Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea Magistrale in Fisica

A Machine Learning Approach to QSM: Quantitative Susceptibility Map Reconstruction with Convolutional Autoencoders

Supervisor:

Prof. Daniel Remondini

Physics and Astronomy, Università di Bologna

Author:

Cristiana Fiscone

Co Supervisor:

Prof.ssa Claudia Testa

Physics and Astronomy, Università di Bologna

Co Supervisor:

Prof. Richard Bowtell

SPMIC, University Of Nottingham

Academic Year 2017/2018

Abstract (ITA)

QSM (Quantitative Susceptibility Mapping) è una promettente applicazione MRI, in grado di riprodurre, a partire da dati di fase ottenuti con sequenze gradient echo, distribuzioni di suscettività magnetica dei tessuti. In questo modo, può essere studiata la loro composizione e microstruttura. Questa tecnica viene utilizzata, ad esempio, per identificare lesioni cerebrali traumatiche e per caratterizzare malattie vascolari e neurodegenerative.
Tuttavia, per ottenere la mappa di suscettività, deve essere risolto un problema mal posto, dovuto alla presenza di singolarità all'interno dello spazio di Fourier. Sono state già implementate diverse tecniche per superare questa difficoltà, divise in due principali categorie: *single-orientation methods* (e.g. TKD), che garantiscono una ricostruzione veloce ma rumorosa, e *multiple-orientation methods* (e.g. COSMOS), che forniscono un'accurata distribuzione di suscettività, anche se i tempi di acquisizione richiesti sono maggiori. Bisogna trovare un compromesso tra accuratezza e velocità per poter applicare QSM all'ambito clinico; una soluzione potrebbe essere un metodo che sfrutti tecniche di *deep learning*.

In questo lavoro, abbiamo costruito una rete neurale con una struttura di *fully convolutional autoencoder*, inserendo *long* e *short skip connections*. La rete è stata addestrata tramite apprendimento supervisionato, utilizzando dati di fase come input, dati di suscettività ottenuti da COSMOS come *labels* e dati di suscettività ottenuti da TKD come controllo.
Sono state utilizzate le immagini dell'encefalo disponibili nel database della QSM_{2016} Challenge; da qui, dati 2D e 3D sono stati estratti e processati. La distribuzione ottenuta con il nostro modello è stata confrontata con quella di COSMOS, considerata come gold-standard per la precisione nella ricostruzione: valutando intensità e contrasto in alcune regioni, sono emersi risultati tra loro consistenti. Le perfomance del network sono migliori di quelle di TKD, la cui distribuzione di suscettività assume invece valori non consistenti rispetto a COSMOS e con incertezza maggiore, con conseguente diminuzione del contrasto.
Il risultato ottenuto è perciò soddisfacente: tramite *machine learning*, tecnica che rientra nella categoria dei *single-orientation methods*, è possibile quindi costruire velocemente una mappa di suscettività magnetica accurata tanto quanto quella risultante da approcci a *multiple-orientation*.

Abstract (EN)

QSM (Quantitative Susceptibility Mapping) is a promising MRI application, aimed to reproduce maps of the susceptibility distribution in tissues from gradient echo phase data. Thanks to this, tissue composition and microstructure can be investigated. The QSM tool can be used, for instance, to analyse traumatic brain injury and to treat vascular and neurodegenerative diseases.
However, an ill-posed inversion problem, due to the presence of singularities in the Fourier space, has to be solved to obtain a magnetic susceptibility map. Different techniques have already been implemented in order to fix this issue. They are divided in two main categories: single-orientation methods (e.g. TKD), providing a fast but noisy reconstruction, and multiple-orientation methods (e.g. COSMOS), requiring a long acquisition time but returning a precise and accurate susceptibility map. A good compromise between the accuracy and the speed of the reconstruction has to be found in order to use the QSM method in clinical applications. A deep learning approach could be a solution.

In this work, we implemented a fully convolutional autoencoder algorithm, including long and short skip connections as residual learning tools. It was trained by supervised learning technique, using phase data as input, susceptibility data from the COSMOS map as labels and susceptibility data from the TKD map as control.
Brain images from the QSM_{2016} Challenge dataset were used in this study; 2D and 3D data were extracted and processed. The susceptibility distribution obtained with our model was compared with the COSMOS one, considered as gold-standard: the two approaches provide consistent results, in according to the intensity and contrast analysis. The network perfomance are better than the TKD one: its distribution shows significant differences with respect to the COSMOS map and lower contrast in all the examined areas.
The final result is satisfactory: using a machine learning approach, which is a single-orientation method, a susceptibility map can be quickly reconstructed, and it is as accurate as the ones from multiple-orientation techniques.

Contents

Introduction

The magnetic susceptibility χ is a property of matter, which measures the degree of magnetization of a material when it is exposed to a magnetic field. In the human body there are a lot of susceptibility sources. Those biomarkers and their specificities, such as the density and the concentration, allow healthy and unhealthy states to be determined and classified. Indeed, to study the tissue susceptibility helps in diagnosis and treatment of disorders.
When a magnetic field is externally applied upon an area containing a heterogeneous object, some changes in that field occur, depending on the boundaries between regions with different susceptibility values. Those field perturbations are directly proportional to the phase ϕ data from MRI measurements. An image, as a result of a magnetic resonance acquisition, pictures a collection of tissues with a spatial susceptibility distribution $\chi(r)$. There is not a direct way to detect the susceptibility value of each voxel; to analyze it, some techniques were developed.

The tool which provides the tissue χ values to be calculated is **QSM (Quantitative Susceptibility Mapping, [1])**. It is a recent MRI application which, starting from ϕ measurements, is able to return the anatomical susceptibility distribution. Unfortunately, the inversion $\chi = \chi(\phi)$ is ill-posed, and that causes the presence of a lot of noise and artifacts in the reconstructed map. Different strategies were implemented to overcome the QSM problem.
There are two classes of approaches: the multiple-orientation and the single-orientation ones. The first type provides a very precise and accurate susceptibility map, thanks to multiple acquisitions of the examined object scanned at different orientations. However, they require a long acquisition time. The second type instead are numerical strategies, faster but less precise in reconstructing the susceptibility distribution.
To use the QSM tool in the clinical area, the susceptibility map has to be derived in a quick and accurate way.

NN (Neural Network) algorithms are, nowadays, widely used to perfom image processing tasks **([2], [3])**. The training stage of these models could require many hours, but it has to be performed once, and after that the processing is very fast. The use of NNs could let the ill-posed QSM problem be fixed and the tool be applied in the clinical area.
The main goal of this work is therefore to implement a machine learning algorithm to process

a ϕ map from a single MRI acquisition in order to obtain a precise and noise-and-artifact free susceptibility map. The model has to present a convolutional autoencoder architecture. We would like to perform it with both 2D and 3D data. We are going to focus on the neuroimaging area.

The thesis is divided into five chapters: two of them are dedicated to the theoretical introduction and the other three are focus on the experimental part of the work. Below, a brief description of each of them is given.

- **Chapter 1: Quantitative Susceptibility Mapping**
  After a spotlight on the magnetic susceptibility sources in the human body and their properties, the main susceptibility-based MRI techniques are introduced. Initially, the **SWI (Susceptibility-Weighted Imaging,** [**4**]) is described: this approach combines ϕ and magnitude information contained in MRI measurements. It allows the susceptibility distribution of the examined tissues to be qualitatively described, but it does not provide anatomical information. The QSM tool does.
  The relationship between the field perturbation and the susceptibility is explained in detail, both in real space and in spatial frequency domain (k-space). After that, the multiple-orientation and single-orientation approaches are introduced. In the first category there are the **COSMOS (Calculation of Susceptibility Mapping through Multiple Orientation Sampling,** [**5**]) and the **STI (Susceptibility Tensor Imaging,** [**6**]) techniques. The second class is divided into non-iterative k-space approaches and iterative optimization techniques. We are going to focus on the first group, in particular on the **TKD (Threshold K-Space Divison,** [**7**]) approach. In our experiments, the COSMOS map is taken as a gold-standard susceptibility reconstruction, while the TKD one is used as a control map.

- **Chapter 2: Convolutional Neural Networks in Image Processing**
  Learning algorithms used in image processing tasks are very complex network structures. Before describing them, a focus on the first implemented artificial neurons, basic unit of every neural network, is made. We are going to describe the **Single-Layer** and the **One-Hidden-Layer Perceptrons** ([**8**]), characterized by learning rules and hyperparameters.
  Algorithms with **Multiple-Hidden-Layer Structure** have also been implemented, with the view to improve the deep learning techniques ([**9**]). There is a specific category of architectures aimed at performing image processing tasks: they are called the **CNNs (Convolutional Neural Netwoks,** [**3**]). When a CNN presents an **Encoded-Decoded** structure, an image reconstruction goal may be achieved.
  Residual learning tools are essential with respect to this purpose: long and short skip connections are going to be introduced.

- **Chapter 3: Materials and Methods**
  In this chapter, the experimental part begins to be described. Focusing on the goal of the work - to implement a convolutional autoencoder model in order to solve an image reconstruction task and alternatively fix the ill-posed QSM problem -, we are going to present the methods used to achieve it.
  First, the images used are going to be described. Input, reference and control maps are all from the free downloadable database of the **QSM Challenge$_{2016}$ ([10])**. It contains ϕ maps with different head orientations; all of them were registered. The pre-processing stages and the dataset construction from those data will be explained. After this, there is a section dedicated to the metric parameters chosen to evaluate the model performance.
  The second part is focused on the model structure and on the optmization of its hyperparameters: a set of preliminary experimentes will be listed to explain the final model we are going to use for the proper susceptibility map reconstruction. In here, we focus on 2D data processing.

- **Chapter 4: 2D Quantitative Susceptibility Reconstruction**
  After the structure optimization, the first two experiments are going to be introduced. This part is dedicated to 2D data training. The axial slab orientation is chosen as direction to be observed.
  Initially, the model is trained using **2D patches from single-orientation** ϕ data, and supervised learning is performed. The COSMOS patches are taken as labels, while the TKD map is taken as the control. The machine learning approach is compared with the other two methods, looking into the susceptibility values and the contrast analysis. The areas we focus on are: the putamen, globus pallidus, caudate, substantia nigra and red nucleus. These are grey matter sub-cortical structures. Also some white-matter-based tissues are explored.
  After that, **2D patches from multiple-orientation ϕ maps** are used to train the model again. The same analysis is carried out. The response of the model changing the direction of the input data is tested.

- **Chapter 5: 3D Quantitative Susceptibility Reconstruction**
  Sagittal, coronal and axial slices are used to build another database, in order to observe a 3D susceptibility map and examine different brain sections. The model has still been trained in a supervised way, with the COSMOS patches as labels and the TKD ones as controls.
  Finally, we present a **3D data experiment**. The model structure has to be adapted to pass from the two-dimensional to the three-dimensional data processing. We conclude with a spotlight on the possible follow-up of the work.

Chapter 1

Quantitative Susceptibility Mapping

QSM - Quantitative Susceptibility Mapping is an MRI technique which can be used to obtain both functional and anatomical images, starting from NMR-GRE phase data. This is possible thanks to the response of a magnetic susceptibility distribution inserted into an externally applied magnetic field. The forerunner of this tool is called **SWI - Susceptibility Weighted Imagining**: this is another susceptibility-based technique, which allows to explore new variables with respect to the traditional NMR techniques, but which does not provide an anatomical reconstruction.
The goal of this chapter is to introduce the QSM method and its applications. The chapter is divided into the following sections:

- Section 1.1: **Magnetic Susceptibility χ in the Human Body** ([11], [12], [13]) - This part describes χ as a physical quantity and its relationship with the magnetic field $\vec{H}$, the magnetic induction field $\vec{B}$ and the magnetization field $\vec{M}$. We are going to divide the matter according to its magnetic properties into: diamagnetic, paramagnetic and ferromagnetic materials. We are going to describe then the main susceptibility sources in the human body, focusing especially on the brain tissues and structures.

- Section 1.2: **Susceptibility and Imaging** - There are two main imaging susceptibility-based techniques: SWI ([14], [4]) and QSM ([1], [15]). We are going to describe these two tools, especially concentrating on the QSM one and on the ill-posed inversion problem involved in the susceptibility map reconstruction. Two approaches to QSM will be explained in detail: the COSMOS ([5]) and the TKD methods ([7]).

- Section 1.3: **Susceptibility Mapping Applications** - This includes a list of some of the possible applications of susceptibility mapping.

1.1 Magnetic Susceptibility χ in the Human Body

The proposal of the next pages is to analyze different MRI techniques which exploit susceptibility-based contrast in tissues. To better understand this, it has to be clear what magnetic susceptibility is and what we measure and describe when we are referring to it.
Magnetic susceptibility χ is a physical quantity which measures the extent to which a material is magnetized by an applied magnetic field ([14]). Depending on the characteristics of this field, one refers to **static** or **dynamic** χ. In MRI, the tissues' response is due to an external, static field $\vec{B_0}$, so in this work we are going to refer to the static magnetic susceptibility. $\vec{B_0}$ is actually the **magnetic flux density** corresponding to the applied **magnetic field** $\vec{H_0}$. The magnetic flux density is also called the **magnetic induction** vector.

1.1.1 Magnetic Field and Tissue Magnetization

Tissues in the human body react when a magnetic field, is applied to them. In fact, they are characterized by magnetic properties. Their reaction is due to the interaction between the atomic electrons and the field, which causes the magnetization of the matter. Matter magnetization is quantitativly described by the **magnetization** $\vec{M}$, defined as ([13]):

$$\vec{M} = \frac{d\vec{m}}{dV} \tag{1.1}$$

where $d\vec{m}$ is the elementary **magnetic moment**. The magnetization $\vec{M}$ and magnetic field $\vec{H}$ are directly related. The susceptibility χ is the proportionality constant ([11], [13]):

$$\vec{M} = \chi\vec{H} \tag{1.2}$$

χ has both isotropic and anisotropic behaviour, depending upon the tissue under examination. In isotropic and homogeneous materials the response is uniform, and it means that χ can be modelled as a scalar quantity. But in some regions, the response of the tissue is not uniform: it depends on the orientation of the field, so χ shows an anisotropic reaction. This behaviour is an effect of the asymmetry of the molecules' structure ([11]). Thus, there are some situations in which the susceptibility is modelled as a tensor function of the applied magnetic field ($\vec{\chi} = \vec{\chi}(\vec{H})$). In such situations, the mathematical formalism for the phenomena description is more complicated. We are going to return to this point later. In the meantime, we consider χ as scalar property of matter.
To make clear the relationship between χ and the magnetic induction $\vec{B}$, remember that the following expression applies:

$$\vec{B} = \mu_0(\vec{H} + \vec{M}) \tag{1.3}$$

where μ_0 is the vacuum permeability constant ($4\pi{*}10^{-7}\frac{H}{m}$). Then:

$$\vec{B} = \mu_0(\vec{H} + \chi\vec{H}) = \mu_0(1+\chi)\vec{H} = \mu_0\mu_r\vec{H} \tag{1.4}$$

μ_r is called the **relative permeability**, and it represents the degree of the magnetization of the tissues ([16], [13]).

In MRI images, the ratio between magnetization and magnetic field defines the **volume susceptibility** ($\chi = \frac{\vec{M}}{\vec{H}}$) ([4]). It is a dimensionless property, which can be measured in the SI (*Systeme Internationale*) or in CGS (Centrimetre-Gram-Second) systems. To pass from SI to CGS values one has to divide by a factor of 4π. For example ([17], [16]):

$$\chi_{water} = -9.035\ ppm\ (SI\ units) = -0.719\ ppm\ (CGS\ units) \tag{1.5}$$

Water is the most abundant molecule in the human body. $\Delta\chi$ between water and other tissues inside the body is on the order of $\pm$ 0.1 ppm ([14], [7]). The biggest gap is between water and air, since χ_{air} is equal to +0.37 ppm (in SI units) (Fig. 1.1).
During a magnetic resonance experiment, a region of interest ROI, containing different tissues, is explored. Different tissues have different values of susceptibility, which are described by the $\chi(r)$ distribution. We are going to explain, in Sec. 1.2, how this distribution affects the NMR measurements.

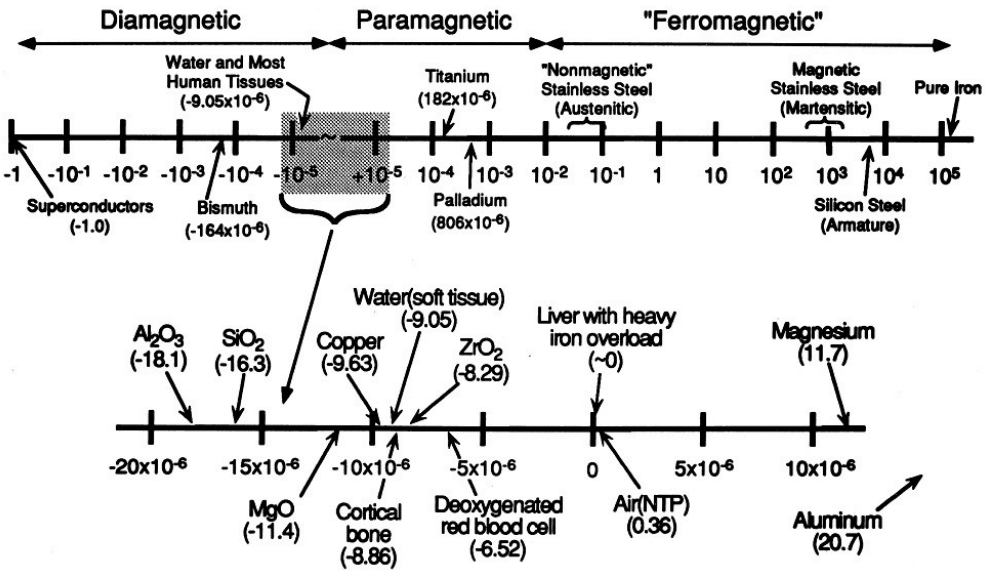


Figure 1.1: Scale of magnetic susceptibility values of some element and compounds in the human body [17]

1.1.2 Magnetic Materials: Dia-, Para- and Ferro-magnetic Conditions

In almost all the materials, without an externally applied magnetic field - $\vec{H} = \vec{B} = 0$ -, the magnetization vector $\vec{M}$ has intensity equal to zero. When instead $\vec{B} \neq 0$ and $\vec{H} \neq 0$, also $\vec{M} \neq 0$ because of the equation Eq. 1.3. The direction, sense and intensity of $\vec{M}$ are related to χ and μ_r, which affect the magnetic response of matter.
Magnetic susceptibility values depend on the atomic and the molecular structure: availability of unpaired electrons, distribution of electron cloud, competition between spin and induction,

and microstructure ([14], [11] [18]). In relation to these characteristics, materials are divided in three main classes: diamagnetic, paramagnetic and ferromagnetic materials.

- **Diamagnetic Materials**

  Diamagnetism is a magnetic property of matter which let a repulsive force be induced when an external magnetic field is applied. All the materials have this property, but sometimes other forms of magnetism, such as paramagnetism or ferromagnetism, dominate the diamagnetic effect.

  The diamagnetic materials have not a permanent magnetic moment $|\vec{\mu}|$. The presence of an external magnetic field induced a magnetization field $\vec{M}$. $\vec{M}$ has a small intensity and is directed in the opposite sense with respect of the field. This is due to electron induction currents which generate a magnetic field directed in the opposite sense with respect of the external one (**Lenz's Law**) ([11], [13])

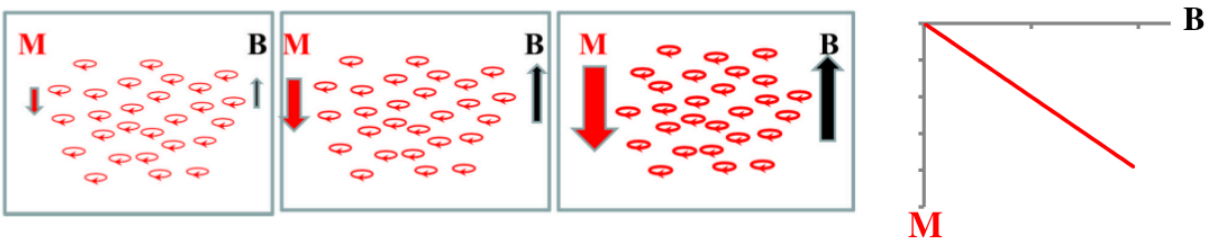


Figure 1.2: Scheme of magnetization vector $\vec{M}$ and external magnetic field $\vec{B}$ in diamagnetic materials ([11]). In balance condition, without an external magnetic field, diamagnetic materials have net magnetic moment equal to zero. When a field $\vec{B}$ is applied, a weak magnetization vector $\vec{M}$ is formed along the same direction of the field but in the opposite sense

In the human body, almost all the substances are diamagnetic ([1]). These diamagnetic materials have χ values small and negative. Water is diamagnetic (χ_{water} = -9.035 ppm), but also calcium, oxyhemoglobin (HbO_2,Fe^{3+})[4] and myelin ([14]). Myelin sheaths envelop the **white matter (WM)** structures in the brain; for this reason WM appears as a primarily diamagnetic tissue.

- **Paramagnetic Materials**

  In contrast to diamagnetic materials, the paramagnetic and ferromagnetic ones are attracted by an externally applied magnetic field. The magnetic moments $\vec{\mu}_i$ in the paramagnetic materials have a non-zero intensity value even if there is not an external field $\vec{B}$ applied. When a $\vec{B}$ is applied, the material is magnetized ($\vec{M} \neq 0$). $\vec{B}$ and $\vec{M}$ are directed in the same way. So, the external magnetic field is reinforced.

  Without an external magnetic field the magnetic moments of the single particles are directed in a random way. To simplify the description, consider only the $\vec{z}$ direction and look into the Fig. 1.3: n_+ is the number of the magnetic moments pointing up, n_- the number of magnetic moments pointing down. In a balanced situation ($\vec{H} = 0 \rightarrow \vec{M} = 0$), $n_+ = n_-$. This balance is broken when an external field is applied. The imbalance $n_+ - n_-$ is proportional to $\frac{\mu B}{k_B T}$ when $|\vec{\mu}||\vec{B}| << k_b T$ - where $\vec{\mu}$ is the total magnetic moment, k_B the Boltzmann constant ($1.23 \cdot 10^{-23} JK^{-1}$) and T is the

temperature. When $\vec{B}$ has a strong intensity ($|\vec{\mu}||\vec{B}| >> k_bT$), the relationship stops to be linear and n_- tends to zero ([11]). All the single magnetic moments tend to line up in the same direction of the field.

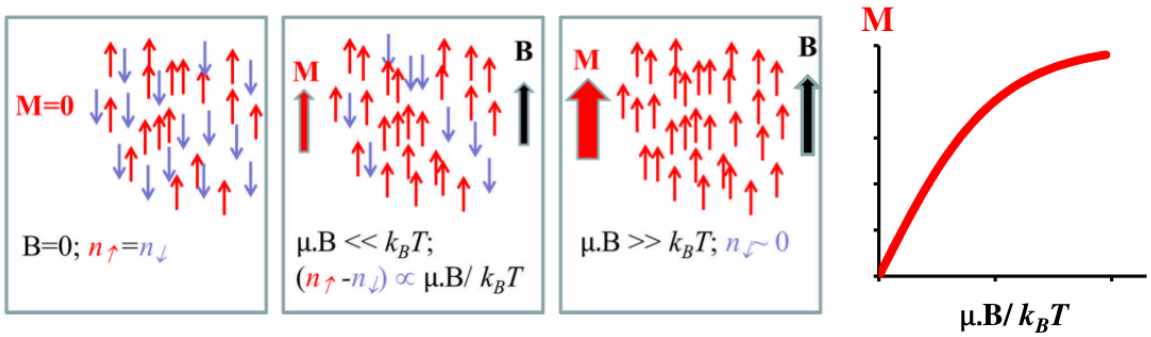


Figure 1.3: Scheme of magnetization vector $\vec{M}$ and external magnetic field $\vec{B}$ in paramagnetic materials ([11]). In absence of an external magnetic field, the number of magnetic moments pointing up n_+ and the ones pointing down n_- balance each other. When an external field is applied, the majority of the magnetic moments starting to allign as the field. When the intensity of the field is strong, the number of the magnetic moments direct in the opposite sense of the field tends to zero

Unpaired electrons produce strong paramagnetic effects. Their spins tend to align themselves along the direction of the field. In this way, they make it stronger in intensity.

In paramagnetic materials, χ assumes small and positive values; the relative permeability μ_r is slighlty greater than 1. Deoxyhemoglobin (Hb,Fe^{2+}) is paramagnetic, and also iron-storage substances (i.e. ferritin and hemosiderin). Iron, inside the brain, is mainly concentrated in the **gray matter GM** structures. Thus, GM is mainly paramagnetic.

- **Ferromagnetic Materials**

  Also ferromagnetic materials react in a constructive manner when an external field is applied. They are characterized by a net magnetic moment, also in absence of an external field. When a field is externally applied, their reaction is stronger than paramagnetic materials: their magnetization is long lasted, and they may become magnets. This happens thanks to a very energetically favourable energy exchange between the material and the field.

  Ferromagnetic materials are predominantly metals. Their magnetic moments are very high in intensity, so the magnetic susceptibility values are positive and very large. There are not elements stored in known ferromagnetic structures in the human body ([14]), even if there are ferromagnetic elements. For example, iron is a ferromagnetic element, but ferritin and hemosiderin, which store iron, are not.

1.1.3 Susceptibility Sources in Human Body

Study of the magnetic properties of materials allows information about the concentration and the structure of many elements in the human body to be obtained. Diseases and disorders can be related with abnormalities and changes in magnetic properties. Thus, using considerations about χ, many symptoms and manifestations can be quantified.

Below, we give a list of the most interesting materials to analyze in evaluating human body state.

- **Iron**
  Iron is a transition metal with the following electron configuration: $[Ar]3d^6s^2$. The four unpaired electrons in the d-shell cause its magnetic response. In the human body, there are around 3-5 g of iron.
  Iron is ferromagnetic, but, as we have already noticed, it is stored in structures which are not. One third of iron is found in non-heme compounds, in which iron is in a ionic form. The two main structures are ferritin (Fe^{3+}) and hemosiderin (Fe^{2+} and Fe^{3+}), which are paramagnetic ([14]). The remaining two thirds ([11]) is stored in hemoglobin, and also does not show ferromagnetic behaviour. We are going to focus on it later.
  Iron deposition is the main reason of the paramagnetic susceptibility values in the brain. One often refers to it as the **brain iron**. Thus, a susceptibility map may be used to describe and quantify iron stores, also in the brain and in the nervous system in general. These stores are mainly concentrated in the gray matter structures. Iron concentration is an age-based parameter, and it is associated with some neurodegenerative conditions - i.e. Parkinson's and Alzheimer's diseases ([1]).

- **Hemoglobin**
  The majority of the human iron is stored in hemoglobin ([11]). It is a metalloprotein, containg four iron atoms, involved in the oxygen transport. It may be found in two forms: **oxyhemoglobin** and **deoxyhemoglobin** (Fig. 1.4). In the first form, the molecule is carrying oxygen ($(HbO_2,Fe^{3+})^4$ - Fig. 1.4, left panel). Iron is temporarily oxidized from Fe^{2+} to Fe^{3+}. It is weakly diamagnetic, thanks to ligand interactions inside the structure, that prevents the unpaired electrons in the iron ion from showing paramagnetic effects ([15]). Instead, the second form of hemoglobin is the structure that the molecule assumes when the oxygen is released during the metabolic consuption ((Hb,Fe^{2+}) - Fig. 1.4, right panel). It is strongly paramagnetic, because it has four unpaired electrons per heme. Also, deoxyhemoglobin may further oxidized in methemoglobin, which has five unpaired electron per heme. It shows a ferromagnetic response ([14]).
  To study hemoglobin magnetic effects aims to study BOLD signals and all disorders connected with the blood flow, i.e. hemorrhages ([14], [11]), thanks to the oxygenation-related susceptibility changes occurring in blood products.

- **Myelin**
  It is a spiralling sheath that wraps around nerves, in the brain and in the spinal cord ([14]). A schematic diagram of the myelin sheath is reported in Fig. 1.5. It is composed of proteins (10%), water (40%) and lipids (50%) ([11]), which are responsible to the electric insulation myelin provides. For most of these components, the volume susceptibility assumes diamagnetic values, some of them are more diamagnetic than

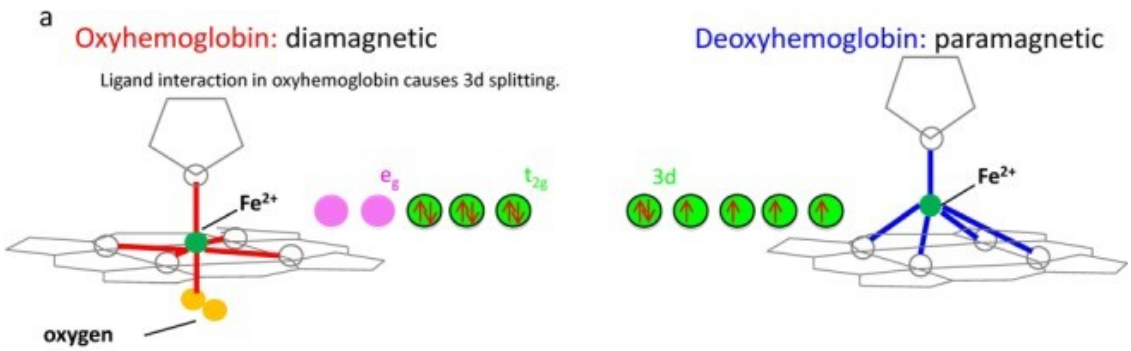


Figure 1.4: Oxy- (left panel) and Deoxy- (right panel) hemoglobin structures ([15])

water - i.e. $\chi_{phospholipids}$ = -9.68 ppm, $\chi_{sphingolipids}$ = -10.03 ppm ([15]).
Myelination and **de-myelination** are age-based processes; loss of myelin is related with some neurodegenerative autoimmune diseases - e.g. Multiple Sclerosis.
Also, the WM tracts in the brain consist of myelinated nerve fibers ([15]). In here, the several layers of bilayers lipid are very sensitive to the particular microstructures of WM. Indeed, here the susceptibility shows anisotropic response. This is the reason why to model χ as a tensor is more appropriate if the white matter inside the brain is the object under examination. WM has a very particular and complex structure. Its study requires more attention also because of its compartmentalization. It is in different compartments, such as the axonal space, the myelin space and extracellular space, and each of them has specific magnetic properties and reacts in a unique way ([14], [19]).

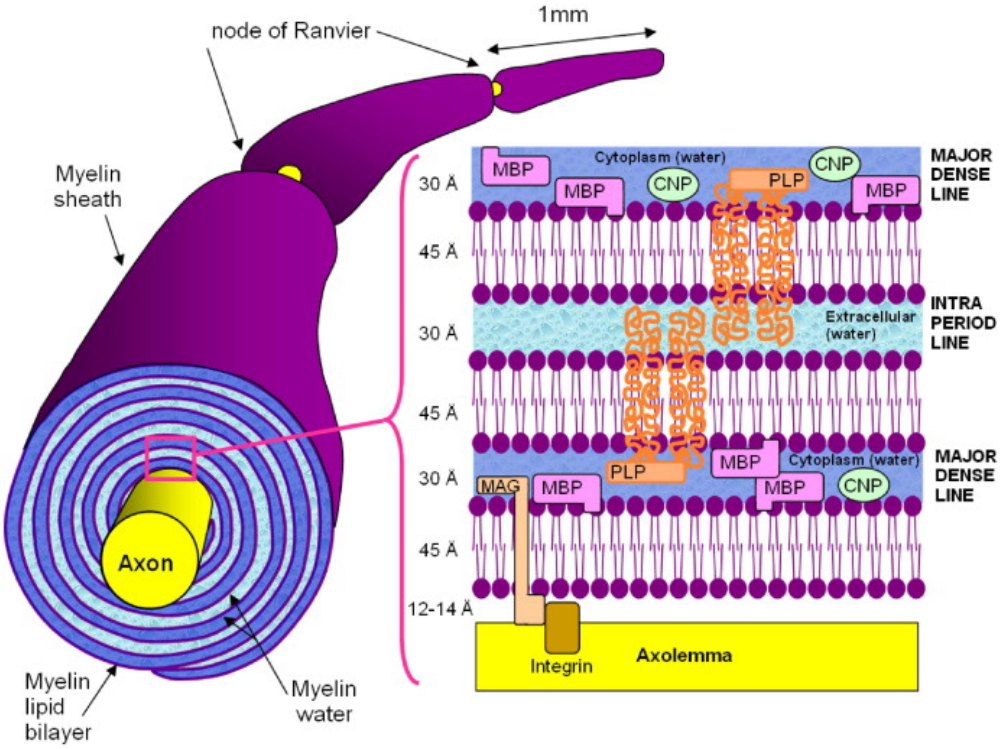


Figure 1.5: Myelin scheme ([20])

Referring to the anisotropic response, a point has to be clarified. In fact, this ex-

pression may indicate two different phenomena: **structural anisotropy** and **susceptibility anisotropy**. The first expression describes the situation in which the orientation-dependence of NMR measurements is due to the geometrical shape of the tissue structure at the cellular and subcellular level. Instead, to talk about susceptibility anisotropy refers to those situations in which the response of the tissues changes with the orientation in NMR measurements, because χ, at the molecular level, is a tensor and not a scalar quantity. In that situation, there are inherent molecular χ anisotropies ([14], [15]).

- **Calcium**
  It is an essential element, involved, for example, in the building of bones and in the synthesis of blood cells. It is stored in different structures in the human body, often in a ionic form (Ca^{2+}). If calcium magnetic properties are under examination, it is actually more precise to speak about **calcification** rather than proper calcium. In fact, there are many studies and insights about calcium-related pathologies (i.e. calcified lesions) but not many about the healthy state conditions ([15]). Calcification in damaged tissues causes the local tissue to appear diamagnetic in MRI susceptibility-based techniques.

To analyze a susceptibility distribution $\chi(r)$ is useful to understand physical mechanism in healthy and unhealthy human tissues. It is possible to investigate $\chi(r)$ with MRI techniques thanks to the relationship between $\chi(r)$ and the field perturbations $\Delta B(r)$ that it produces. The latter quantity is strictly related with the $\phi(r)$ distribution measured during gradient echo NMR experiments.

In the next section, we are going to explain different MRI techniques which exploit this phenomenon and allow us to investigate magnetic effects. The first one is **SWI - Susceptibility-Weighted Imaging**, in which phase and magnitude data from NMR measurements are combined to enhance susceptibility-based contrast. This MRI tool does not allow a proper anatomical susceptibility map $\chi(r)$ to be produced. To achieve this, proper calculations have to be performed on the raw data. The technique with this regard is called **QSM - Quantitative Susceptibility Mapping**, thanks to which it is possible to renconstruct $\chi(r)$ map in the spatial domain, starting from measured data ($\Delta\phi(r)$ or $\Delta B(r)$). We are going to focus on it, studying the mathematical aspects and the possible applications.

1.2 Susceptibility and Imaging

In a lot of research fields, especially in the medical area, there are many advantages in analyzing human tissues magnetic properties and responses. There are biomarkers, such as iron, calcium, and myelin, whose changes in concentration and state are strictly related with many disorders and pathologies. They are marked by different magnetic properties. Thus,

to study those properties means to study and analyze human body diseases.
Some MRI techniques are specific for these kind of studies, using tissues' magnetic susceptibility χ as the principal observable. Instead of other tools, to use magnetic resonance is better because it implies a non-invasive procedure. Susceptibility-based techniques use information in phase data from gradient echo MRI measurements. Before these tools became available, only magnitude information was used; ϕ was ignored, even if it contained information about local χ changes between tissues.
The main problem in using the raw ϕ data is that they show some artifacts and they do not contain only local voxel information, but also some from effects of background fields. So, before using phase data in χ-based imaging techniques, the data need to be pre-processed by **unwrapping** and **background field removal** algorithms. Without them, the χ-induced field perturbations in MRI may be considered only source of noise and artifacts, without any significant meaning.
After the ϕ pre-processing, the proper χ-based map can be obtained.
We are going to describe the pre-processing stages first, and then the two techniqes SWI ([4], [14]) and QSM [18], [1], [15], [12], [21]).
Below, a brief representation of the GRE sequence is given.

Gradient Echo Sequence

The most frequently used sequence in susceptibility-based imaging techniques is called the **Spoiled Gradient-Recalled-Echo - SPGR or GRE sequence**. A scheme is reported in Fig. 1.6. About the magnitude images (Eq. 1.6), in the first row, a single- or a multi-exponential decay occurs, in order to allow the use of single echo or multiple echoes respectively. About the phase images ϕ (Eq. 1.6), in the second row, they measure the local frequency offsets relative to the Larmor frequency ([14]), which is related to the magnetic field perturbations.

The measured NMR signal at echo time TE is ([12]):

$$S(TE) \propto e^{-\frac{TE}{T_2^*}} \cdot e^{i\phi} \tag{1.6}$$

$$\phi = \phi_0 - \gamma \Delta B TE \tag{1.7}$$

ϕ_0 is an offset related to the RF phase and sequence adjustment. ΔB represents the main magnetic field perturbations. To estimate them, multiple measurements have to be performed.
The MRI theory and acquisition sequence explanation is behind the proposal of this work. Deeper descriptions can be found in [22], [18] and [23].

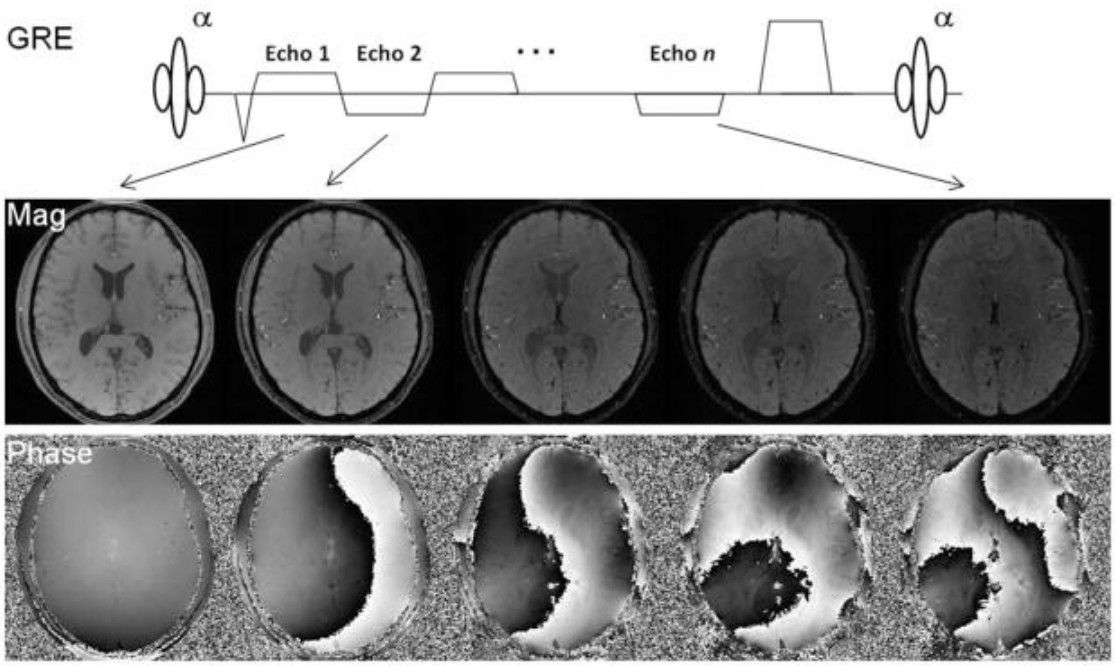


Figure 1.6: A spoiled multiecho gradient sequence ([14]). Alternating readout polarities are illustrated. The contrast in the magnitude images evolves as the echo time increases due to T_2^* decay. Phase values increase as TE increases, thus more phase wraps appear at later echoes. SWI and QSM use either single or multiple echos

1.2.1 Raw Data Processing: Phase Unwrapping and Background Field Removal

The first operation to perform on raw ϕ data is the elimination of artifacts coming from the discontinuities caused by phase wrapping ([24]). In fact, trigonometric functions, such as sine and cosine, are periodic, with period $T = 2\pi$. During the acquisition, all the angles outside of the range $[-\pi, +\pi]$ will be folded back into this range. An example of a raw ϕ map is reported in Fig. 1.7, left panel. As can be noticed from the picture, the map is not clear at all and to extract useful information from that is very difficult. Another problem to handle in this stage, in addition to the discontinuities, is the low values of **SNR - Signal-to-Noise Ratio**.

Furthermore, the background field contribution in NMR measurements worsens the wrapping effect. In fact, in order to the shimming set during the acquisition, the NMR signal contains also information of the area where the region of interest is. The total field information B_T is the sum of two contributions, the local one and the one from the background ([25]):

$$B_T = B_L + B_B \tag{1.8}$$

The origin of the background field perturbations is related to the main field inhomogeneities, increasing with the intensity of the field, with the inaccuracies in the shimming setting and with the external χ sources ([14], [1]). So, to obtain the local $\chi(r)$ distribution, namely whose of the region of interest, the external contribution has to be eliminated.
Ideally, the background effect may be removed with a previous reference scan, in which the

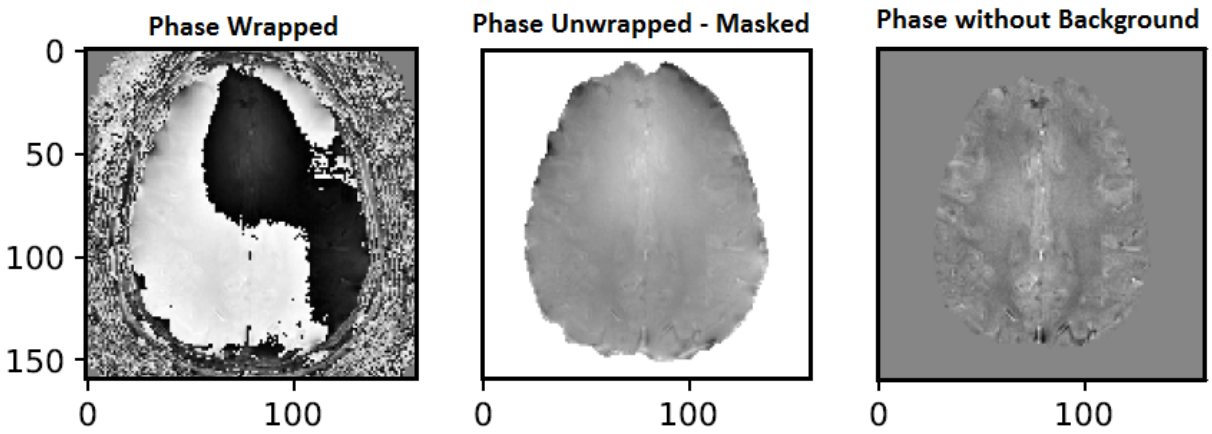


Figure 1.7: Left panel: raw ϕ data from NMR-GRE measurements. Central panel: same data after the application of an unwrapping algorithm. The data were also masked. Right panel: same data after the application of a background removal algorithm (LBV) - data from [10]

region of interest is replaced with a uniform region with known susceptibility. However, this approach is impracticable in the clinical area. So, other post-acquisition techniques have to be implemented.
Fortunately, B_B and B_L have different characteristics, thanks to which it is not too difficult to separate these two contributions, after the acquistion, and to eliminate the background one. In fact, B_L is small in intensity with fast spatial variations. On the contrary, B_B is high in intensity and the spatial variations are slower. The main difference, exploited directly or indireclty in background removal algorithms, is that B_T may be described as harmonic function ([26]). Then, background removal algorithms are often solvers for differential equations. In fact, Laplace's equation may be solved to obtain the background field function B_B ([25]):

$$\Delta B_B = 0 \tag{1.9}$$

or Poisson's equation may be solved to obtain the local field expression B_L:

$$\Delta B_L = \Delta B_T \tag{1.10}$$

There are also algorithms which work as high-pass filters, exploiting the fact that the signal variations which we are interested in are faster varying in space than the ones associated to the background, so they correspond to high spatial frequencies range.

