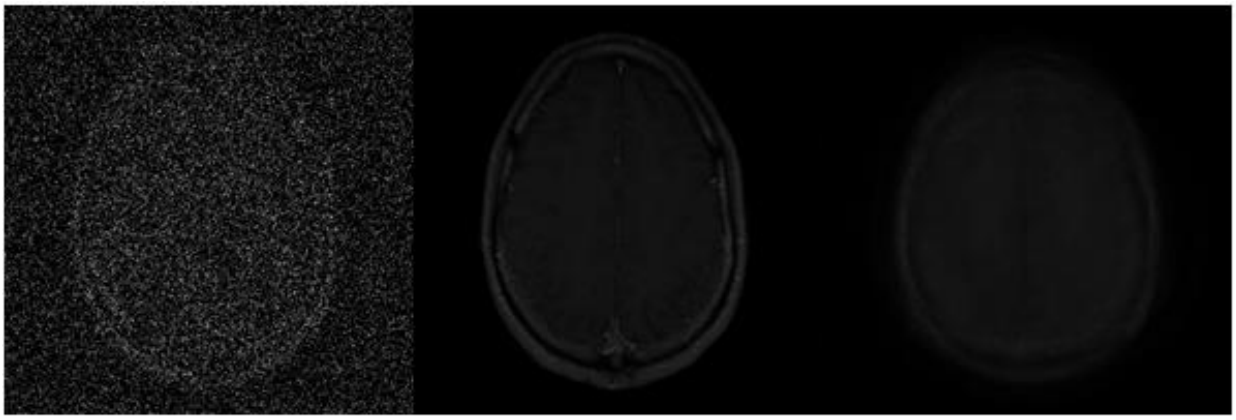

Figure 4.15 Reconstruction results for batch size = 16, 100 epochs, noise factor 0.3, only T_1 weighted scans

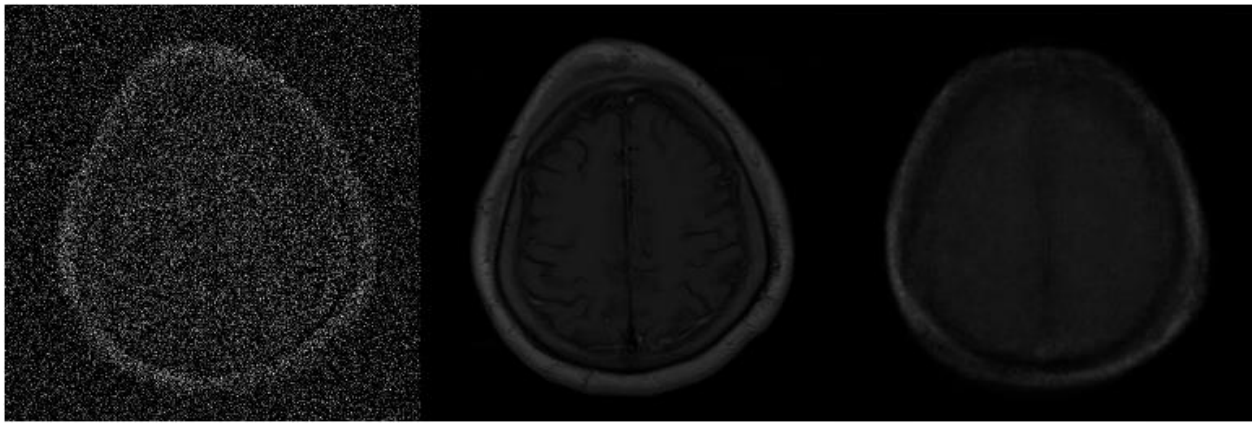

Figure 4.16 Reconstruction results for batch size = 16, 500 epochs, noise factor 0.3, only T_1 weighted scans

A degradation of results can be observed. This is the caused by underfitting, respectively overfitting of the model. An underfit model occurs when the network did not learn sufficiently from the training data, while an overfit model will have a reduced capacity to generalize to the new data.