Some of the most used background field removal algorithms are: **PDF - Projection onto Dipole Fields**, ([27],[28]), **SHARP - Sophisticated Harmonic Artifact Reduction for Phase data** and **RESHARP - Regularization Enabled SHARP** ([29]), **LBV - Laplacian Boundary Value** ([25]). In Fig. 1.7, right panel, a ϕ map after a background removal algorithm is shown. In that case, the LBV technique was used.

1.2.2 Susceptibility Weighted Imaging: SWI

Susceptibility-Weighted Imaging (SWI) ([14] , [4]) uses both T_2^* magnitude and ϕ data from gradient echo sequences ([14], [4]). Taken individually, these two maps carry only half of the

useful information contained in a NMR acquisition (Eq. 1.6).
Before, we have noticed that the construction of a χ-based map is possible thanks to the relationship between χ and the measured ϕ data. Actually, it is not only the ϕ map which contains χ-based contrast information; also the magnitude map does ([14]). Using both, tissue contrast is reinforced. To enhance the intensity variations, many stages have to be followed, to combine magnitude and phase information. This combination is not unique, there are different ways to do this. In Fig. 1.8, a scheme of the most common procedure used nowadays is shown.

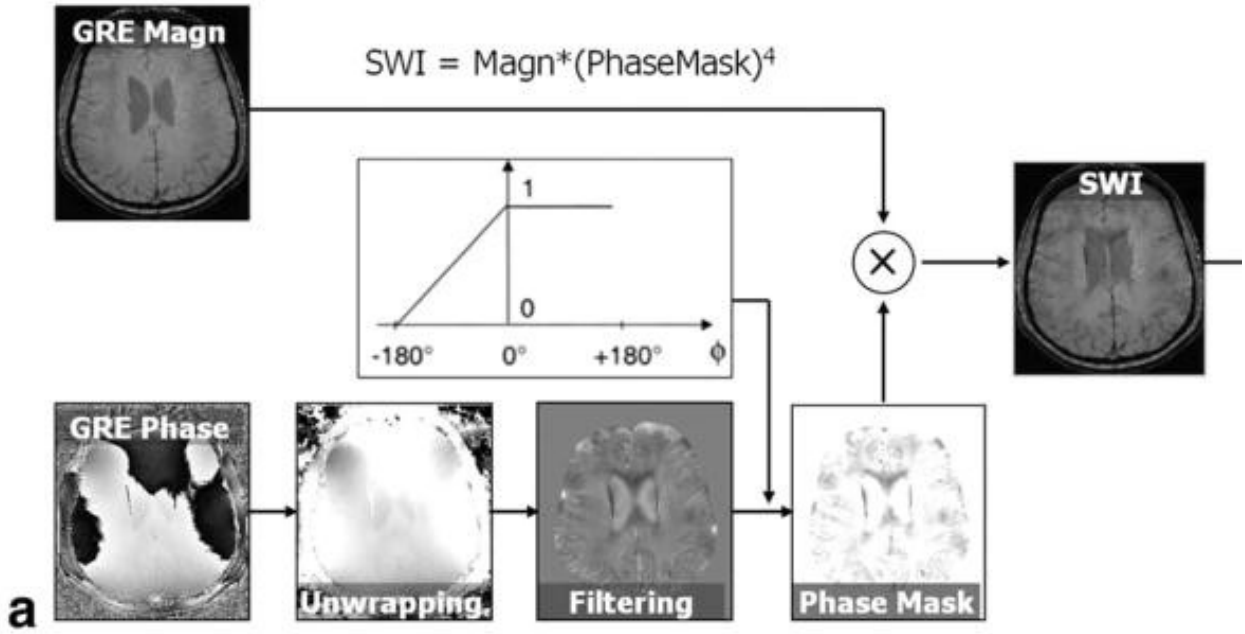


Figure 1.8: Scheme of operations which have to be performed in SWI technique ([14]). GRE phase data are processed with unwrapping and background field removal (high-pass filtering) algorithm. The returned map is transformed in a mask with intensity values between 0 and 1. Then, a pointwise multiplication is done between this mask and the GRE magnitude mask, n times. In here, n equal to 4

The GRE phase map has to be pre-processed, with an unwrapping and background field removal algorithm. Specifically here, a high-pass filtering operation is performed. This kind of filter exploits the fact that the spatial variations of the background field are slow, at least slower than the signal under consideration. The correspondent frequencies are indeed deleted with a high-pass filter. Then, the resulting ϕ map is transformed into a mask ϕ_{mask} with intensity values between 0 and 1. A pointwise multiplication between GRE magnitude map M and this mask is performed, n times ($[M * \phi_{mask}]_n$). The value of n depends on the phase differences and o the contrast-to-noise ratio values. If there is no any specification, like in this situation, it is often set to be equal to 4 ([14]).

A SWI map provides the evidence of particular contrast, different from the one based on the spin density, T_1, T_2 or T_2^*. It helps, for example, to enhance the differences in contrast between WM and GM, or between fat and water. The main drawback of SWI is that it does not provide a quantitative susceptibility map: it is like a weighted-sum of the magnetic

properties of the tissues. Thus, SWI does not allow the local magnetic properties of the single voxels to be explored. To do this, the phase map has to be deconvolved and a Quantitative Susceptibility Mapping algorithm has to be performed.

1.2.3 Susceptibility and Field Perturbation

A particle (molecule, atom, ..), polarized by an external magnetic field $\vec{B_{ext}}$, generates in turn a magnetic field $\vec{B_\mu}$, that interacts with the external enviroment depending on the properties of the material. This generated field is called the **chemical-shift shielding field** from the internal point of view, while the χ **inhomogeneity field** forms the external one. The induced field $\vec{B_\mu}$ is related to the magnetic properties looked for, and it also adds information to the NMR phase measurements. For this reason, MRI is a good tool to investigate χ: magnetic properties are almost directly available from measurements and a non-invasive procedure has to be performed on the patient.
With GRE sequences, shifts in the magnetic field ΔB coming from the chemical-shift shielding field which can be evaluated. ΔB is direcitly proportional to the phase shifts $\Delta\phi$:

$$\Delta\phi = \frac{\gamma}{2\pi}\Delta B = 42.57 \cdot 10^6 \Delta B \tag{1.11}$$

γ is the ^{1}H gyromagnetic factor.
Each voxel in an image represents a particle, and it generates a field that influences the surrounding voxels. Also, the surrounding voxels generate fields which influence it. In other words, the magnetic field at any voxel is a superposition of all the dipole fields generated by the neighborhood. For simplicity, each voxel is modelled as a dipole. A scheme of the dipole field ($\vec{B_\mu}$) lines is reported in Fig. 1.9, left panel. In the central panel of the same picture, the z-component of the dipole field is reported. To evaluate the effect of the field in the $\vec{r}$ position, the angular distance θ between $\vec{r}$ and the $\vec{z}$-axis has to be known. As we can notice looking into the right panel, the effect of the field is equal to zero at the magic angle θ_m, equal to 54.7°.

$\chi(r)$ distribution is related to the magnetization field $\vec{M}$; $\phi(r)$ distribution is related to the induction field $\vec{B}$. $\vec{M}$ and $\vec{B}$ are related each other. Considering that dipole fields, voxel representatives, are shift invariant, χ and ϕ are related each other by a convolution operation in the space domain. The response function of this operation is the **unit dipole field**, which from here on out we are going to call $D(r)$. Before, we referred to it as B_μ. Below, the mathematical formalization of these concepts is described.

When a material is inserted in a strong, externally applied, static magnetic field, the material reacts. The magnetic field which can be measured takes into account this reaction. It depends not only on the specific properties of the area under examination, but also on the shape and the orientation of regions with different χ relative to the external static field ([11], [12]). The behaviour depends on the **Maxwell's Equations**, in a current free region:

$$\nabla \cdot \vec{B} = 0 \ , \ \nabla \times \vec{H} = 0 \tag{1.12}$$

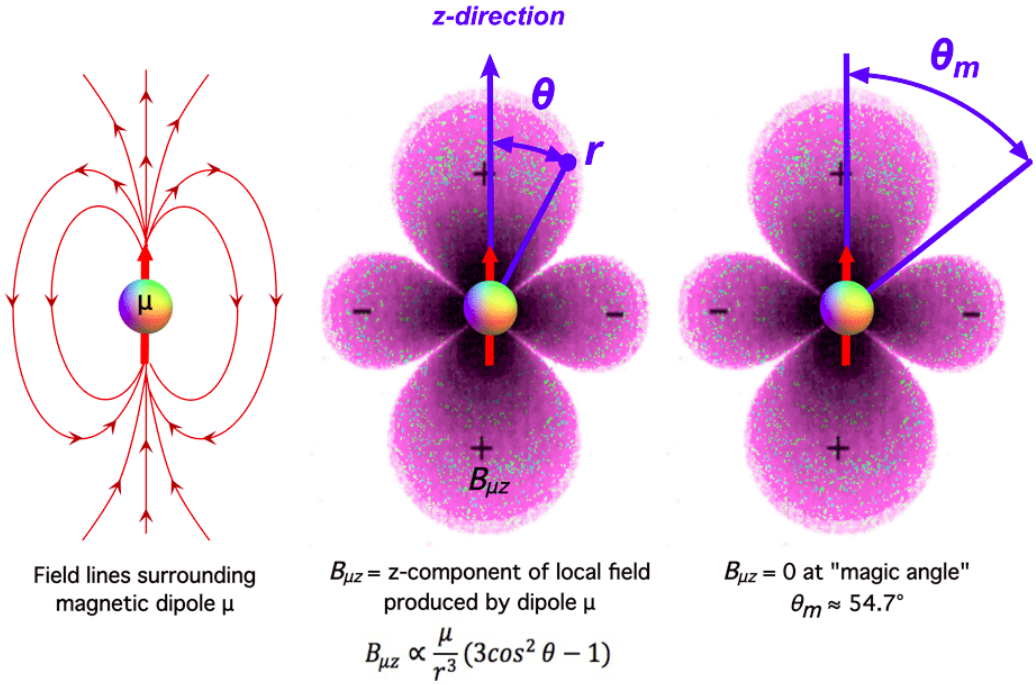


Figure 1.9: Scheme of the dipole magnetic field $B_{\mu z}$ produced by a magnetic dipole ([30]). Left panel: field lines. Central panel: relationship between the effect of the field in the $\vec{r}$ position and the angular distance from the $\vec{z}$ axis. Right panel: emphasis on the magnetic angle $\theta_m = 54.7°$, where the effect of the field is null

Thus, $B_\perp$ and $H_\parallel$ must be continuos. The direction of the perpendicular and parallel components of the fields has to be seen with respect the shape of the boundaries. If in the object under examination there are two regions, 1 and 2, with different susceptibility values, χ_1 and χ_2, then:

$$B_{1,\perp} = B_{2,\perp}\ ,\ H_{1,\parallel} = H_{2,\parallel} \tag{1.13}$$

If $\nabla \cdot \vec{B} = 0$, then $\nabla \cdot (\vec{H} + \vec{M}) = 0$. Let $\vec{H} = H_0\hat{z}$ be the external magnetic field applied. Also, consider objects with susceptibility values $\chi << 1$, that is likely if we are studying human tissues (Fig. 1.1). Under these assumptions - small χ values and field applied along the $\vec{z}$ axis:

$$\nabla \cdot \vec{H} = -\nabla \cdot (\chi H_0 \hat{z}) \tag{1.14}$$

because $\vec{M} = M\hat{z} = \chi H_0 \hat{z}$. The term $\chi \cdot H_0 \cdot \hat{z}$ is called the **magnetic charge** ([12]).
To find the solution of this equation, that allows us to find the relationship between susceptibility and field or phase shift distributions, we are going to make explicit the shape of $\vec{M}$ and $\vec{B}$, the magnetic induction field. In the situation described ($\vec{H_0} = H_0\hat{z}$) only M_z has a significant contribution ($\vec{M} = [M_x\ M_y\ M_z] = [0\ 0\ M_z(r)]$) ([31], [32]):

$$M_z(r) = \chi(r)H_0 = \chi(r)\frac{B_0}{\mu_0\mu_r(r)} = \chi(r)\frac{B_0}{\mu_0(1+\chi(r))} \tag{1.15}$$

Using the assumption $\chi << 1$, the previous equation becomes:

$$M_z(r) = \chi(r)\frac{B_0}{\mu_0} \tag{1.16}$$

From the resulting magnetization distribution $M_z(r)$ it is possible to evaluate the magnetic field generated by this distribution, $\Delta B_z(r)$. This is due to the sum of the dipolar fields generated by each element of $M_z(r)$. To do this in the spatial domain entails the use of a complex and nonlocal espression:

$$\Delta B_z(r) = \frac{\mu_0}{4\pi} \int \frac{1}{|r-r'|^3} \times \left(3\frac{M_z(r')(r-r')}{|r-r'|^2}(r-r') - M_z(r') \right) d^3r' \tag{1.17}$$

This is a convolution operation between the magnetization distribution $M_z(r)$ and the expression of the unit dipole field $D(r)$, the response function, in the spatial domain. In the k-space it becomes:

$$\Delta B_z(r) = \mu_0 \cdot M_z(r) \otimes D(r) \longrightarrow \frac{\Delta B_z(r)}{B_0} = \chi(r) \otimes D(r) \;\; (\mu_0 M_z(r) = \chi(r) B_0) \tag{1.18}$$

To solve this expression in the k-space is easier, thanks to the **Convolution Theorem**. We have to perfom the Fourier Transform $\mathcal{F}$ of $\chi(r)$ and $D(r)$ distributions, in order to have a simple point-wise multiplication problem to solve ([31], [32]):

$$\frac{\mathcal{F}(\Delta B_z(r))}{B_0} = \frac{\Delta B_z(k)}{B_0} = \chi(k) \cdot D(k) = \mathcal{F}(\chi(r)) \cdot \mathcal{F}(D(r)) \tag{1.19}$$

$$D(k) = -\frac{1}{3}\left[\frac{3k_z^2}{k_x^2 + k_y^2 + k_z^2} - 1 \right] = \left[\frac{1}{3} - \frac{k_z^2}{k^2} \right] \tag{1.20}$$

Then, to obtain the required distribution only the inverse-transformation $\mathcal{F}^{-1}$ on $\Delta B_z(k)$ has to be perfomed:

$$\Delta B_z(r) = \mathcal{F}^{-1}(\Delta B_z(k)) \tag{1.21}$$

The solution of the equation which relates $\Delta B(k)$ and $\chi(k)$ in k-space is possible thanks to the homogeneity hypothesis, which guarantees isotropic reaction of the tissue, the χ small values hypothesis ($\chi << 1$) and the use of the **Sphere of Lorentz** concept ([12], [18], [33]).

In the k-space, to solve the problem is easy also when small rotations of the magnetic field have to be taken into account. For example, if the magnetic field is rotated clockwise by an angle θ with respect the $\vec{x}$-axis, then the dipole field expression (Eq. 1.20) becomes ([31]):

$$D(k) = -\frac{1}{3}\left(3\frac{(k_z \cdot \cos\theta - k_y \sin\theta)^2}{k_x^2 + k_y^2 + k_z^2} - 1 \right) \tag{1.22}$$

We have just analyzed the **forward problem** linking the suceptibility distribution and the field shift distribution, namely how to obtain $\Delta B(r)$ starting from $\chi(r)$ data. The **inverse problem** is ill-posed. There are in fact some mathematical issues in inverting the relationship in equation Eq. 1.19 and so in obtaining a $\chi(r)$ map starting from $\Delta B(r)$ data. The quantitative susceptibility mapping is an MRI technique that allows to fix those issues. Actually, there is more than one strategy.
To understand the ill-posed nature of this problem, to describe the shape of the unit dipole field in k-space $D(k)$ is necessary. After that explanation, we are going to analyze the main strategies to fix the ill-posed inversion problem.

1.2.4 Ill-posed Problem and Quantitative Susceptibility Mapping

Using the raw phase data, directly available with GRE sequences, to obtain a high-contrast image is possible. This is due to differences in susceptibility $\Delta\chi$, which provide differences in the NMR frequencies. The intrinsic problem in using phase data is that ϕ is non-local and orientation dependent, so it is not easily reproducible. In the SWI technique, ϕ data are not processed taking into account the relationship between ϕ and χ. It may happen that, depending on the orientation-dependence, the same structures appear different in two different images. So, with SWI it is not always possible to recognise diamagnetic and paramagnetic substances. QSM instead provides a way to investigate the intrinsic property of the tissues.

We have already explained the theoretical foundations of the QSM technique. When a susceptibility distribution $\chi(r)$ is exposed to an external magnetic field, $\chi(r)$ produces changes in $\Delta B(r)$. $\Delta B(r)$ is directly related to the $\Delta\phi(r)$ distribution, which is directly measurable with a GRE sequence. If the direct problem just introduced ($\chi(r) \longrightarrow \Delta B(r)$) is formalized in the space domain, then it needs to be solved using a non local complex expression (Eq. 1.17). The use of **Fourier-based methods**, thanks to the Convolution Theorem, allows the same problem to be solved in the k-space with a local and simple equation (Eq. 1.19) ([31]).
Below, a scheme of the stages to follow (forward-problem: $\chi(r) \longrightarrow \Delta B(r)$:

1. The susceptibility distribution in the domain space $\chi(r)$ is the input data. Its Fourier transform has to be performed: $\chi(r) \longrightarrow \mathcal{F}(\chi(r)) = \chi(k)$.

2. Using Eq. 1.19, $\Delta B(k)$ can be calculated easily solving a point-wise multiplication. It is possible to solve the problem in k-space thanks to the convolution theorem and the application of Fourier-based methods.

3. To obtain the field perturbation distribution in the space domain, the Fourier inverse-transform of $\Delta B(k)$ has to be performed ($\Delta B(r) = \mathcal{F}^{-1}(\Delta B(k))$. The returned $\Delta B(r)$ is the field distribution due by a susceptibility distribution inserted in an external strong magnetic field.

Actually, we usually do not know the susceptibility distribution of the tissue. We do know the field perturbation map, related with the ϕ data measured, and we would like to know the susceptibility values of the tissues in order to obtain an even higher-contrast map. So we have to solve an inverse problem ($\Delta B(r) \longrightarrow \chi(r)$).
To work in k-space is convenient, as before. Inverting the equation Eq. 1.19 the following expression is obtained ([11], [12], [1]):

$$\chi(k) = \frac{\Delta B(k)}{B_0 \cdot D(k)} \ , \ D(k) = \left[\frac{1}{3} - \frac{k_z^2}{k^2}\right] \tag{1.23}$$

The function $D = D(k)$ has some zero-values in k-space, along the surface of a double cone oriented along the $\vec{z}$ direction and with its surface inclined by an angle $54.7°$ with respect to

$\vec{z}$. This is the **magic angle** θ_m, already mentioned in explaining the effect of a dipole field with respect the angular distance (see Fig. 1.9). The shape of this cone is shown in Fig. 1.10.

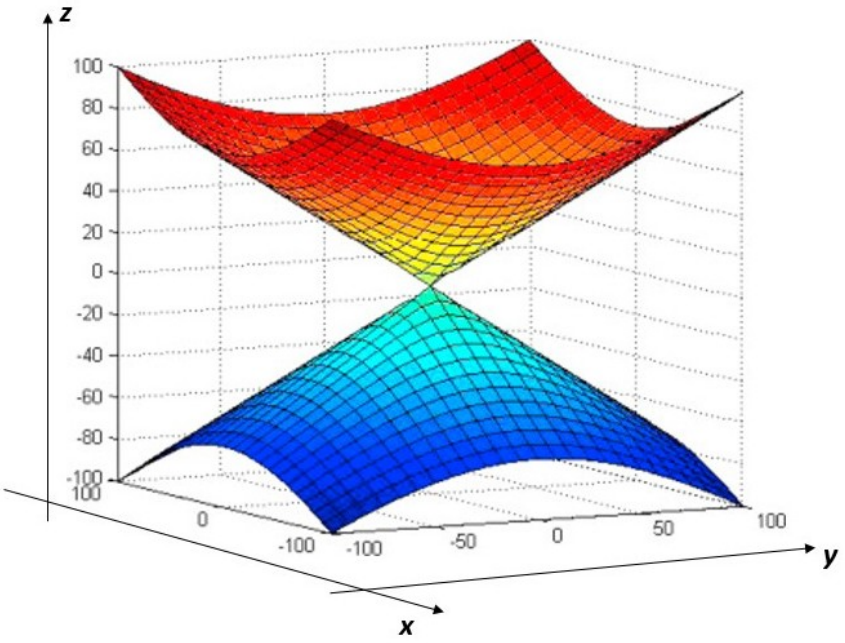


Figure 1.10: Double cone surface which solves the equation $D(k) = 0$. $D(k)$ is the unit dipole field in k-space. The cone axis is directed along $\vec{z}$ axis, the same direction of the externally magnetic field applied. The cone is inclined from the $\vec{z}$ axis by the magic angle θ_m 54.7°

The inverse problem is ill-posed because, according to the equation 1.23, we have to divide by an expression which assumes zero-values in some points of the k-space. Actually, the imaging problem is discretized; it is possible to choose the sampling points in order to avoid the precise zero-values of the double-cone. But, even if we do it well, also the discrete problem is still ill-conditioned: the dipole response in k-space may be arbitrarily close to zero, and this causes a large amplification of noise, errors and artifacts.

To reproduce a quantitative susceptibility map, finding a solution to the ill-posed inversion, different approaches have been implemented. They may be divided in two main classes: **multiple-orientation** and **single-orientation methods** ([7], [1], [15]).
The first type compensates the impossibility to sample the entire k-space in one acquisition using more than one scan of the same object. In each aquisition, the object changes its orientation with respect to the main magnetic field. Sampling the k-space many times N, with $N > 3$ ([5]), it is possible to recover all the values of k-space and to obtain the required susceptibility map. This class of techniques provides very accurate and precise susceptiblity maps. As a drawback, they require a quite long acquistion time, because more than one acquisition has to be done for each susceptibility map. Also, the mathematical processing to obtain the susceptibility map requires to know the affine transformation describing the movements with a pretty good precision.

The other category of methods is instead based on a single measurement. In here, regularization or conditioning strategies are necessary to select a unique solution for a given field. They are in turn divided into two categories ([7]):

1. **Non-Iterative K-Space Techniques** or Threshold-Based Single Orientation Methods (**TSO**)

2. **Iterative Image Space-Based Optimization Techniques** or Regularised Single-Orientation Methods (**RSO**)

They are numerical strategies, faster than the other ones because of the shorter acquisition time needed. On the other hand, the returning susceptibility map loses in precision and accuracy.
We are going to analyze, in the following sections, two specific techniques: **COSMOS** and **TKD**. The focus on these two approaches is due to the fact that these specific reconstructions were used as a control during the experimental stages of this work.

1.2.5 Multiple-Orientation Approach: COSMOS

The **COSMOS -Calculation Of Susceptibility through Multiple Orientation Sampling** technique is the gold-standard multiple-orientation approach used to fix the ill-posed QSM problem ([7], [5], [14]). As we have already noticed, the idea behind this technique is to image the same object at multiple angles θ with respect the main magnetic field. In this way we can recover all the values in the k-space, overcoming the zero-values issue. To fix the problem is possible with numerical approaches as well. But with them the unit dipole field function $D(k)$ has to be changed, some k-space points are not taken into account or *a priori* knowledge has to be used. In this way, a susceptibility distribution is still obtained, but not using the true k-space values.

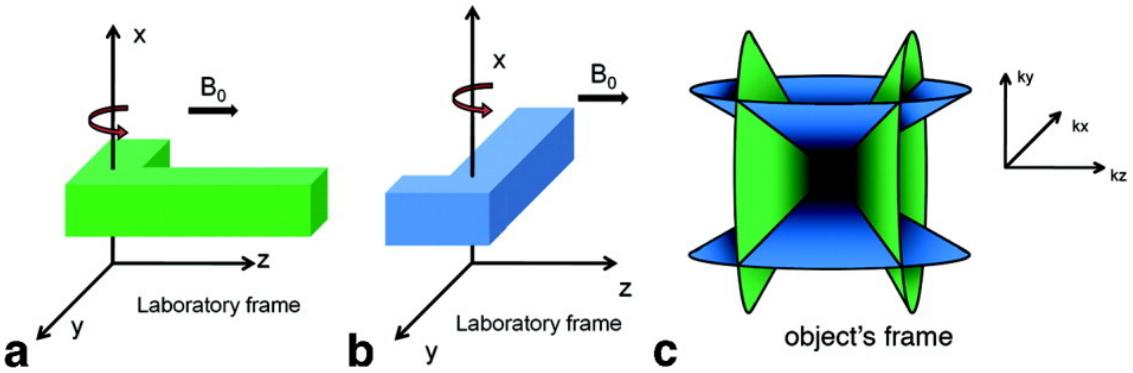


Figure 1.11: Scheme of the COSMOS approach to solve the ill-posed QSM problem ([5])

A scheme of the idea behind COSMOS is reported in Fig. 1.11. When the frame of reference of the object is the same as that of the laboratory - with $\vec{z}$ directed along the main magnetic field $\vec{B_0}$ - the point-wise multiplication to solve in k-space is:

$$\Delta B(\vec{k}) = \left(\frac{1}{3} - \frac{k_z^2}{k^2}\right) \cdot \chi(\vec{k}) \tag{1.24}$$

where $\vec{k}$ is the magnitude of a Fourier domain coordinate vector and k_z is the projection of $\vec{k}$ onto the $\vec{z}$-axis, that matches the direction of the magnetic field.
If the object is rotated with respect $\vec{B_0}$, Eq. 1.24 becomes ([7], [5]):

$$\Delta B(\vec{k}) = \left(\frac{1}{3} - \frac{k_{zp}^2}{k^2}\right) \cdot \chi(\vec{k}) \tag{1.25}$$

where z_{zp} is the projection of the object coordinate vector $\vec{k}$ onto the direction of the main magnetic field. For the sake of simplicity, let's assume that the rotations of the object are only around the $\vec{x}$-axis by angle θ. Then:

$$k_{zp} = k_z \cdot \cos\theta + k_y \cdot \sin\theta \tag{1.26}$$

If the object is rotated by N different angles $\{\theta_i\}_{i=1}^N$, then we have:

$$\begin{bmatrix} \left(\frac{1}{3} - \frac{k_z \cdot \cos\theta_1 + k_y \cdot \sin\theta_1}{k^2}\right) \\ \left(\frac{1}{3} - \frac{k_z \cdot \cos\theta_2 + k_y \cdot \sin\theta_2}{k^2}\right) \\ .... \\ \left(\frac{1}{3} - \frac{k_z \cdot \cos\theta_N + k_y \cdot \sin\theta_N}{k^2}\right) \end{bmatrix} \cdot X(\vec{k}) = \begin{bmatrix} \Delta B_1 \\ \Delta B_2 \\ .... \\ \Delta B_N \end{bmatrix} \tag{1.27}$$

It has been proven that there is a set of angles $\{\theta_i\}_{i=1}^N$ to fullfill this criterion for every $N \geq 3$ ([5]).
In Fig. 1.13 an example of COSMOS susceptbility reconstruction is reported. The data are from [10]. For this particular reconstruction, 12 different head-orientations were used. In Fig. 1.12, there is the first ϕ map, after the unwrapping and the background field removal. The map reported was acquired setting the $\vec{z}$ axis of the head in the same direction of the main magnetic field $\vec{B_0} = B_0\hat{z}$.

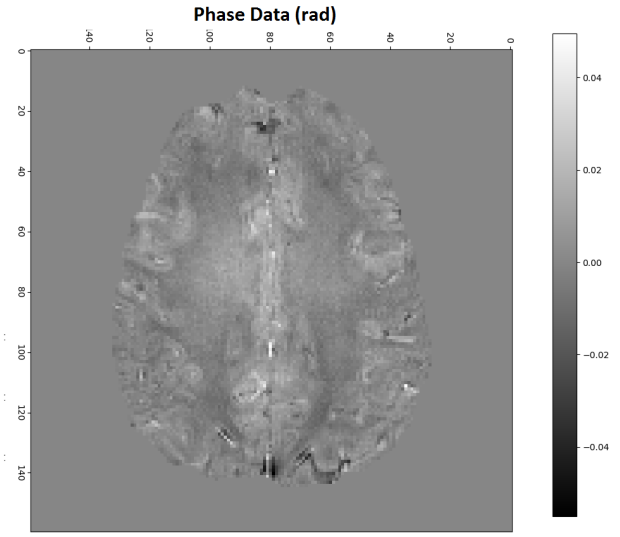


Figure 1.12: ϕ brain map, after unwrapping and background field removal ([10])

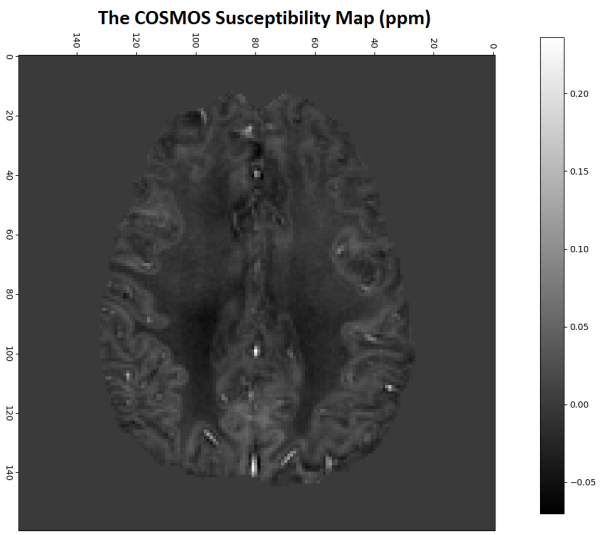


Figure 1.13: χ COSMOS map ([10]), obtained with 12 acquisitions

COSMOS and STI: Different χ Models

The COSMOS approach models the susceptibility as a scalar and isotropic property, in order to solve the set of equations reported in Eq. 1.27. At the beginning of this chapter, we have explained that there are some situations in which this assumption is correct, namely when an homogeneous material answers in a uniform way to the application of an external magnetic field. In this case, it is right to consider χ as a scalar property, taking into account only the isotropic component of the susceptibility.
However, in other situations this assumption is not properly matched with the actual physical behaviour of the tissue. Some regions react in a different way in order to the relative direction between the main magnetic field and the tissue. The susceptibility values measured in this case should be considered being tensors. There are some studies which highlight the anisotropic χ behaviours of some human structures, especially in the central nervous system. One example is the white matter microstructure ([19], [6]).
There are some techniques, related to the QSM problem, which model χ as a tensor. They are called **STI - Susceptibility Tensor Imaging** techniques.

Also STI, as the COSMOS method, uses multiple orientation scans of the same object to reconstruct its χ map. COSMOS, without considering the orientation dependence of χ, actually returns the average value of the susceptibility along the multiple acquisitions. There is not a one-to-one correspondence between the phase and susceptibility map. In some situations it is better to use other techniques, such as STI, because they take into account the orientation dependence of the susceptibility, considering it as a tensor ([14], [6]):

$$\bar{\chi} = \begin{bmatrix} \chi_{11} & \chi_{21} & \chi_{31} \\ \chi_{12} & \chi_{22} & \chi_{32} \\ \chi_{13} & \chi_{23} & \chi_{33} \end{bmatrix} \tag{1.28}$$

Let $\hat{n} = [n_x \ n_y \ n_z]$ be the direction of the main field. Then, the field pertubation distribution becomes ([10]):

$$\Delta B(k) = \frac{1}{3}\hat{n} \cdot \bar{\chi} \cdot \hat{n}^T - \hat{n}^T \cdot \vec{k}\frac{\vec{k}^T \cdot \bar{\chi} \cdot \hat{n}^T}{\vec{k}^2} \tag{1.29}$$

where $\vec{k}$ is a vector of k-space coordinates. If $\hat{n} = [0\ 0\ 1]$, that means that the main field is directed along the $\vec{z}$ axis, the last expression becomes:

$$\Delta B(k) = \left(\frac{1}{3} - \frac{k_z^2}{k^2}\right)\chi_{33} - \frac{k_z}{k^2}(k_x\chi_{13} + k_y\chi_{23}) \tag{1.30}$$

In no-tensor analysis, the off-diagonal terms χ_{13} and χ_{23} are considered equal to zero.
In the solution described, with $\hat{n}$ alligned with the $\vec{z}$ axis, $\chi_{33}(r)$ is usually considered as final susceptibility map.

Consider χ a symmetric tensor ($\chi_{21} = \chi_{12}$, $\chi_{31} = \chi_{13}$, $\chi_{32} = \chi_{23}$). In that case, χ is a second order, or rank-2, tensor ([14]). So, six independent variables have to be determined. The STI application requires at least 6 different measurements, changing the orientation between the main field and the object ([14]). To provide less measurements is also possible, but to assume χ cylindrical symmetry and to combine DTI measurements is necessary ([14], [34]). The STI tool provides a potential fiber-tracking method, as an alternative to diffusion-based tractography ([6]).

Both COSMOS and STI have advantages and drawbacks, depending on the measurement to be performed. Both are more precise and accurate than numerical methods, because they provide the susceptibility reconstruction without numerical regularization, spatial smoothing or incorporation of prior information. Thanks to the multiple acquisitions, they return high-SNR maps. The main drawback relates to the long acquisition times needed.
In Fig. 1.14, there is a comparison between the returned map of these two methods: the STI reconstruction on the left and the COSMOS one on the right.

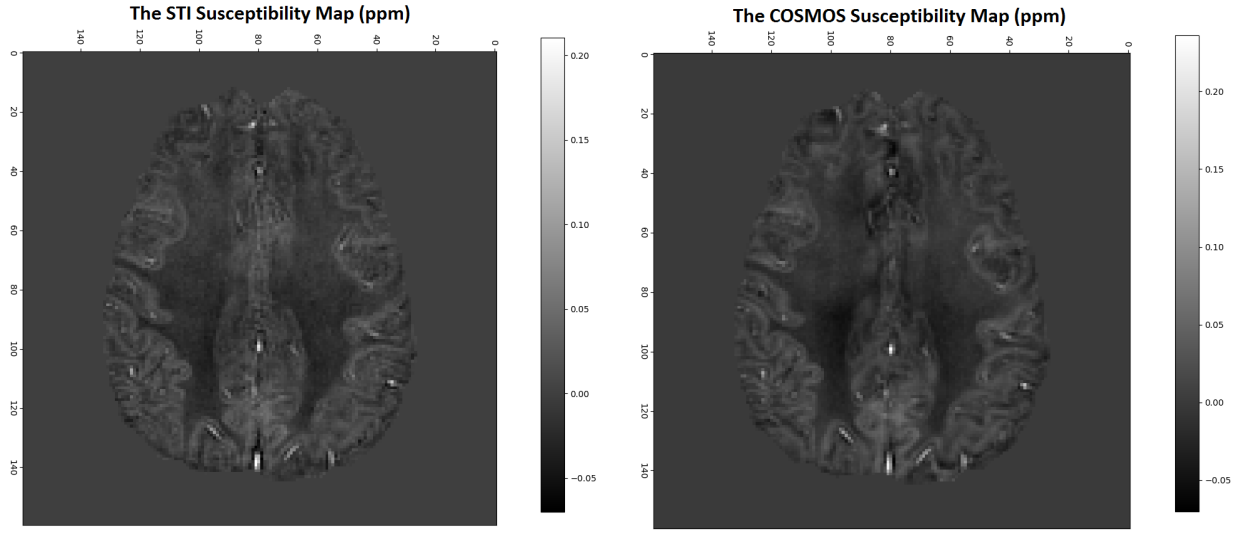


Figure 1.14: Left panel: χ STI map ([10]). Right panel: χ COSMOS map ([10]). Both of the susceptibility reconstructions were obtained with 12 different acquisitions

1.2.6 Single-Orientation Approach: TKD

The **TKD - Threshold K-space Division** is a numerical, non-iterative technique, which is user-friendly and fast to solve the ill-posed problem in the susceptibility map reconstruction. It consists in changing the dipole kernel function $D(k)$ in k-space for regions which are close to the zero-double-cone surface ($D(k) = 0$, Fig. 1.10). With the TKD approach, Eq. 1.23 is solved using the function $\tilde{D}(k)$ instead of $D(k)$. This new function is defined as ([7], [15]):

$$\tilde{D}(k) = \begin{cases} \left(\frac{1}{3} - \frac{k_z^2}{k^2}\right) \ , \ |D(k)| < \alpha \\ sign\left(\frac{1}{3} - \frac{k_z^2}{k^2}\right) \cdot \alpha \ , \ |D(k)| > \alpha \end{cases} \tag{1.31}$$

$$|D(k)| = \left|\frac{1}{3} - \frac{k_z^2}{k^2}\right| \tag{1.32}$$

α is a threshold value in k-space set by the user.
If α is too small, then the algorithm is forced to consider values in k-space very close to the zero-double cone (see. Fig. 1.10). In this situation, it is possible to store the majority of the k-space values, but errors and noise are more evident in the resulting map, because values very close to zero are taken into account.
In the opposite situation, where the threshold α is set too large, only values far from the cone are considered, so the returning map is not noisier as before. On the other side, the result loses a lot of details, because the algorithm is not considering a broad range of the high-values in the k-space. They are representative of the high frequencies, which are in turn representative of the image details. The risk is to obtain a too smooth map. Also, a systematic understimation of χ values occurs with increasing α values.
A compromise has to be found, as when **edge detection** and **noise reduction filters** are used in a classical image processing. Remember that there is not an absolute compromise: the choice of α always depends on the kind of object or tissue that has to be analyzed.

In Fig. 1.15, three susceptbility maps are reported, all of them obtained with the TKD tool, but the choice of the threshold α is varied. Starting from the left, it was set respectively equal to 0.01, 0.1 and 0.4. To obtain those maps, we used the ϕ data from [10] as input and we performed the TKD algorithm (Eq. 1.31) setting different values of threshold α.

- Left panel, $\alpha = 0.01$: here the set threshold is too small, in fact noise and artifacts are so evident in the resulting map.

- Central panel, $\alpha = 0.1$: the set threshol is a good compromise, there is not too much noise and also the main structures and details in the image can be recongnised.

- Right panel, $\alpha = 0.4$: the set threshold is too high, the resulting map is too much smooth to recognise any structure in the image.

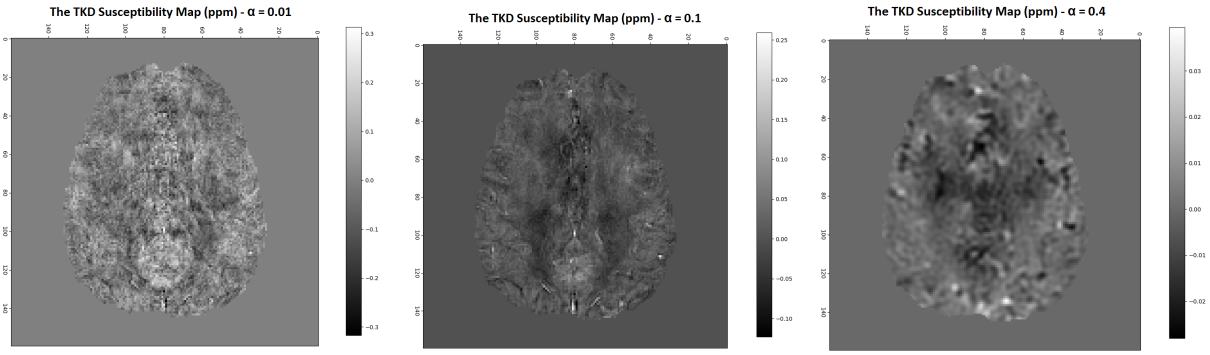


Figure 1.15: χ TKD map obtained with different values of the threshold α. Left panel: $\alpha = 0.01$. Central panel: $\alpha = 0.1$. Right panel: $\alpha = 0.4$. Maps were obtain using the ϕ map from [10] as input and performing on those the TKD algorithm setting different values of α

Figs. 1.16, 1.17 and 1.18 shows the k-space of the three TKD reconstructions - respectively with α equal to 0.01, 0.1 and 0.4.

In these three figures, projections of the 3D k-space from the panels in Fig. 1.15 are shown. In the first and second rows, there are $\vec{x}$ and $\vec{y}$ projections. They look very similar each other because the axis of the cone is directed along the $\vec{z}$ direction. With these projections, we can observe the sagittal and the coronal sections. In the third row, $\vec{z}$ projections are shown, to observe axial sections.
We can notice, as expected, how much the extent of thickness of the double cone increases with the increasing of α (Eq. 1.31, Fig. 1.10).

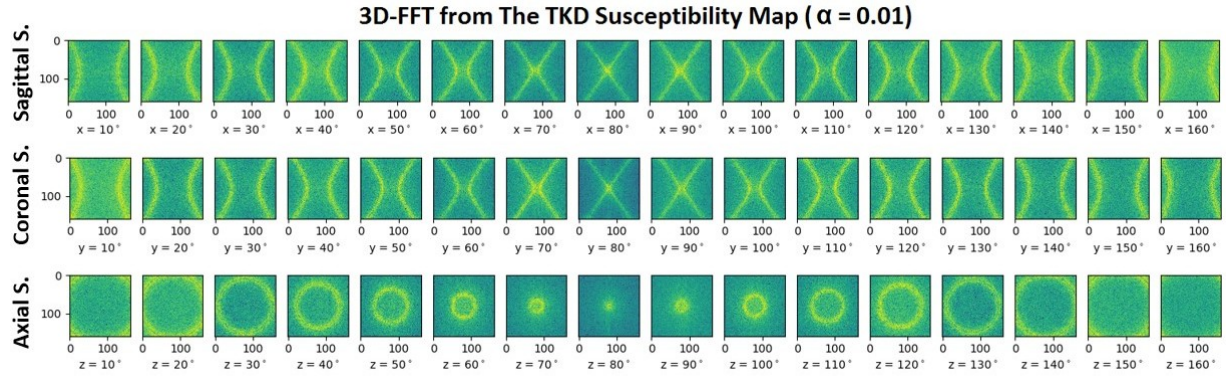


Figure 1.16: The 3D Fourier transform of a TKD suscepibility map obtained setting $\alpha = 0.01$. Sagittal, coronal and axial slices are reported respectively in the first, second and third row. The starting image has size (160,160,160), the same is for the Fourier transform. One axial slice of the original map is in Fig. 1.15, left panel

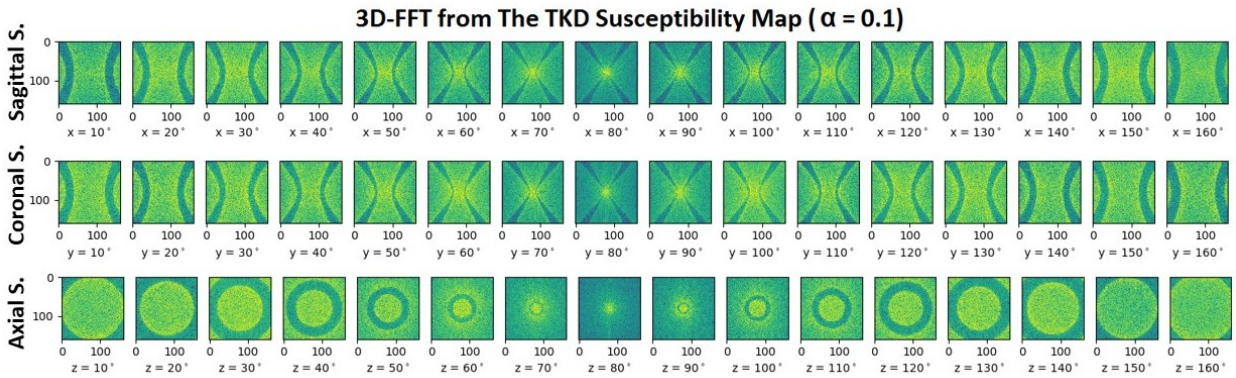


Figure 1.17: The 3D Fourier transform of a TKD suscepibility map obtained setting $\alpha = 0.1$. Sagittal, coronal and axial slices are reported respectively in the first, second and third row. The starting image has size (160,160,160), the same is for the Fourier transform. One axial slice of the original map is in Fig. 1.15, central panel

In Fig. 1.19, a comparison between single- (TKD) and multiple- (COSMOS) orientation approaches to QSM is reported. We can notice, as we have already said, that the COSMOS map provides a more accurate and less noisy map than the TKD technique.

There are other single-orientation approaches to the QSM problem, which use an iterative strategy. The starting point of these approaches is the assumption that modulus, phase and susceptibility maps have similar edges to the underlying structure. So, they use *a priori* knowledge from the modulus map.
The most known ones are **MEDI - Morphology Enabled Dipole Inversion** ([35]) and

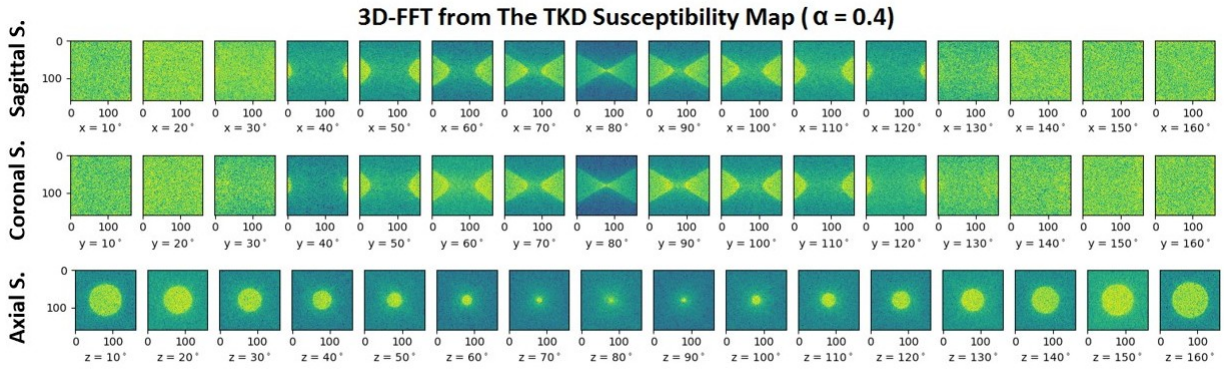


Figure 1.18: The 3D Fourier transform of a TKD suscepibility map obtained setting $\alpha = 0.4$. Sagittal, coronal and axial slices are reported respectively in the first, second and third row. The starting image has size (160,160,160), the same is for the Fourier transform. One axial slice of the original map is in Fig. 1.15, right panel

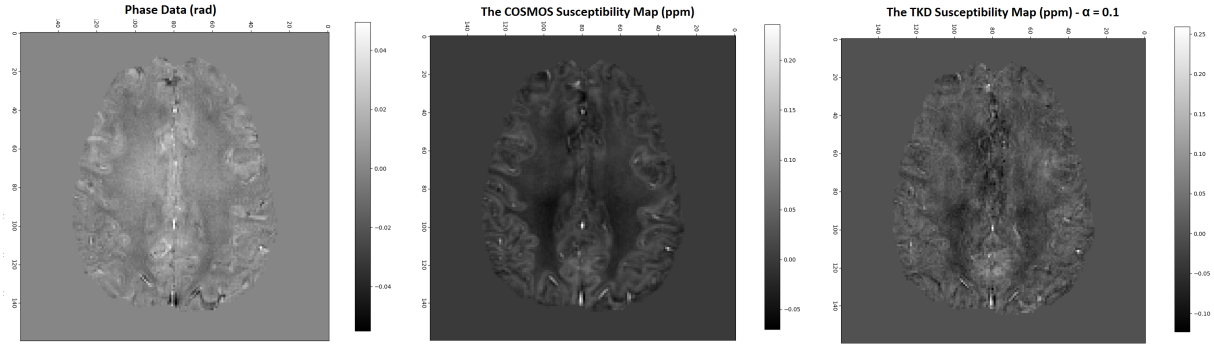


Figure 1.19: Left panel: ϕ brain map, after unwrapping and background field removal ([10]) . Central panel: the COSMOS χ map, obtained with 12 acquisitions ([10]). Right panel: the TKD χ map, with threshold $\alpha = 0.1$ ([10])

HEIDI - Homogeneity Enabled Incremental Dipole Inversion ([36]). To explore in details these techniques is beyond the proposal of this work.
A good overview may be found in [37].

1.3 Susceptibility Mapping Applications

In this chapter, we have described the magnetic susceptibility quantity and its potential as a biomarker which makes interesting to study this property (Sec. 1.1). Then, we have explored the MRI technique to observe it; in particular, we have focused on Quantitative Susceptibility Mapping (Sec. 1.2.3 and 1.2.4) and on two specific strategies, COSMOS (1.2.5) and TKD (Sec. 1.2.6).
To conclude this chapter, a list of some susceptibility mapping applications is provided. A more detailed overview can be found in [1], [15] and [14].

- **Cerebral Vascular Pathologies** ([14], [1])
  Susceptibility-based techniques, especially QSM, are used for the diagnosis of cerebrovascular disorders. They are usually divided in three main classes: vascular malformations (i.e. **AVM: ArterioVenous Malformations**), restrictions in blood flow (i.e.

DVA: Developmental Venous Anomaly, **CCM: Cerebral Cavernous Malformations**) and hemorrhages (i.e. **CAA: Cerebral Amyloid Angiopathy**, **TBI: Traumatic Brain Injury**).

The source of contrast in detecting hemorrhages is due to the χ changes in blood products oxygenation-related (Fig. 1.20). Hemoglobin is normally carried by red blood cells bounded with oxygen. In this form, it is called oxyhemoglobin, which is diamagnetic. During a hemorrhage, blood degradation happens ([15]): hemoglobin changes its magnetic properties and oxyhemoglobin becomes deoxyhemoglobin (weakly paramagnetic with 4 unpaired electrons in Fe^{2+}), methemoglobin (strongly paramagnetic with 5 unpaired electrons in Fe^{3+}) and hemosiderin (superparamagnetic with possible magnetic domain formation or ferromagnetic).

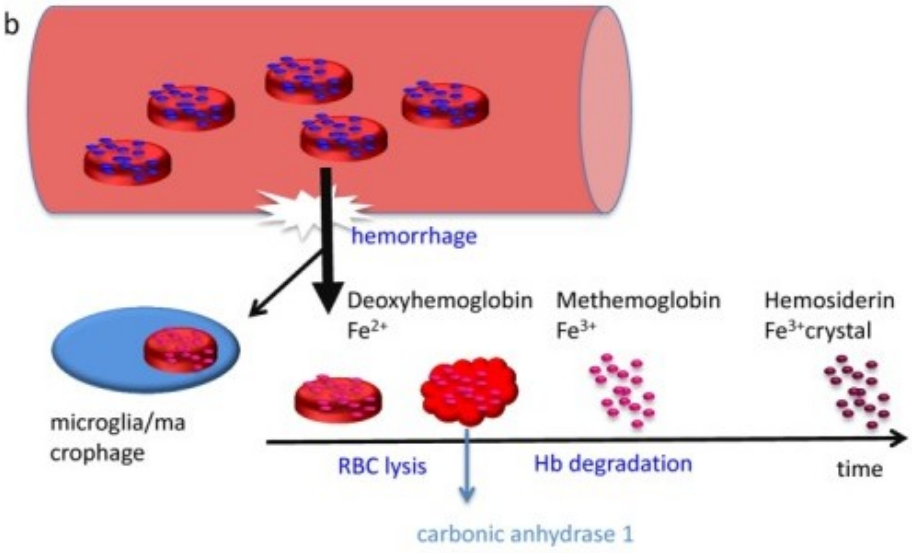


Figure 1.20: A scheme of blood degradation which occurs during a hemorrage. Oxyhemoglobin, diamagnetic, carryed by the red blood cells in a normal state, overflows changing its magnetic properties. It becomes deoxyhemoglobin, paramagnetic with 4 unpaired electrons per heme, methemoglobin, paramagnetic with 5 unpaired electron per heme, and hemosiderin, superparamagnetic with possible magnetic domain formation or ferromagnetic ([15])

Another blood disorder which may be studied with the QSM recontruction is β-**thalassemia** ([15]). It is a disease that implies the loss of hemoglobin, involved in oxygen transport.

- **Lesion Classification** ([14], [1], [15])

  Another clinical application that benefits from the QSM technique is lesion distinction. In fact, an anatomical susceptibility map allows to separate the different kind of intracerebral lesions, like those due to TBI, ischemia, tumors and so on. An important point is to make a distinction between calcification and blood-product-based lesions, that means to make a distinction between para- or dia-magnetic injuries. As a convention, brighter intensity in a susceptbility map represents paramagnetic susceptibility and darker intensity represents diamagnetic material.

  A lot of experiments confirmed this, using the **CT - Computer Tomography** tech-

nique as control map, which is considered the non-invasive gold-standard for this application. An advantage in using the QSM technique is that it is more accurate and also able to detect lesions in their first stages. An example of the comparison of the two techniques is shown in Fig. 1.21.

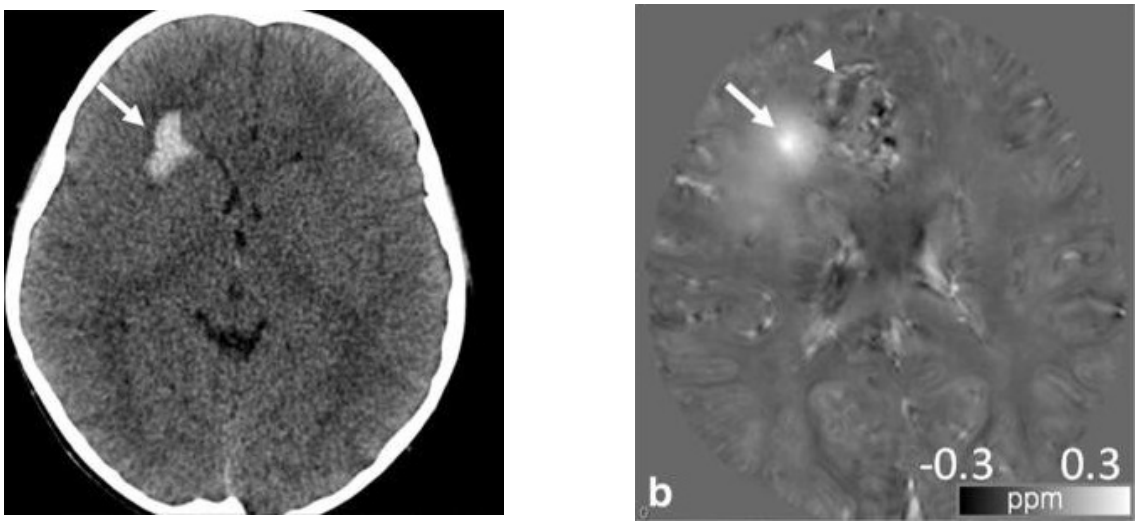


Figure 1.21: An 8-year-old with hemorrhage from an arteriovenous malformation ([14]). Left panel: CT obtained at presentation showed right frontal lobe hemorrhage (arrow). Right panel: QSM obtained a 3T shortly after the CT showed bright signal abnormality, confirming the presence of iron/hemorrhage (arrow). Additional areas of curvilinear and nodular bright signal suggest additional areas of hemorrhage, thromboses or abnornal vasculature.

- **Mineral Deposition** ([14])
  Abnormal mineral deposition is frequently related to neurological diseases. Susceptibility-based techniques are sensitive to the accumalation of compounds containing minerals. In the brain, **iron** abnormalities are called **Brain Iron Accumulation Disorders**, and they generally involve gray matter structures. Specifically, the subcortical GM is involved, including the basal ganglia, thalamus, substantia nigra and dentate nucleus. An excessive **copper** accumulation is related to **WD - Wilson's disease**. The excess is due to an abnormality in copper metabolism. It involves subcortical GM structures, especially the globus pallidus, putamen, thalamus, dentate nucleus, pons and midbrain. As we described before, studying **calcium** accumulation allows the localisation and evaluation of calcification-based lesions.

- **Neurodegenerative Diseases** ([1], [14])
  Abnormalities in mineral metabolism may be related to neurodegenerative disordes. The QSM tool provides a way to identify and quantify those abnormalities, and it also allows other biomarkers connected with neurological diseases to be studied.

  – **PD - Parkinson's Disease**: this pathology is associated with iron accumulation in specific regions, e.g. substantia nigra, and loss of dopaminergic neurons.

  – **AD - Alzheimer's Disease**: it is related with a massive presence of iron, which forms neuritic plaques and neurofibrillary tangles. This is the perfect condition for the protein β-amyloid aggregation and neurotoxicity.

– **MS - Multiple Sclerosis**: there is evidence of correlation between MS and increasing iron deposition in deep GM nuclei and lesion in WM and cortical GM. An example of a multiple-sclerosis patient brain reconstruction is reported in Fig. 1.22.

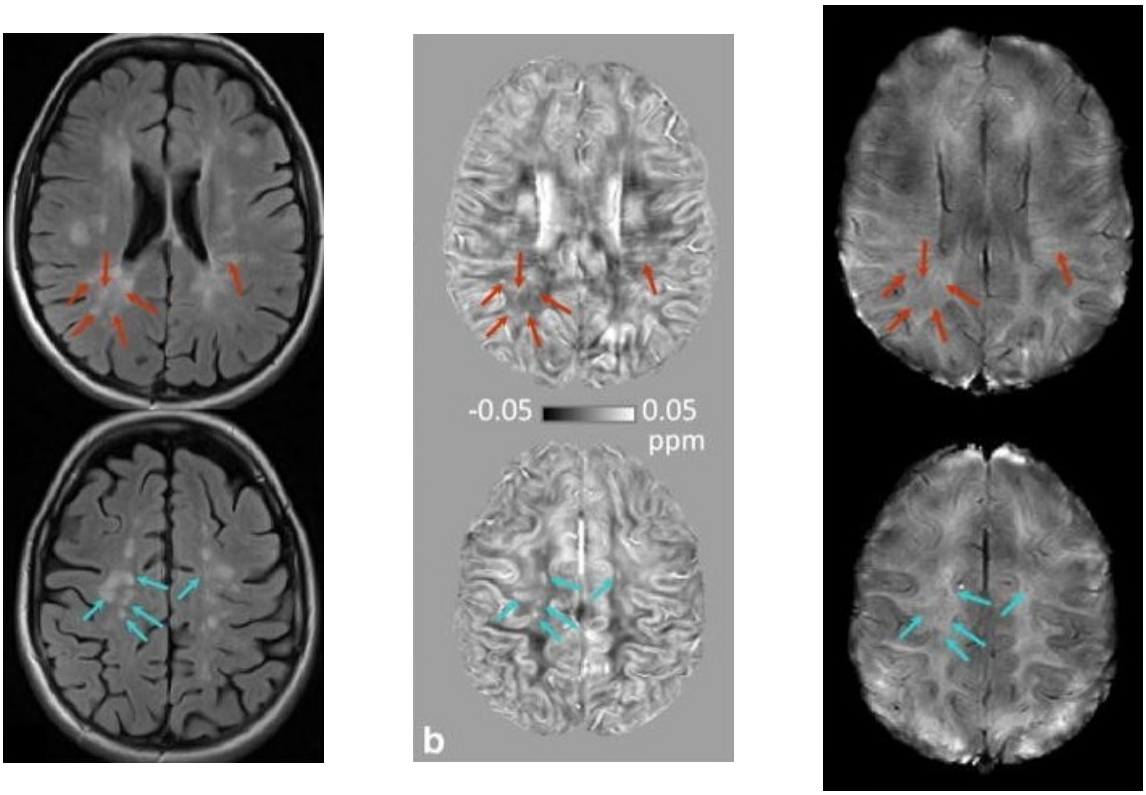


Figure 1.22: A 49-year-old multiple sclerosis patient ([14]. Left panel: MS plaques are usually best visualized on FLAIR images. Central panel: Hyperintense lesions on FLAIR may exhibit increased magnetic susceptibility on the QSM images (arrows). The size and boundaries of hyperintense lesions appear differently on QSM. This increased susceptibility may indicate demyelination or iron deposition. Right panel: SWI without minIP shows the perivenular distribution of the lesions and venous involvement

- **Non-invasive Brain Area Visualization**
  We have noticed yet that the QSM is a non-invasive MRI technique, thanks to which a lot of areas - e.g. substantia nigra pars reticulata, ventral intermediate thalamic nucleus, subthalamic nucleus and the substructures of the internal globus pallidus, common targets for **DBS - Deep Brain Stimulation** ([1]) - may be observed in details. This is a technique frequently used in neurosurgery, because, placing electrodes precisely in brain structures, it can be used to alleviate motor disorders.

Chapter 2

Convolutional Neural Networks in Image Processing

The goal of this work is to reproduce an anatomical suceptibility map of a human brain starting from GRE-MRI phase data. This procedure is called Quantitative Susceptibility Mapping (QSM), and in the previous chapter we described the physical phenomena behind it and the different approaches that can be used to reproduce the sought map. In this work, we want to reconstruct the susceptibility distribution with a neural network, specifically with a **convolutional autoencoder** with an **image reconstruction** task.
In this chapter we are going to provide an introduction to learning algorithms and to some deep learning tools. In particular:

- Section 2.1: **Neural Networks and Learning Algorithms** ([38], [3]) - In this section, a historical review of the first learning algorithm developments may be found. Single- and multiple-layer structures are introduced.

- Section 2.2: **Network Details and Hyperparameters** - In network design a lot of details and hyperparameters have to be taken into account. In this section, a list of the main features to consider is proposed: activation and loss functions, optimizer, learning rate and regularization techniques.

- Section 2.3: **Deep Learning** ([3], [9]) - After the first models, also more complex structures, with a deeper architecture, started to be implemented. The use of deep networks needs a few adjustments, but they show good performance in image processing tasks. We are going to focus on **Convolutional Neural Networks (CNNs)** and on **U-Net structures**, which we will use then to perform the susceptibility map reconstruction.
  We are going to illustrate also the **residual learning technique**, which helps in handling deep-learning algorithms.

2.1 Neural Networks and Learning Algorithms

ANNs - Artificial Neural Networks are **learning algorithms**, which are inspired by the behaviour of neurons and synapses inside the brain. They employ many artificial neurons which are organized in network structures. They are flexible systems, adapting to the data given as input ([38]).

First, the concept of a learning algorithm has to be formalised. What does it mean that an informatic system is learning?
To learn is a skill which implies a twofold ability. First of all, the algorithm has to be able **to fit** the input data, extracting patterns and features and recognizing trends and categories. After that, it has **to generalize** to a new set of unseen data. Usually, the second stage is the most difficult to achieve.
In 1997, the American computer scientist Tom Mitchell, who was working at the Machine Learning Department at the Carnegie Mellon University (Pittsburgh, Pennsylvania), proposed the following definition of learning ([39], [3]):

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Thus, one may say that the characteristic aspects of a learning algorithm are: task, performance and experience.

- **Task T**: it is the goal of the algorithm, which determines the way to process the input data (**feature vectors** $\mathbf{x} \in \mathbb{R}^n$) and the output of the network. Tasks are divided into two main categories: **classification** ($\mathbb{R}^n \longrightarrow \{1, 2, 3, ..., m\}$, where m is the number of classes) and **regression** ($\mathbb{R}^n \longrightarrow \mathbb{R}$).

- **Performance P**: it is a quantitative way to evaluate the efficiency and the accuracy of the algorithm. Its choice and computation depends on the data that can be accessed. They may be in fact labelled data or not. We are going to explain this point by speaking about the network experience.
  The function chosen to evaluate the perfomance is called the **error, cost** or **loss function**.

- **Experience E**: this aspect refers to the availability of data.
  When a dataset with input ($\{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x_i} \in \mathbb{R}^n$) and desired output ($\{\mathbf{y_i}\}_{i=1}^N$, $\mathbf{y_i} \in \mathbb{R}^m$) is accessible, then to implement a **supervised learning** algorithm is possible. N is the dataset size, n is the input space dimension, m is the output space dimension. n, m do not have necessarily to be equal. $\{\mathbf{y_i}\}_{i=1}^N$ are called the **labels**. In this situation, the algorithm looks for the conditional probability distribution $p(\mathbf{y}|\mathbf{x})$.

Instead, when there are no labels, the computational process may be based on **unsupervised learning**. This involves evaluating the probability distribution $p(\mathbf{x})$.

There is not a strict division between these two different processes: supervised-learning problems may be reduced to unsupervised-learning ones, and vice versa ([3]).
To pass from an unsupervised to a supervised learning algorithm, the probability distribution $p(\mathbf{x})$, with $\mathbf{x} \in \mathbb{R}^n$, can be decomposed as:

$$p(\mathbf{x}) = \Pi_{i=1}^{n}\, p(x_i|x_1, x_2, ...x_n) \tag{2.1}$$

using the chain rule of probability. So now, instead solving one unsupervised learning algorithm, n supervised learning processes have to be performed.
On the other hand, if the goal is to find the conditional probability $p(\mathbf{y}|\mathbf{x})$ in a supervised algorithm, the use of the traditional unsupervised learning approaches is possible. Initially, the joint distribution $p(\mathbf{x}, y)$ may be learned, and then the sought distribution could be obtained by inferring:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{\sum_{y'} p(\mathbf{x}, y')} \tag{2.2}$$

Techniques, which are called **semi-supervised**, are based on both supervised and unsupervised tools. To investigate them is beyond the goal of this thesis; interesting works are introduced in ([40], [41]).
For the sake of simplicity, from now on we are going to use the mathematical formalism relevant to supervised learning.

2.1.1 Training, Validation and Test Dataset

There are three stages in a learning process: training, validation and test. In order to perfom them, the available data have to be divided into three corresponding datasets.
We have not talked yet about the proper process that lets the algorithm learn. We are going to do this in the next section, introducing the concept of a **learning rule** and exploring its mathematical application in some cases. In the meantime, a qualitative idea may help in understanding the three steps just named.
Data are used to optimize the perfomance, related to the specific task, of the network. The optimization computation depends on the choice of the loss function. In order to progress the loss function, the network parameters, called the **weights of the network**, have to be updated with a mechanism called **back propagation** ([42], [43]).
This is what happens, in different procedures, in the three stages.

- **Training Stage**
  The training dataset, which is normally the largest one, is the only one actually used to adjust the parameters of the network. In particular, the value of the loss function after the training is exploited to adapt them. The update is done many times. Each time

is called an **epoch**. We are going to see how it is possible to update the parameters more than once per epoch, by dividing the training dataset into **batches** (Sec. 2.2).

- **Validation Stage**
  This is a control stage, used to check the occurrence of **overfitting** during the training. Overfitting means that the algorithm is fitting not on the proper data, but on the noise (see Fig. 2.1). It may occur due to the inadequacy of the model or to the amount of available data. A network runs the risk to learn irrelavant properties from the input dataset, properties which hamper the acquisition of the generalization skill. The validation stage is a good tool to take this possibility under control.
  The error function is, therefore, evaluated also on the validation dataset, unseen for the weights adjustment computed during the training.

- **Test Stage**
  At the end of the training and the validation stages, the error function has to be evaluated on the test dataset, which is previously completely unseen by the algorithm. If the error on the test dataset is almost equal to the one resulting from the training, it means that the network performed a good fit and the model is able to generalize on new data.

The main challenge, for a learning algorithm, is the last step. The parameter adjustment is done by evaluating the perfomance of the network on the training dataset, but what we actually want to make optimal is the error on the test dataset. Some assumptions about the data-generating process have to be made. These are called the **independent identical distributed assumptions (i.i.d.)** ([8]).
It is expected that the performance evaluated on the training dataset (P_{train}) are better than the one on the test dataset (P_{test}), because the training data are used for the parameter update and the others are not. If P_{train} is high, it means that the network is working in an **underfitting** condition. So, the first approach is to reduce as much as possible P_{train}. The second goal is to lower the gap between $P_{training}$ and P_{test}, to be sure to not to be in a overfitting condition (see Fig. 2.1).
This is the reason why it is essential to divide the initial dataset, for the purpose of letting the network learn. The use of the entire dataset only for the training stage is incorrect, even if the available data sample is limited. In that situation, there are some strategies that could be adopted. The most famous is the **Cross Validation** ([44]).

2.1.2 Historical Review: Single- and Multi- Layer Structures

In the next few pages, a brief historical review is provided, focused on the birth of neural network algorithms and their first developments. Starting with the description of the first artificial neuron, a lot of models were implemented in order to perform an algorithm able

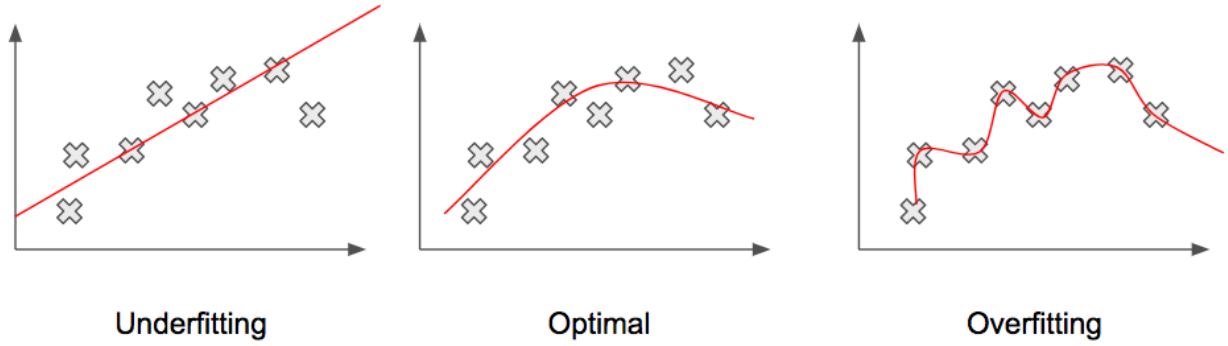


Figure 2.1: Left panel: underfitting condition - the data are not enough, or the model is not appropriate, to fit the real function. Central panel: optimal fit condition. Right panel: overfitting condition - the algorithm is fitting the noise and so extracting irrelevant features from the data

to learn in different and complex situations. The key ideas are the **weights introduction** and the **learning rule application**.

1943, McCulloch and Pitts's Neuron - TLU

The first mathematical model of a neuron was developed by McCulloch and Pitts in 1943 ([45], [38], [46]). It has been taken as a principal unit for all the following models and theories in the **Artificial Intelligence - AI** area.

A scheme is reported in Fig. 2.2, left panel. The neuron models a function $f_a(I, E) = Y$. I and E are the input data, divided into: **excitatory** input $\{E_i\}_{i=1...E}$ and **inhibitory** input $\{I_j\}_{j=1...I}$. $\{E_i\}_{i=1...E}$ make the neuron active, or 'fired', while $\{I_j\}_{j=1...I}$ deactivate the neuron. E, I, Y are binary ($\in \{0, 1\}$) ([8]). The function f_a, called the **activation function**, follows the expression:

$$\begin{cases} Y = 1 \; if \; \sum_{j=1}^{I} I_j = 0 \; and \; \sum_{i=1}^{E} E_i > T \\ Y = 0 \; otherwise \end{cases} \tag{2.3}$$

where T is a threshold set by the user.
This is the Heaviside step function, used by the model to pass the information through. The McCulloch and Pitts' model is also known as the **Thresholding Logic Unit - TLU** or **Linear Thresholding Unit - LTU**. At the beginning, it was introduced to perform basic Boolean operations.
At that time, many TLU variants were suggested. One of the most famous is the one proposed by the Hungarian mathematician John von Neumann, in 1956 ([8]). The latter and TLU differ in the possible I_j values. In von Neumann's model, they can also be negative. So, the condition to make the neuron active changes:

$$\sum_{i=1}^{E} E_i - |\sum_{j=1}^{I} I_j|) > T \longrightarrow Y = 1 \tag{2.4}$$

It is possible to organize these simple units, the artificial neurons, in a good structure to mimic the processes of animal and human brains. However, whatever the structure is, it

does not have any learning skill. In order to perfom this function, parameters to control the relationship between input and output have to be introduced; they are called the **network weights**. Then, a proper **learning rule** is needed.

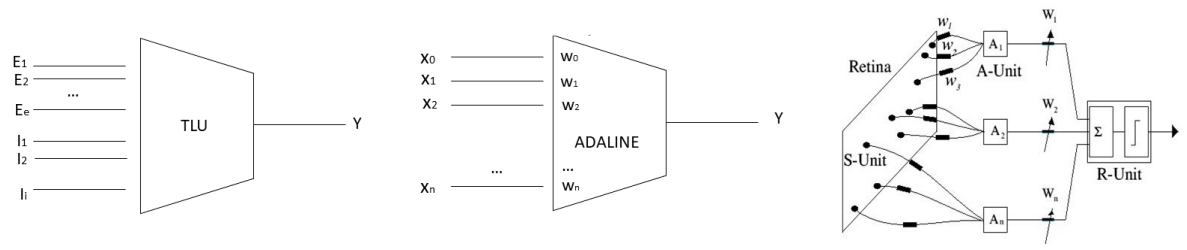


Figure 2.2: Left panel: the Pitt and McCulloch's neuron ([45]). Central panel: the Widrow and Hoff's model - ADALINE ([47]). Right panel: The Rosenblatt's perceptron ([8])

1956, Windrow and Hoff's Model - ADALINE

In 1960, the American engineer Bernard Widrow and his doctoral student, Ted Hoff, implemented a new neuron model ([47], [8], [38]), which incorporeted weighted input. It is called **ADALINE - ADAaptive LInear NEuron** or later **ADAptive LINear Element**.
A scheme is reported in Fig. 2.2, central panel. The model includes n input $\{x_i\}_{i=1...n}$ and n correspondent weights $\{w_i\}_{i=1...n}$. $\{x_i\}_{i=1...n}$ are binary, and their possible values are $\pm$ 1 ($\in \{-1, +1\}$). Each weight w_i provides information on the importance of the correspondent input x_i. $\{w_i\}_{i=1}^n \in \mathbb{R}$ and they are in the range $[-1, 1]$. There is an extra couple of fixed parameters, (x_0, w_0), which set an offset on $+1$.
Also here n is the dimension of the input space: it is not representative of the dataset size. The input space is composed of vector $\mathbf{x} \in \mathbb{R}^n$. $\mathbf{w} \in \mathbb{R}^n$ as well, so each component of the input vector, namely each **feature**, has an associated weight.
The activation function f_a of this model is the **Hard-Limiter** non-linear function:

$$\begin{cases} Y = 1 \; if \; net > 0 \\ Y = -1 \; if \; net < 0 \end{cases} \tag{2.5}$$

$$net = \sum_{i=0}^{n} w_i x_i \tag{2.6}$$

How do $\{w_i\}_{i=1}^n$ change in order to perform the task as well as possible? Below, it is explained the first mathematical fomalization of learning rules and its application in the ADALINE model.

Hebbian and Delta Learning Rules

The first learning theory was developed by the Canadian psychologist Donald Olding Hebb in 1949 ([8], [38]). The original statement is ([48]):

When an axon of a cell A is near enough to excite a cell B and repeatedly or persistenly takes place in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

Let $\mathbf{x}$ and $\mathbf{y}$ be binary vectors, not necessarily of the same size ($\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m$). $\mathbf{x}$ are the input, $\mathbf{y}$ the labels. $\{x_i\}_{i=1}^n$ and $\{y_j\}_{j=1}^m$ possible values are $\{0, 1\}$. $\mathbf{w}$ is a matrix, with size (n, m), which associates a weight to each component of the input vector with respect to each component of the output vector. The matrix form of the rule is:

$$[\mathbf{w}] = [\mathbf{x}]^t[\mathbf{y}] \tag{2.7}$$

In other words, the **Hebbian rule** states that a weight w_{ij} is reinforced when the correspondent input x_i is fired ($= 1$) and the label y_j also is.
Hebb influenced all the following discussions about network mathematical models and theoretical learning. However, this formulation of the rule only works with orthogonal or linearly independent inputs ([8]). In the other cases, it does not guarantee satisfactory results. This is because the actual ouput of the model is not taken into account, rather only the label data are.

Thus, a new parameter was introduced to evaluate the gap between the label and the network proper output. Let $\mathbf{y}$ be the label data, $\hat{\mathbf{y}}$ the network output, and delta $\boldsymbol{\delta} = (\mathbf{y} - \hat{\mathbf{y}})$. The learning statement was modified; it is known as the $\boldsymbol{\delta}$ **rule**. Its purpose is to reduce the distance between the expected result $\mathbf{y}$ and the one returned by the algorithm $\hat{\mathbf{y}}$.
The first implementation of this method was applied to ADALINE's model, with the Hard-Limiter function as f_a. This is not a differentiable function at zero, and so $\boldsymbol{\delta}$ had to be re-defined as:

$$\boldsymbol{\delta} = \mathbf{y} - \mathbf{net}(\mathbf{x}) \tag{2.8}$$

with $\mathbf{net} \in \mathbb{R}^m$ and $net_j(\mathbf{x}) = \sum_{i=1}^n w_{ij}x_i$, $j = 1, 2, .., n$. Intuitively, the adjustment of the weight w_{ij} has to be proportional to the correspondent input x_i and δ_j ($\Delta w_{ij} \propto x_i\delta_j$). To define the proportionality constant, let's introduce the cost function **LMS - Least Mean Square** - δ rule is also known as LMS rule ([8]):

$$E = \frac{1}{m}\sum_{j=1}^{m} e_j = \frac{1}{m}\sum_{j=1}^{m} \delta_j^2 = \frac{1}{m}\sum_{j=1}^{m}[y_j - \sum_{i=0}^{n}(w_{ij}x_i)]^2 \tag{2.9}$$

If the task is a classification one, then m represents the number of patterns or classes in the database under examination.
E is a quadratic function in terms of all the weights, so only one minimum with regard each w_{ij} should be. We are looking for the w_{ij} value which makes the error as low as possible. To reach this minimum, the **gradient descent method** is used: weights are corrected going along the same direction of the gradient, but opposite in sign. Calling k the proportional constant, this means:

$$\Delta w_{ij} = -k\frac{\partial E}{\partial w_{ij}} \tag{2.10}$$

$$\Delta w_{ij} = \frac{2k}{m} \sum_{k=1}^{m} (\delta_k x_i)_j = \frac{\eta}{m} \sum_{k=1}^{m} (\delta_k x_i)_j \tag{2.11}$$

η is the **learning rate**, which is a **hyperparameter** of the model. Hyperparameters are distinct from parameters because they are not updated during the training, but they are fixed by the user before starting the learning process.
Originally, η was set to $\frac{1}{n+1}$. It turned out that this is a too high value, which does not often allow a loss function minimum to be found. Thus, η has to be set lower than this value.
For now, let's consider **w** initialized to random small values. Actually, the **weights initialization** is an issue in all learning algorithms, especially in deep learning ones. We are going to discuss this point later.

This δ rule application is not a completely satisfactory result, because the proper output is still not considered. In order to do this, a function that is differentiable at zero has to be introduced.

1958, Rosenblatt's Perceptron - Mark 1

The term **perceptron** was used for the first time by the American psychologist Frank Rosenblatt, in 1958 ([38], [46]). It was then used to describe a large number of different neural models. At first, referring to a perceptron meant referring not to the mathematical model but to the implementation of the algorithm in software. The first perceptron was developed for supervised learning of binary classifiers. **Mark 1** is its name (see Fig. 2.2, right panel). It is composed of three layers: 1) the **sensor layer S**, which contains the input data, digitalized images with $\{0, 1\}$-pixel values; 2) the **association layer A**, a logical feature detector which groups of random inputs from S are connected to; 3) the **response layer R**, to which A's output are connected. R is actually the only layer with adjustable weights, so Mark 1 may be considered a single layer perceptron.

The activation function f_a has to be differentiable, in order to evaluate δ between the actual output of the network and the ground truth values. The first function introduced for this purpose was the sigmoid function σ:

$$\sigma = \frac{1}{1 + e^{-x}} \quad , \quad \frac{d\sigma}{dx} = \sigma(1 - \sigma) \tag{2.12}$$

δ_j is now defined as ([8]):

$$\delta_j = (y_j - \hat{y}_j) = \left(\frac{1}{1 + e^{-net_j}} \right) \tag{2.13}$$

LMS is used as before to evaluate the loss function E. The gradient of E with respect the weight w_{ij} becomes:

$$\frac{\partial E}{\partial w_{ij}} = \frac{1}{m} \sum_{k=1}^{m} \left(\frac{\partial e_k}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial w_{ij}} \right)_j = \frac{1}{m} \sum_{k=1}^{m} \left(-2(y_k - \hat{y}_k)] \frac{\partial \hat{y}_k}{\partial net_k} \frac{\partial net_k}{\partial w_{ij}} \right)_j \tag{2.14}$$

$$\Delta w_{ij} = \frac{2k}{m} \sum_{k=1}^{m} \left(x_i \hat{y}_k (1 - \hat{y}_k)(y_k - \hat{y}_k) \right)_j = \frac{2k}{m} \sum_{k=1}^{m} \left(\delta_k x_i \right)_j \tag{2.15}$$

One of the most important results connected to the perceptron's development is the **Perceptron Convergence Theorem** ([38]). It guarantees that a perceptron algorithm - working with a couple of elements $\{(\mathbf{x}, \mathbf{y})\}_{i=1}^{N}$ which describe a linearly separable dichotomy - always finds a solution in a finite number of weights updates. The proof of this theorem was given by Marvin Minsky and Seymour Papert in 1989.
To be able also to solve the problem in non-linearly separable **pattern spaces**, multi-layer perceptrons were developed.

Multi-Layer Neural Networks

With a multi-layer perceptron it is possible to implement even non-linearly separable functions. The first multi-layer algorithms had three layers: **input I**, **hidden H** and **output O**. Then, also models with more than one hidden layer were implemented. A single-hidden-layer network scheme is shown in Fig. 2.3.
Notice that the model that we are going to describe in here is a **fully-connected** network. It means that all the neurons of each layer are connected to all the neurons in the previous layer and in the following ones. Not all network models work in this way.

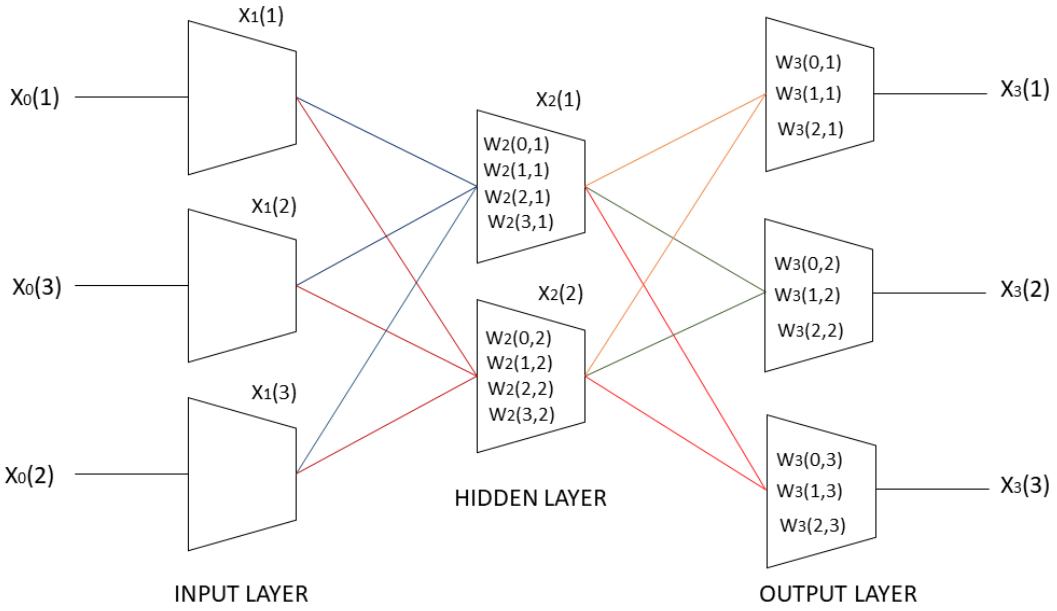


Figure 2.3: Multi-layer perceptron model (single-hidden-layer structure)

Let L be the number of the layers and l_i the number of neurons in the i^{th} layer. The i^{th} layer has as output dataset $\{x_j\}$, with $j = 1, 2, ..., m_i$. The number of weights associated with each neuron in the i^{th} layer is equal to l_{i-1}. The activation function is σ. To the a^{th} neuron in the i^{th} layer corresponds the output $x_i(a)$, which is defined as:

$$x_i(a) = net_i(a) = \sigma(\sum_{k=0}^{l_{i-1}} w_i(k,a)x_{i-1}(k)) \tag{2.16}$$

Let's evaluate $\Delta w_2(r,q)$, namely the r^{th} weight of the q^{th} neuron in the 2^{nd} layer. We assume that there is only one output neuron $x_3(1)$, so $E = [y(1) - x_3(1)]^2$. Then ([8]):

$$\frac{\partial E}{\partial w_2(r,q)} = \frac{\partial E}{\partial x_3(1)} \cdot \frac{\partial x_3(1)}{\partial x_2(q)} \cdot \frac{\partial x_2(q)}{\partial w_2(r,q)} \tag{2.17}$$

$$\Delta w_2(r,q) = \eta x_1(r) \delta_2(q) \tag{2.18}$$

$$\eta = 2k \tag{2.19}$$

$$\delta_2(q) = x_2(q)(1 - x_2(q)) w_3(q,1) \delta_3(1) \tag{2.20}$$

$$\delta_3(1) = x_3(1)(1 - x_3(1))(y(1) - x_3(1)) \tag{2.21}$$

If there are many output neurons, the error fuction E would be:

$$E = [y(1) - x_3(1)]^2 + [y(2) - x_3(2)]^2 + ... + [y(O) - x_3(O)]^2 \tag{2.22}$$

So, the expression for $\delta_2(q)$ changes:

$$\delta_2(q) = x_2(q)(1 - x_2(q)) \sum_{i=1}^{O} w_3(q,i) \delta_3(i) \tag{2.23}$$

If there is more than one hidden layer, the expression of $\delta_p(q)$ - q^{th} unit of p^{th} layer - is:

$$\delta_p(q) = x_p(q)(1 - x_p(q)) \sum_{i=1}^{O} w_{p+1}(q,i) \delta_{p+1}(i) \tag{2.24}$$

This is an example of how **Backpropagation Algorithms** works. Backpropagation is a technique that is commonly used for the weights adjustment in neural networks. It is a shorthand for **backward propagation of errors**. This algorithm has been introduced since the 1960s by different research groups, but it was taken seriously into account in 1986 thanks to the work of David Rumelhart, Geoffrey Hinton and Ronald Williams ([42], [43]).

To sum up, the stages in a learning algorithm are:

1. **Initialization** of the parameters, which are going to be updated during the training.
2. **Forward Propagation**
3. **Perfomance Evaluation**, related to the choice of the loss function E
4. **Loss Gradient Evaluation** $\frac{\partial E}{\partial w_i}$
5. **Back Propagation**, with the weight w_{ij} adjustment in the opposite sign of $\frac{\partial E}{\partial w_{ij}}$.