The authors of [42] conducted experiments on the same database, but on both knee and brain scans. They tested their algorithm against state-of-the art approaches for MRI reconstruction. Their results, along with the ones obtained in this thesis are presented in Table 4.3. Since they considered the whole database, regardless of the weight of the scans, the results considered are for the same approach.

Network	PSNR [dB]	SSIM
Zero-filled	29.61	0.657
KIKI-net	31.38	0.712
U-net	31.78	0.720
Cascade-net	31.97	0.719
PD-net	**32.15**	0.729
AE	31.82	**0.820**

Table 4.3 Comparison between methods [42]

As it can be seen, the proposed method manages to offer the highest SSIM. The PSNR is not as high as the one obtained by other methods, but the value is close enough to be considered of similar quality.

The last experiment performed involved the image reconstruction from the segmented scans. Gaussian noise of mean 0, variance 1 and noise factor 0.1 was considered the distorting factor. The network was trained for 100 epochs, the batch size being chosen 16. In this case, the average SSIM obtained was 0.564. Figure 4.17 presents the reconstruction result for this special case.

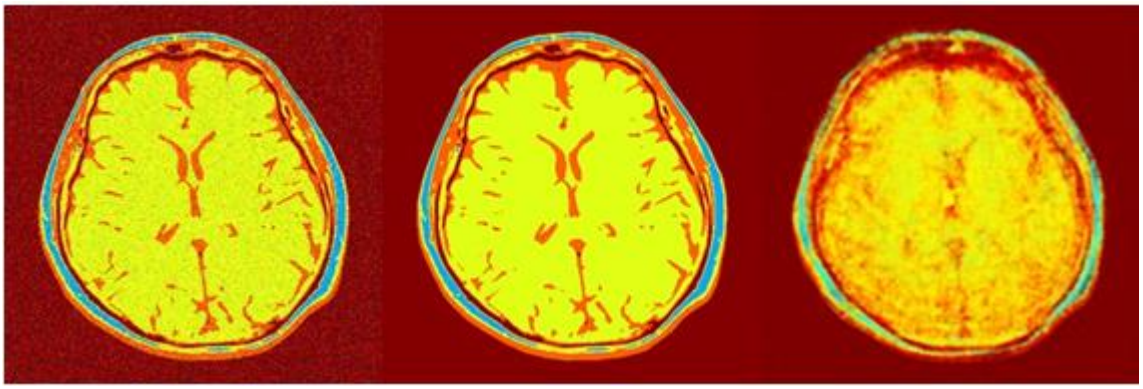

Figure 4.17 Reconstruction result for segmented scans, batch size = 16, 100 epochs

The reconstruction results obtained are worse than the ones obtained for the gray scale images, having the same training parameters. A factor that leads to this case is the dimension of the training dataset; as mentioned in Chapter 3, the segmentation was applied on a reduced number of scans.

Conclusions

General conclusions

In the last few years, the utilization of deep learning strategies for medical applications increased dramatically. A focus on increasing the image quality has been observed and medical images benefit enormously from this rise in interest. Indeed, a better quality for such scans can assist medical doctors to give a better diagnostic to patients. The goal is to process the input image in such a way as to obtain a very high quality reconstructed output. However, in the medical filed the challenge with applications that involve putting a diagnosis is the intolerance for mistakes.

Although good results were obtained using mathematical implementations, the lack of flexibility poses major problems. Indeed, studies have shown the difficulty in mapping a large database using traditional approaches. Still, the robustness of such methods makes them desirable if the input data allows their use.

The Artificial Intelligence techniques changed the approach regarding image processing. Since results in present time for object detection, face recognition, gesture recognition, image restoration applications are promising, offering state-of-the art accuracies of over 95%, today's tendency is to use such solutions also for medical images. Despite that, almost all these high-performance classifiers are very resource-consuming, using very complex architectures as DNNs or CNNs. This complexity makes them difficult to integrate in real-time applications.