2.2 Network Details and Hyperparameters

Before starting to describe more complex models of networks, a list of the principal network details and hyperparameters is detailed below. There are in fact many aspects and variants to be taken into account in designing a learning algorithm. We have already mentioned some of them, such as the loss function or the learning rate.

- **Activation Function**
  It is the function which connected the weighted sum $net_j = \sum_{i=1}^{n} w_{ij}x_i$ with the output of the network $\hat{y}_j$. It has to be differentiable to apply the backpropagation tool. The first choice was the **logistic** or **sigmoid function** σ (Eq. 2.12), but there are also other possibilities.
  Activation functions are basically divided into linear and non-linear. The specific choice depends on the kind of problem which has to be solved, and there is not a deterministic way to decide it. When the development of NNs began, almost all algorithms used non-linear functions as f_a. Then, especially with the development of more complex structures, also the linear ones were started to be taken into account as good candidates. In fact, recent research highlights that an excessive saturation of the activation function does not help the convergence of the algorithm ([49]).
  Fig. 2.4 shows a list of the activation functions that are most usually used ([50]).

- **Loss Function** ([51])
  Also the choice of the loss function influences the convergence of the network, because the backpropagation process is based on its gradient, and that process is the key stage for the parameter adjustment. There are many functions which may be considered when calculating the error; the most frequently used are:

  1. **Mean Squared Error MSE** $:= \frac{1}{N}\sum_{i=1}^{N}(net(x_i) - y_i)^2$; x_i are the input data, y_i are the labels, $net(x_i) = f_a(x_i)$. N is the size of the database.
  2. **Cross Entropy Loss CE** and **Binary Cross Entropy Loss BCE**
     $CE := -\sum_{i=1}^{N} y_i * log(net(x_i))$
     $BCE := \sum_{i=1}^{N} [y_i * log(net(x_i)) + (1 - y_i)log(1 - net(x_i))]$
     Using entropy to quantify error measurements comes from the information theory ([52]). Models for probabilistic classification were developed, their goal was, through training, to make their prediction $f_a(x_i)$ closer and closer to the ground truth probabilities y_i. x_i are the input data, and y_i are the labels.
     BC and **BCE** functions are used for classification tasks instead of MSE, and they provide good results also in some experiments of image reconstruction ([53]).
  3. L_1 **Loss** or **Mean Absolute Error** $:= \sum_{i=1}^{N} |y_i - net(x_i)|$

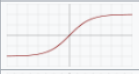
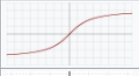
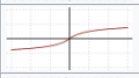
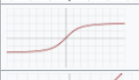
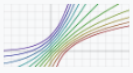
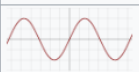
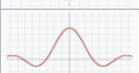
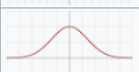
Name	Plot	Equation	Derivative (with respect to x)	Range	Order of continuity
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	C^∞
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$[1]	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	C^∞
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$	C^∞
Softsign[7][8]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$	$(-1, 1)$	C^1
Inverse square root unit (ISRU)[9]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3$	$\left(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}}\right)$	C^∞
Rectified linear unit (ReLU)[10]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$	C^0
Leaky rectified linear unit (Leaky ReLU)[11]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0
Parameteric rectified linear unit (PReLU)[12]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$[2]	C^0
Randomized leaky rectified linear unit (RReLU)[13]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$[3]	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0
Exponential linear unit (ELU)[14]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\alpha, \infty)$	$\begin{cases} C_1 & \text{when } \alpha = 1 \\ C_0 & \text{otherwise} \end{cases}$
Scaled exponential linear unit (SELU)[15]		$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ with $\lambda = 1.0507$ and $\alpha = 1.67326$	$f'(\alpha, x) = \lambda \begin{cases} \alpha(e^x) & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\lambda\alpha, \infty)$	C^0
S-shaped rectified linear activation unit (SReLU)[16]		$f_{t_l, a_l, t_r, a_r}(x) = \begin{cases} t_l + a_l(x - t_l) & \text{for } x \leq t_l \\ x & \text{for } t_l < x < t_r \\ t_r + a_r(x - t_r) & \text{for } x \geq t_r \end{cases}$ t_l, a_l, t_r, a_r are parameters.	$f'_{t_l, a_l, t_r, a_r}(x) = \begin{cases} a_l & \text{for } x \leq t_l \\ 1 & \text{for } t_l < x < t_r \\ a_r & \text{for } x \geq t_r \end{cases}$	$(-\infty, \infty)$	C^0
Inverse square root linear unit (ISRLU)[9]		$f(x) = \begin{cases} \frac{x}{\sqrt{1+\alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$\left(-\frac{1}{\sqrt{\alpha}}, \infty\right)$	C^2
Adaptive piecewise linear (APL)[17]		$f(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$	$f'(x) = H(x) - \sum_{s=1}^{S} a_i^s H(-x + b_i^s)$[4]	$(-\infty, \infty)$	C^0
SoftPlus[18]		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$	$(0, \infty)$	C^∞
Bent identity		$f(x) = \frac{\sqrt{x^2+1}-1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2+1}} + 1$	$(-\infty, \infty)$	C^∞
Sigmoid-weighted linear unit (SiLU)[19] (a.k.a. Swish[20])		$f(x) = x \cdot \sigma(x)$[5]	$f'(x) = f(x) + \sigma(x)(1 - f(x))$[6]	$[\approx -0.28, \infty)$	C^∞
SoftExponential[21]		$f(\alpha, x) = \begin{cases} -\frac{\ln(1-\alpha(x+\alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x}-1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1-\alpha(\alpha+x)} & \text{for } \alpha < 0 \\ e^{\alpha x} & \text{for } \alpha \geq 0 \end{cases}$	$(-\infty, \infty)$	C^∞
Sinusoid[22]		$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$	C^∞
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{for } x \neq 0 \end{cases}$	$[\approx -.217234, 1]$	C^∞
Gaussian		$f(x) = e^{-x^2}$	$f'(x) = -2xe^{-x^2}$	$(0, 1]$	C^∞

Figure 2.4: List of possible activation functions f_a in NNs ([50])

- **Optimizer** ([54])
  This is the technique used to optimize the algorithm. The basic one is called the **Gradient Descent**, we have introduced it speaking about the delta rule in Sec. 2.1. In this approach, the **w** values are updated in the opposite direction of $\nabla_{\mathbf{w}} E$. The size of the steps along the gradient direction is called the **learning rate** η:

$$w_{i+1} = w_i - \eta \nabla_{w_i} E \tag{2.25}$$

  In this way local and global minima of the cost function E may be reached.

- **Batch Gradient Descent, Stochastic and Mini-Batch** ([54])
  Normally, during one epoch, weights are updated once, after the network looks at the entire dataset. To increase the number of the updates, one possibility is to define a **batch size**, to divide the training dataset with respect to this size and to update the weights $\frac{training\ size}{batch\ size}$ times during one epoch. This technique, called the **Batch Gradient Descent** (BGD), increases network convergence speed.
  If the batch size is equal to 1, this technique is called **Stochastic Gradient Descent** (SGD). In this case, parameter adjustment happens for each training example. This technique is faster than the others and can be used for online learning. But, considering that the parameter updates are performed each time, they have high variance and the loss function heavily fluctuates.
  Mini-batch Gradient Descent (MGD) is a compromise between BGD and SGD. It allows the variance of the parameter updates to be reduced and leads to more stable convergence. Normally, the mini-batch size is set from 32 to 256, depending on the size of the input data and the specific task of the network. The smaller the batch size is, the more sensitive to noise and overfitting the network becomes.
  Another benefit is that the MGD allows the use of less memory.

  There are many **GD optimization algorithms**, a good overview of them may be found at [54]. We focus on the **RMSprop optimizer**, which is used for the training processes in this work.
  The magnitude of $\nabla_{w_i} E$ can significantly vary in according to w_i, and this makes hard to choose a single global learning rate η - next on the list. The **Rprop optmizer** proposes only the sign of the gradient to be used during the weight udjustment stage. This algorithm combines this concept with the idea of adapting the step size separately for each weight, looking into the signs of its last two gradients. According to them, weights have to be divided or multiplied by a specific factor.
  The Rprop algortithm works well since the batch size is quite wide, but not for mini-batch algortihms. The more the batch is small, the more the factor by which we have to divide or multiply changes very quickly, taking stability away from the process. The RMSprop force this factor to be very similar for adjacent mini-batches.

- **Learning Rate** η

  This parameter controls how much weights have to be adjusted with respect to the loss function gradient. It affects how quickly the model can converge to a minimum. With a low η, the network explores the entire loss function gradient space, avoiding becoming stuck in any local minima. On the other hand, moving in this way it needs a lot of time to converge, especially if there are some plateau regions. If the learning rate is instead too high, it may occur that a minimum is never reached.
  In Fig. 2.5 different shapes of the loss function as a function of time (number of epochs) are shown. The shape changes affect the choice of the learning rate. If the choice is right, the loss function descreases (red line), not too slowly (blue line), not too quickly (green line).

  The learning rate η can also be modified and updated during the training. There are different ways to do this:

  – **decay updating**: $\eta(n) = \frac{1}{1+\tau*n} * \eta_0$, where n is the number of the current epoch and τ the chosen dacay rate

  – **exponential updating**: $\eta(n) = \tau^n * \eta_0$

  – **discrete updating**: $\eta(n) = \frac{Constant}{\sqrt{n}} * \eta_0$ or $\eta(t) = \frac{Constant}{\sqrt{t}} * \eta_0$, where t is the current batch number

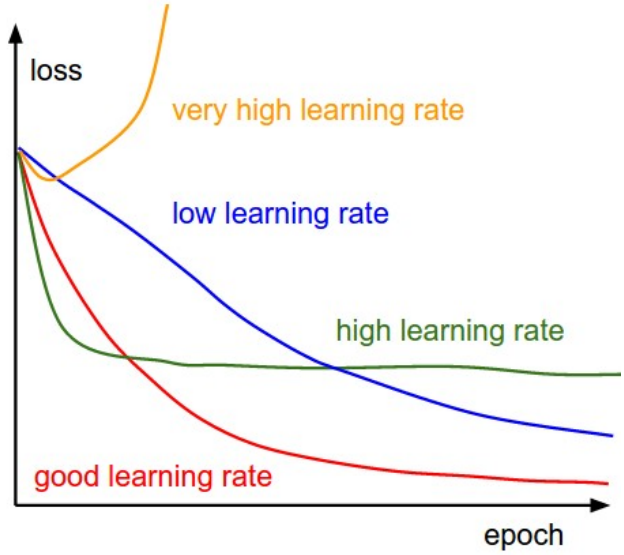


Figure 2.5: Loss function E shape with different learning rate η ([55]

- **Regularizer** ([3], [54])

  There are some techniques, called **regularization tools**, which change the formalization of the learning rule, without changing the basic idea, in order to avoid an overfitting condition. They introduce new elements to reduce the test but not the training error of the model. Below, we introduce two of them. In the next chapter, the **dropout technique** ([3]) is also described.

– **Weights Decay Rate** λ

$$w_{i+1} = w_i - \eta \nabla_{w_i} E - \eta \lambda w_i \tag{2.26}$$

When there is no regularization, λ is set equal to zero.
This technique produces a preference for weights with small values. As λ is big, the **w** values are forced to become smaller.

– L_2 **Regularization**
It involves modifying the error function by including the λ_2 parameter. This is an attempt to find a compromise between finding small weights and minimizing the cost function. With L_2 regularisation the original E_0 becomes:

$$E = E_0 + \frac{\lambda_2}{2n} \sum \mathbf{w}^2 \tag{2.27}$$

where n is the training dataset size.
If **w** is small, the network behaviour will not change too much with a small random input change. This makes it hard to learn the effect of local noise and easy to learn the type of useful evidences that are often seen across the input dataset. If **w** is large, the network could learn a complex model that carries a lot of information also about the noise of the training data.

2.3 Deep Learning

In the previous sections, the theoretical aspects of single- and multi-layer perceptrons were explored, focusing in particular on the **single-hidden-layer** structures (Fig. 2.3). This category was used for a long time as a unique neural network algorithm, because of its success in both practical and theoretical applications. In fact, it has been proven that a feedforward neural network with a single, huge, hidden layer is a universal approximator of Borel measurable functions ([56]). Then, also **multiple-hidden-layer** structures started to be used. It was observed that they do produce better results than their shallow counterparts, despite using the same number of computational units ([56], [57]).

Deep neural networks are learning algorithms, which allow complex data to be managed, with a **coarse-grained representation** - i.e. 2D and 3D visual data, time signals, .. - and non trivial information to be extracted from them. Their development started in the 1980s, but they have been massively adopted after the ImageNet Large Scale Challenge, in 2012 ([2]), and also after the development of **GPU** tools, that are 10 time faster than the CPU ones ([9]). They have showed brilliant perfomance in **image processing** and in **natural language processing**, which is generally referred to as NLP. Their being so interesting also arises because they are inspired by biological and human learning mechanisms. For example, there is a class of deep structures called **Convolutional Neural Networks** which were inspired by the mechanism of simple and complex cells in neuroscience, and by

the LGN-V1-V2-V4-IT hierarchy in the visual cortex ventral pathway ([9]). We are going to speak about them later, because they are much exploited in visual area treatments.
Complex neural networks are based on the idea of the **feature hierarchy** and **representative learning**. Going through the **stacked layers**, namely going deeper into the network, it is possible to reach an even more abstract level of representation for each step. We may think about these networks as **high-level feature** extractors. For example, if an image is being processed with a deep network, at the first level of depth the algorithm could recognize edges and relevant structures; at the second level of depth, motifs; at the third one, assembly of motifts into larger combinations, and so on ([9]). They have to be able to separate the important features from the irrelavant ones, such as variation of the orientation or the accent of speech, to avoid an overfitting condition (Fig. 2.1). Remember that in that condition the generalization skill of the network is compromised. The algorithm has to be insensitive to pointless variation of the data. This is possible because the layers are not fully connected, at least not at all the levels of depth. Thanks to this property, each concept is represented by many neurons, and each neuron participates in the representation of many concepts. With this **distributed representation**, deep structures react equally to complicated patterns of different inputs.

Parameters Initialization Rules

The **parameter initialization rule** influences a lot the convergence of the model. Sometimes it happens that a right decision for a **shallow structure** is not optimal for the deep counterpart ([58]).
The random commonly used rule is:

$$w_i \sim U\left[-\frac{1}{\sqrt{n}}, +\frac{1}{\sqrt{n}}\right] \tag{2.28}$$

- where $U[-a, +a]$ is the uniform distribution in the interval $(-a, +a)$ and n is the size of the layer under examination. It has been shown that this rule implies that the variance of the backpropagated gradient changes a lot with respect to the index of the layer i (see Fig. 2.6). It might vanish or explode as we consider deeper networks. This condition is known as the **vanishing/exploding gradient problem** ([58]), and it is an issue for the deeper networks convergence. One way to fix it, is to change the inizialization rule. The use of the **normalized distribution** was proposed (see Fig. 2.6):

$$w_i \sim U\left[-\frac{1}{\sqrt{n_i + n_{i+1}}}, +\frac{1}{\sqrt{n_i + n_{i+1}}}\right] \tag{2.29}$$

There are other ways to keep the flow of information controlled:

- **unsupervised pretraining** ([59])

- **residual learning** approach (Sec. 1.4.2)

- **activation function** change. A smooth function indeed helps to avoid saturation during the training ([58])

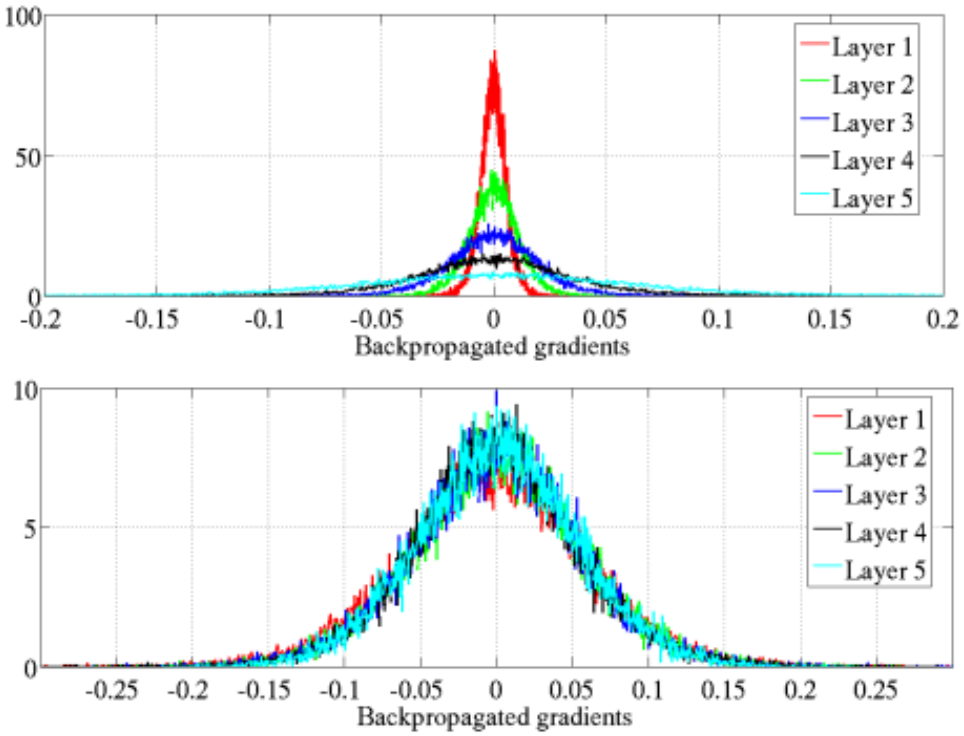


Figure 2.6: Back-propagated gradients normalized histograms with hyperbolic tangent activation, with standard initialization (Eq. 2.28, top panel) and normalized initialization (Eq. 2.29, bottom panel). Tope panel: 0-peak decreases for higher layers ([58])

2.3.1 Convolutional Networks

Convolutional Neural Networks - CNNs are a class of **locally connected** net structures, that are very useful to use with data with a known **grid-like topology** ([9]). They were introduced by the French scientist Yann LeCunn in 1989 ([60]), who was interested in the problem of invariant visual perception and its solution by learning methods. He first used these structures for shape recognition tasks, and then also for recognition of handwritten characters.

Since 2012, which was the year of the Large Scale Visual Recognition Challenge ([2]), CNNs have been heavily used in the 2D and 3D visual area. For the sake of semplicity, we are going to focus on the mathematical formalism with 2D data. The extension to three-dimensional data is quite intuitive.

They are based on representative learning, which allows the pre-processing stage that is necessary in traditional learning algorithms to be avoided. In fact, **shallow structures** require that a good feature extractor is performed, in order to solve the **selectivity-invariance dilemma** ([9]). With representative tools, we do not have to know the important features, patterns or trend in the input data.

A convolutional network is characterized by the use of peculiar kinds of layers: convolutional and pooling. Below, we give a description of both of them.

Convolutional Layers

In traditional image processing **punctual** and **local operations** may be performed ([61]). The first are functions $f(i) = o$, where i is the intensity value of the input pixel and o is the intensity value of the output pixel. o, in the returned map, depends only on the intensity of the same pixel in the original map. This kind of operation is usually a grey-level histogram transformation, like **thresholding** or γ**-correction**.
Local operations are instead functions in which o is not related only to the intensity of the correspondent pixel in the original map, but also to neighbouring pixels values. To perform this operation, to define the shape and the dimension of the neighborhood is necessary. Usually, squared matrices with dimension (3,3) or (5,5) are chosen. They are called **filters** or **kernels**. A local operation consists of a convolution between an image and a specific kernel.

Convolution is a mathematical linear operation, defined as:

$$s(t) = \int x(a)w(t-a)da = (x \otimes w)(t) \tag{2.30}$$

in a monodimensional space.
It can be performed for any functions $x(t), w(t)$ for which the integral in Eq. 2.30 is defined. In a discrete space, it becomes:

$$s(t) = \sum_{-\infty}^{+\infty} x(a)w(t-a) \tag{2.31}$$

To perfom a convolution between an image S and a kernel K, the formalism has to be extended to 2D space ([3]):

$$S(i,j) = (I \otimes K) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.32}$$

Convolution is commutative, so:

$$I \otimes K = \sum_m \sum_n I(m,n)K(i-m,j-n) = \sum_m \sum_n I(i-m,j-n)K(m,n) = K \otimes I \tag{2.33}$$

In some programming languages, **cross correlation** is used instead of convolution, and it is defined by the expression:

$$\sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.34}$$

Local operators are widely used in image processing, both in the spatial and frequency domains ([61]). Depending on the task, the kernels used are different. In the spatial domain there are:

- **Edge Detection Filters**
  They recognize high-contrast structures in the input image. They are gradient and Laplacian operators. Some examples are: **Sobel, Robert** and **Prewitt filters** and **Laplacian filters**.

- **Noise Reduction Filters**
  They are used to reduce noise and errors in the input image, with essentially smoothing operations. Some examples are: **Gaussian filters** and **Running Average filters**

- **Edge Enhancement Filters**
  They are used to find a compromise between noise reduction and loss of details. Usually they consist of applying an edge detection filter and then adding the original image to the returned map.

Convolutional operations may also be performed in the frequency domain.

- **High-Pass Filters**
  They are filters which stop the low frequencies, so are edge enhancement filters. In the high-frequency domain there is information related both to the details and to the noise. The risk in applying high-pass filters, and in using edge detection/enhancement filters in general, is to stress also the noise in the returned map.

- **Low-Pass Filters**
  They are filters which stop high frequencies, so are noise reduction filters. If some of these filters are applied, the returned map has also some loss in details and edges with respect to the input map.

The use of these filters in NNs is almost the same, so convolutional layers are actually feature detectors. The difference is that the features to extract are not known *a priori*. So, values of the kernel matrix are initialized, and then they are adjusted during the training stage to extract the information in which we are interested. They are the weights of the network. Also, in each convolutional layer there is more than one kernel, that means that they can manage more features in the same moment, one for each kernel. The number of kernels has to be set before the training stage starts. The result of each convolution operation is called a **feature map**.

Modern networks can have more than one million input units. To perferm a convolution operation between I and K means actually to transform in Fourier space I and K, perform a point-wise multiplication and then transform back to the spatial domain (**Convolution Theorem**):

$$I \otimes K \longrightarrow \mathcal{F}(I) * \mathcal{F}(K) = \mathcal{R} \longrightarrow \mathcal{F}^{-1}(\mathcal{R}) \tag{2.35}$$

If K is a separable matrix, namely which may be written as a product of d vectors, one for each dimension, it is more computationally convenient to perform d different convolutions ($\mathcal{O}(w^d) \longrightarrow \mathcal{O}(w * d)$).

Let $I(M, N)$ be the input image, $K_p(m, n)$ one of the P kernel of a convolutional layer ([3]). There are P resulting feature maps:

$$O_p(M, N) = I(M, N) \otimes K_p(m, n), p = 1, 2, .., P \tag{2.36}$$

The size of a O_p map is $(M - m + 1, N - n + 1)$, but some padding options may be set to have the output size equal to the input one.
If there are Q input maps, then each output O_p is equal to:

$$O_p(M, N) = \sum_{j=1}^{Q} I_j(M, N) \otimes K_p(m, n) \tag{2.37}$$

A kernel is also called a **filter bank**. It is a band-pass filter that separates the input signal in multiple components. Each feature maps in a layer use a different filter bank. In this way, the algorithm can recognize local correlations between pixels and it becomes invariant to feature location.

Convolutional layers give to the learning algorithm three important properties ([3]):

- **Sparse Interaction**
  Fully-connected neural networks have a structure in which each unit in each layer interacts with all those in the previous layer and all those in the subsequent layer. In a convolutional layer this does not happen, they are locally connected networks. The kernel size is in fact smaller than the input one, so it is possible to perform sparse interactions between units. One may refer to this process also using the term **sparse connectivity** or **sparse weights** ([9]). In this way, the number of parameters to be stored can be reduced, and with it the computational effort. Also, the statistical efficiency increases. If m is the input size, n the output , k the number of connection that each output may have, then the computational complexity is $\mathcal{O}(k * n)$, instead of $\mathcal{O}(m * n)$.

- **Parameter Sharing**
  In a convolutional layer, kernel values are the parameters of the network. Kernels are used along all the image area. It means that each parameter is used for more than one unit. Kernel values are called **tied weights**: each member of the kernel is used for every position of the input - without considering the pixels on the border, where the operations performed change in response to the padding settings. This fact has no effect on the runtime, but it allows the number of stored parameters to be reduced.

- **Equivariant Representation**
  Thanks to the convolution properties, a convolutional layer is **equivariant to translation**. Specifically, it means that the order of application of two functions can be changed without changing the result ($f(g(x)) = g(f(x))$). For example, let I be the

input image and $I' = g(I) = I(x-1, y)$ an object with the same dimension. Let K be a kernel.

$$g(I) \otimes K = g(I \otimes K) \tag{2.38}$$

A convolutional layer performs several convolutions to produce a set of linear activations. As activation function f_a, ReLU is usually used (Tab. 2.4). It is recommended, using this function, to normalize the input values in the range $[0, 1]$.

Pooling Layers

In CNNs, there are many convolutional layers in a row. K, the number of kernels, may change from one layer to another, but the feature map size does not change if the padding options are fixed.
Pooling layers, also called the **sampling layers**, changes instead the matrix size to let the network be invariant to translational disortions of the input. If I want the network to recognize a particular structure, I do not want it to be sensitive to the position of this structure in the frame. In other words, I want the algorithm to be sensitive to **high-level features**.
This class of layers is also useful to handle inputs of varying size.

A pooling operation may be a **subsampling** or an **upsampling** one. Both of them process the input map with a specified-size matrix, let it be (2,2). In a subsampling operation, the original image size is halved; it is divided into matrices (2,2) and they are replaced with an unique pixel with intensity equal to the max value (**Max Pooling**) or the mean value (**Average Pooling**) of the initial four pixels. Subsampling pooling layers have the role of merging similar features into one, to let the learning system become invariant to small shifts and distortions. The **coarse-grained representation** is exploited.
In a upsampling layer, the map size increases; so each pixel is replaced by a matrix (2,2). This operation has a blurring effect on the input map.

Organizing alternate convolutional and pooling layers, flexibly deformed structures can be recognized as variations of the same structural theme. High-level categories are fully independent of the position of the object within the frame.

2.4 Convolutional Autoencoders

Convolutional neural networks are widely used in image processing. They are usually organized in **deep structures**, which allow to perform tasks in the visual area such as: **segmentation**, **recognition**, **reconstruction**, **denoising**, just to name a few.
In the first section, we divided the possible tasks into two categories: **classification** ($\mathbb{R}^n \to \{1, 2, .., m\}$) and **correlation** $\mathbb{R}^n \to \mathbb{R}$). But we may observe that in some situations, such

as reconstruction and denoining tasks, the algorithm has to fit a function $f : \mathbb{R}^n \to \mathbb{R}^n$, in which input and output have the same size. In those situations, networks with **autoencoder** structures are required.

2.4.1 U-Net: U Structure

Convolutional autoeconders are also called **U-Networks** ([62]). Thir structure combines a **contracting or encoding path** and a **expanding or decoding path** (Fig. 2.7).
They may be thought as a composition of functions ([63]). Let $f_1 : \mathbb{R}^{n^2} \to \mathbb{R}^{m^2}$, with $m < n$, be the function fit in the encoding path, and $f_2 : \mathbb{R}^{m^2} \to \mathbb{R}^{n^2}$ the one fit in the decoded path. So $y = f(x) = f_2 \circ f_1(x) = f_2(f_1(x))$. Input and output have the same size: $f : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$, with n^2 the image size we are working with.
A convolutional autoencoder processes the input data in order to reduce the number of variables under consideration - **dimension reduction task** - or in order to change them in others which we are more interested in.

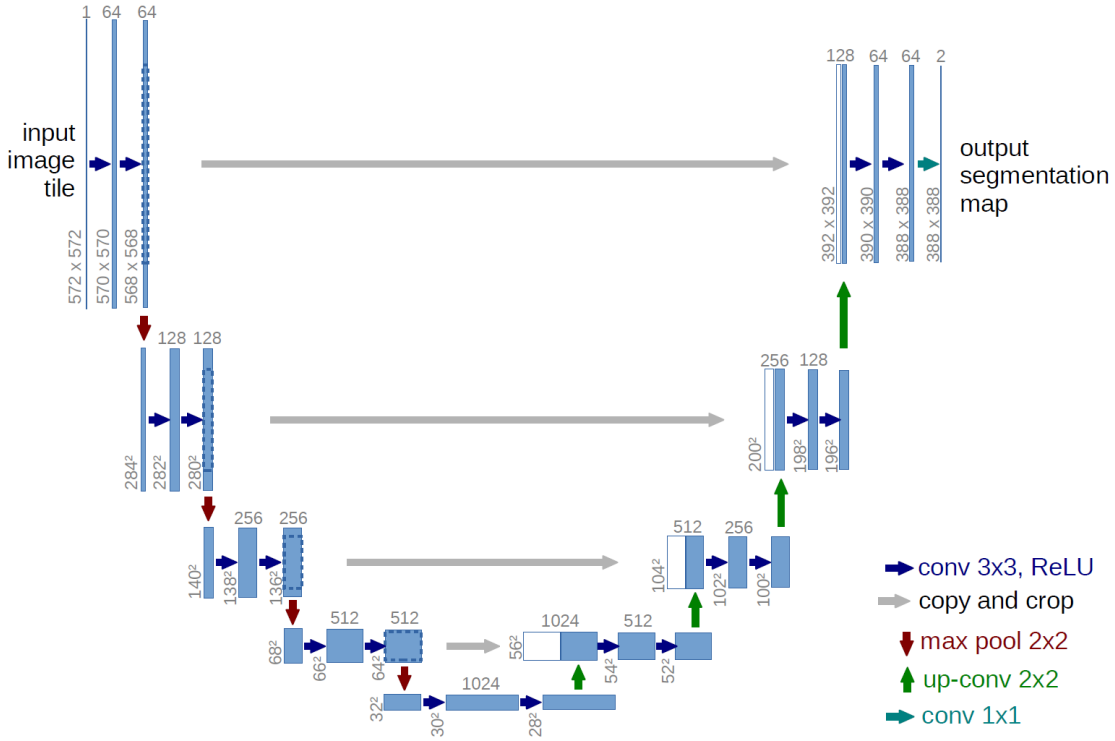


Figure 2.7: U-net architecture ([62]). Each blue box corresponds to a multi-channel feature max. The number of the maps ide denoted on the top of the box

- **Encoding Path**.
  It is the first part of the network, composed by two ore more level of depth. Along the encoding path the feature maps size decreases while the number of feaure maps increases. In the deeper level, called the **latent representation**, the data have the minimum size and the maximum number of feature maps.

This part is used to the high-feature detection, and it is essentially the **capture context** part.
In each level of depth there is a **block**. In each block, there is a sequence of sub sampling and convolutional layers. A subsampling layer signs the beginning of a new level of depth, in which the maps size is smaller than the previous one.

Classification task networks end with the latent representation, this it usually composed of a mono-dimensional array: the input data are flattened to be then classified. Examples of image classification may be found in [64] and in [65].
If the task is different from classification and requires input and output with the exact same size, the expanding path has to be added to the algorithm.

- **Decoding Parh**.
This second path is necessary to reconstruct an output of the same dimension of the input data. It is symmetric to the encoding parh, and this is an important property: without the symmetry, the reconstruction is still possible but the results loss quality. In the expanding path of Fig. 2.7, for each block there are one upsampling layer and two convolutional layers in sequence.
This parh is useful to recover the input data size, but we have to take into account that, along the contracting path, a lot of high-resolution information is lost. For many tasks, we need to combine high and low resolution information. This is the reason why there are some elements called **skip connections**, which link maps with the same size between the encooding and the decoding part. They are specifically **long skip connections**, separated by **short skip connections** ([53]). We are going to come back to them in the following section.

An example of the use of convolutional autoencoders in image segmentation may be found in [66] and in [67]. In there, they refer to the structure as **convolutional encoder-decoder structure**.

2.4.2 Residual Learning: Skip Connections

We have already highlighted, at the beginning of this section, the advantages in using deep structures instead of the shallow ones because of their ability in combining low- and high-level features. However, there are some drawbacks, like the computational effort and the **convergence issue**. In fact, deep neural networks sometimes experience some difficulties in reaching the global minimum in the gradient space because of the presence of plateau regions or many local minima. That implies an increase of the error values, as the number of level of depth grows. For example, see Fig. 2.8. In there, 20-layer and 56-layer structures were trained on CIFAR-10. This is a dataset from **CIFAR - Canadian Institute For Advanced Research**, used to train machine learning and computer vision algorithms, consisted in a collection of 60000 (32,32) images divided in ten classes.

This issue is also called the **degradation problem** ([57]). The accuracy of the algorithm gets saturated, increasing the number of stacked layers, and then quickly decreases. This phenomenum is related to the vanishing/exploding gradient problem ([58]), presented in the introduction of Sec. 2.3.

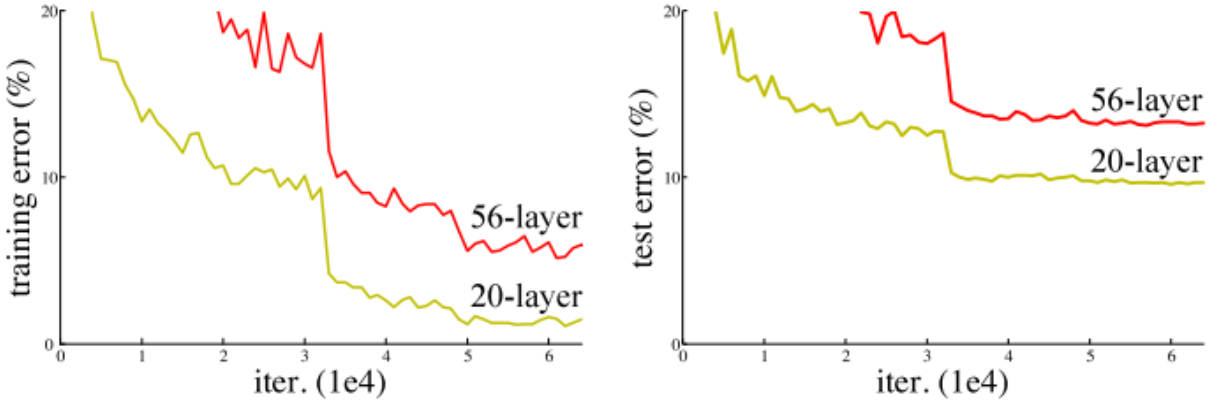


Figure 2.8: Training error (left panel) and test error (right panel) on CIFAR-10 dataset with 20-layer and 56-layer networks ([57], [68])

Theoretically speaking, it is difficult to understand why a deep structure has a bigger error than its shallow counterpart. There is in fact a solution by construction to the deeper model: the added layers are identity mapping, and the other layers are copied from the learned shallower model ([57]). So, there is no reason to have an higher error in deeper algorithm training, but actually we have experienced it. It is a counterintuitive phenomenon, which indicate that deep structures have trouble in fitting identity map. The **residual learning** techniques try to overcome this issue.

Let be $\mathcal{H}(\mathbf{x})$ the underlying function to be fit by a few stacked layers. $\mathbf{x}$ are the input data. Let's assume that a multiple non-linear layer structures can easily asymptotically approximate whatever function, no matter what its complexity ([56]). There is no difference, since $\mathbf{x}$ are known, in approximating $\mathcal{H}(\mathbf{x})$ or the **residual function** $\mathcal{F}(\mathbf{x})$ ([57], [68]):

$$\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x} \tag{2.39}$$

To fit the residual function, a shortcut is added to merge the input and the output of the block of layers considered. A scheme of the **residual building block** is shown in Fig. 2.9.

This formulation is used when the input and output have the same dimension. When this is not the case, there are different strategies to handle the problem, for example performing convolution operations to match the size or added extra-zero values to increase the dimension and to not add extra parameters.
A common choice is to add a residual learning to every few stacked layers. A building block may be formalized as ([57]):

$$\mathcal{H} = \mathcal{F}(\mathbf{x}, \{w_i\}) + \mathbf{x} \tag{2.40}$$

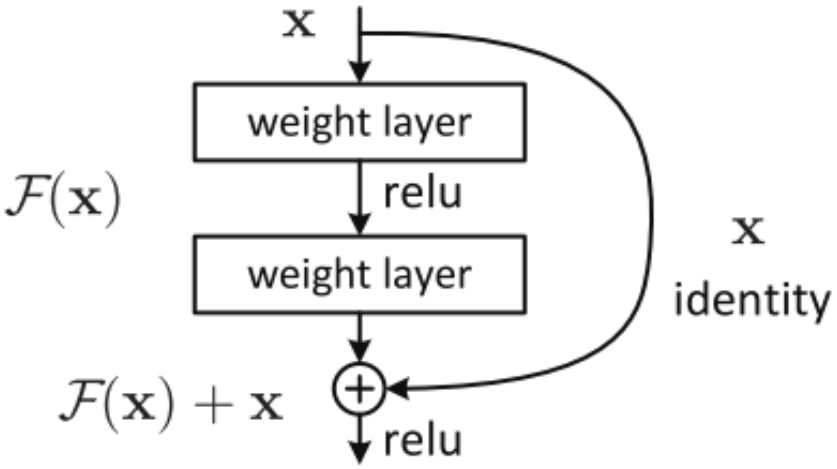


Figure 2.9: Residual learning building block ([57], [68])

It have been experienced that $\mathcal{F}$ is easier to fit than $\mathcal{H}$, because the network has just to let the weights tend to zero-values ([57]).
Using residual learning approach, when non-linear layers are added to the network, the identity map is easier to approximate and hidden-weights are more controlled. That technique work in a good way when the identity function is the optimal function, but also it shows good results for functions close to it ($\mathcal{H} \sim \mathbb{I}$).

In U-Net structures (see Fig. 2.7, [62]), residual learning concepts are exploited. In particular, the skip connections added may be:

- **Long skip connections**: they merge feature maps with the same size of the encoded and the decoded path. They keep together high and low resolution information.

- **Short skip connections**: they merge feature maps with the same size in the same building block.

An example is shown in Fig. 2.10, to perfom a segmentation task ([53]). The use of both these connections guarantees a decreasing in the error values and an increase of the speed of convergence.

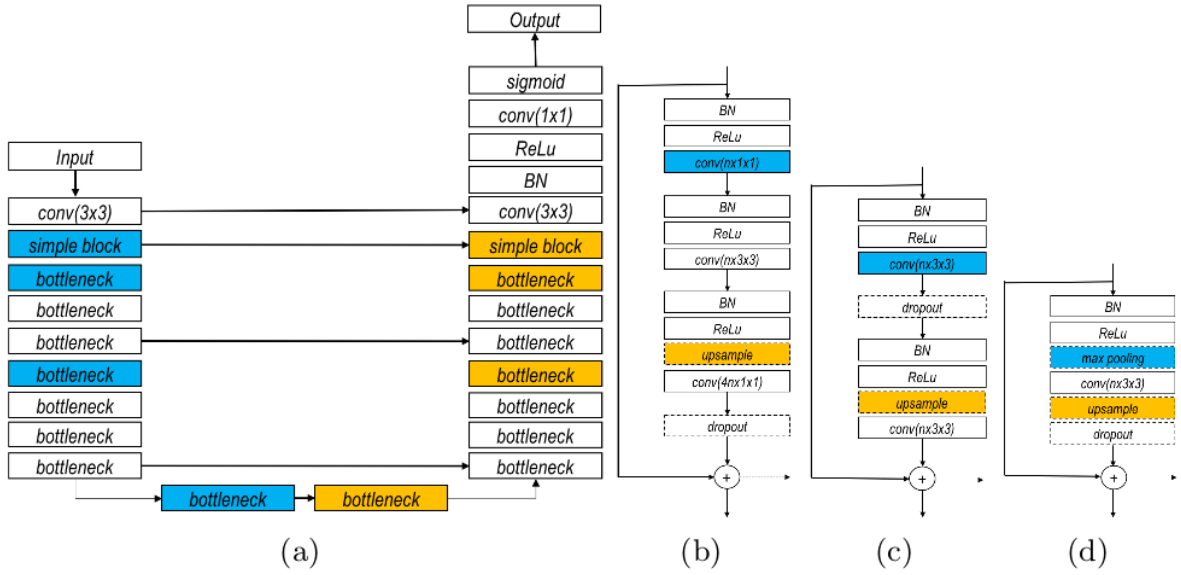


Figure 2.10: An example of residual network for image segmentation ([53]. (a) Residual Network with long skip connection, built from bottleneck blocks, (b) bottleneck block, with short skip connections (c) basic block and (d) simple block. Blue color indicate where a downsampling is optionally performed, yellow color depicts the (optional) upsampling blocks, dashed arrow in figures (b), (c) and (d) indicates possible long skip connections

Chapter 3

Materials and Methods

In the first two chapters of this thesis we made the theoretical introduction required to explain the susceptibility map reconstruction task (Chapter 1) with convolutional neural networks (Chapter 2). In view of that, the next pages describe the implemented experiments. We are going to focus on **2D** (Chapter 4) and **3D reconstruction** (Chapter 5). The **dataset** used and the **network structure development** will be explained in detail.

Chapter 3 is divided into the following sections:

- Section 3.1: **Susceptibility Map Reconstruction with Convolutional Autoencoders** - The ill-posed QSM problem ([1], [15]) could be addressed using learning algorithms designed for image reconstruction task. We are going to use a convolutional autoencoder architecture. In this section, the description of the approach used and a literature review is provided.

- Section 3.2: **Database** - Data from the QSM_{2016} Challenge ([10]) are described. Metrics chosen to evaluate the model performance are introduced. The last paragraph is about the dataset construction, and the **data augmentation** ([69]) techniques adopted to build it.

- Section 3.3: **Network Structure Optimization** - Starting from the U-Net structure ([62]), the model has to be optimized for the proper task. Supervised and unsupervised networks performances are compared. Then, the following aspects are analysed: long and short skip connections, loss function, batch normalization and dropout layers, batch size. In this chapter and in the following one, 2D experiments are described. Thus, the dataset construction and the network optimization are specific for the 2D reconstruction task.

- Section 3.4: **Conclusions**

3.1 Susceptibility Map Reconstruction with Convolutional Autoencoders

The magnetic susceptibility χ is a property of matter, which is useful for investigating healthy and unhealthy conditions in the human body. Starting from MRI measurements, it is possible to obtain an anatomical χ map: in fact, the $\chi(r)$ and the $\phi(r)$ distributions are linked by a mathematical relationship ([1], [15]). Fourier based methods are used to solve the direct problem ($\chi(r) \longrightarrow \phi(r)$): in the k-space, a simple point-wise multiplication between $\chi(k)$ and the dipole kernel $D(k)$ has to be performed ([31], Eq. 1.19).
In contrast, the inversion problem ($\phi(r) \longrightarrow \chi(r)$) is ill-posed. $D(k)$ (Eq. 1.20) assumes zero-values in some points in k-space. $D(k) = 0$ is the expression of a double-cone surface oriented along the $\vec{z}$ direction (Fig. 1.9). To find $\chi(k)$, $\phi(k)$ has to be divided by $D(k)$, which is null over the double cone surface. This is the issue associated with the QSM tool.
To overcome the ill-conditioned problem, single- and multiple- orientation approaches have been implemented ([7]). One example, respectively for the first and the second category, is the TKD (Sec. 1.2.6) and the COSMOS method ([5], Sec. 1.2.5). The TKD method, as all the other numerical approaches, requires a short acquisition time, but it returns a noisy susceptibility map. On the other hand, the COSMOS reconstruction is more precise and less noisy, but it takes a lot of time because more than one acquisition - three at least - is needed in order to obtain one susceptibility map.
With a view to introducing the QSM tool in the clinical area, the $\chi(r)$ reconstruction has to be fast. It could not require multiple scans for a single image. Also, the χ map has to be clear and noise-and-artifact free as more as possible. Both the TKD and the COSMOS approaches have advantages and drawbacks in respect of this aim.