Contributions

Among the author's contributions, the most notable are:

- Selection of a suitable database and analysis for the most realistic corruption mechanism in such images.

- Implementation of a segmentation algorithm that can be used to obtain the interest regions in the brain. The requirement for this task was to have the smallest complexity possible.

- Implementation for the Autoencoder
- Analysis for quality variation with some parameters

Future work

This work has proven that neural networks with a simple structure can be successfully used in the image reconstruction task. However, as shown in the thesis, the results could be further improved to reach medical image imposed standards.

An important part of the future work consists in analyzing the segmentation results and improving the method such as to obtain the best classification. The justification for continuing with this task is represented by the important applications it could have, as it enables visualizing the anatomical structure of the brain and also brain changes which can be a lot easier detected than from the whole image. Also, another direction for the future work that requires a good segmentation for the brain scan is represented by the classification of regions depending on the presence of a tumor. A good classification with this regard can be of invaluable help to medical doctors.

Another objective is to optimize the method and propose new approaches. Different neural networks can be considered for this task, each introducing its own benefits that should be further exploit in order to have the best reconstruction results.

In the end, the principles from this thesis could be used for applications involving other types of medical images, the goal being to have minimum changes in the workflow.

References:

[1] Haimiao Zhang, Bin Dong, A Review on Deep Learning Medical Image Reconstruction, June 2019.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intel., 39(12):2481–2495, 2017.

[3] ThorstenMBuzug. Computed tomography: from photon statistics to modern cone-beam CT. Springer Science & Business Media, 2008.

[4] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. Physica D, 60(1):259–268, 1992.

[5] Luis Alvarez and Luis Mazorra. Signal and image restoration using shock filters and anisotropic diffusion. SIAM J. Numer. Anal., 31(2):590–605, 1994.

[6] Stephane Mallat. A Wavelet Tour of Signal Processing, The Sparse Way. Academic Press, Burlington,MA, third edition, 2009.

[7] https://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamentals-neural-networks/ accessed on 01/17/2020

[8] Cosmin Toca. Interfete Vizuale Om-Masina. Lecture notes. 2020

[9] Facundo Bre, Juan M. Gimenez, Víctor D. Fachinotti. Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks. Energy and Buildings 158, November 2017

[10] http://mathworld.wolfram.com/SigmoidFunction.html , accessed on 01/17/2020

[11] Giovanni Barrero. PREDICCION DE LA CALIDAD DE SOFTWARE DESARROLLADO EN IBM RPG USANDO DEEP LEARNING, 2018

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. arXiv:1409.0575, 2014.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. IEEE, December 2016

[14] https://mc.ai/resnet-architecture-explained/ , accessed on 01/15/2020

[15] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In NeurIPS, pages 1137–1144, 2007.

[16] Imran Salam. Autoencoders – Guide and Code in TensorFlow 2.0. RedBuffer, August 2019

[17] Hu Chen, Yi Zhang, Mannudeep K Kalra, Feng Lin, Yang Chen, Peixi Liao, Jiliu Zhou, and Ge Wang. Low-dose CT With a Residual Encoder-Decoder Convolutional Neural Network. IEEE Trans. Med. Imaging, 2017.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015.

[19] Kyong Hwan Jin, Michael T. Mccann, Emmanuel Froustey, and Michael Unser. Deep convolutional neural network for inverse problems in imaging. IEEE Trans. Image Process., 2017.

[20] Jian Sun, Huibin Li, Zongben Xu, Yin Yang. Deep ADMM-Net for compressive sensing MRI. NeurIPS, 2016.

[21] Jonas Adler and Ozan Oktem. Learned primal-dual reconstruction. IEEE Trans. Medical Imaging, 2018

[22] Bin Dong, Jia Li, and Zuowei Shen. X-ray CT image reconstruction via Wavelet frame based regularization and Radon domain inpainting. Journal Scentific. Computing, 2013.