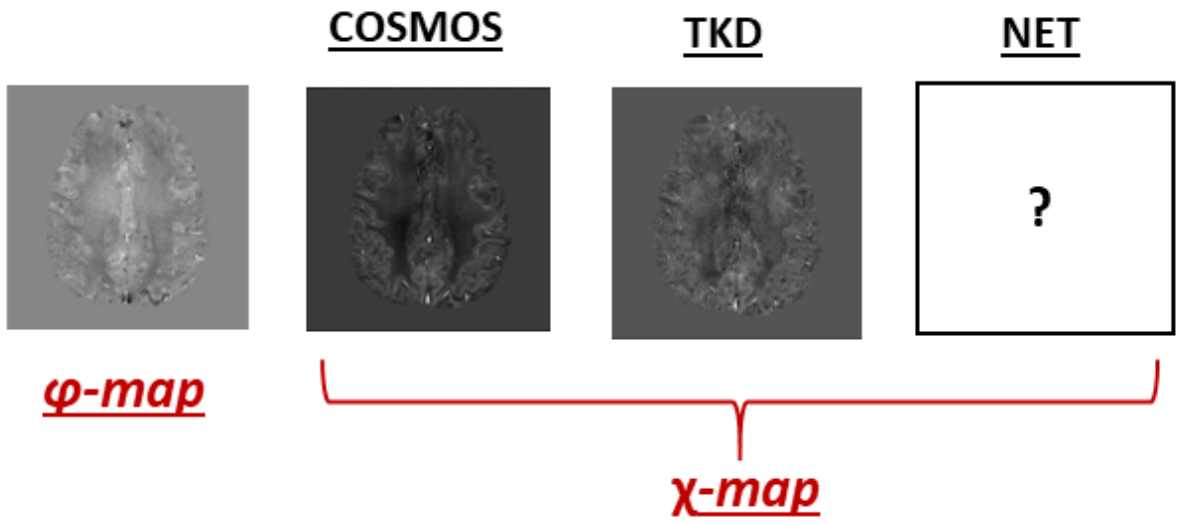


Figure 3.1: Phase data (left panel) were used to train a convolutional autoencoder in order to reproduce a quantitative susceptibility map. The results were compared with the COSMOS (central-left panel), multiple-orientation approach, and the TKD (central-right panel), single-orientation approach, reconstruction. The data used are from the QSM_{2016} Challenge ([10])

CNNs have been mainly used for image processing, also for image reconstruction tasks, for a few years - e.g. ImageNet Large Scale Visual Recognition Challenge, 2012 ([2]).
The idea behind this work is to exploit a **machine learning approach** to reproduce a **quick and artifact-and-noise free anatomical susceptibility map**, in order to allow the QSM technique to be applied in the clinical area. We have focused on brain scans, working in the **neuroimaging area**.
Thus, a convolutional autoencoder was trained to obtain an anatomical $\chi(r)$, both with 2D and 3D data. GRE-NMR phase data were used as input, with the COSMOS susceptibility reconstruction as reference, the TKD susceptibility reconstruction as control map (Fig. 3.1).

3.1.1 Literature Review

This work was inspired essentially by two works in the literature. Both of them were proposed in 2018: the first is from KGB Rasmussen et al. ([70]), and the other from J. Yoon et al. ([71]). Below, a brief description is given.

1. **KGB Rasmussen et al., 2018** ([70])
   They trained a U-Net model with only long skip connections. Input and output data are 3D synthetic patches with size (48,48,48).
   70400 synthetic examples were simulated and used as ground truth. They were convolved with the dipole kernel $D(k)$:

$$\mathcal{F}(D(r)) = D(k) = \frac{1}{3} - \frac{k_z^2}{k_x^2 + k_y^2 + k_z^2} \tag{3.1}$$

   The resulting patches were used as input.
   The structure has a level of depth of five, and the number of feature maps ranges from 16 to 256. The kernel size of the convolutional and sampling layers is respectively (5,5,5) and (2,2,2).
   ReLU (Fig. 2.4, [50]) was used as the activation function, ADAM ([54]) as optimizer with learning rate $\eta = 0.001$, MSE as loss function, dropout fraction was set to 0.1 - we will return later to the dropout layer function.
   Single-orientation data were used to train the model.
   After the training, it was tested with real single-orientation phase data, in healthy and unhealthy - Multiple Sclerosis disease - conditions.

2. **J Yoon et al.** ([71]
   They trained a U-Net structure with only long skip connections. Input and output are 3D real patches with size (64,64,64). The training dataset is composed by 16800 patches, from 3T-acquisitions.
   The network has five levels of depth. The number of feature maps ranges from 32 to 512, the kernel size of convolutional and sampling layers is respectively (5,5,5) and (2,2,2).

The loss function was built *ad hoc*, as a weighted sum of the three components E_{Model}, E_{L1} and $E_{Gradient}$:

$$E_{tot} = w_1 E_{model} + w_2 E_{L1} + w_3 E_{Gradient} \tag{3.2}$$

E_{Model} takes into account the difference between the convolution of the dipole and the label and the convolution of the dipole and the output; E_{L1} looks into the difference between the label and the output of the model; $E_{Gradient}$ preserves the edge information in the reconstructed map. The weights w_1, w_2, w_3 are respectively 1,1 and 0.1.
Even if the physical model behind the QSM problem is considered in building the loss function, supervised learning techniques have still been used to perform the training.

3.2 Database

In this section, we introduce the data used for the training, validation and test dataset construction. Then, the metrics parameters adopted to evaluate the model performance and the strategies necessary to build an efficient dataset will be described. In the next section, we are going to focus on the network architecture.

3.2.1 QSM_{2016} Challenge

To train the model, we built a dataset using three dimensional maps from the QSM_{2016} Challenge ([10]). They consist of MRI data, acquired in a healthy female volunteer, 30 years of age, using a 3T system (Tim Trio Siemens Health-care GmbH, Erlangen, Germany). The images have original size (160,160,160), and they were acquired with 1.06 mm isotropic resolution. These data are completly freely downloadable, and contain:

- 3D GRE magnitude (Fig. 3.2, left panel) and wrapped phase (Fig. 3.3, left panel) images acquired with axial slab orientation. To perform the multiple-orientation susceptibility reconstructions (the COSMOS and the STI maps), 12 head-orientation scans were aquired.

- A brain mask obtained from FSL (https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/) Brain Extraction Tool ([72])

- Magnitude images masked (Fig. 3.2, right panel)

- Phase images were unwrapped and masked (Fig. 3.3, central panel), and processed with LBV ([25]), a background removal algorithm (Fig. 3.3, right panel)

- The COSMOS susceptibility map (Fig. 3.4, left panel), obtained with the 12 pre-processed phase maps

- The STI susceptibility map (Fig. 3.4, central panel), obtained with the 12 pre-processed phase maps. Referring to the Eq. 1.28, for each voxel χ_{33} is taken as the final susceptibility value

- 12 TKD susceptibility map (Fig. 3.4, right panel), one for each orientation. The threshold α was set to 0.1

The susceptibility maps were scaled from -0.1 to 0.25 ppm (SI units), the raw phase from $-\pi$ to $+\pi$ radians, and the LBV-phase image from -0.05 to + 0.05 radians.

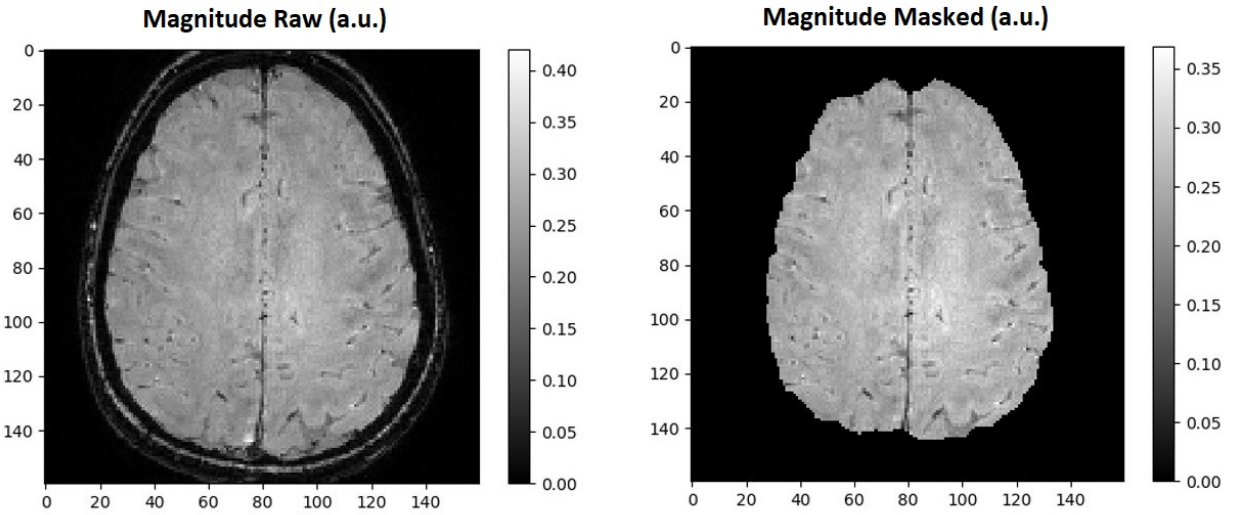


Figure 3.2: Axial section from a 3D GRE magnitude image, raw (left panel) and masked (right panel). The original images are 3D maps with size (160,160,160), from the QSM_{2016} Challenge ([10])

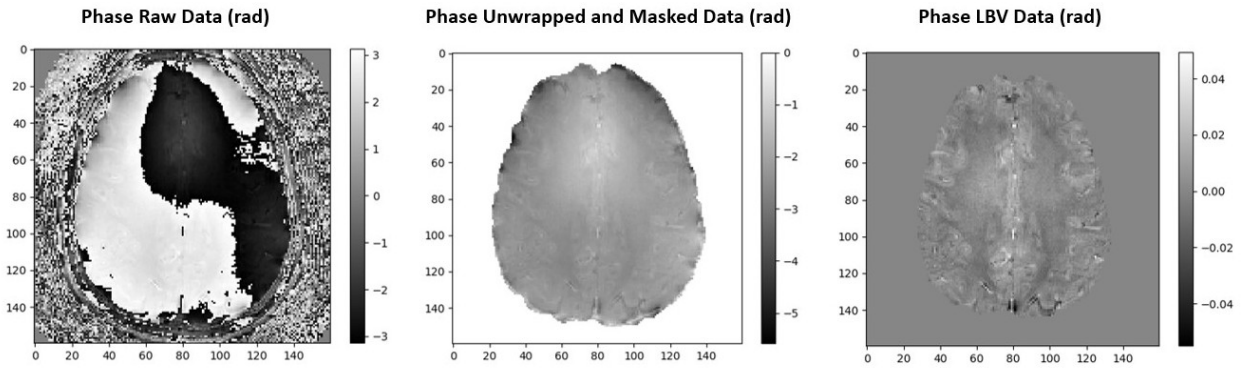


Figure 3.3: Axial section from a 3D GRE phase image, wrapped (left panel), unwrapped and masked (central panel) and background-field corrected (right panel). The background removal algorithm used is LBV ([25]). The original images are 3D maps with size (160,160,160), from the QSM_{2016} Challenge ([10])

In Fig. 3.5, a scheme of the brain sections with respect to the reference system of the frame is reported. The axial slices are on the $\mathbf{xy-plane}$, the sagittal ones on the $\mathbf{yz-plane}$, the coronal ones on the $\mathbf{zx-plane}$. In all the images, even if they were acquired changing the head orientation with respect to the main magnetic field $\vec{1}1B_0$, the $\vec{z}_{head}$ axis of the head is coincident with the vertial axis of the frame. In fact, the images were affine-registered ([73]) to the axial-slab orientation.

Registration

The registration is a pre-processing operation frequently computed after the acquisition. It consists in rotating and translating the images in order to have the head direction and

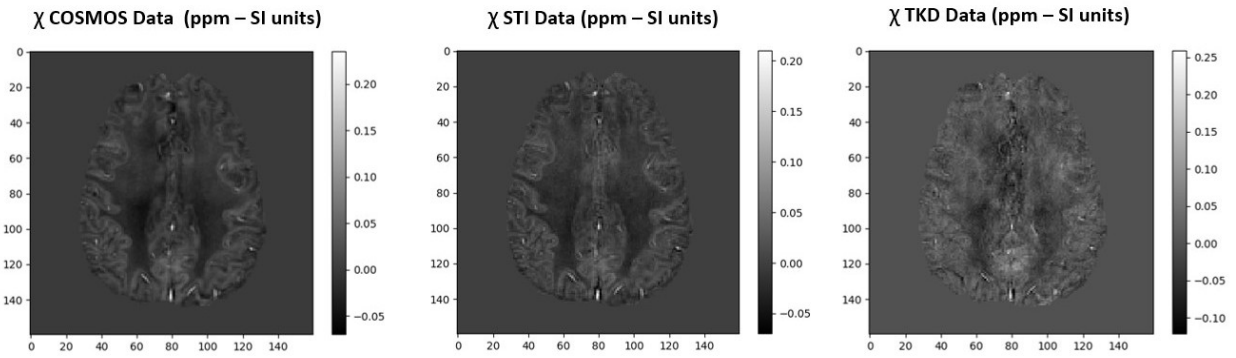


Figure 3.4: Axial section from 3D susceptibility map, obtained with: COSMOS ([5], left panel), STI ([6], central panel, TKD ([7], right panel). The COSMOS and STI (χ_{33}, Eq. 1.28) reconstructions were obtained from 12 head-orientation scans. The threshold set for the TKD reconstruction was 0.1. The original images are 3D maps with size (160,160,160), from the QSM_{2016} Challenge ([10])

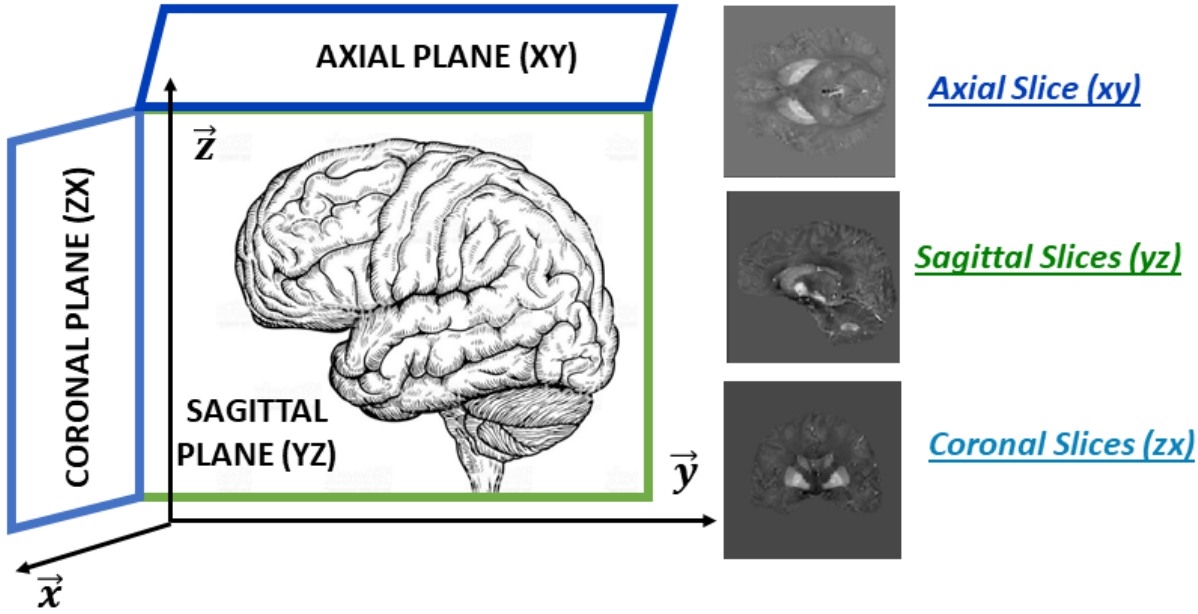


Figure 3.5: Scheme of the brain sections with respect to the reference system of the frame

position fixed with respect to the reference system of the frame ([73]).
Referring to the scheme in Fig. 3.6, let $\vec{B_0}$ be the main magnetic field, $\vec{z}$ the direction of the head, $\vec{n}$ the vertical direction of the frame. Let ϑ be the angular distance between $\vec{B_0}$ and $\vec{z}$. In the **0^{th}-direction** situation (left panel), $\vartheta = 0$, so the field, the head and the frame are all oriented in the same way. In this case, no registration operation is required.
When the head is rotated inside the scan ($\vartheta \neq 0$), without registering the image the frame is still directed along the static magnetic field ($\vec{B_0} \parallel \vec{n}$, central panel). To register means to rotate the image by an angle $(-\vartheta)$, to have $\vec{z} \parallel \vec{n}$ (right panel).
The images in the QSM_{2016} Challenge database ([10]) were registered. The mutual position between the images and the frame is fixed. Also, the static magnetic field $\vec{B_0}$ is not always in the same direction with respect to the reference system of the frame. In the dataset, the rotation matrices of the affine transformations computed are available.

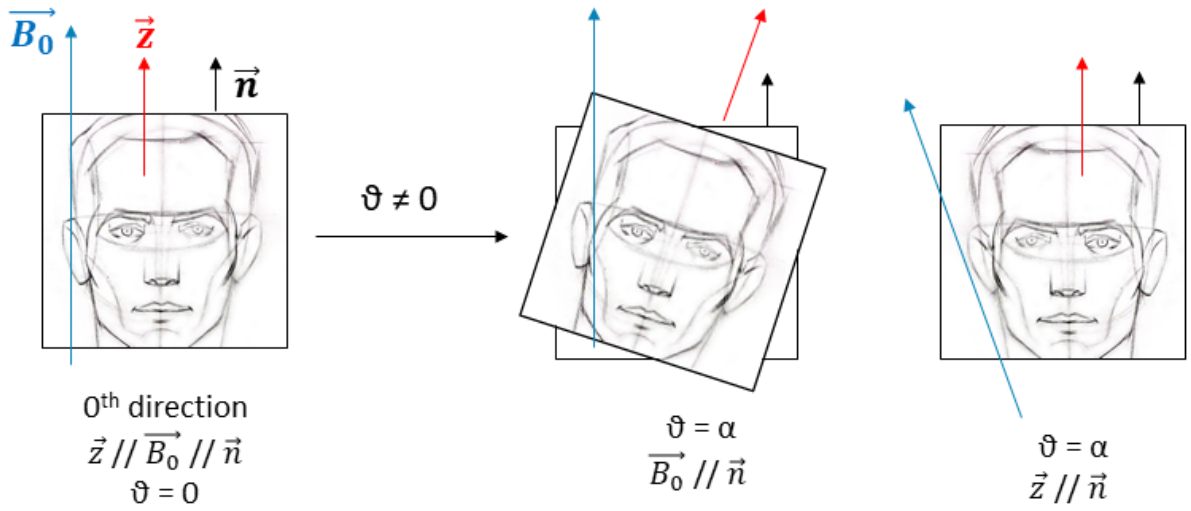


Figure 3.6: Left panel: acquisition with main magnetic field B_0, $\vec{z}$ axis of the head and $\vec{n}$ axis of the frame all aligned. Right panels: acquisition with an angular distance $\vartheta = 0$ between B_0 and $\vec{z}$. The difference between the acquisition with and without registration is shown

3.2.2 Metrics

To quantify the similarity between the single-orientation methods, i.e. the TKD and the machine learning approach, and the gold-standard COSMOS method, the following parameters were used ([74]). In the succeeding definitions, $\mathbf{y}$ is the reference map and $\mathbf{x}$ the variable being calculated. They are multi-dimensional arrays with size N.

- **RSME - Root Mean Square Error**

$$MSE(\mathbf{x},\mathbf{y}) = \frac{1}{N}\sum_{i=0}^{N-1}(x_i - y_i)^2 \tag{3.3}$$

$$RMSE(\mathbf{x},\mathbf{y}) = \sqrt{MSE(\mathbf{x},\mathbf{y})} \tag{3.4}$$

$$RMSE(\mathbf{x},\mathbf{x}) = 0\ ,\ RMSE(\mathbf{x},\mathbf{y}) = RMSE(\mathbf{y},\mathbf{x}) \tag{3.5}$$

- **pSNR - peak Signal-to-Noise Ratio**

$$pSNR(\mathbf{x},\mathbf{y}) = 10\log_{10}\left(\frac{max(\mathbf{x})}{MSE(\mathbf{x},\mathbf{y})}\right) \tag{3.6}$$

$$pSNR(\mathbf{x},\mathbf{x}) = 1\ ,\ pSNR(\mathbf{x},\mathbf{y}) \neq pSNR(\mathbf{y},\mathbf{x}) \tag{3.7}$$

- **SSIM - Structural SIMilarity index**

$$SSIM(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.8}$$

where μ_i and σ_i are respectively the mean value and the standard deviation of the i^{th} image, $\sigma_{i,j}$ is the covariance index. $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$ are two variables introduced to stabilise the division in case of weak denominators, k_1, k_2 are constant

and usually set equal to 0.01, 0.03. L is the dynamic range.
SSIM is useful to compare local patterns of pixel intensities. This parameter includes luminance $l(\mathbf{x},\mathbf{y})$, contrast $c(\mathbf{x},\mathbf{y})$ and structure $s(\mathbf{x},\mathbf{y})$, which are respectively defined as:

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C1}{\mu_x^2 + \mu_y^2 + C1} \tag{3.9}$$

$$c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{3.10}$$

$$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, C_3 = C_2/2 \tag{3.11}$$

$$SSIM(\mathbf{x},\mathbf{y}) = l^\alpha c^\beta s^\gamma, \alpha = \beta = \gamma = 1 \tag{3.12}$$

$$SSIM(\mathbf{x},\mathbf{x}) = 1\ ,\ SSIM(\mathbf{x},\mathbf{y}) = SSIM(\mathbf{y},\mathbf{x}) \tag{3.13}$$

- **HFEN - High Frequency Error Norm**

$$HFEN(x,y) = \frac{\|LoG(x) - LoG(y)\|_2}{\|LoG(y)\|_2} \tag{3.14}$$

where $LoG(i)$ is a rotationally symmetric Laplacian of Gaussian filter capturing detailed textures.

$$HFEN(\mathbf{x},\mathbf{x}) = 0\ ,\ HFEN(\mathbf{x},\mathbf{y}) \neq HFEN(\mathbf{y},\mathbf{x}) \tag{3.15}$$

We have also computed contrast analysis for the TKD, COSMOS and the network reconstruction. The **CNR - Contrat-to-Noise Ratio** is used; it is defined as:

$$CNR = \frac{|I_i - I_{bck}|}{\sigma_{bck}} \tag{3.16}$$

I_i is the mean intensity of the region insterested in, I_{bck} and σ_{bck} are respectively the mean intensity and the standard deviation of the background. The background area has to be chosen as uniform as possible, in order to hove σ_{bck} as small as possible. CNR is very sensitive with respect the fluctuations of the background.

3.2.3 Data Augmentation

The **generalization** is the most important skill to reach for a learning algorithm. If the model works well only with the data used during the training stage, it means that it has to be trained every time the dataset changes. This means that the learning algorithm loses its power and becomes unuseful. Possible reasons for the inability in generalizing are the **overfitting** and the **underfitting** conditions (Sec. 2.1.1, Fig. 2.1).

In the medical area, a wrong fit is usually due to the poor availability of data. This is a big deal for machine learning experiments: if any failure occurs during the training, the model is not able to generalize on validation and test dataset ([75]).
There are many strategies to adopt when designing network structures - e.g. the addition of **dropout** ([3]) or **batch-normalization** ([76]) layers. For instance, the dataset to use has to be built properly. In fact, the amount of data could be increased using information only from the original dataset. This is called **data augmentation**. It consists in applying geometrical or color transformations to the available data. These operations fall under the category of **data warping strategies**. There are a lot of image warping methods ([69]), parametrics and non-parametrics - e.g. Bayesian-approach-based transformations.
Usually, only geometric parametric transformations are used to create datasets for a machine learning algorithms ([53]). Below, a brief list is reported.
Let (x, y) be the coordinates of a pixel in the input image, and (u, v) the correspondent coordinates in the returned map. Then:

- **Translation** - either along rows, columns or a combination of both:

$$u = x + a_{00} \ , \ v = y + b_{00} \tag{3.17}$$

- **Procrustes Transformations** - thay allow changes in magnification and rotations of ϑ degrees:

$$u = cx\cos\vartheta + cy\sin\vartheta + a_{00} \ , \ v = -cx\sin\vartheta + cy\cos\vartheta + b_{00} \tag{3.18}$$

  The constant c performs a rescaling

- **Affine Transformations** - they are six-parameter transformations, which are the generalization of the Procrustes functions. They permit: stretching along rows and columns, flipping and rotations, shearing. They are the most general linear transformations:

$$u = a_{10}x + a_{01}y + a_{00} \ , \ v = b_{10}x + b_{01}y + b_{00} \tag{3.19}$$

- **Bilinear Transformations** - they are eight-parameter transformations, generalization of the affine functions. They include also mixed terms:

$$u = a_{10}x + a_{01}y + a_{11}xy + a_{00} \ , \ v = b_{10}x + b_{01}y + b_{11}xy + b_{00} \tag{3.20}$$

- **Polynomial Transformations** of order p are defined by:

$$u = \sum_{i=0}^{p}\sum_{j=0}^{p-i} a_{ij}x^i y^j \ , \ \sum_{i=0}^{p}\sum_{j=0}^{p-i} b_{ij}x^i y^j \tag{3.21}$$

3.2.4 Datasets Construction Pipeline

- **2D Single-Orientation Dataset Construction - Axial Slices**:
  Patches (x_p, y_p)
  Z_{cut} axial slices (Fig. 3.5) are selected from the 0^{th}-orientation map (Fig. 3.6). Each slice is rotated by K random angles. In each rotated slice N couples (x_i, y_i) are randomly extracted, and the patches $(x_i : x_i + x_p, y_i : y_i + y_p)$ compose the dataset. Its dimension is: $s = Z_{cut} \cdot K \cdot N$.

- **2D Multiple-Orientation Dataset Construction - Axial Slices**:
  Patches (x_p, y_p)
  D different head-orientation maps are selected. Z_{cut} axial slices (Fig. 3.5) are selected from each map. Each slice is rotated by K random angles. In each rotated slice N couples (x_i, y_i) are randomly extracted, and the patches $(x_i : x_i + x_p, y_i : y_i + y_p)$ compose the dataset. Its dimension is: $s = D \cdot Z_{cut} \cdot K \cdot N$.

- **2D Multiple-Orientation Dataset Construction - Axial, Sagittal and Coronal Slices**:
  Patches (x_p, y_p)
  D different head-orientation maps are selected. X_{cut} sagittal slices, Y_{cut} coronal slices and Z_{cut} axial slices (Fig. 3.5) are selected from each map. $C = X_{cut} + Y_{cut} + Z_{cut}$. Each s^{th} slice, with $s = 1, 2, .., C$, is rotated by K random angles. In each rotated slice N couples (x_i, y_i) are randomly extracted, and the patches $(x_i : x_i + x_p, y_i : y_i + y_p)$ compose the dataset. Its dimension is: $s = D \cdot C \cdot K \cdot N$.

- **3D Multiple-Orientation Dataset Construction**:
  Patches (x_p, y_p, z_p)
  D different head-orientation maps are selected. Each map is rotated K times by random angles, with respect the $\vec{x}$, $\vec{y}$ or $\vec{z}$ axis. In each rotated map, N groups (x_i, y_i, z_i) are randomly extracted, and the patches $(x_i : x_i + x_p, y_i : y_i + y_p, z_i : z_i + z_p)$ compose the database. Its dimension is: $s = D \cdot K \cdot N$.

The **random extractions** performed are all from **uniform distributions**. This is important, in order to avoid overfitting and loss of generalization capability.
The datasets thus obtained are used to build the training dataset ($s_{train} = s \cdot f_{train}$), the validation dataset ($s_{val} = s \cdot f_{val}$) and the test dataset ($s_{test} = s \cdot f_{test}$).

The exact same operations are computed at the same moment on the ϕ, the χ-COSMOS and the χ-TKD map, for the purpose of having coincident patches to use as input data (ϕ), label data (χ-COSMOS) and control data (χ-TKD). The ϕ data used are unwrapped and background-field corrected (Fig. 3.3, right panel).

We decided to use ReLU (Sec. 2.2, Fig. 2.4) as activation function, so the histogram of all the maps has to be scaled into the range ($min_{new} = 0, max_{new} = 1$) (**histogram stretching**):

$$r = \frac{(s - min_{old})}{(max_{old} - min_{old})} * (max_{new} - min_{new}) + min_{new} \tag{3.22}$$

r is the intensity value of the pixel in the returned map, s the intensity value of the same pixel in the original map. This operation has to be performed on the entire set of 3D images, even if the algorithm is then designed for 2D data.

3.3 Network Structure Optimization

The first implemented model (**NET1**) is shown in Fig. 3.7. Its goal is to reproduce a 2D anatomical susceptibility map, starting from phase data. It does not take the entire brain map as input, but patches with size (x_p, y_p).
NET1 consists of a four-level-depth autoencoder architecture (Sec. 2.4). In each block of the structure, convolutional (kernel size (3,3)) and pooling (kernel size (2,2)) layers alternate. The size and the number of the feature maps are indicated in the figure. The input and the output data are 2D patches (x_p, y_p, k) = (64,64,1). The k coordinate represents the number of feature maps, other than one going deeper in the structure. In the latent representation, the size of the processed data is (8,8,256): (8,8) is the minimum map size reached in this model, 256 is the maximum number of feature maps.
Only long skip connections were included to NET1 ([53], Sec. 2.4.2). Remember that they are necessary to guarantee an sufficiently detailed reconstruction, because they keep together high- and low- resolution information.

3.3.1 Supervised and Unsupervised Learning

Initially, we verified the requirement for performing a supervised training. NET1 was trained twice, with and without labels. Patches from the COSMOS map were used as labels. The training details are in Table 3.1.

The top panels of Fig. 3.8 show the shape of the loss function during the training, both for the training and the validation dataset. The left panel refers to the unsupervised training, the right panel to the supervised one. NET1 performs a good fit with both of them. But, looking at the same picture on the bottom, it should be noticed that, without including label data during the training (unsupervised learning), the model output is pretty much equal to the input. So, from now on, we are going to perform only **supervised learning experiments**.

The **speed of convergence** is another aspect to be observed during the performance evaluation. It is related to the decay constant of the loss shape along the epochs.

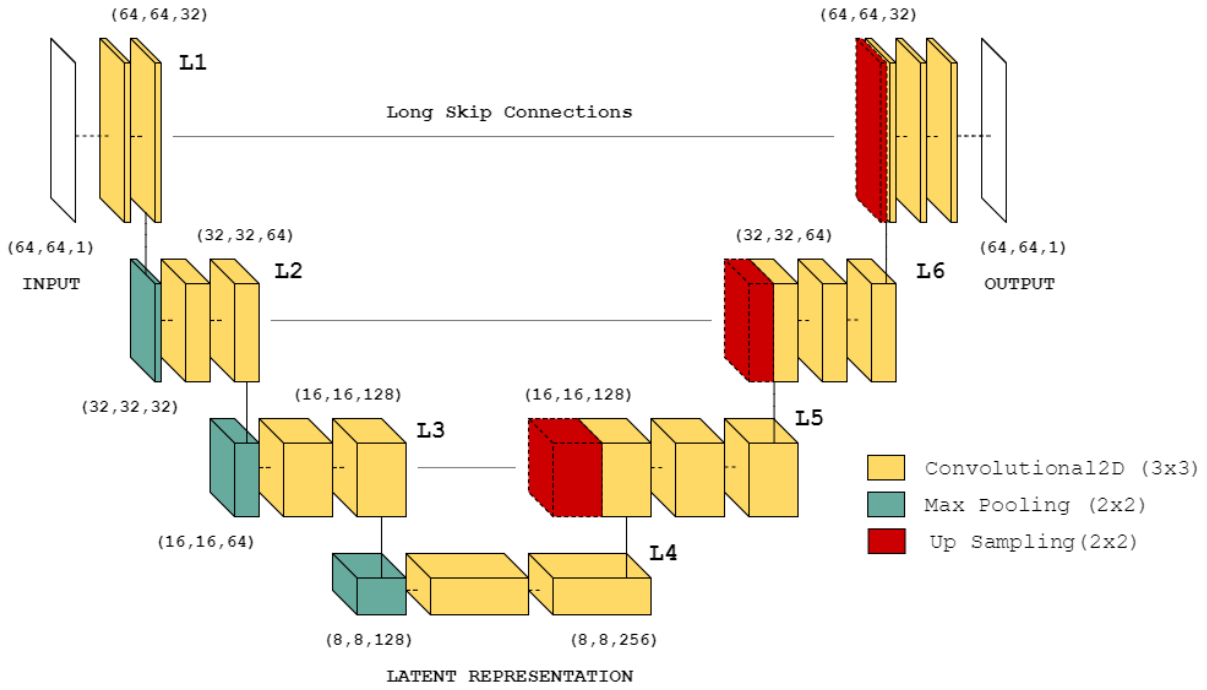


Figure 3.7: NET1: convolutional autoencoder with 4 levels of depth. Input and output have size (64,64,1). Convolutional and pooling (sub- and up- sampling) layers form the structure. The number of feature maps is from 32 to 256. Long skip connections merge feature maps with the same size between the contracting and the expanding paths

NET1
Dataset size: 6400 patches (64,64), axial slices, single-orientation data
RMSProp optimizer, $\eta = 0.001$
MSE loss function
Batch size: 128
Epochs:100
Training time: 1h, 10 min
Keras 2.2.0, Tensorflow 1.9.0 Backend, Python 3.6
GPU GeForce 940MX - nVdia 384, Cuda 9.0

Table 3.1: Training details of Sec. 3.3.1 experiments. NET1 was trained twice, with (supervised learning) and without (unsupervised learning) labels

In Fig. 3.9, there is a comparison between the unsupervised-training- (blue line) and the supervised-training- (orange line) loss function. Notice that the model trained without labels is faster in converging than the other one.

Similarity parameters (Sec. 3.2.2) were evaluated from the test dataset, only for the supervised training. The test dataset size is equal to 640 ($s = 6400$, $f_{test} = 0.1$). The results are shown in the Table 3.2.
The NET1 performance was compared with that of the TKD method; for both, the COSMOS map was used as reference. Our model produces better results than the second one, in accordance with the parameters chosen.
In Fig. 3.10, a random patch from the test dataset is shown in: ϕ, TKD, NET1 and COSMOS maps. Looking at these, the NET1 reconstruction is clearly closer to the COSMOS

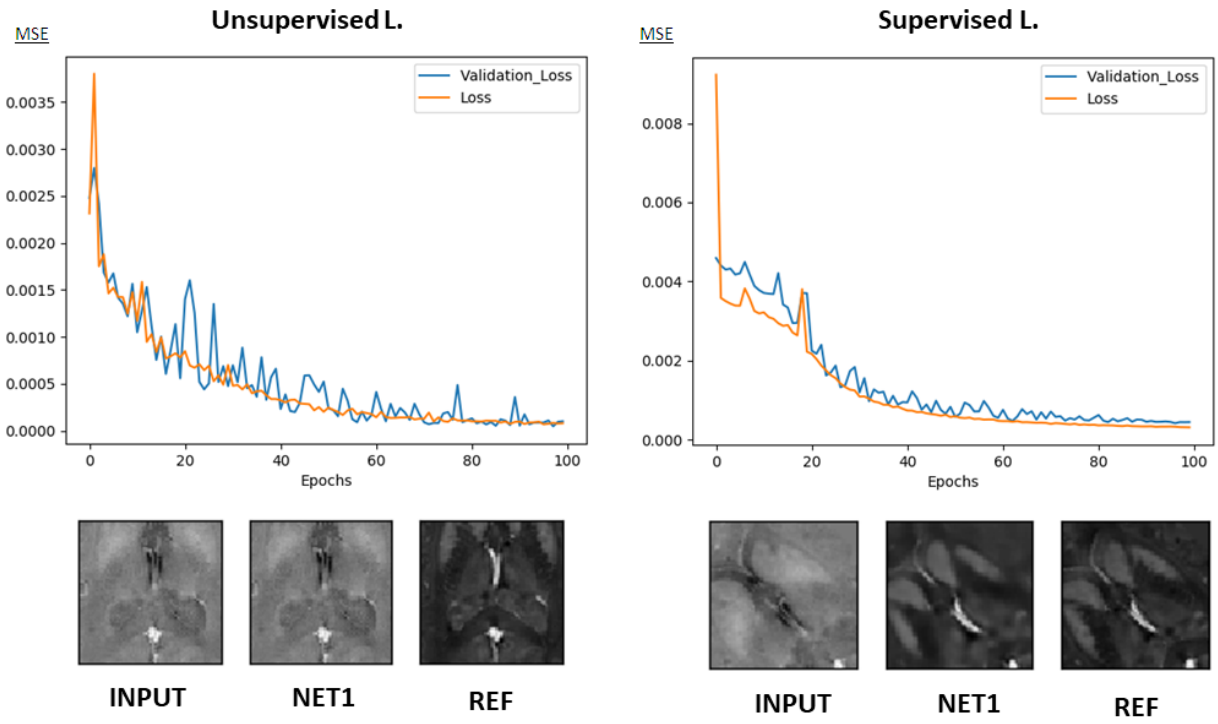


Figure 3.8: Left panel: on the top, validation (blue) and training (orange) loss function shape in unsupervised training; on the bottom, an example of a random patch, from the test dataset, in input, output and label stage. Right panel: on the top, validation (blue) and training (orange) loss function shape in supervised training; on the bottom, an example of a random patch, from the test dataset, in input, output and label stage. Training details in Tab. 3.1

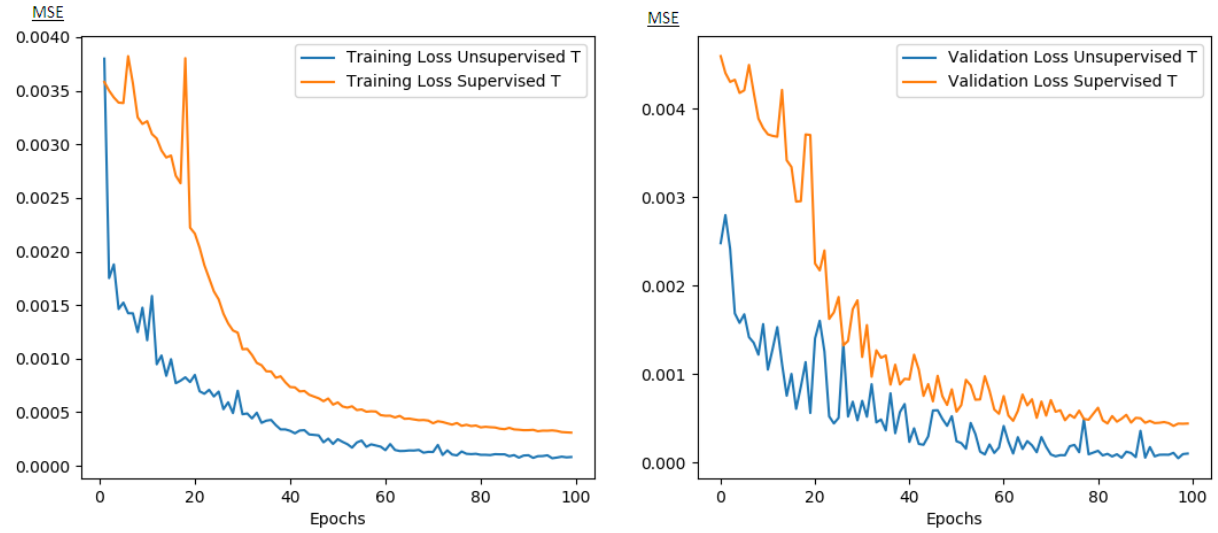


Figure 3.9: Loss function shape in unsupervised (blue) and supervised (orange) training. Left panel: training loss. Right panel: validation loss. Training details in Tab. 3.1

data than TKD, and it also definitely less noisy.

The NET1 map is smoother than the COSMOS one. This is not surprising, since convolutional autoencoders are also used to perform **denoising** operations.

In this specific patch, there are three important brain structures: the putamen, caudate and globus pallidus. We will focus on those ROIs, after the network optimization (Chapter 4 and 5).

S T	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
ϕ	0.0176	0.0056	15.4	1.6	0.75	0.09	1.43	0.25
NET1	0.0004	0.0001	31.7	1.3	0.95	0.01	0.63	0.16
TDK	0.0023	0.0005	23.3	1.3	0.89	0.03	0.73	0.08

Table 3.2: RSME, pSNR, SSIM and HFEN, for supervised training, details in Tab. 3.1. The values indicated are the average values along the test dataset (size 640)

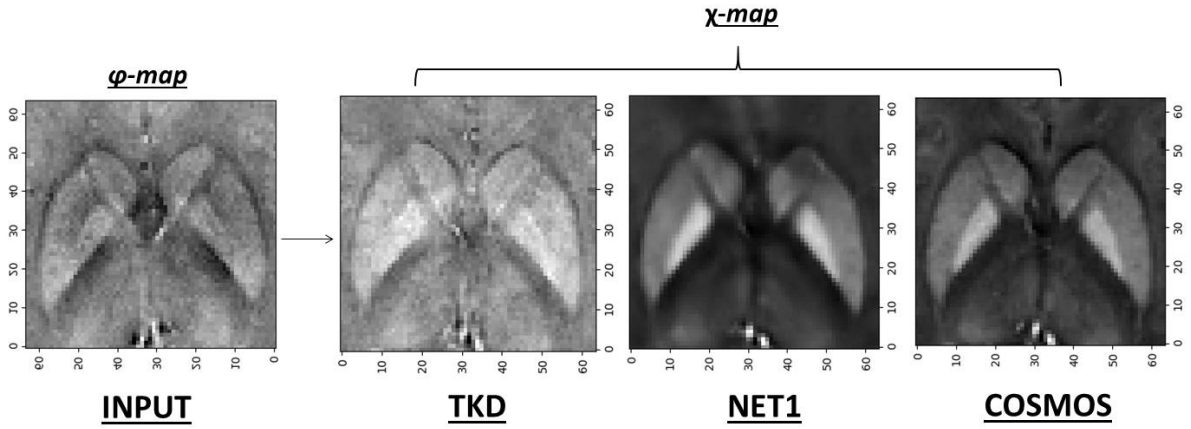


Figure 3.10: Random patch from the test dataset in: ϕ map (left), χ-TKD map (central-left), χ-NET1 (central-right), χ-COSMOS map (right). Training details of NET1 (Fig. 3.7) in Tab. 3.1

Conclusions

- As expected, a model trained using unsupervised learning techniques and having ϕ data as input, can not reconstruct a susceptibility map. So, from now on, we are going to use only a supervised learning approach.

- It could be interesting to try to perform an unsupervised training using an appropriate cost function, which takes into account the physical model behind the reconstruction task. In [71], they introduce an original loss function (Eq. 3.2), but they still use supervised learning techniques.

- A supervised training of NET1 lets the model reproduce a χ-map that is more similar to the reference (COSMOS) than the TKD method. Similarity parameters along the test dataset are reported in Tab. 3.2. An example of recostruction may be found in Fig. 3.10.