[23] Ruohan Zhan and Bin Dong. CT image reconstruction by spatial-Radon domain data-driven tight frame regularization. SIAM Journal Imaging Scientific, 2016.

[24] Dufan Wu, Kyungsang Kim, Bin Dong, Georges El Fakhri, and Quanzheng Li. End-to-end lung nodule detection in computed tomography. Lecture Notes in Computer Science book series (LNCS, volume 11046), 2018.

[25] https://camelyon16.grand-challenge.org/Evaluation/ , accessed on 01/17/2020

[26] Grover VP, Tognarelli JM, Crossey MM, Cox IJ, Taylor-Robinson SD, McPhail MJ. Magnetic Resonance Imaging: Principles and Techniques: Lessons for Clinicians. J Clin Exp Hepatol. 2015;5(3):246-255. doi:10.1016/j.jceh.2015.08.001

[27] http://mriquestions.com/proton--nucleus--spin.html , accessed on 06/03/2020

[28] https://www2.cs.sfu.ca/~stella/papers/blairthesis/main/node11.html, accessed on 06/05/2020

[29] https://teachmeanatomy.info/the-basics/imaging/magnetic-resonance-imaging-mri/ , accessed on 06/05/2020

[30] https://www.radiologyinfo.org/en/info.cfm?pg=bodymr , accessed on 06/05/2020

[31] Goyal B, Dogra A, Agrawal S, Sohi B. S. Noise Issues Prevailing in Various Types of Medical Images.Biomed Pharmacol J 2018;11(3).Available from: http://biomedpharmajournal.org/?p=22526

[32] Gudbjartsson H, Patz S. The Rician distribution of noisy MRI data [published correction appears in Magn Reson Med 1996 Aug;36(2):332]. Magn Reson Med. 1995;34(6):910-914. doi:10.1002/mrm.1910340618

[33] https://en.wikipedia.org/wiki/Rice_distribution#/media/File:Rice_distributiona_PDF.png , accessed on 06/07/2020

[34] https://fastmri.med.nyu.edu/ , accessed on 06/08/2020

[35] Zbontar, Jure & Knoll, Florian & Sriram, Anuroop & Muckley, Matthew & Bruno, Mary & Defazio, Aaron & Parente, Marc & Geras, Krzysztof & Katsnelson, Joe & Chandarana, Hersh & Zhang, Zizhao & Drozdzal, Michal & Romero, Adriana & Rabbat, Michael & Vincent, Pascal & Pinkerton, James & Wang, Duo & Yakubova, Nafissa & Owens, Erich & Murrell, Tullie. (2018). fastMRI: An Open Dataset and Benchmarks for Accelerated MRI.

[36] Nameirakpam Dhanachandra, Yambem Jina Chanu. A survey on Image Segmentation Methods using Clustering Techniques. EJERS, European Journal of Engineering Research and Science, Vol. 2, No. 1, January, 2017

[37] Ionescu B. Data Analysis and Machine Learning. Lecture notes, 2019. Available on http://campus.pub.ro/lab7/bionescu/courses.html#tacai

[38] R. Jensi and G. Wiselin Jiji. Hybrid Data Clustering approach using K-means and Flower Pollination Algorithm. Advanced Computational Intelligence: An Internal Journal (ACII), vol 2, No 2, April 2015.

[39] Nilesh Bhaskarrao Bahadure, Arun Kumar Ray and Har Pal Thethi. Image Analysis for MRI Based Brain Tumor Detection and Feature Extraction Using Biologically Inspired BWT and SVM. International Journal of Biomedical Imaging, volume 2017

[40] https://github.com/ShaunCayabyab/k-means.py , accessed on 02/20/2020

[41] Ivana Despotović, Bart Goossens, and Wilfried Philips. MRI Segmentation of the Human Brain: Challenges, Methods, and Applications. Computational and Mathematical Methods in Medicine Volume 2015, Article ID 450341

[42] Ramzi, Z.; Ciuciu, P.; Starck, J.-L. Benchmarking MRI Reconstruction Neural Networks on Large Public Datasets. *Appl. Sci.* 2020, vol *10*, article no 1816.