3.3.2 Short Skip Connections

The use of **Skip Connections SCs** is fundamental in image processing tasks ([53], Sec. 2.4.2). They may merge feature maps in the same block (**short SCs**) or between the encoded and decoded paths (**long SCs**). In NET1 (Fig. 3.7), only the latter were added to the model. The presence of the short SCs, in addition to the long ones, helps in **improving the speed of convergence** of the model and in **fixing further the vanishing/exploding gradient problem** (Sec. 2.3), which affects multi-hidden-layer structures, especially in the deepest levels ([53]). So, we built a new model, including short skip connections in each block. The structure is shown in Fig. 3.11, left panel (**NET2**).
NET2 was trained. The details are reported in Tab. 3.3. The training time, because of the short SCs addition, is slightly higher than before (cf. Tab. 3.1).

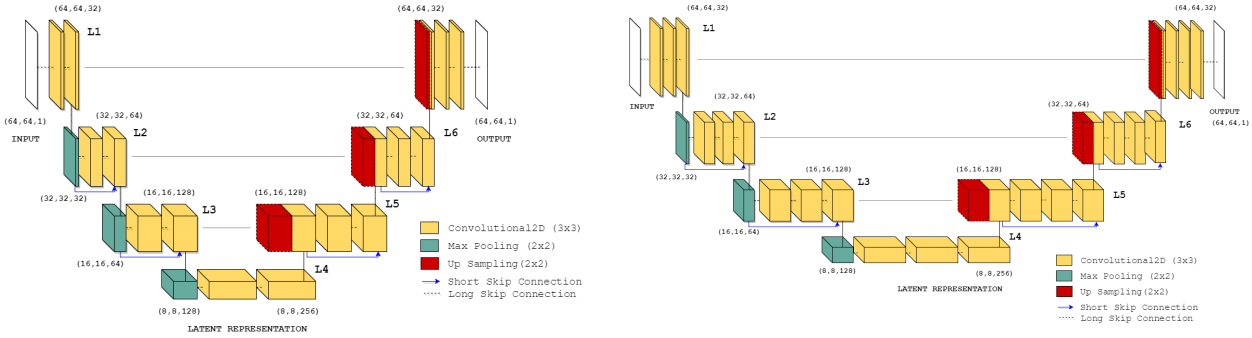


Figure 3.11: Left panel: NET2 - convolutional autoencoder with 4 levels of depth. Input and output have size (64,64,1). Convolutional and pooling (sub- and up- sampling) layers compose the structure. The number of feature maps is from 32 to 256. Long and short skip connections merge feature maps with the same size. Right panel: NET2.0 - with respect NET2, there is an extra convolutional layer in each block

NET2 | NET2.0
Dataset size: 6400 patches (64,64), axial slices, single-orientation data
RMSProp optimizer, $\eta = 0.001$
MSE loss function
Batch size: 128
Epochs:100
Training time: 1h, 50 min | 3h, 40 min
Keras 2.2.0, Tensorflow 1.9.0 Backend, Python 3.6
GPU GeForce 940MX - nVdia 384, Cuda 9.0

Table 3.3: Training details of Sec. 3.3.2 experiments. NET2 and NET2.0 were supervised trained with single-orientation data

After adding short SCs, the speed of convergence actually upgrades, but there were no improvements with respect to the loss gradient issue. To evaluate the behaviour of the cost gradient, we exploited the fact that (Sec. 2.1.2, Eq. 2.10):

$$\Delta w_i \propto \nabla_{w_i} E \tag{3.23}$$

There is a Keras tool which allows the values of the weights of each layer to be returned over the training time. Thus, we could design for the r^{th} layer the colour map with: number of neurons i on the $\vec{x}$ axis, number of epochs t of the $\vec{y}$ axis, $|\Delta w_i(t)| = |w_i(t) - w_i(t+1)|$ on the $\vec{z}$ axis. In this way, we realized that there were some regions, in the second, third and fourth levels of depth, in which the weights did not change ($|\Delta w_i(t)| = |w_i(t) - w_i(t+1)| = 0$, $\forall\, t$). That happened during the training of NET1 (Fig. 3.7). The addition of the short skip connections (NET2, Fig. 3.11) did not completely fix this issue.

To eliminate or at least to reduce the number and the size of zero-update patterns, we added another convolutional layer for each block at each level (Fig. 3.11, right panel, **NET2.0**). NET2.0 was trained, the training details are reported in Tab. 3.3. The training time, on an equal database size, is significantly higher than before. In fact, convolutional layers entail a big computational effort.
This amendment improves the weight update issue, reducing the zero-update patterns.

Conclusions

- Short skip connections were added to the model (**NET2**), and a convolutional layer was added to each block (**NET2.0**, Fig. 3.11). In this way, the speed of convergence of the network improves and the vanishing/exploding gradient problem is reduced.

- The similarity parameters in the experiments described in this section (Tab. 3.3), obtained with a model including both long and short SCs, are consistent with those obtained with NET1 (Tab. 3.2), which was a structure containing only long SCs. We aim to design a solid model, so it is important also to take into account other aspects of the training, such as the speed of convergence and the weight updates, as well as the final results. The model has to be tight because we are going to broaden the range of the dataset, adding multiple-orientation measurements and working with 3D data.

- Note that the addition of a convolutional layer is not always a benefit. We tested the model by adding another convolutional layer to each block, and the results were worse than before.

2D Multiple-Orientation Dataset and Skill Generalization

NET2.0 was also trained using a multiple-orientation dataset, the details are in Tab. 3.4. The training time improved because the size of the dataset is bigger than before. The dataset used contains **ten-head-orientation** maps. **Two-head-orientation** maps were left out on purpose, to test, after the training, the generalization skill of the network.

The shape of the loss function, comparing single-orientation- (blue line) and multiple-orientation-dataset (orange line) training, is shown in Fig. 3.12 (left panel: training error,

NET2.0
Dataset size: 8000 patches (64,64), axial slices, single-orientation data
RMSProp optimizer, $\eta = 0.001$
MSE loss function
Batch size: 128
Epochs:100
Training time: 4h, 30 min
Keras 2.2.0, Tensorflow 1.9.0 Backend, Python 3.6
GPU GeForce 940MX - nVdia 384, Cuda 9.0

Table 3.4: Supervised training details of NET2.0 with 10-orientation data

right panel: validation error). From the first to the second one, the speed of convergence improves, thanks to the broadening of the input data range and the addition of multiple-orientation data. The dataset construction has become more random, and that helps the generalization skill.

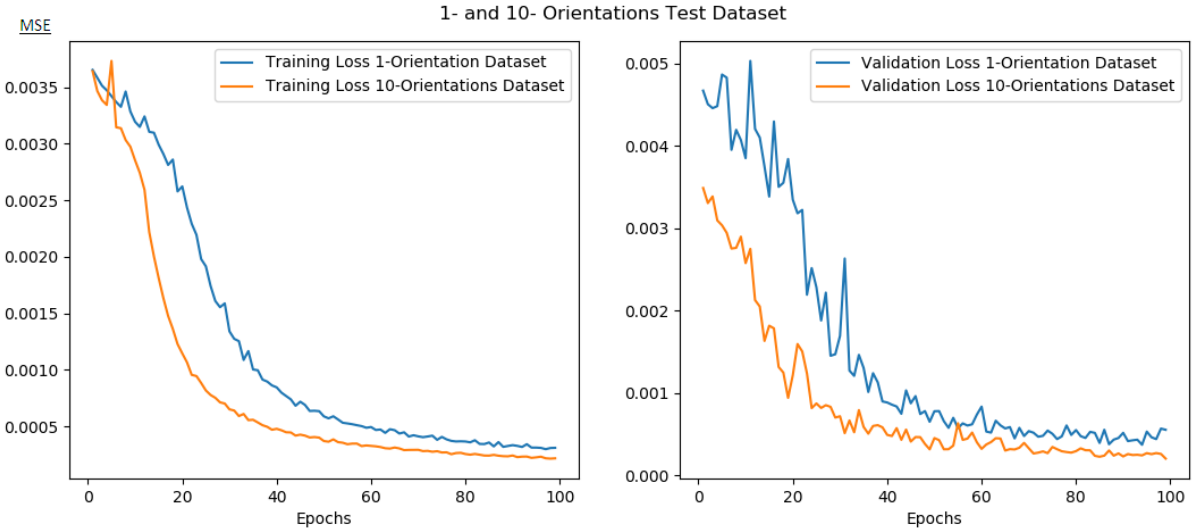


Figure 3.12: Loss function shape of NET2.0 with single-orientation (blue line) and multiple-orientation (orange line) dataset. The training details are respectively in Tab. 3.3 and 3.4. Left panel: training loss. Right panel: validation loss

Fig. 3.13 shows reconstructed patches, to evaluate the model reconstruction skill. First row: random patch from the ten-orientation-, used during the training, test dataset. Second row: random patch from the two-orientation-, unused during the training, dataset.
Tab. 3.5 details the similarity parameters - average values along the two datasets - respectively in the first and in the second block. The parameters in the first block are better than before (cf. Tab. 3.2), while the ones in the second block are basically the same. In an ideal situation, the model has to behave in the same way both for already seen and unseen head-orientation data.

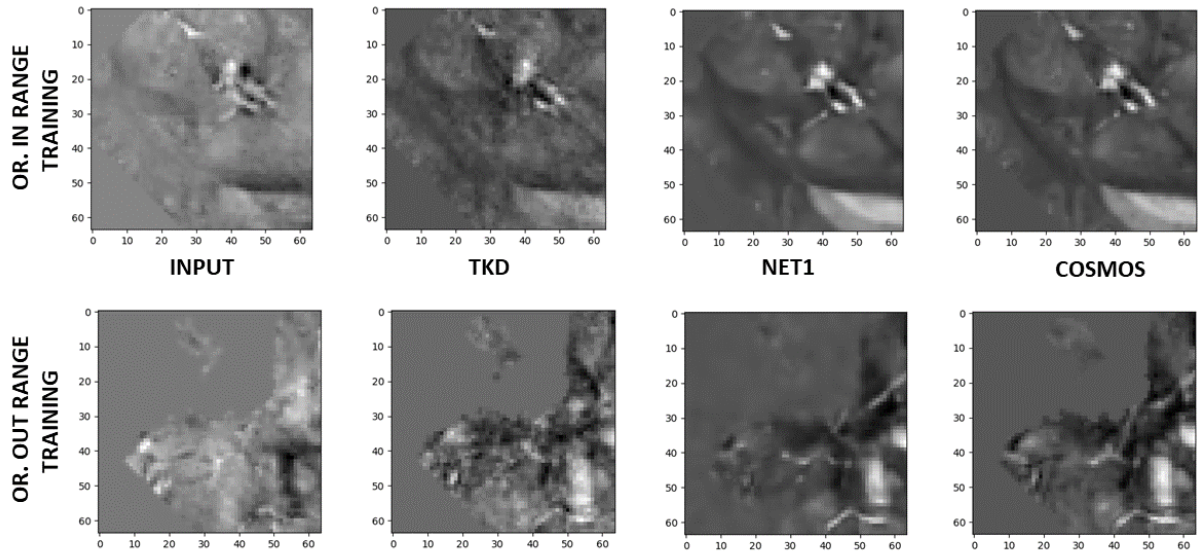


Figure 3.13: Random patch from the test dataset in: ϕ map (left), χ-TKD map (central-left), χ-NET2.0 (central-right) and χ-COSMOS map (right). Training details of NET2.0 in Tab. 3.4. First row: random patch from the ten-orientation, used during the training, test dataset. Secon row: random patch from the two-orientation, unused during the training, dataset

	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
ϕ	0.0179	0.0005	15.4	2.1	0.76	0.09	1.20	0.22
NET2.0	0.0002	0.0001	35.3	1.7	0.97	0.01	0.37	0.13
TKD	0.0012	0.0006	27.5	1.9	0.89	0.04	0.79	0.20
ϕ	0.0066	0.0032	20.1	2.3	0.78	0.10	1.21	0.23
NET2.0	0.0005	0.0001	30.7	1.3	0.97	0.01	0.41	0.12
TKD	0.0013	0.0007	27.2	2.0	0.88	0.04	0.85	0.21

Table 3.5: RSME, pSNR, SSIM and HFEN for multiple-orientation training with NET2.0. Details in Tab. 3.4. First block: random patch from the ten-orientations, used during the training, test dataset. Secon block: random patch from the two-orientations, unused during the training, dataset

Conclusions

- After adding multiple-orientation data, the speed of convergence of the model significantly improves. This is due to the further randomisation of the training dataset, which helps the generalization skill
- Similarity parameters, shown in Tab. 3.5, are better than before (cf. Tab. 3.2).
- NET2.0 was tested using data with different orientations to those used for the training. The model, with the hyperparameters listed in Tab. 3.4, does not behave in the same way with seen and unseen head-orientation data. In an ideal situation, that should not to happen.

3.3.3 Binary Cross Entropy

In literature, there are interesting results about the use of **binary cross entropy** for the loss function, instead of the **mean square error** ([53]), used until now.

Thus, NET2.0 was trained again on the ten-orientation dataset, changing only the loss function and leaving all the other hyperparameters fixed. The details are given in Tab. 3.6.

NET2.0
Dataset size: 8000 patches (64,64), axial slices, single-orientation data
RMSProp optimizer, $\eta = 0.001$
BCE loss function
Batch size: 128
Epochs:100
Training time: 4h, 30 min
Keras 2.2.0, Tensorflow 1.9.0 Backend, Python 3.6
GPU GeForce 940MX - nVdia 384, Cuda 9.0

Table 3.6: NET2.0 supervised training with 10-orientation data

In Tab. 3.7, the metric values along the ten-orientation and the two-orientation datasets are shown. The use of binary cross entropy as loss function let the network behave similarly with the ten-orientation- and the two-orientation- datasets.

	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
ϕ	0.0167	0.005	15.8	2.1	0.76	0.09	1.16	0.22
NET2.0	0.0002	0.0001	34.5	1.5	0.97	0.01	0.35	0.12
TKD	0.0012	0.0006	27.5	1.9	0.89	0.04	0.76	0.18
	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
ϕ	0.0066	0.0032	20.1	2.3	0.78	0.10	1.21	0.23
NET2.0	0.0003	0.0001	33.7	1.5	0.97	0.01	0.37	0.12
TKD	0.0013	0.0007	27.2	2.0	0.88	0.04	0.85	0.21

Table 3.7: RSME, pSNR, SSIM and HFEN for multiple-orientation training with NET2.0. Details in Tab. 3.6. First block: 10 head orientation dataset, used also during the training. Second block: 2 head orientation dataset, completly unseen before.

Conclusions

- The use of BCE as the loss function allows the network to reach equally good results also for orientations different than those used during the training. It aims to improve the model generalization skill.

3.3.4 Batch Normalization and Dropout

The last modification is the addition of **Dropout - D** and **Batch Normalization - BN** layers.

Dropout ([3]) is a regularization technique (Sec. 2.2). It involves ignoring some randomly chosen units during the training. The **dropout fraction** d is set ($d \in (0, 1)$). If the layer

under examination has N neurons, then for each epoch $d \cdot N$ neurons are not examined during the forward or backward pass.

A batch normalization ([76]) layer is often coupled with a dropout one. It allows the activation function to be controlled, keeping the mean value close to 0 and the standard deviation close to 1, to reduce the risk of overfitting and to guarantee a faster convergence. It can be used to reduce the **covariance shift** and allows each layer of the model to learn by itself a little bit more independently of the other layers.
Let $\{\mathbf{x}\}_{i=1}^{N}$ be the input data. $\mathbf{x_i}$ is a n-dimensional array, with $n \geq 1$. Let $\{\mathbf{y}\}_{i=1}^{N}$ be the output data, where, $\mathbf{y}_i = BN_{\gamma,\beta}(\mathbf{x}_i)$. β and γ are parameters which the network has to find during the training, using mean and standard deviation of the input data $\{\mathbf{x}\}_{i=1}^{N}$.

$$\mu = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i, \quad \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mu)^2 \tag{3.24}$$

$$\hat{\mathbf{x}_i} = \frac{\mathbf{x}_i - \mu}{\sqrt{\sigma^2}} \tag{3.25}$$

$$BN(\mathbf{x}_i) = \mathbf{y}_i = \gamma * \hat{\mathbf{x}_i} + \beta \tag{3.26}$$

$\mathbf{x}$ and $\mathbf{y}$ have the same dimensions.
At the end of the training, the final model includes, for each batch normalization layer, a 4D array, (β, *gamma*, μ and σ). The length of this array is the size of the input dataset. For example, if a batch normalization layer follows a convolutional one, it means that the dimension of the returned array is equal to k, number of feature maps of the previous convolutional layer.

D and BN layers are usually added to prevent **overfitting**, improve the **speed of convergence** and to handle the **gradient problem**. The scheme of the model (**NET3**), is reported in Fig. 3.14.

In Tab. 3.8 the training details are shown. The dropout fraction was set to 0.05. The training time does not change too much with respect to the previous training. So, the addition of D and BN layers does not affect significantly the computational effort.
We have verified that dropout and batch normalization layers definitely improve the speed of convergence and the weight updating issue.
Metric values, both for the ten-orientation and the two-orientation datasets, are almost the same as before (Tab. 3.7).

Conclusions

- Dropout and batch normalization layers were added to the model (NET3, Fig. 3.14). They help in fixing the gradient problem and in improving the speed of convergence.

- Their addition does not increase the computational effort as much as the addition of convolutional layers.

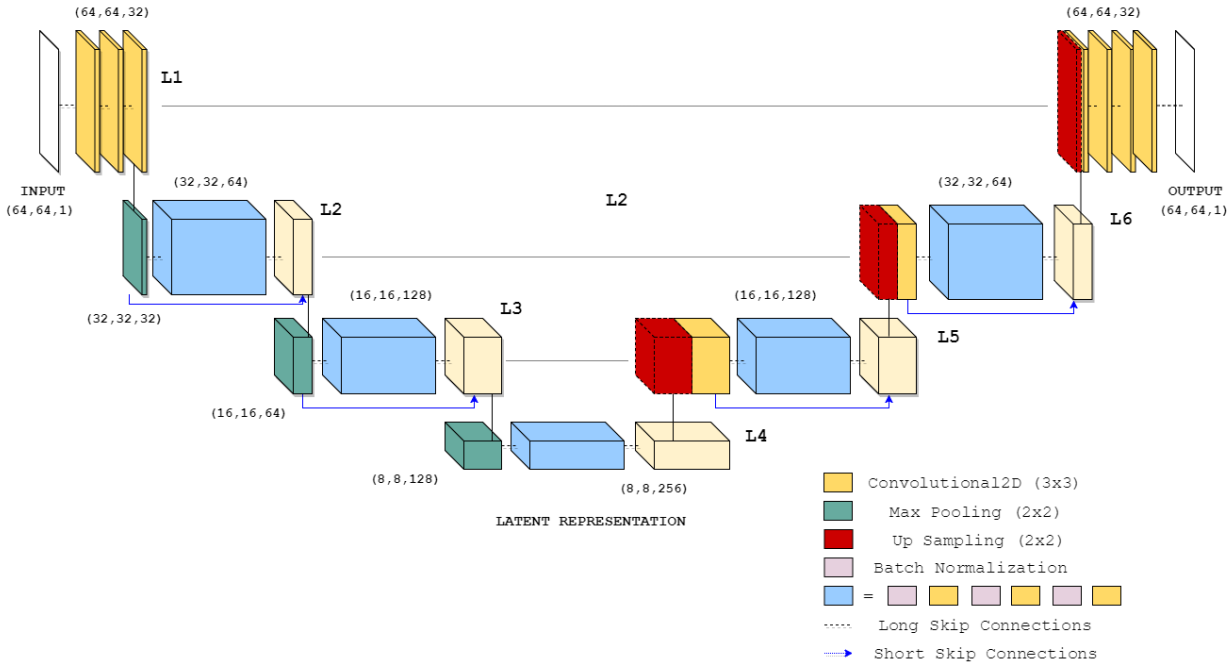


Figure 3.14: NET3: convolutional autoencoder with 4 levels of depth. Input and output have size (64,64,1). Convolutional, pooling (sub- and up- sampling), dropout and batch normalization layers compose the structure. The number of feature maps is from 32 to 256. Long and short skip connections merge feature maps with the same size

NET3
Dataset size: 8000 patches (64,64), axial slices, single-orientation data
RMSProp optimizer, $\eta = 0.001$
BCE loss function
Dropout fraction: 0.05 Batch size: 128
Epochs:100
Training time: 4h, 40 min
Keras 2.2.0, Tensorflow 1.9.0 Backend, Python 3.6
GPU GeForce 940MX - nVdia 384, Cuda 9.0

Table 3.8: NET3 supervised training with 10-orientation data

- The benefts of NET3 were verified. The metric performance substantially did not change, staying almost equal to those listed in Tab. 3.7.

3.3.5 Weight Updates: Batch Size

The batch size is a quantity associated with the number of weight updates that the network makes during the training. If the training dataset size is N and the batch size is set equal to M, in Q epochs the network makes $\frac{N}{M} * Q$ updates (Sec 2.2).
With the 8000-patches training dataset and 128 batch size, the number of updates in 100 epochs is 6250. But, in the literature we found that the number of updates required for this kind of experiment is around 20 or 30 thousand ([70], [71]).
To improve this number, we reduced the batch size from 128 to 32. So now, for a 8000-dataset

and a 100-epoch training, the number of updates is 25000.

3.3.6 Metrics Parameters Errors

The relative errors of the similarity parameters are high in all the described experiments, especially those related to RMSE and HFEN. Looking into Tab. 3.7:

- pSNR and SSIM relative errors are respectively 4% and 1%
- RMSE and HFEN relative errors are respectively 50% and 35%

This means that the quality of the reconstruction significantly changes along the test dataset. This is not only related to our model reconstruction. The same behaviour occurs also in the TKD reconstruction (Fig. 3.15). Similar oscillations may be observed. Some regions of the brain seem to be easier to reconstruct than others.
Note that the performance of NET3 is better then the TKD method for all the parameters along the entire test dataset.

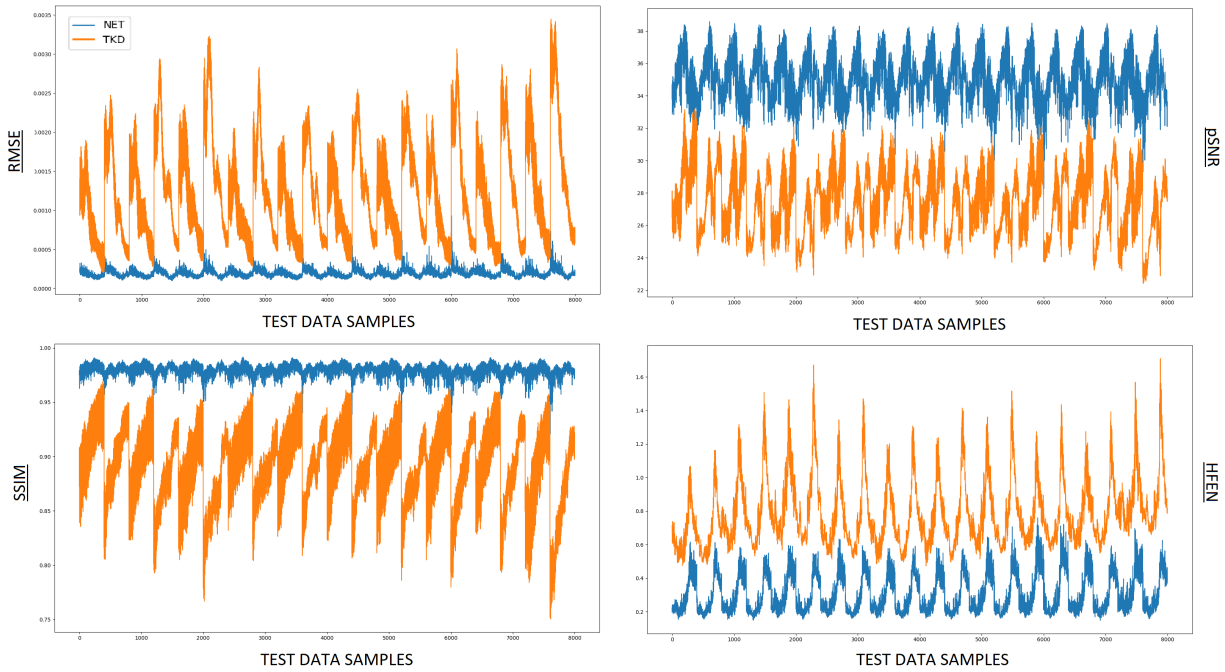


Figure 3.15: Metric parameters shape for NET3 (blue line) and TKD (orange line) reconstruction. NET3 training details in Tab. 3.8

In Fig. 3.16, left panel, the shape of HFEN is shown, zooming on 400 samples. Z_{cut} was set to 50; the first 200 samples are from the first 25 slices, the others from the second 25 slices. For the first 25 slices the HFEN average is ~ 0.2, for the subsequent slices it increases to ~ 0.5. If a smaller range of slices is considered, there are still oscillations, but these are reduced in amplitude (Fig. 3.16, right panel).

In Tab. 3.9 results for both 50-slice (first block) and 25-slice (second block) training are reported. In the second one, the RMSE error, both for the NET3 and the TKD reconstruction, is $\sim 20\%$. The HFEN relative error is respectively 19% and 12%.

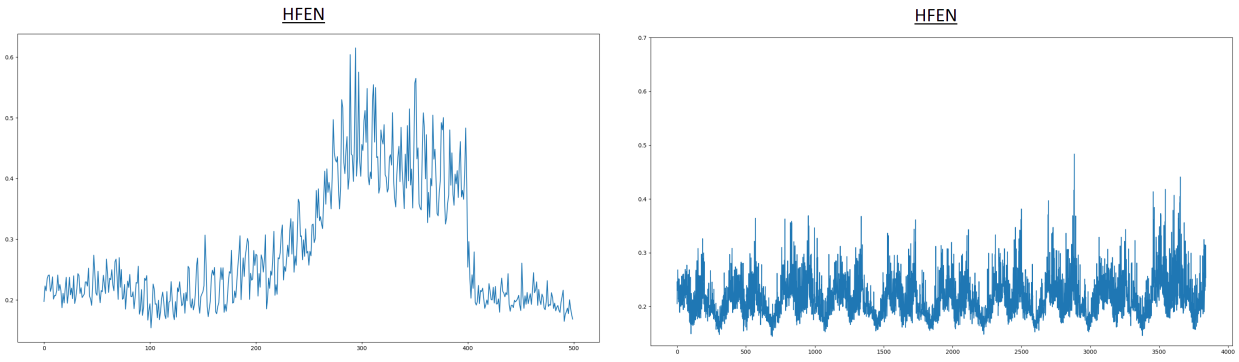


Figure 3.16: Left panel: zoom of the top-left panel in Fig. 3.15 (HFEN). The average value change from the first 25 slices ($\sim$ 0.2) to the subsequent ones ($\sim$ 0.5). Right panel: the HFEN shape of NET3 after a training with only 25 slices

50-slices D.	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
Original	0.016	0.004	16.0	2.1	0.77	0.09	1.17	0.22
Net	0.0002	0.0001	35.0	1.4	0.98	0.01	0.30	0.11
Tkd	0.0012	0.0006	27.6	1.9	0.89	0.04	0.79	0.19
25-slices D.	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
Original	0.015	0.005	17.1	1.8	0.71	0.08	1.01	0.16
Net	0.00037	0.00007	32.9	1.17	0.975	0.007	0.21	0.04
Tkd	0.0018	0.0004	26.4	1.6	0.87	0.03	0.65	0.08

Table 3.9: RMSE, pSNR, SSIM and HFEN for 50-slices (first block) and 25-slice (second block) training

3.4 Conclusions

In this chapter, we have focused on the network algorithm implemented and on the preliminary measurements used to optimize the model, when working with 2D data.

- The original database, from the QSM_{2016} Challenge ([10]), is described (Sec. 3.2.1), with a specific spotlight on the registration operation.

- Similarity (RMSE, pSNR, SSIM and HFEN) and contrast (CNR) parameters are studied (Sec. 3.2.2), to evaluate the performance of the model and to compare its results with other reconstruction methods - e.g. TKD and COSMOS.

- Data augmentation techniques are necessary to produce an enough broad and diversified dataset (3.2.3). In fact, in the medical area it often happens that there are not many available data, and this is a problem for machine learning algorithms.

- Starting from the U-Net structure with long skip connections (NET1, Fig. 3.7), the architecture of the model was modified in order to achieve better perfomance

    1. The need for supervised learning, without an *ad hoc* loss function, was verified.
    2. Already with the use of the simplest structure (NET1), the returned susceptibility map of the model is less noisy and more artifact-free than the TKD one (Sec. 3.3.1).

3. Short skip connections were added to the model (NET2). Then, an extra convolutional layer was added to each block (NET2.0, Fig. 3.11). It was verified that these help in improving the speed of convergence of the network and the weight update issue (Sec. 3.3.2). Also, the model was tested with both single-orientation and multiple-orientation datasets. Its perfomance improves when the training dataset is more varied.
4. The loss function was changed from MSE to BCE (Sec. 3.3) and batch size was changed from 128 to 32.

In Tabs. 3.10 and 3.11 the details of each layer in NET3 (Fig. 3.14) are reported, delivering the kind of layer, the input and the output size: 1) convolutional layers (Conv2D) have $x_{kernel} * y_{kernel} * k_{in} * k_{out}$ weights associated with, where (x_{kernel}, y_{kernel})=(3,3) is the kernel size, k_{in} and k_{out} are the number of the feature maps rispectively in the input and in the output; 2) the number of weights related to each batch normalization layer (BN) is $4 * k$, where k is the number of feature maps, equal in the input and in the output - for each of them 4 parameters have to be updated (β, γ, μ and σ); 3) subsampling (Sub2D) and upsampling (Up2D) layers perform operations independent from any parameters, so there are no weights associated to them; 4) a dropout layer does not influence neither the size of the maps neither the number of feature maps, but it controls the number of samples taken into account during the training - they were not insered.

Block	Layer	Input	Output
1	Conv2D (1)	(64,64,1)	(64,64,32)
	Conv2D (2)	Conv2D(1)	(64,64,32)
	Conv2D (3)	Conv2D(2)	(64,64,32)
2	Sub2D (1)	Conv2D(3)	(32,32,32)
	BN (1)	Sub2D (1)	(32,32,32)
	Conv2D (4)	BN (1)	(32,32,64)
	BN (2)	Conv2D (4)	(32,32,64)
	Conv2D (5)	BN (2)	(32,32,64)
	BN (3)	Conv2D (5)	(32,32,64)
	Conv2D (6)	BN (3)	(32,32,64)
	Conc (1)	Conv2D (6), Sub2D (1)	(32,32,96)
3	Sub2D (2)	Conc (1)	(16,16,96)
	BN (4)	Sub2D (2)	(16,16,96)
	Conv2D (7)	BN (4)	(16,16,128)
	BN (5)	Conv2D (7)	(16,16,128)
	Conv2D (8)	BN (5)	(16,16,128)
	BN (6)	Conv2D (8)	(16,16,128)
	Conv2D (9)	BN (6)	(16,16,128)
	Conc (2)	Conv2D (9), Sub2D (2)	(16,16,224)
4	Sub2D (3)	Conc (2)	(8,8,224)
	BN (7)	Sub2D (3)	(8,8,224)
	Conv2D (10)	BN (7)	(8,8,256)
	BN (8)	Conv2D (10)	(8,8,256)
	Conv2D (11)	BN (8)	(8,8,256)
	BN (9)	Conv2D (11)	(8,8,256)
	Conv2D (12)	BN (9)	(8,8,256)

Table 3.10: Net3 (Fig. 3.14): Encoding path and latent representation

Block	Layer	Input	Output
5	Up2D (1)	Conv2D (12)	(16,16,256)
	Conv2D (13)	Up2D (1)	(16,16,128)
	Conc (3)	Conv2D (13), Conc (2)	(16,16,352)
	BN (10)	Conc (3)	(16,16,352)
	Conv2D (14)	BN (10)	(16,16,128)
	BN (11)	Conv2D (14)	(16,16,128)
	Conv2D (15)	BN (11)	(16,16,128)
	BN (12)	Conv2D (15)	(16,16,128)
	Conv2D (16)	BN (12)	(16,16,128)
	Conc (4)	Conv2D (16), Conc (3)	(16,16,480)
6	Up2D (2)	Conc (4)	(32,32,480)
	Conv2D (17)	Up2D (2)	(32,32,64)
	Conc (5)	Conv2D (17), Conc (1)	(32,32,160)
	BN (13)	Conc (5)	(32,32,160)
	Conv2D (18)	BN (13)	(32,32,64)
	BN (14)	Conv2D (18)	(32,32,64)
	Conv2D (19)	BN (13)	(32,32,64)
	BN (15)	Conv2D (19)	(32,32,64)
	Conv2D (20)	BN (14)	(32,32,64)
	Conc (6)	Conv2D (20), Conc (5)	(32,32,224)
7	Up2D (3)	Conc (6)	(64,64,224)
	Conv2D (21)	Up2D (3)	(64,64,32)
	Conc (7)	Conv2D (21), Conv2D (3)	(64,64,64)
	Conv2D (22)	Conc	(64,64,32)
	Conv2D (23)	Conv2D (22)	(64,64,32)
	Conv2D (24)	Conv2D (23)	(64,64,1)

Table 3.11: Net3 (Fig. 3.14): Decoding path

Chapter 4

2D Quantitative Susceptibility Reconstruction

We are going to use the NET3 structure (Fig. 3.14), introduced in Chapter 3, to reconstruct 2D suceptibility maps. The model consists of a **4-level-depth fully convolutional autoencoder**. Convolutional, sampling (sub- and up-pooling), batch normalization and dropout layers compose the structure. Both long and short skip connections were added to the model, as residual learning tools.
Input and output data are 2D patches, with size (64,64). This is the maximum map size, the minimum is (8,8) in the latent representation. The number of feature maps ranges from 32, in the first level, to 256, in the deepest level. In the table below, a summary of the hyperparameters and details set is given.

RMSProp optimizer, $\eta = 0.001$
BCE loss function
Dropout fraction: 0.05
Batch size: 32
Epochs:100
Keras 2.2.0, Tensorflow 1.9.0 Backend, Python 3.6
GPU GeForce 940MX - nVdia 384, Cuda 9.0

Table 4.1: NET3 (Fig. 3.14) hyperparameters and details summary, as result of the optimization in Chapter 3

Chapter 4 is divided into the following sections:

- Section 4.1: **2D Data Training, Single-Orientation Data, Axial Slices** - NET3 (Fig. 3.14) is trained using (64,64)-patches taken from axial slices. A single-orientation ϕ map is used as input. NET3, TKD and COSMOS performances are compared, by consideration of: similarity parameters (RMSE, pSNR, SSIM, HFEN), intensity,

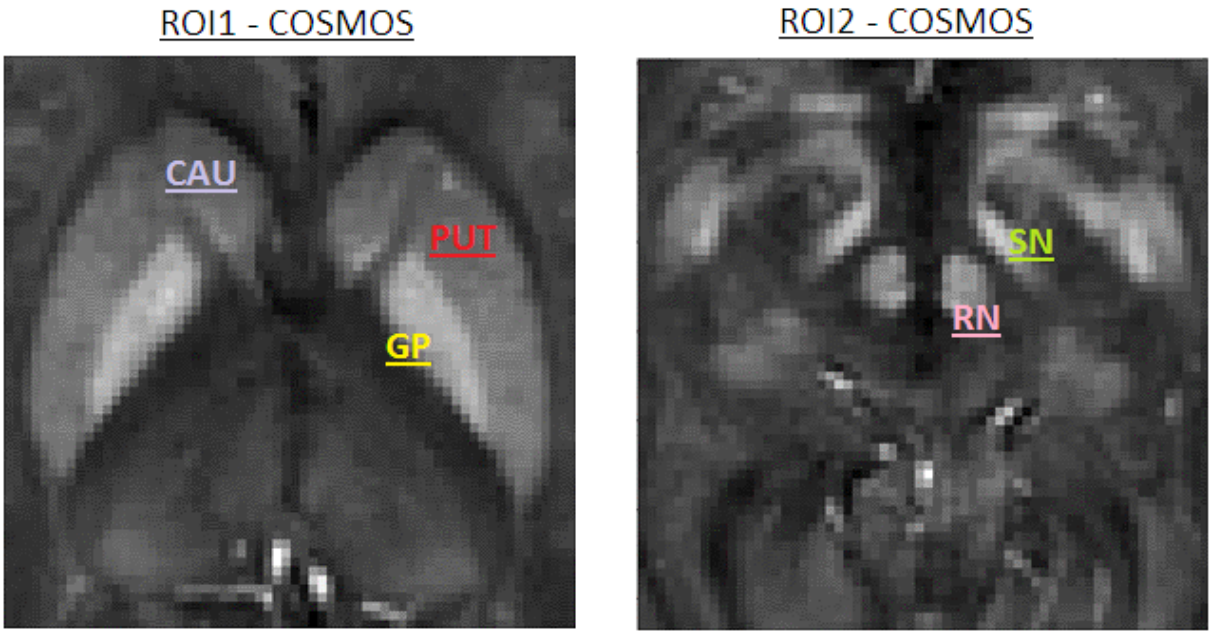


Figure 4.1: Left panel: ROI1 with the caudate (CAU), globus pallidus (GP), putamen (PUT). Right panel: ROI2 with the substantia nigra (SN) and red nucleus (RN). This two patches are from the χ-COSMOS map in [10]

standard deviation and contrast values.
We will focus on the two **Regions of Interest (ROIs)** shown in Fig. 4.1. ROI1, left panel, contains the **caudate** (CAU), **globus pallidus** (GP), **putamen** (PUT). In ROI2, right panel, there are the **substantia nigra** (SN) and **red nucleus** (RN).

- Section 4.2: **2D Data Training, Multiple-Orientation Data, Axial Slices** - NET3 (Fig. 3.14) is trained using (64,64)-patches drawn from axial slices. Multiple-orientation ϕ maps are used as input. NET3, TKD and COSMOS performances are compared, by consideration of: similarity parameters (RMSE, pSNR, SSIM, HFEN), intensity and constrast analysis.

- Section 4.3: **Fast Fourier Transform and K-space** - Each approach provides a 3D susceptibility map. Their appearance in k-space, after 3D Fourier transformation, is studied and compared.

- Section 4.4: **Generalization Skill** - The model perfomance is tested outside of the training range (Sec. 4.4.1), to test its generalization capability. Sagittal, coronal and axial slices are used.

- Section 4.5: **Conclusions**

4.1 2D Data Training, Single-Orientation Data, Axial Slices

The NET3 model (Fig. 3.14) was trained, using two-dimensional patches. They are taken from axial slices in a single-orientation ϕ map (0^{th}-orientation, Fig. 3.6). The magnetic field $\vec{B_0}$, the vertical axis of the frame and the axis of the head are oriented along the same direction in these data. The size of the input and output patches is (64,64). Referring to Sec. 3.2.4: Z_{cut}=10 ($Z \in [65, 75]$), K=30 (angles $\in$ [-20°, +20°]), N=30. The size of the dataset is 9000 (f_{train}=0.8, $f_{val} = f_{test}$=0.1). The other details and hyperparameters are reported in Tab. 4.1. The training stage lasted $\sim$ 5 h, 30 min.

4.1.1 Results: Training and Metrics

The shape of both training and validation loss functions is reported in Fig. 4.2. The validation loss shows more irregularaties during the first twenty epochs than before, but after that there is no atypical behaviour and convergence is easily achieved.

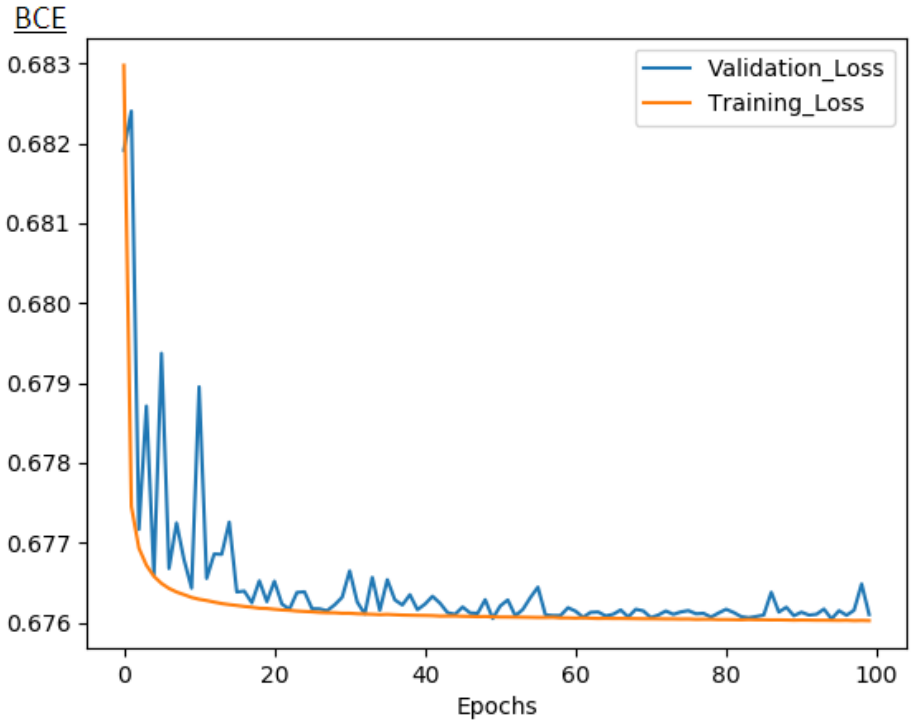


Figure 4.2: Training (orange line) and validation (blue line) loss function. NET3 was trained by setting (Sec. 3.2.4): Z_{cut}=10 (from 65 to 75), K=30 (angles between -20° and +20°), N=30. The size of the dataset is 9000 (f_{train}=0.8, $f_{val} = f_{test}$=0.1). The other details are reported in Tab. 4.1. The training stage lasted $\sim$ 5 h, 30 min

The average values of the similarity parameters (RMSE, pSNR, SSIM and HFEN) over patches drawn from the test dataset were calculated, and they are reported in Tab. 4.2. All the parameters improve (cf. Tab. 3.7), as the number of updates increases. Also, their relative error considerably reduces, respectively to 25 %, 3%, <1% and 27%. This result is

expected, as an effect of the Z_{cut} range reduction (Sec. 3.3.6) - from 50 to 10.
The reported values are not related to single samples. To punctually explore the NET3 reconstruction along the entire test dataset, graphs in Fig. 4.3 were drawn - RMSE, pSNR, SSIM and HFEN respectively in the top-left, top-right, bottom-left and bottom-right panel. The COSMOS map was used as a reference, both for the TKD (orange line) and the NET (blue line) reconstruction. Note that the machine learning approach is better than the TKD one all over.