[43] Srinjay Paul. Convolutional denoising autoencoder for images. 2018. Online article available here: https://srinjaypaul.github.io/Convolutional_autoencoders_for_images/

[44] Nathan Hubens. Deep inside: Autoencoders. Feb 25, 2018. Online article available here: https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

[45] L. Gondara, "Medical Image Denoising Using Convolutional Denoising Autoencoders," 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, 2016, pp. 241-246, doi: 10.1109/ICDMW.2016.0041.

[46] Mehta, Janki & Majumdar, Angshul. (2016). RODEO: Robust DE-aliasing autoencOder for Real-time Medical Image Reconstruction. Pattern Recognition. 63. 10.1016/j.patcog.2016.09.022.

[47] Vitaly Bushaev. Adam — latest trends in deep learning optimization. Oct 22, 2018. Accessible here: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

[48] Jason Brownlee. Difference Between a Batch and an Epoch in a Neural Network. July 20, 2018. Accessible here: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

[49] https://github.com/crackwebai1024/denoising-autoencoders-keras-tensorflow, accessed on 02.18.2020

Annex

Segmentation:

```
def converged(centroids, old_centroids):

        epsilon = 1

        for i in range(0, len(centroids)):
                cent = centroids[i]
                old_cent = old_centroids[i]

                if ((int(old_cent[0]) - epsilon) <= cent[0] <= (int(old_cent[0]) + epsilon)) and
((int(old_cent[1]) - epsilon) <= cent[1] <= (int(old_cent[1]) + epsilon)) and ((int(old_cent[2]) - epsilon) <=
cent[2] <= (int(old_cent[2]) + epsilon)):
                        continue
                else:
                        return False

        return True


def getMin(pixel, centroids):
        minDist = 1e5
        minIndex = 0

        for i in range(0, len(centroids)):
                d = np.sqrt(int((centroids[i][0] - pixel[0]))**2 + int((centroids[i][1] - pixel[1]))**2 +
int((centroids[i][2] - pixel[2]))**2)
                if d < minDist:
                        minDist = d
                        minIndex = i

        return minIndex


def assignPixels(centroids):
        clusters = {}

        for x in range(0, img_width):
                for y in range(0, img_height):
                        p = px[x, y]
                        minIndex = getMin(px[x, y], centroids)
                        clusters[minIndex].append(p)

        return clusters

def adjustCentroids(centroids, clusters):
        new_centroids = []
        keys = sorted(clusters.keys())

        for k in keys:
                n = np.mean(clusters[k], axis=0)
```

```
                new = (int(n[0]), int(n[1]), int(n[2]))
                new_centroids.append(new)

        return new_centroids


def startKmeans(K):
        centroids = []
        old_centroids = []

        for k in range(0, K):
                cent = px[np.random.randint(0, img_width), np.random.randint(0, img_height)]
                centroids.append(cent)

        i = 1
        while not converged(centroids, old_centroids) and i <= 20:
                i += 1

                old_centroids = centroids
                clusters = assignPixels(centroids)
                centroids = adjustCentroids(old_centroids, clusters)

        return centroids


def drawWindow(result):
  // save segmented image


k_input = 5
path = os.getcwd()
path = path +'/db'
subdir = os.listdir(path)

path_results = path+'/results_filt'
results = os.mkdir(path_results)

for subdirectory in enumerate(subdir):
        new_path = path+"/"+subdirectory[1]
        files = os.listdir(new_path)
        for file in enumerate(files):
                file = file[1]
                filename = new_path+'/'+file
                ds = dicom.dcmread(filename)
                im = ds.pixel_array
                img_height, img_width = im.shape
                im = im/np.max(im)
                px = Image.fromarray(np.uint8(cm.gray(im)*255))
                px = px.load()
                result = startKmeans(k_input)
                max_count = 0
                while((len(result) != k_input) and max_count<=1):
                        result = startKmeans(k_input)
```

```
                print(max_count)
                max_count = max_count+1

        if(max_count > 1 and (len(result) != k_input)):
                print("Couldn't make ", k_input, "classes, only ", len(result), "could be found." )

        drawWindow(result)
```