	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
ϕ	0.025	0.002	14.4	1.0	0.72	0.07	1.10	0.18
NET3	0.00012	0.00003	37.9	1.1	0.986	0.002	0.22	0.06
TKD	0.0015	0.0005	27.0	1.1	0.91	0.02	0.60	0.07

Table 4.2: RMSE, pSNR, SSIM and HFEN. Average values over patches drawn from the test dataset. NET3 was trained by setting (Sec. 3.2.4): Z_{cut}=10 (from 65 to 75), K=30 (angles between -20° and +20°), N=30. The size of the dataset is 9000 (f_{train}=0.8, f_{val} = f_{test}=0.1). The other details are reported in Tab. 4.1. The training stage lasted $\sim$ 5 h, 30 min

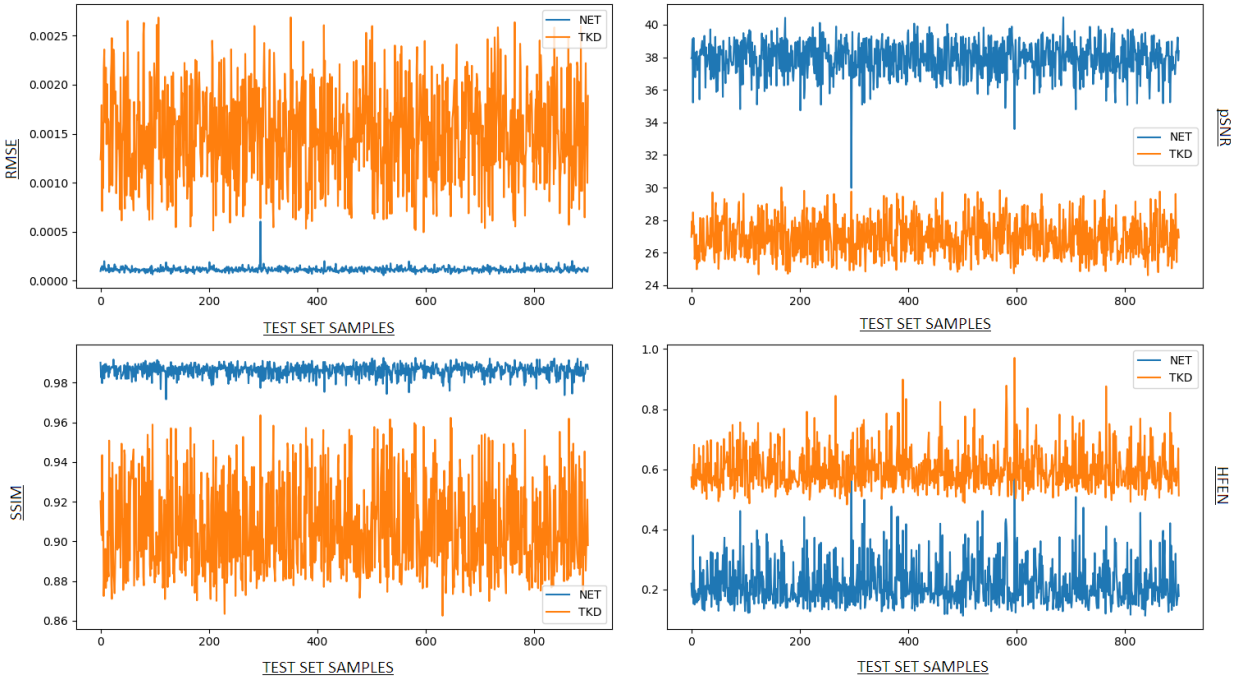


Figure 4.3: RMSE (top-left), pSNR (top-right), SSIM (bottom-left) and HFEN (bottom-right) along the test dataset are reported, both for the NET (blue line) and the TKD (orange line) reconstruction. The COSMOS map was used as reference. NET3 was trained by setting (Sec. 3.2.4): Z_{cut}=10 (from 65 to 75), K=30 (angles between -20° and +20°), N=30. The size of the dataset results to be 9000 (f_{train}=0.8, f_{val} = f_{test}=0.1). The other details are reported in Tab. 4.1. The training stage lasted $\sim$ 5 h, 30 min

4.1.2 Results: ROIs

The ROI1 and ROI2 from the input ϕ map are reported in Fig. 4.4. They are from the 0^{th} head- orientation map (Fig. 3.6), with the magnetic field $\vec{B_0}$, the head axis and the vertical direction of the frame directed in the same way.

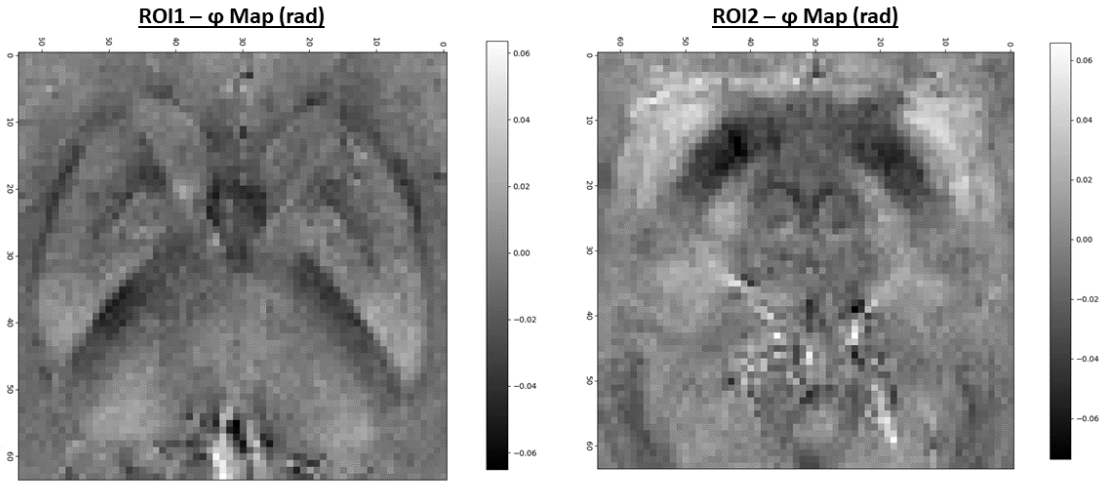


Figure 4.4: ROI1 and ROI2 from the 0^{th} head orientation ϕ map(Fig. 3.6)

- Figs. 4.5 and 4.6 report ROI1 and ROI2 from the three susceptibility maps (TKD, NET3 and COSMOS). The NET3 reconstructions are closer to the ones from the COSMOS method than the patches returned from the TKD approach. Remember that we are comparing single-orientation methods, NET3 and TKD ($\alpha = 0.1$), with a multiple-orientation one, COSMOS, considered as the gold-standard susceptibility reconstruction.

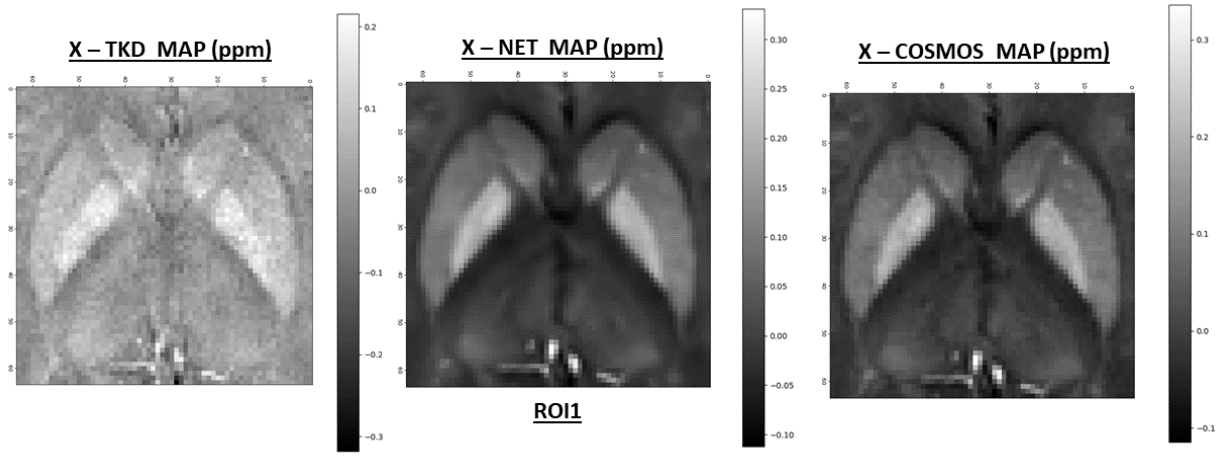


Figure 4.5: ROI1 from susceptibility maps: TKD (left panel, $\alpha = 0.1$), NET3 (central panel) and COSMOS (left panel). NET3 was trained by setting (Sec. 3.2.4): Z_{cut}=10 (from 65 to 75), K=30 (angles between -20° and +20°), N=30. The size of the dataset results to be 9000 (f_{train}=0.8, $f_{val} = f_{test}$=0.1). The other details are reported in Tab. 4.1. The training stage lasted $\sim$ 5 h, 30 min

- To further confirm the better performance of the machine learning approach, we also report the grey-level histograms (Fig. 4.7), both of ROI1 (left panels) and ROI2 (right panels). The COSMOS and NET3 reconstructions match one other very well, while the grey-level distribution from TKD, especially in ROI1, assumes a different shape from the others. It loses some values, especially in the right part of the histogram. This is probably due to the modification of the $D(k)$ expression (Sec. 1.2.6).

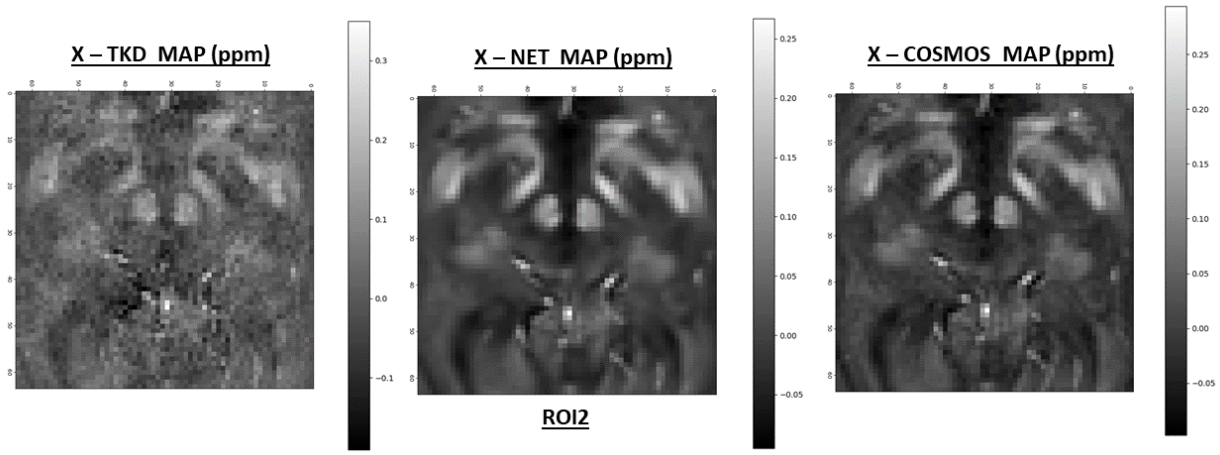


Figure 4.6: ROI2 from susceptibility maps: TKD (left panel, $\alpha = 0.1$), NET3 (central panel) and COSMOS (left panel). NET3 was trained by setting (Sec. 3.2.4): Z_{cut}=10 (from 65 to 75), K=30 (angles between -20° and +20°), N=30. The size of the dataset results to be 9000 (f_{train}=0.8, $f_{val} = f_{test}$=0.1). The other details are reported in Tab. 4.1. The training stage lasted $\sim$ 5 h, 30 min

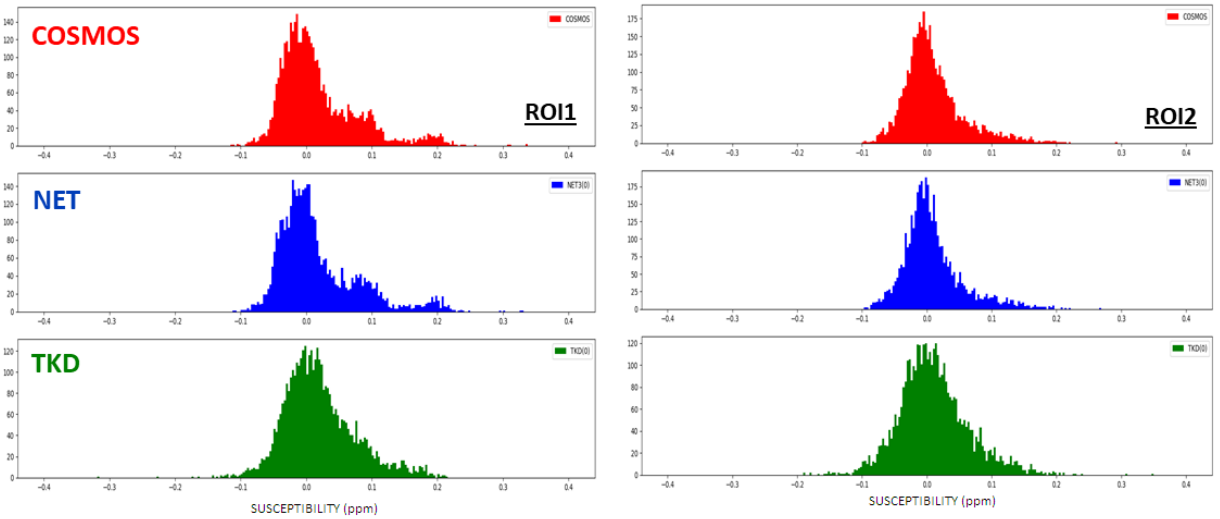


Figure 4.7: Grey-level histograms from COSMOS (red), NET3 (blue) and TKD (green) susceptibility maps. Left panels: ROI1 (Fig. 4.5; right panels: ROI2 (Fig. 4.6)

- Fig. 4.8 includes scatter plots of **NET3-values vs COSMOS-values** and **TKD-values vs COSMOS-values**, in ROI1 (Fig. 4.5) and ROI2 (Fig. 4.6). The TKD values are spread out with respect to the identity function, representing the COSMOS values. Instead, the NET3 outcome shows a good agreement with the gold-standard map, the dispersion is reduced and the entire range of susceptibility values is covered.

4.1.3 Results: Intensity and Contrast Analysis

The image processing sofware **ImageJ** was used to extract mean μ_i and standard deviation σ_i values from PUT, GP, CAU, RN, SN (Fig. 4.1) in the three χ-distributions (TKD, NET3 and COSMOS, Figs. 4.5 and 4.6). The maps were converted to 8-bit type images, with the intensity level range equal to [0,255].

In Tab. 4.3 and in Fig. 4.9, left panel, the results are reported. The $\vec{y}$ axis of the graph is

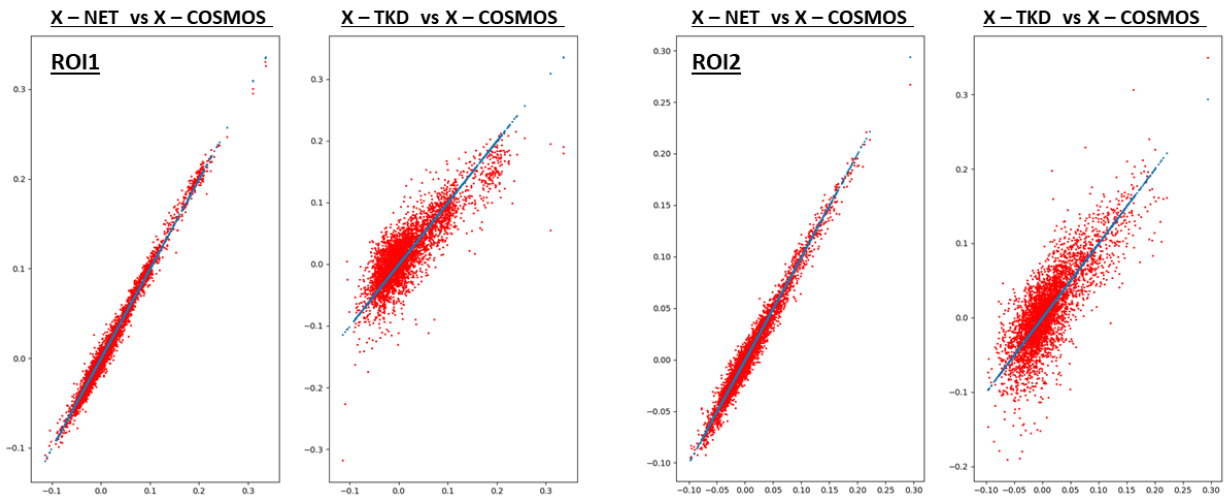


Figure 4.8: Scatter plots: χ-NET-values *vs* χ-COSMOS-values and χ-TKD-values *vs* χ-COSMOS-values. Left panles: ROI1 (Fig. 4.5); right panels: ROI2 (Fig. 4.6)

in arbitrary units, representing the intensity levels.
Also the values related to the background were extracted, let them be B(1) and B(2). B(1) is the background of the ROI1, and it is mainly composed of **general white matter**. B(2) instead is related to the ROI2 and it is composed of **white matter tracts**.
There is a good agreement between the NET3 and the COSMOS reconstructions in all the areas examined. However, the TKD one is not consistent with them in almost all regions considered. Note also that, except for the SN, the σ_i values for NET3 and COSMOS are lower, while the TKD reconstruction is noisier. This is one of the main drawbacks in using simple numerical approaches to fix the ill-posed QSM problem.

	$(a.u.)$	PUT	GP	CAU	B(1)	RN	SN	B(2)
COSMOS	μ	102.4	121.9	99.4	82.6	107.6	128.9	66.6
	σ	1.9	3.3	1.8	0.9	1.4	9.8	3.5
NET3 (0)	μ	102.6	123.6	99.3	81.3	109.6	134.0	63.5
	σ	1.6	2.3	1.7	1.4	1.3	11.7	1.8
TKD (0)	μ	125.8	150.7	122.8	106.8	110.8	111.36	97.6
	σ	7.4	8.6	5.4	0.9	2.2	4.5	2.4

Table 4.3: Mean intensity values in COSMOS, NET3 and TKD reconstructions, from ROI1 and ROI2 in Figs. 4.5 and 4.6. For NET3 and TKD, single-orientation data are taken into account (0^{th} head direction). The corrispondent plot is in Fig. 4.9, left panel

Mean and standard deviation values were used to evaluate the contrast of the regions with respect to the background and its error through the propagation of uncertainty. Results are reported in Tab. 4.4 and in Fig. 4.9, right panel. The contrast between the globus pallidus and putamen (GP-PUT) and the caudate and putamen (CAU-PUT) was also calculated.
There is agreement in the GP-PUT and CAU-PUT points. Elsewhere, there is not. In the PUT-B, GP-B and CAU-B points, the NET3 model works slightly less well than the TKD

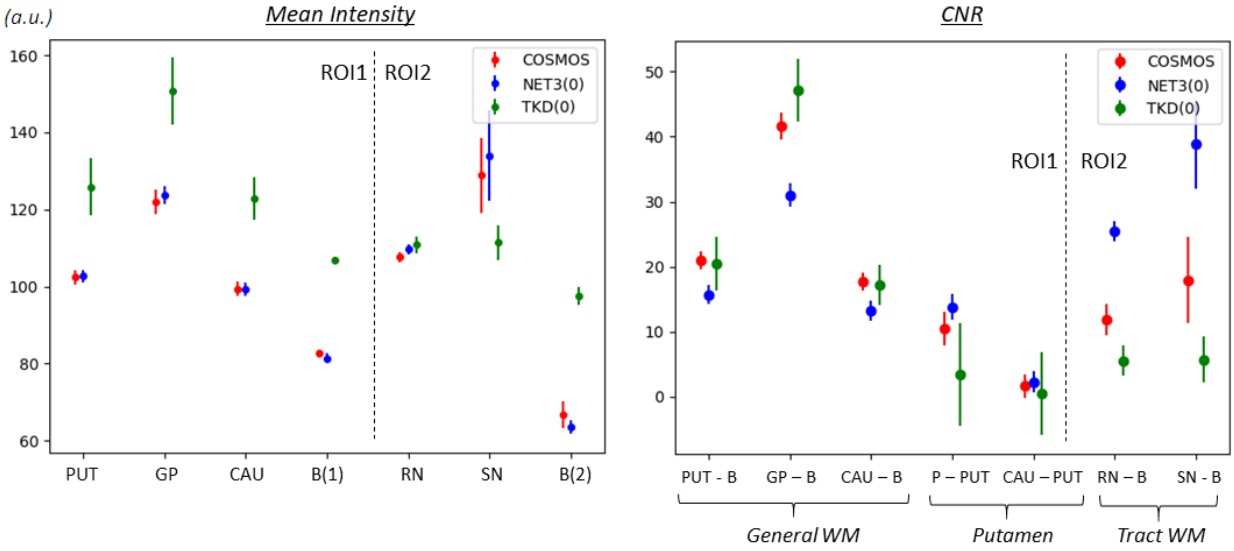


Figure 4.9: Left panel: mean intensity values in COSMOS, NET3 and TKD reconstructions, from ROI1 and ROI2 in Figs. 4.5 and 4.6. For NET3 and TKD, single-orientation data are taken into account. Data values are in Tab. 4.3. Right panel: CNR values in COSMOS, NET3 and TKD reconstruction, from ROI1 and ROI2 in Figs. 4.5 and 4.6. For NET3 and TKD, single-orientation data are taken into account. Data values are in Tab. 4.4

and COSMOS techniques. They correspond to comparisons involving the general-WM-background areas. In the RN-B and SN-B points, in which the background is composed of white matter, the machine learning approach provides significantly better results. The discrepancy in WM-background-related values may be due to the fact that COSMOS models the χ susceptibility as a scalar property, and the NET3 tool learns from it. But susceptiblity shows also anisotropic behaviour in white matter (Sec. 1.1.3).

Despite this, the recognition ability of the NET3 is never compromised. In fact, the regions in which our model works worse than the others are high-contrast areas. Look at the scatter plots in Fig. 4.10, in which there are CNR- NET3 and TKD values *vs* CNR-COSMOS values. The discordant points are all over the level CNR=10.

		PUT-B	GP-B	CAU-B	GP-PUT	CAU-PUT	RN-B	SN-B
COSMOS	CNR	21.0	41.6	17.8	10.4	1.6	11.8	17.9
	ϵ	1.4	2.1	1.4	2.6	1.8	2.4	6.6
NET3 (0)	CNR	15.7	31.0	13.2	13.8	2.2	25.4	38.8
	ϵ	1.4	1.9	1.5	1.9	1.6	1.5	6.8
TKD (0)	CNR	20.4	47.1	17.2	3.4	0.4	5.5	5.7
	ϵ	4.1	4.8	3.2	8.0	6.4	2.3	3.5

Table 4.4: CNR values in COSMOS, NET3 and TKD reconstruction, from ROI1 and ROI2 in Figs. 4.5 and 4.6. For the NET3 and TKD methods, single-orientation data are taken into account (0^{th} head direction). The correspondent plot is in Fig. 4.9, right panel

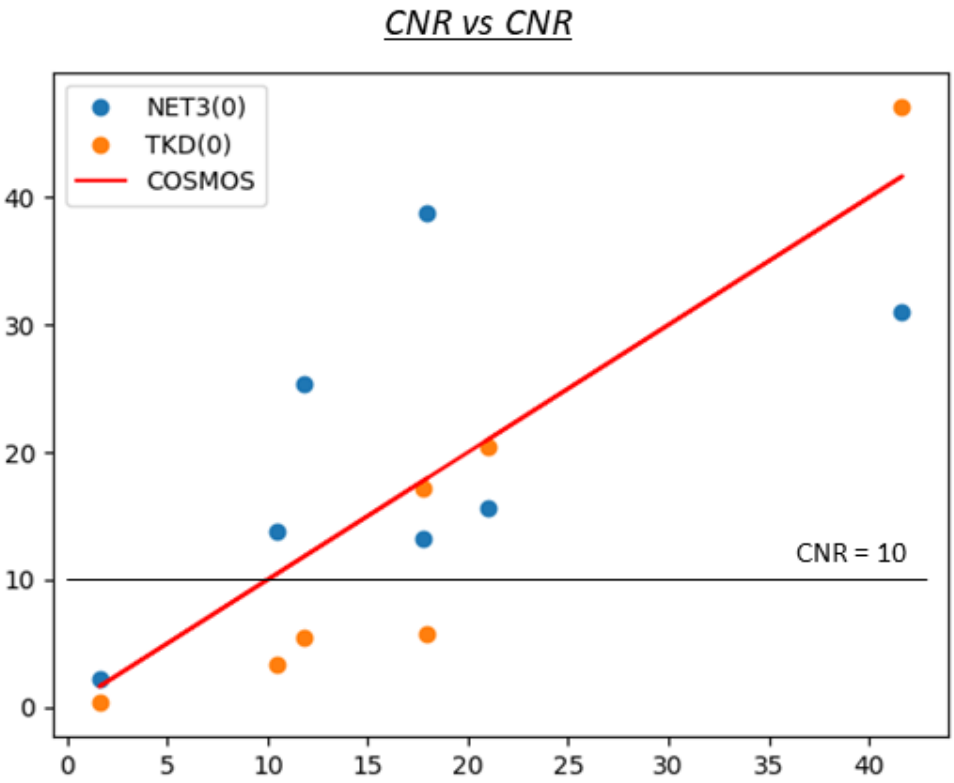


Figure 4.10: NET3-, TKD- and COSMOS CNR values *vs* COSMOS CNR values (Tab. 4.4). Single-orientation data are used for the NET3 and TKD methods

4.1.4 Conclusions

- The NET3 model (Fig. 3.14) was trained using (64,64)-patches from axial slices of a ϕ map. Single-orientation data were used as input (0^{th} head orientation). Both the training and validation loss functions converge (Fig. 4.2), and the similarity parameters confirm a good behaviour of the model (Tab. 4.2, Fig. 4.3). The results of the NET3 reconstruction is closer to the COSMOS than the TKD one.

- The ROI1 and ROI2 (Fig. 4.1) were analyzed, comparing the three techniques. The ϕ input data are shown in Fig. 4.4, in Figs. 4.5 and 4.6 there are the susceptibility maps. We have also considered the grey-level histograms (Fig. 4.7) and the χ-values-scatter plots (Fig. 4.8). The trained model always shows better agreement with the COSMOS than the TKD method.

- Mean intensity values were extracted from: the putamen, globus pallidus, caudate (ROI1) and red nucleus, substantia nigra (ROI2) areas. Also, the intensity of the background was measured. ROI1 (B1), it is composed of general WM, while in ROI2 (B2) it consists of WM tracts. Standard deviation values were used as error.
  Results are reported in Fig. 4.9, left panel, and in Tab. 4.3. Single-orientation data were used for the NET3 and the TKD reconstruction. Our model and the COSMOS one are consistent each other in all the examined points. The TKD values are far from the other reconstructions in almost all the areas. Also, the TKD map is, as expected,

noisier than the others.

- The contrast analysis was also carried out. Results are shown in Fig. 4.9, right panel, and in Tab. 4.4. The NET3 model is not always consistent with the COSMOS reconstruction; it is better in some points (general white matter as background) and worse in others (white matter tracts as background). This may be due to the modelling of the susceptibility as a scalar property of matter. However, the model shows good performance also evaluating the contrast: in fact, its reconstruction has never prevented the recognition of any area, because the areas of disagreement are in a high-contrast regions (CNR>10). It works better than the other single-orientation method.

4.2 2D Data Training, Multiple-Orientation Data, Axial Slices

The model we are working on is aimed to equally react regardless of the direction of the input ϕ map. Conceptually, there are some issues related to this goal. The ϕ map is actually sensitive with respect to the direction of the static field $\vec{B}_0$, and in the presence of anisotropy so it is the susceptibility. Mathematically, there is not a one-to-one orientation-independent correspondence between the measured field shift ΔB - from which the ϕ information is derived - and χ.
Remember also that we are performing a single-orientation method. Multiple-orientation data are using only during the training stage. Then, for the proper reconstruction, a single ϕ map is used as input to obtain a single susceptibility map. In contrast, multiple-orientation methods - e.g. the COSMOS and STI techniques (Sec. 1.2.5) - have more than one ϕ map as input and they return a single susceptibility map.

The NET3 model was trained again, to explore its response to changing the head direction of the input data. This time, slices from multiple-orientation ϕ maps were used. We are still working with two dimensional and axial slices.
Input patches have, as before, size equal to (64,64). Referring to Sec. 3.2.4: D=5, Z_{cut}=10, ($Z \in [65, 75]$), K=15 (angles $\in$ [-20° to +20°]), N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test} = 0.1$. The other details are reported in Tab. 4.1. The total training time was $\sim$ 6h, 50 min.

4.2.1 Results: Training and Metrics

The variation of the training and validation loss functions is shown in Fig. 4.11. Their relative shape is different than before (cf. Fig. 4.2): both of them achieve convergence, but the training loss is slightly higher than the validation one. We expected the opposite: the validation loss has to be a little higher than the training one, because the training data are not used for the weight updates. It makes sense that the model performance evaluated on

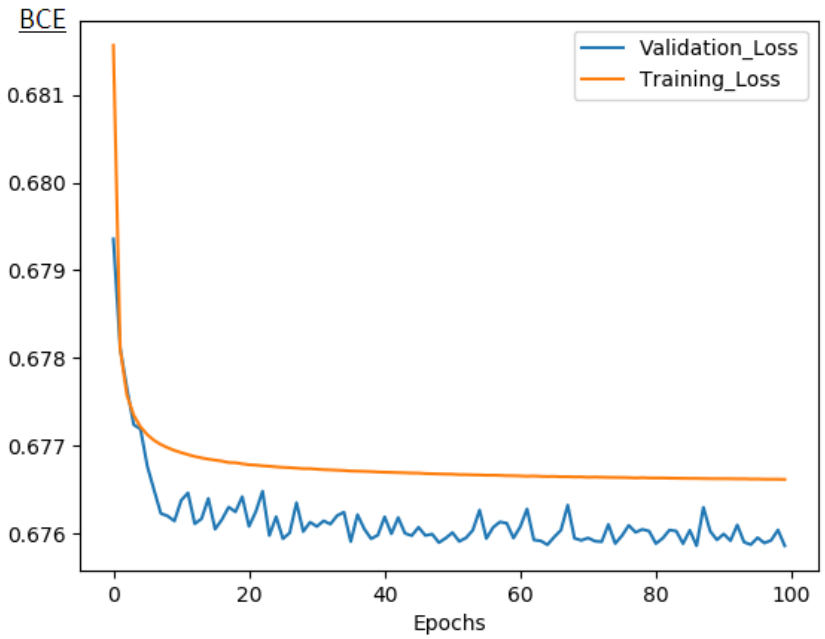


Figure 4.11: Training (orange line) and validation (blue line) loss functions. NET3 was trained by setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was ∼ 6h, 50 min

the validation dataset is worse. When it is not like this, as in this situation, particular care has to be taken during the rest of the analysis.

The similarity parameters are reported in Tab. 4.5 and in Fig. 4.12. In the table, metric mean values along the test dataset are shown. In the graphs, the outline of the parameters along the test dataset figures. Results related to RMSE, pSNR, SSIM and HFEN could be found respectively in the top-left, top-right, bottom-left and bottom-right panel. The patches from COSMOS were used as a reference, both for the TKD (orange line) and the NET3 (blue line) reconstructions.
The results are almost the same as before (cf. Tab. 4.2). The NET3 perfomance, considering both the average and the individual values, is better than the TKD one for all the metric parameters along the entire dataset.

	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
ϕ	0.020	0.002	15.5	1.0	0.73	0.07	1.09	0.15
NET3	0.00012	0.00003	37.7	1.2	0.986	0.003	0.22	0.06
TKD	0.0021	0.0007	25.4	1.1	0.88	0.03	0.71	0.12

Table 4.5: RMSE, pSNR, SSIM and HFEN. Average values along the test dataset. NET3 was trained by setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was ∼ 6h, 50 min

4.2.2 Results: 3D χ Reconstruction from 2D Trained Model

A 2D training allows a three dimensional susceptibility reconstruction to be obtained. Following the scheme in Fig. 4.13:

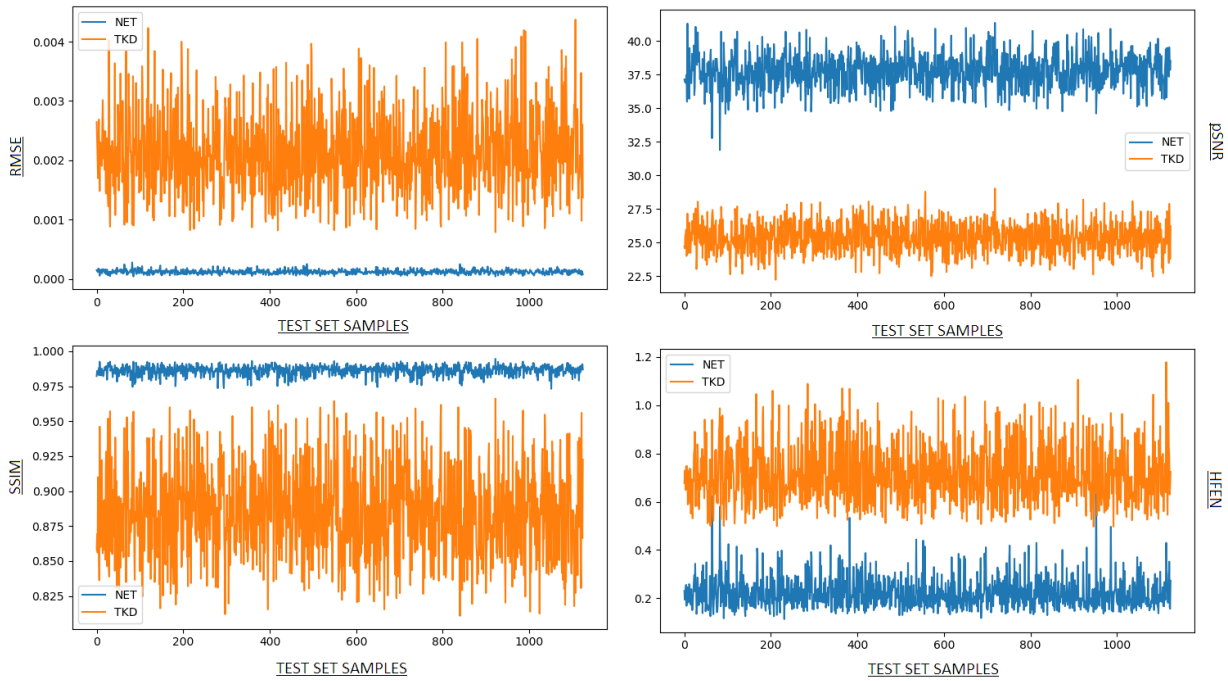


Figure 4.12: RMSE (top-left), pSNR (top-right), SSIM (bottom-left) and HFEN (bottom-right) along the test dataset are reported, both for the NET (blue line) and the TKD (orange line) reconstructions. The COSMOS map was used as a reference. NET3 was trained by setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was $\sim$ 6h, 50 min

- 2D axial slices are extracted from the 3D ϕ map.

- N patches are extracted from each slice, and they are processed by the trained model. It was trained with (64,64)-patches, so the input data should have the same size - at least, they could be smaller.
  N is not necessary fixed. The set of the patches has to cover the 2D map, that means that N has to be sufficiently large to make the patches enclose the entire map. With (x_{tot}, y_{tot})=(160,160) and (x_p, y_p)=(64,64), N_{min} is equal to 9:
  $$N_{min} = int\left(\frac{x_{tot}}{x_p} + 0.5\right) \cdot int\left(\frac{y_{tot}}{y_p} + 0.5\right) \tag{4.1}$$
  Also the choice of the patch coordinates is not fixed, since the just-named condition is respected.

- The output patches are pieced together in order to obtain a 2D susceptibility map. A **re-patching** operation is required. Its formalization clearly depends on the number and position of the patches extracted from the 2D ϕ map.

- The 3D χ map is obtained putting together all the 2D axial susceptibility maps.

- From the 3D reconstruction, different orientation slices - axial, sagittal and coronal - may potentially be extracted and observed.

Ten axial slices were used to train NET3 in the just presented experiment ($Z \in [65, 75]$). Initially, a 3D map with size (160,160,10) was built. We are going to refer to it as the

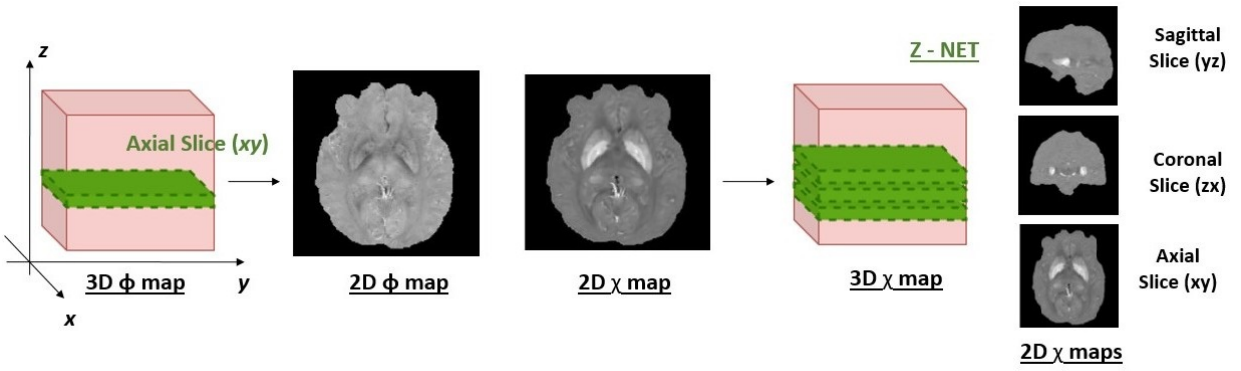


Figure 4.13: Scheme of a three dimensional χ map reconstruction from a two dimensional axial slices processing

Z-NET.
From it, the axial slices containing ROI1 and ROI2, which are inside the training range - $Z = 65$ and $Z = 75$ respectively -, were extracted and compared with the other reconstruction techniques. They are reported in Fig. 4.14. The SLICE1, containing the first region of interest, is in the first row, the SLICE2 in the second one. The quality and the accuracy of the Z-NET maps are better than the TKD method. They are less noisy and closer to the reference maps.

The NET3 performance was also tested when changing the direction of the input data. Z-NET$_i$ maps - $i = 0, 1, 2, 3, 4$, one for each direction used during the training - were built. As an example, the SLICE1 is reported in Fig. 4.15. There are four images, which correspond to $i = 1, 2, 3, 4$. The $i = 0^{th}$ is in Fig. 4.14. In according to a qualitative analysis, they look very similar each other.

Results: ROIs

ROI1 and ROI2 were analyzed for all the examined head-orientation maps. As an example, only the results related to the 0^{th} direction are reported.
The outcomes are summarized in Figs. 4.16 and 4.17, respectively associated with the first region and the second region of interest. The figures show: the proper reconstruction (panel a), the grey level histograms (panel b) and the χ-value scatter plots (panel c).
The Z-NET reconstructions are close to the gold-standard COSMOS, observing both the maps and the grey-level histograms. The result of the TKD method more noisy than the reference and the susceptibility map assumes values less consistent with the other reconstructions.