Autoencoder:

```
class Autoencoder:

    def build(width, height, depth):
        inputShape = (height, width, depth)
        filters=(32,64)
        filter_dim = (3,3)
        latentDim=16
        chanDim = -1

        inputs = Input(shape=inputShape)
        x = inputs

        for f in filters:
            x = Conv2D(f, filter_dim, strides=2, padding="same")(x)
            x = LeakyReLU(alpha=0.2)(x)
            x = BatchNormalization(axis=chanDim)(x)

        volumeSize = backend.int_shape(x)
        x = Flatten()(x)
        latent = Dense(latentDim)(x)

        encoder = Model(inputs, latent, name="encoder")

        latentInputs = Input(shape=(latentDim,))
        x = Dense(np.prod(volumeSize[1:]))(latentInputs)
        x = Reshape((volumeSize[1], volumeSize[2], volumeSize[3]))(x)

        for f in filters[::-1]:
            x = Conv2DTranspose(f, filter_dim, strides=2,padding="same")(x)
            x = LeakyReLU(alpha=0.2)(x)
            x = BatchNormalization(axis=chanDim)(x)

        x = Conv2DTranspose(depth, filter_dim, padding="same")(x)
        outputs = Activation("sigmoid")(x)
        decoder = Model(latentInputs, outputs, name="decoder")

        autoencoder = Model(inputs, decoder(encoder(inputs)),name="autoencoder")
        return (encoder, decoder, autoencoder)
```

Evaluation:

```
def compute_psnr(original, compressed):
   mse = np.mean((original - compressed) ** 2)
   if(mse == 0):
      return 100
   max_pixel = 255.0
   p = 20 * log10(max_pixel / sqrt(mse))
   return p

PATH = os.getcwd()
path = PATH+'/k-means.py-master/result'
imgs = os.listdir(path)

psnr = [[] for i in range(int(len(imgs)/3))]
i = 0
f = open(path+'/../result.txt', 'w+')
f.write("PSNR [dB] and SSIM\n")
while i<int(len(imgs)/3):
   pos1 = imgs.index(str(i)+'.jpeg')
   pos2 = imgs.index(str(i)+'_orig.jpeg')
   pos3 = imgs.index(str(i)+'_recon.jpeg')
   img1 = Image.open(path+'/'+imgs[pos1])
   img2 = Image.open(path+'/'+imgs[pos2])
   img3 = Image.open(path+'/'+imgs[pos3])

   img1 = (img1.resize((256,256), Image.ANTIALIAS))
   img2 = (img2.resize((256,256), Image.ANTIALIAS))
   img3 = (img3.resize((256,256), Image.ANTIALIAS))

   psnr[int(i)].append(imgs[pos2]+" & "+imgs[pos3])
   f.write("\n")
   f.write(str(imgs[pos2]+" & "+imgs[pos3]+": psnr = "))
   p = compute_psnr(img_to_array(img2), img_to_array(img3))
   psnr[int(i)].append(p)
   f.write(str(p))

   s = compare_ssim(img2,img3)
   psnr[int(i)].append(s)
   f.write(" || ssim = "+str(s))
   f.write("\n")
   psnr[int(i)].append(imgs[pos1]+" & "+imgs[pos2])
   f.write(str(imgs[pos1]+" & "+imgs[pos2]+": psnr = "))
   p = compute_psnr(img_to_array(img2),img_to_array(img1))
   psnr[int(i)].append(p)
   f.write(str(p))
   s = compare_ssim(img2,img1)
   psnr[int(i)].append(s)
   f.write(" || ssim = "+str(s))
   f.write("\n\n")

   i = i+1
f.close()
```