Results: Intensity and Contrast Analysis

- **Single-Orientation Data: NET3 and TKD**
  First, we introduce the results obtained using single-orientation data as input, both for the NET3 and the TKD reconstructions. We have expected to confirm the outcome of the previous section, even if for this experiment multiple- and not single-orientation

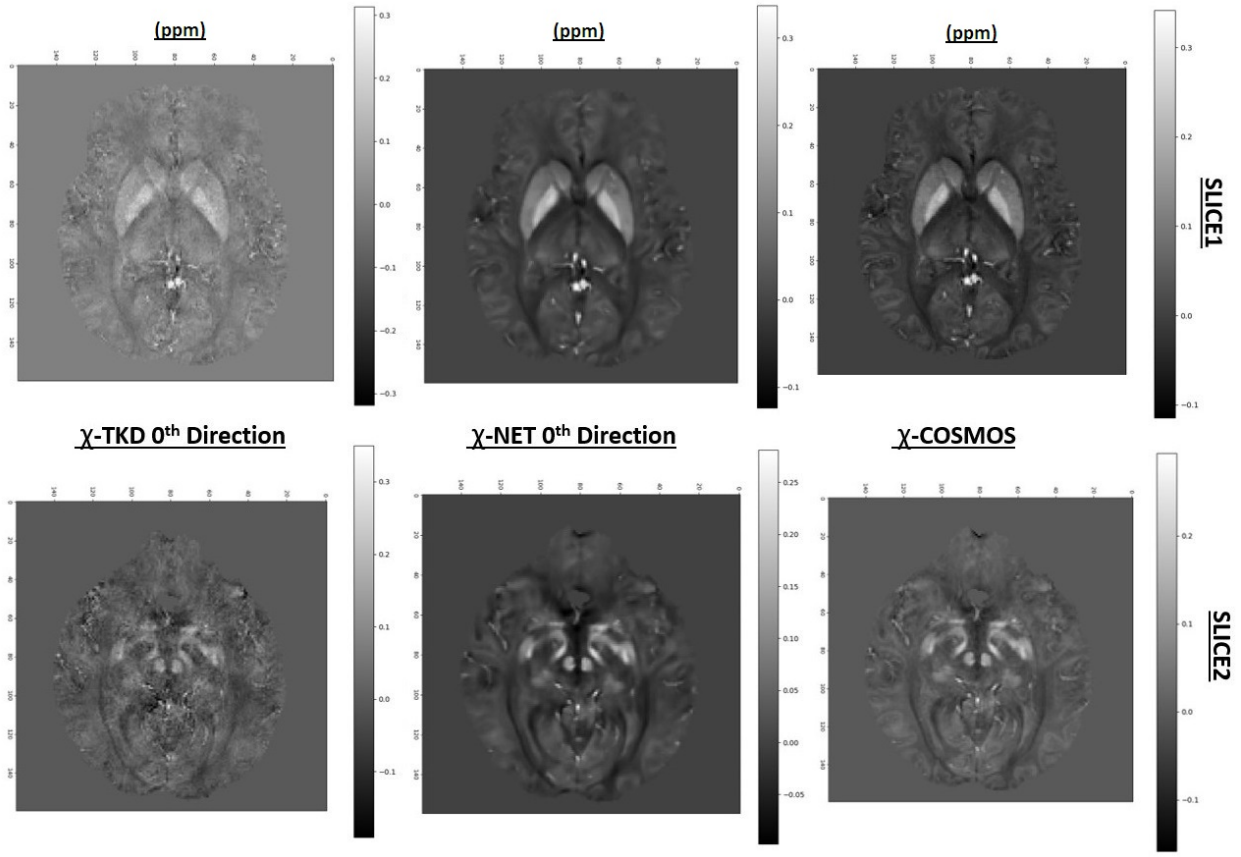


Figure 4.14: Susceptibility maps from the TKD (left panels), NET-Z (central panels) (Fig. 4.13) and COSMOS (right panels) maps. First row: SLICE1, which contains ROI1 (Fig. 4.1, left panel). Second row: SLICE2, which contains ROI2 (Fig. 4.1, right panel). NET-Z was obtained with the NET3 model. NET3 was trained by setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was $\sim$ 6h, 50 min

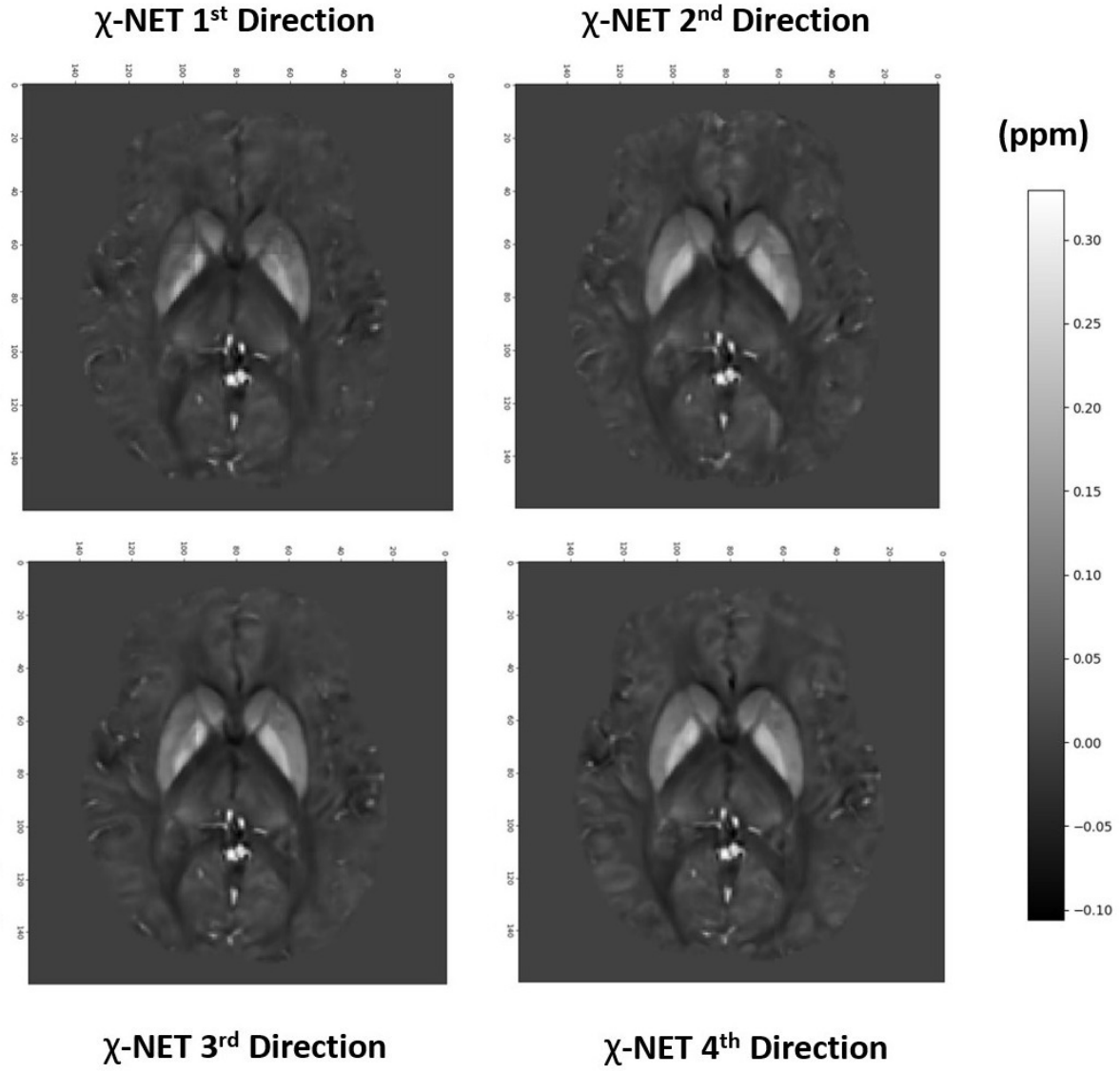


Figure 4.15: Z-NET susceptibility maps, obtained using different orientation data. Z-NETs were gathered with the NET3 model. NET3 was trained by setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was $\sim$ 6h, 50 min

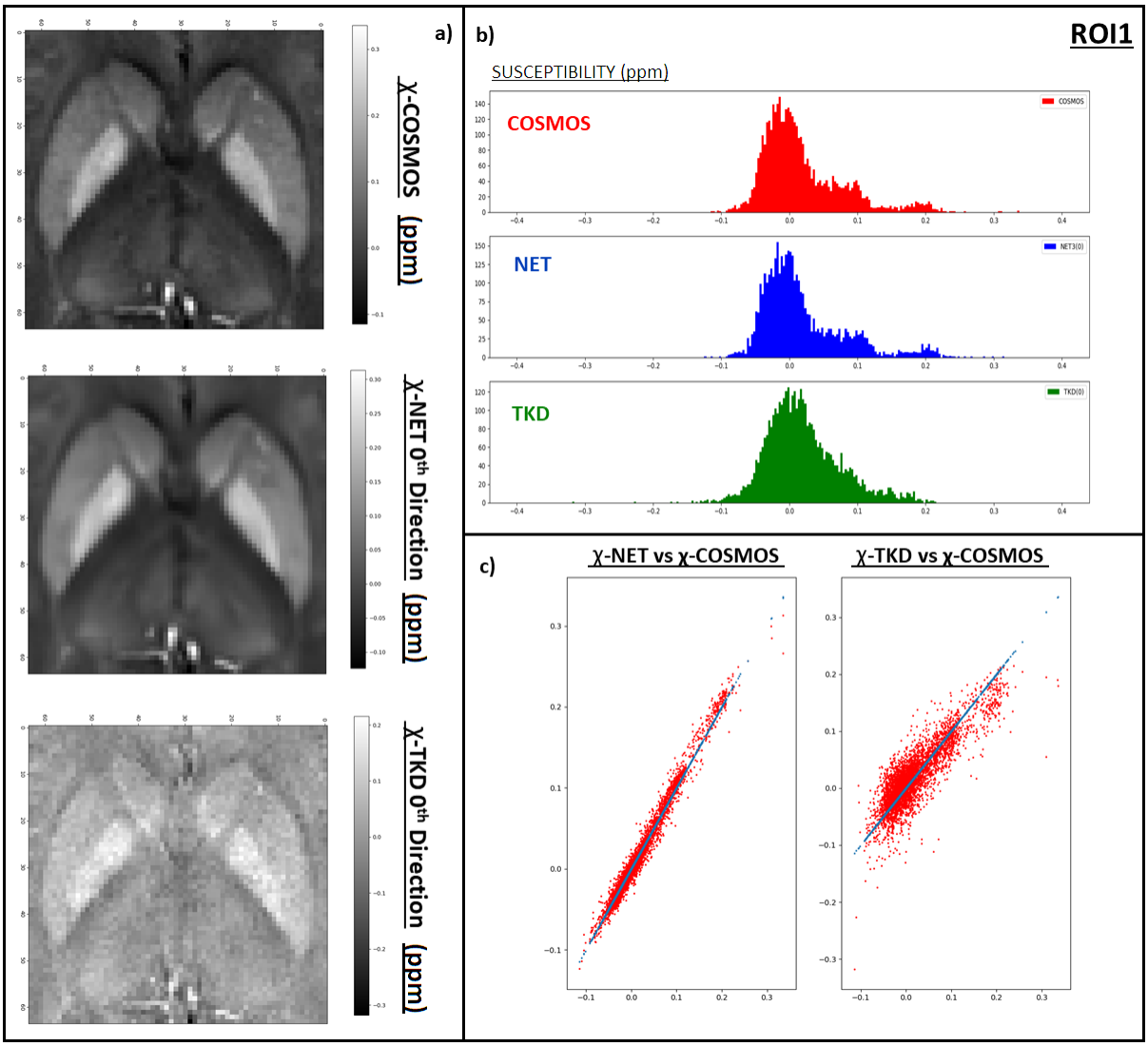


Figure 4.16: ROI1 analysis. a) patch from COSMOS (top), Z-NET (central) and TKD (bottom) map. b) grey-level histograms from COSMOS (red), Z-NET (blue) and TKD (green). c) scatter plots: Z-NET- *vs* COSMOS-values (left panel) and TKD- *vs* COSMOS-values (right panel). Z-NET was obtained with the NET3 model. NET3 and TKD used the 0^{th} input data. NET3 was trained by setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was ∼ 6h, 50 min

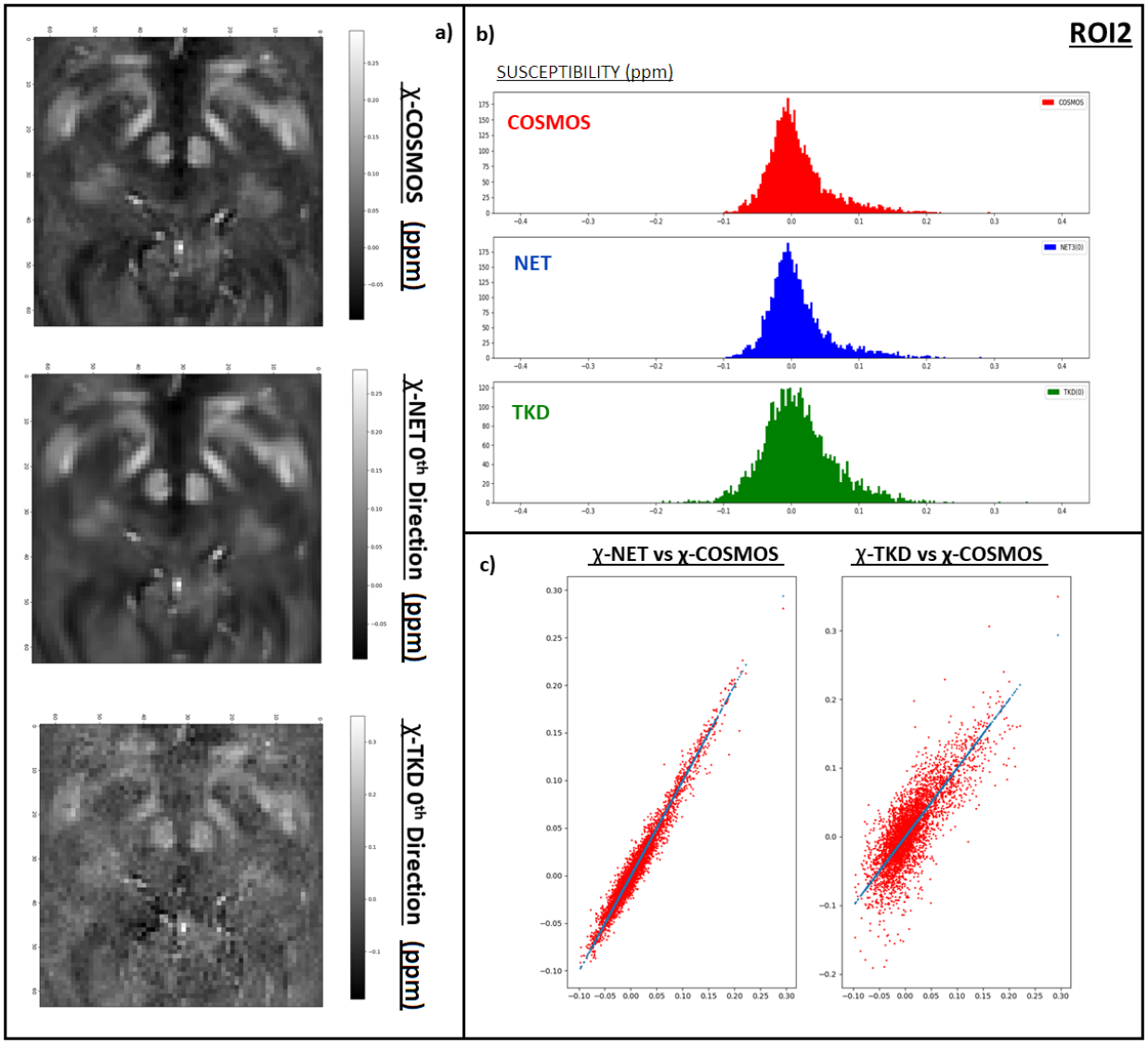


Figure 4.17: ROI2 analysis. a) patch from COSMOS (top), Z-NET (central) and TKD (bottom) map. b) grey-level histograms from COSMOS (red), Z-NET (blue) and TKD (green). c) scatter plots: Z-NET- *vs* COSMOS-values (left panel) and TKD- *vs* COSMOS-values (right panel). Z-NET was obtained with the NET3 model. NET3 and TKD used the 0^{th} input data. NET3 was trained by setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was $\sim$ 6h, 50 min

data were used for the training stage. Mean, standard deviation and CNR values were extracted and evaluated in the putamen, globus pallidus, caudate, red nucleus, substantia nigra and in the WM-composed background (Fig. 4.1).
In Fig. 4.18, left panel, the mean intensity values are reported; the correspondent values are reported in Tab. 4.6. The standard deviation was used as error. The software ImageJ was used for this part, and the intensity values reported are in arbitrary units. The dynamic range is [0,255].
The values from the NET3 and COSMOS methods are consistent with one another, except for the GP value. However, also in that point, the two reconstructions do not show significant discrepancies ($\sim 2\ \sigma_{COSMOS}$, $\sim 2.5\ \sigma_{NET3}$). The TKD values are instead inconsistent with respect to the others, in almost all the points. Notice also that, especially in GP and B(2), the error is substantially higher. Remember that B(2) is the background in the ROI2, and it is mainly composed of white matter tracts.

	$(a.u.)$	PUT	GP	CAU	B(1)	RN	SN	B(2)
COSMOS	μ	106.4	176.4	95.33	39.1	147.0	195.9	19.6
	σ	8.7	5.7	8.4	2.7	9.8	10.3	11.9
Z-NET (0)	μ	114.4	189.7	109.0	41.4	157.8	196.0	9.8
	σ	7.8	5.1	8.5	2.1	8.0	14.7	4.6
TKD (0)	μ	186.5	228.0	179.0	132.8	152.4	156.5	64.5
	σ	9.7	13.7	10.5	6.8	10.3	11.7	17

Table 4.6: Mean intensity values in COSMOS, Z-NET and TKD reconstructions, from ROI1 and ROI2 (Figs. 4.16 and 4.17). For the Z-NET and TKD, single-orientation data are taken into account as input. Z-NET was obtained using the model NET3. It was trained with multiple-orientation data. The correspondent plot is Fig. 4.18, left panel

Mean and standard deviation values were used to implement the contrast analysis. Results are reported in Fig. 4.18, right panel, and in Tab. 4.7. COSMOS and Z-NET values are consistent each other for all the areas considered in ROI1. In ROI2 they are not. Z-NET shows a higher contrast than the gold-standard technique. The TKD values are lower than the other two reconstructions almost everywhere. They are consistent only in the low-constrast area (GP-PUT and CAU-PUT).
The same values are reported in scatter-plot form, in Fig. 4.21, left panel. Notice that the NET3 values (blue points) are almost everywhere above the red line, which represents the CNR-COSMOS values. In contrast, the TKD results are under the level CNR=10, except for the last point on the right (CNR > 60).

- **Multiple-orientation Data: NET3**

Consider now the response of our model when multiple-orientation data are used as input.
The mean intensity values are reported in Fig. 4.19 (left panel) and in Tab. 4.8. The maps NET(i)=NET-Z$_i$, with $i = 0, 1, 2, 3, 4$, correspond to the maps obtained with the

		PUT-B	GP-B	CAU-B	GP-PUT	CAU-PUT	RN-B	SN-B
COSMOS	CNR	31.1	63.6	26.1	8.0	1.2	10.8	14.9
	ϵ	5.4	3.9	5.3	7.2	8.6	10.8	11.1
Z-NET (0)	CNR	34. 6	70.3	32.0	9.7	0.7	32.2	40.5
	ϵ	5.0	3.6	5.3	6.4	8.2	6.3	10.0
TKD (0)	CNR	7.9	14.0	6.8	4.3	0.8	5.2	5.4
	ϵ	8.3	10.3	8.6	11.7	10.1	13.7	14.3

Table 4.7: CNR values in COSMOS, Z-NET and TKD reconstructions, the from ROI1 and ROI2 (Figs. 4.16 and 4.17). For the Z-NET and TKD, single-orientation data are taken into account as input. Z-NET was obtained using the model NET3. It was trained with multiple-orientation data. The correspondent plot is Fig. 4.18, right panel

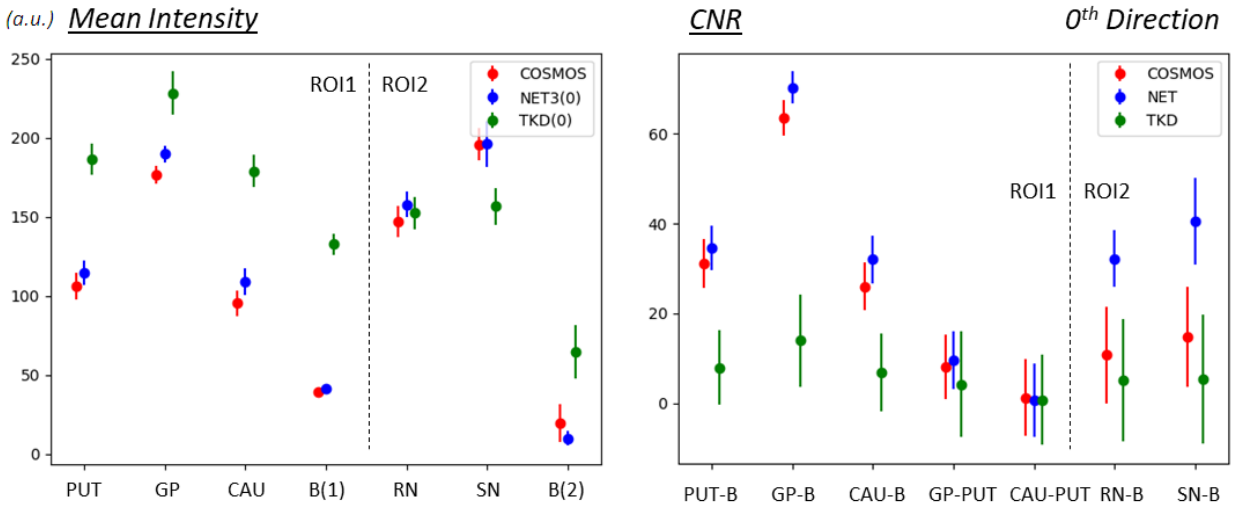


Figure 4.18: Left panel: mean intensity values in COSMOS, Z-NET and TKD reconstruction, from ROI1 and ROI2 in Figs. 4.16 and 4.17. Right panel: CNR values in COSMOS, Z-NET and TKD reconstruction, from ROI1 and ROI2. For Z-NET and TKD, single-orientation data are taken into account

five-orientation data used to train the model. There is a good agreement between the different reconstructions. There is only one inconsistent point (GP in NET(1)).

	$(a.u)$	PUT	GP	CAU	B(1)	RN	SN	B(2)
NET(1)	μ	93.4	134.1	93.7	42.1	175.4	209.6	12.3
	σ	9.5	21.8	10.7	1.6	10.8	14.6	4.6
NET(2)	μ	108.0	189.4	101.3	31.3	170.7	216.6	15.1
	σ	10.4	5.8	8.8	1.8	7.4	9.6	3.8
NET(3)	μ	105.2	178.7	98.1	34.3	168.8	207.0	11.3
	σ	8.6	5.1	9.1	1.1	10.0	17.1	5.2
NET(4)	μ	108.3	184.9	102.0	35.8	153.7	193.2	8.8
	σ	7.3	5.0	9.7	2.1	7.5	10.2	4.3

Table 4.8: Mean intensity values in NET(i) = NET-Z$_i$ reconstructions obtained with different orientation data as input. The correspondent plot is Fig. 4.19, left panel

The contrast analysis is reported in Fig. 4.19, right panel, and in Tab. 4.9. Even if the

mean values are consistent with one other, the small differences cause in some points a disagreement evaluating the CNR parameter. The bigger dispersion is related to the B(1)-background couples (PUT-B, GP-B, CAU-B). B(1) is the background of the first region of interest and it composed of general white matter. They also correspond to the points where CNR assumes the highest values. In Fig. 4.21, central panel, it can be observed the outline of the NET(i) values with respect to the COSMOS ones (red line). The dispersion around the identity function increases as CNR increases.
It is worth remembering the χ-modelling consideration: the COSMOS method models the susceptibility as a scalar property of matter, and the NET3 learns using the patches from COSMOS as a reference. But, there is evidence of anisotropic behaviour of white matter. In this sense, it is not surprising that something changes when we consider regions which contain white matter in multiple-orientation data.

		PUT-B	GP-B	CAU-B	GP-PUT	CAU-PUT	RN-B	SN-B
NET(1)	CNR	31.5	56.5	31.7	4.3	0.4	35.5	42.9
	ϵ	5.6	11.7	6.1	15.7	10.1	7.7	9.6
NET(2)	CNR	43.6	89.8	39.8	7.8	0.7	40.9	53.0
	ϵ	6.1	3.8	5.3	8.1	9.6	5.6	6.7
NET(3)	CNR	62.2	126.7	55.9	8.6	0.8	30.5	37.8
	ϵ	4.9	3.1	5.1	6.8	8.9	7.6	11.1
NET(4)	CNR	35.4	72.7	32.3	10.4	0.9	34.0	43.3
	ϵ	4.7	3.5	5.9	6.2	8.5	5.9	7.2

Table 4.9: CNR values in NET(i) = NET-Z$_i$ reconstructions obtained with different orientation data as input. The correspondent plot is Fig. 4.19, right panel

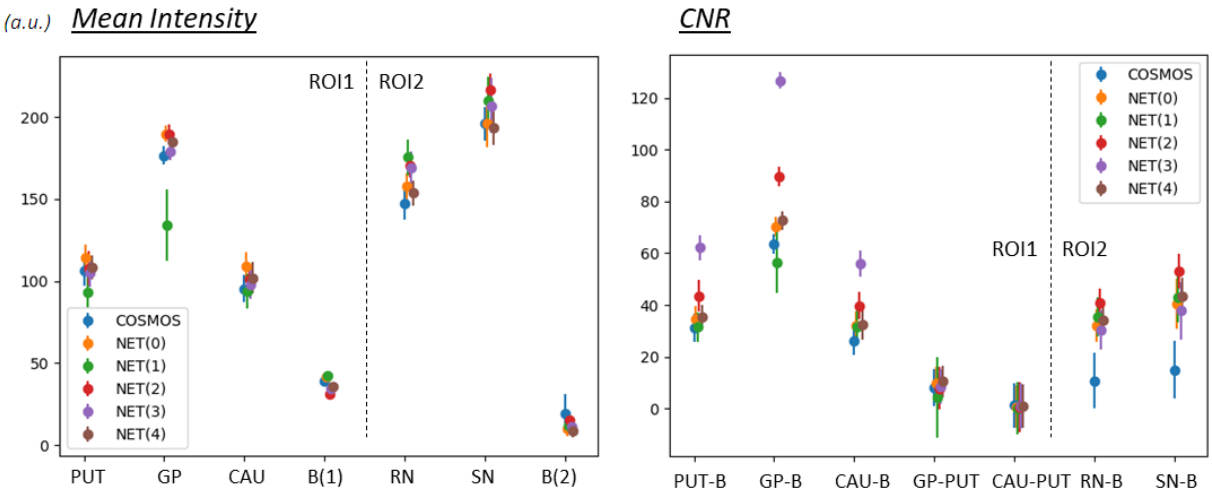


Figure 4.19: Mean intensity (left panel) and CNR (right panel) values in COSMOS and in NET(i) = NET-Z$_i$. ROI1 and ROI2 are analyzed. For NET(i), multiple-orientation data are taken into account. NET(i) were obtained with the NET3 model. It was trained by setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time is 6h, 50 min

- **Multiple-orientation Data: TKD**

Also the TKD technique was tested using multiple-orientation data. The results related to mean intensity and standard deviation values are reported in Fig. 4.20, left panel, and in Tab. 4.10. A bigger dispersion occurs than before, using the NET3 model.

	$(a.u.)$	PUT	GP	CAU	B(1)	RN	SN	B(2)
TKD(1)	μ	121.4	158.2	101.7	83.5	199.9	208.9	105.8
	σ	8.3	11.6	9.6	12.0	11.4	18.2	19.8
TKD(2)	μ	113.4	147.8	118.9	58.1	145.3	200.9	55.0
	σ	8.6	11.9	8.1	7.8	10.3	24.0	28.2
TKD(3)	μ	121.7	114.0	98.3	67.5	155.0	170.5	44.6
	σ	9.7	11.0	9.7	10.8	15.3	15.9	16.6
TKD(4)	μ	140.5	178.5	159.5	114.9	165.2	205.9	84.2
	σ	12.6	20.5	12.4	9.7	17.9	27.3	10.2

Table 4.10: Mean intensity values in TKD reconstructions obtained with different orientation data as input. The correspondent plot is Fig. 4.19, left panel

Mean intensity values are inconsistent each other, and that makes the CNR values show significant differences, as well. But, they are not inconsistent. This happens because the standard deviation values, used as error on the mean values, are high. So, also the CNR error is high - we in fact evaluated that using the propagation of uncertainty. The contrast analysis is reported in Fig. 4.20, right panel, and in Tab. 4.11.
Look into the scatter plot with the different TKD reconstructions with respect to the COSMOS values - Fig. 4.21, right panel. They assume lower values with respect to the COSMOS and also to the NET3 reconstruction.

		PUT-B	GP-B	CAU-B	GP-PUT	CAU-PUT	RN-B	SN-B
TKD(1)	CNR	3.2	6.2	1.5	4.4	2.4	4.8	5.2
	ϵ	10.2	11.8	10.6	10.0	8.8	15.6	19.0
TKD(2)	CNR	7.1	11.5	7.8	4.0	0.6	3.2	5.2
	ϵ	8.2	9.8	7.9	10.2	8.3	19.2	26.1
TKD(3)	CNR	5.0	7.1	2.9	2.3	2.4	6.8	7.7
	ϵ	10.2	10.9	10.2	10.3	9.7	15.8	16.1
TKD(4)	CNR	2.7	6.6	4.6	3.0	1.5	7.9	11.9
	ϵ	11.1	15.1	11.0	16.5	12.5	14.1	18.8

Table 4.11: CNR values in TKD reconstructions obtained with different orientation data as input. The correspondent plot is Fig. 4.20, right panel

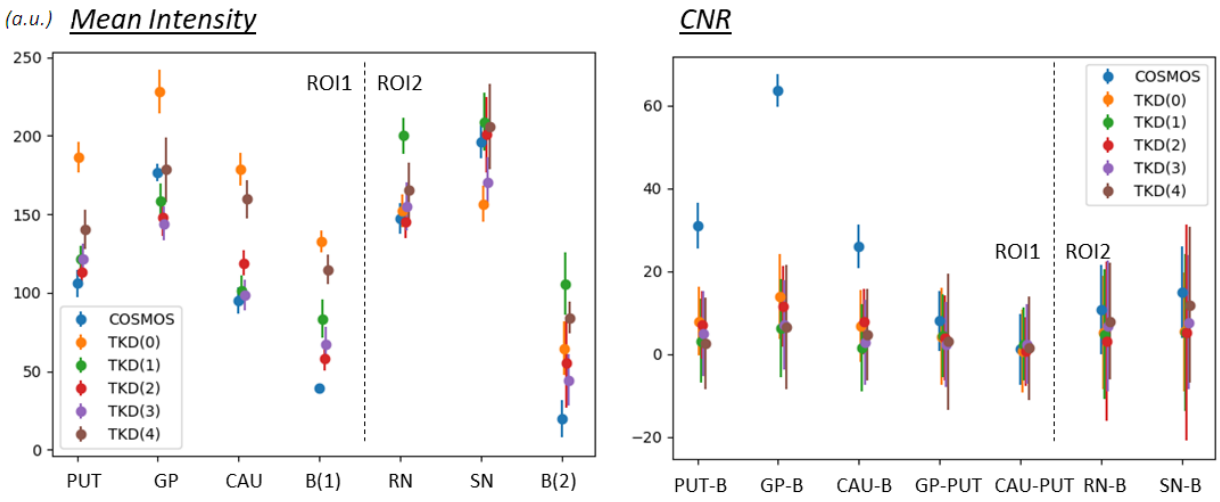


Figure 4.20: Mean intensity (left panel) and CNR (right panel) values in COSMOS and in TKD different reconstruction. ROI1 and ROI2 are analyzed. For TKD, multiple-orientation data are taken into account

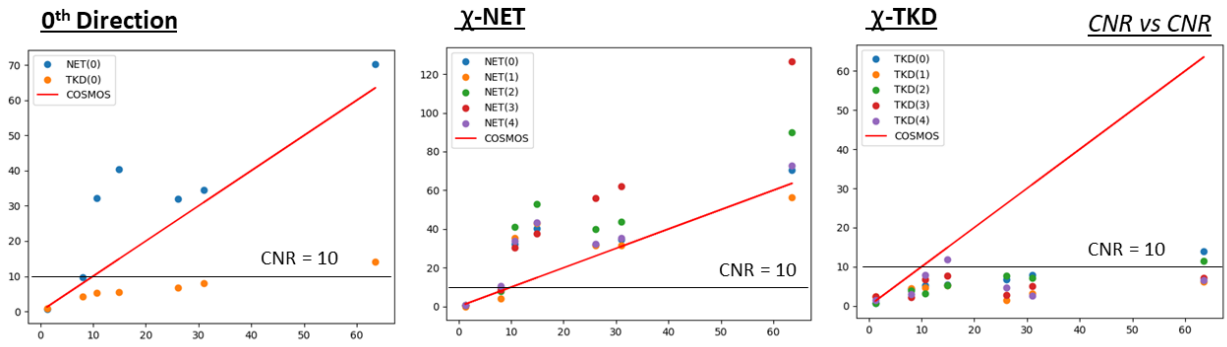


Figure 4.21: Left panel: Z-NET-, TKD- and COSMOS-CNR values *vs* COSMOS-CNR values. For Z-NET and TKD, single-orientation data are taken into account. Central panel: Z-NET- and COSMOS-CNR values *vs* COSMOS-CNR values. Multiple-orientation input data was used to reproduced the five χ-Z-NET maps. They were obtained with the NET3 model. It was trained by setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was $\sim$ 6h, 50 min. Right panel: TKD- and COSMOS-CNR values *vs* COSMOS-CNR values. Multiple-orientation input data was used to reproduced the five χ-TKD maps. The threshold α was set equal to 0.1 for all the considered orientations

4.2.3 Conclusions

- The NET3 model (Fig. 3.14) was trained again, using multiple-orientation data. Five different head-orientations were used. The input data were (64,64)-patches from ten axial slices. Referring to Sec. 3.2.4: D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was $\sim$ 6h, 50 min.

  The training and validation loss functions easily convergence (Fig. 4.11), and the similarity parameters (Tab. 4.5 and Fig. 4.12) show that the NET3 model achieves better perfomance than the TKD method, also with a multiple-orientation training. In the previous chapter, we have already notice how adding multiple-orientation data to the training dataset allows the generalization skill of the network to be increased.

- The entire 2D axial χ maps are reported in Fig. 4.14, from the 3D Z-NET reconstruc-

tion (Fig. 4.13). The susceptibility maps obtained with the trained model are accurate and with a low level of noise. Maps from multiple-orientation data were analyzed (Fig. 4.15).

- ROI1 and ROI2 were analyzed, looking into the proper patches, the gray level histograms and the scatter plots (Figs. 4.16 and 4.17). The outcome confirms the previous one.

- Intensity and constrast analysis were performed for COSMOS, 0^{th}-NET and 0^{th}-TKD maps (Fig. 4.18, Tabs. 4.7 and 4.6).
  There is a good agreement between the COSMOS and NET mean intensity values. The constrast analysis does not show perfect consistency everywhere. It may be due to the χ-modelling. In the points in which they assume discordant values, NET-CNR values are actually higher than the COSMOS- ones.

- Intensity and contrast analysis was performed also considering multiple-orientation input data. In Fig. 4.19 and Tabs. 4.8, 4.9 the NET performance is reported, in Fig. 4.20 and Tabs. 4.10, 4.11 the TKD results are proposed.
  Mean intensity values of NET_i, with $i = 0, 1, 2, 3, 4$ are consistent each other. Evaluating the contrast, a bigger dispersion occurs. This is not what we desire for our model: we want a network able to reconstruct the same susceptibility map starting form a phase map with any orientation. However, even if the values are spread out, they are higher and at least consistent with the gold-standard ones. The model recognition ability is still not compromised.
  The TKD mean values are inconsistent each other. Despite this, since their errors are high, the CNR values are actually consistent each other. But they have high inaccuracy - CNR's error is evaluated with the propagation of uncertainty - and they are lower than the ones from the other reconstructions.

4.3 Fast Fourier Transform and K-Space

K-space is a mathematical formalism that refers to a data matrix containing raw MRI data. To compute it, the Fourier Transform of the image, whether 2D or 3D, has to be calculated. Usually the **Fast Fourier Transform - FFT** is used. This is a slightly modified **Discrete Fourier Transform - DFT**. The FFT uses sparse matrix properties, factorizing the DFT matrix into a product of sparse matrices, to reduce the computational complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n * log(n))$ - where n is the dimension of the input image.
From the **Nyquist-Shannon Theorem**, we know that there is a relationship between the resolution in the spatial domain and that in the frequency domain, investigated in k-space. For example, consider an image $f(x, y)$ with dimension $N * M$, where the pixel size is Δx along the $\vec{x}$ axes and Δy along the $\vec{y}$ axis. Then, in the Fourier Transform $F(u, v)$, $\Delta u = \frac{2\pi}{N\Delta x}$

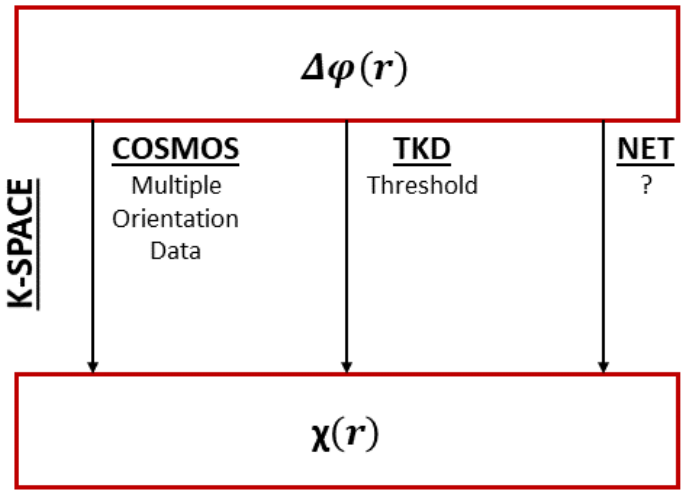


Figure 4.22: The QSM reconstruction is an ill-posed problem. How does a learning algorithm overcome it, being trained from space-domain-data?

and $\Delta v = \frac{2\pi}{M\Delta y}$. If we want the same resolution along all the axes in the Fourier space, it is necessary to resize the starting image in the spatial domain.

The $\phi(r) \longrightarrow \chi(r)$ relationship could be analyzed in k-space, where it is ill-posed. In fact, $\phi(k)$ has to be divided by the dipole $D(k)$, which assumes null values in some points (Fig. 1.10). The expression $D(k) = 0$ represents a double cone surface inclined from the $\vec{z}$ axis by the magic angle θ_m=54.7°. The COSMOS tool overcomes this problem by acquiring multiple-orientation scans (Sec. 1.2.5), while the TKD method changes the $D(k)$ expression setting a threshold α over which $D(k)$ has to be considered flat (Sec. 1.2.6, Eq. 1.31). It could be interesting to observe the k-space of the network susceptibility reconstruction, even if we have not used k-space data to the proper training stage.
To use k-space data is an alternative idea of training, already used in other inversion issues ([77]).

We performed the 3D-FFT transform of the χ Z-NET map (Fig. 4.13), and the correspondent maps from COSMOS and TKD. They have size (160,160,10). The size of the three axes has to be the same, in order to have the same resolution in the Fourier space. So, the matrices were padded with zeros to the size (160,160,160). The FFT $\mathcal{F}(\chi(r))$ is a 3D matrix with dimension (160,160,160). In this section, we are going to show only the axial slices of those matrices.
Fig. 4.23 shows axial slices from the three k-space matrices - respectively first row: COSMOS, second row: TKD, third row: Z-NET. For the Z-NET and the TKD reconstruction, the 0^{th}-direction input data were used. The shape of the double cone, as expected, can be seen in the TKD k-space. Instead, clearly there is no obvious evidence of the double cone in the COSMOS or in the Z-NET k-space.

We have also evaluated the differences between the COSMOS in k-space data and the Z-NET and TKD. These differences are shown in Fig. 4.24. In particular: the first row contains

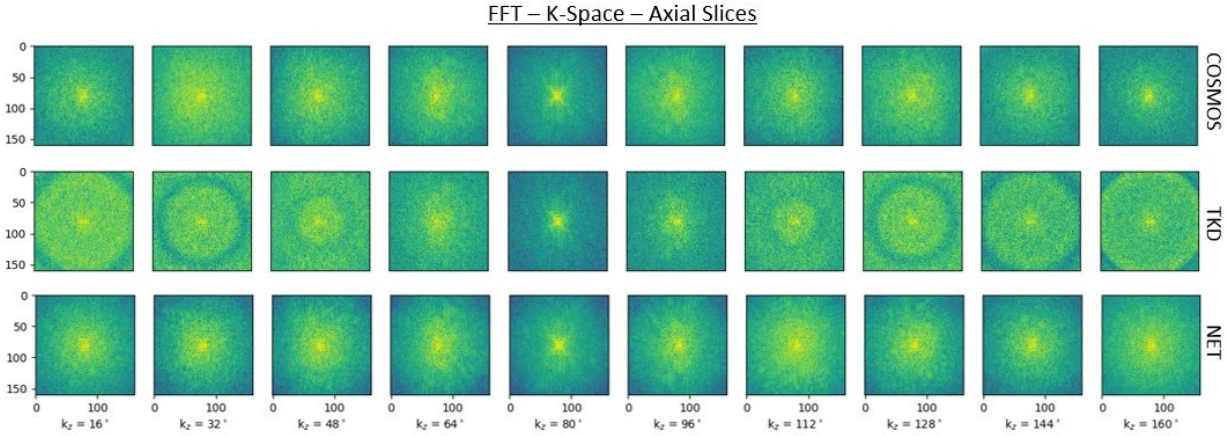


Figure 4.23: The 3D FFT of COSMOS (first row), TKD (second row) and Z-NET (third row) susceptibility map. Only axial slices of the k-space are reported. Z-NET and TKD used 0^{th} input data. Z-NET was built using the trained NET3 model. NET3 was trained setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was ∼ 6h, 50 min

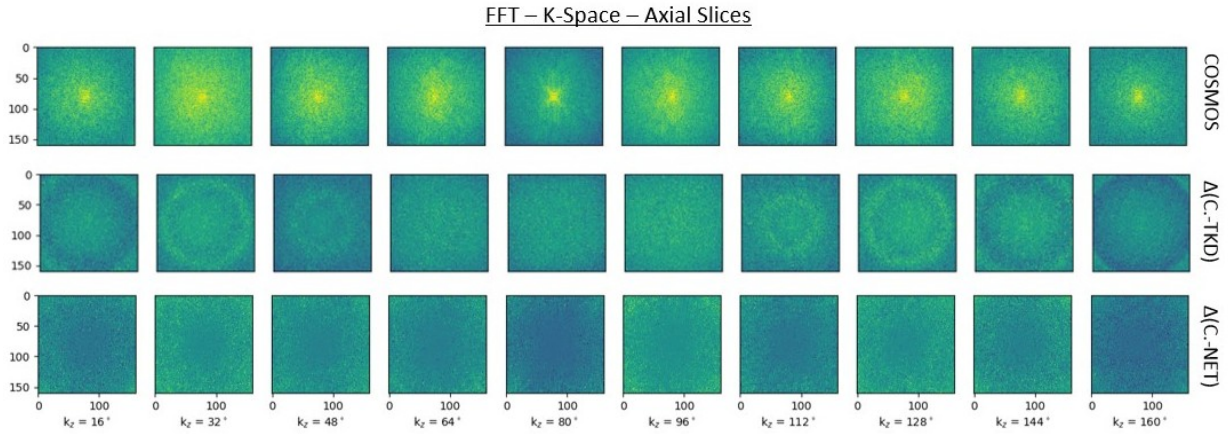


Figure 4.24: First row: the 3D FFT from COSMOS susceptibility map. Only axial slices of the k-space are reported. Second and third rows: differences between the first row and the 3D FFT from TKD (second row) and Z-NET (third row) susceptibility map. Z-NET was built using the trained NET3 model. NET3 was trained setting (Sec. 3.2.4): D=5, Z_{cut}=10, from 65 to 75, K=15, random angles from -20° to +20°, N=30. The resulting size is 11250, with f_{train}=0.8 and $f_{val} = f_{test}$=0.1. The other details are reported in Tab. 4.1. The total training time was ∼ 6h, 50 min

axial sections $k_z = i^{th}$ of the COSMOS k-space; the second row the differences between $k_z = i^{th}$ and the correspondent slices in the TKD k-space; the third row the differences between $k_z = i^{th}$ and the correspondent slices in the Z-NET k-space. The COSMOS and the Z-NET matrices are very similar around the centre of the k-space, while more different from one other in the high-frequency area. That part of the k-space corresponds to the detail of the reconstruction. Details and edges are the most difficult part to reconstruct, also because a convolutional neural network intrinsically performs a denoising operation.

The database which we are using ([10]) contains images which were registered (Sec. 3.2.1). It means that the magnetic field $\vec{B_0}$ is directed in the z-direction of the frame only in the 0^{th} image (Fig. 3.6). In the others, the $\vec{z}_{head}$ axis of the head is directed along the $\vec{z}_{frame}$, but the magnetic field is not because the head has been rotated in the scanner.

The dipole kernel $D(k)$ is directed along the direction of the magnetic field $\vec{B}_0$. Analyzing the TKD k-space using multiple-orientation data, the double cone $D(k) = 0$ is not always directed along the same direction inside the frame. To verify this, in Fig. 4.25, k-space axial slices from the TKD reconstructions, obtained using multiple-orientation data as input, are reported. The direction of the cone is actually not coincident with the vertical axis of the frame, excluding the 0^{th} direction.

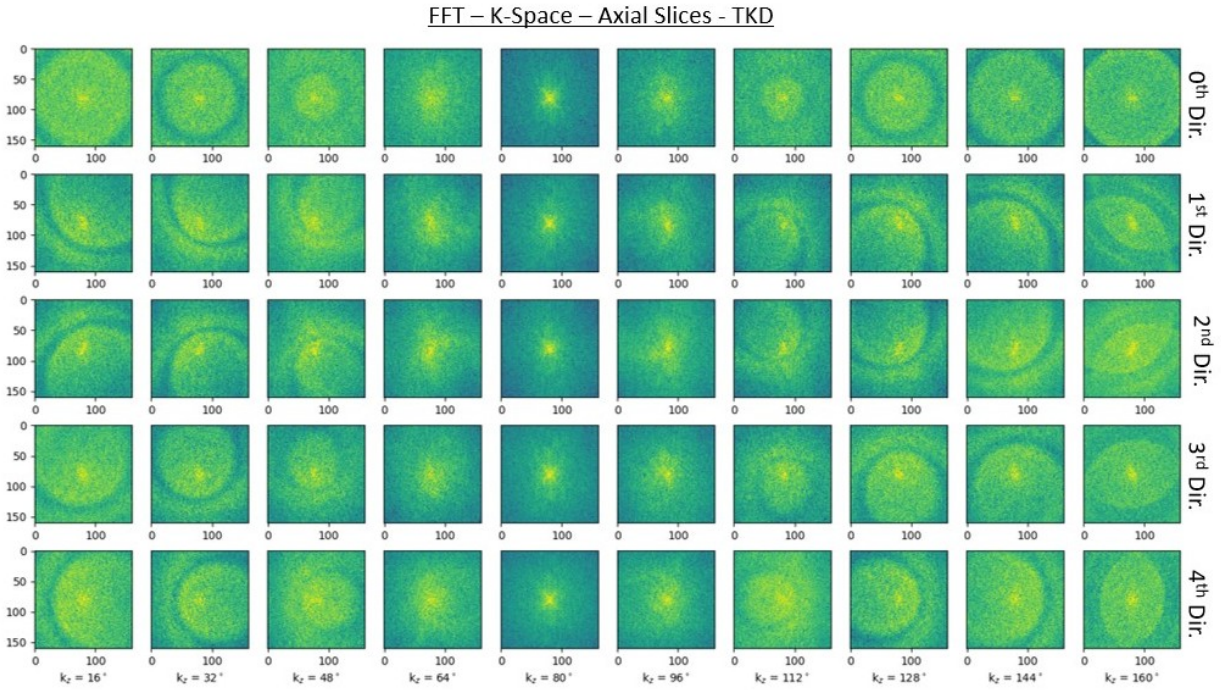


Figure 4.25: 3D FFT from TKD susceptibility maps. Only axial slices of the k-space are reported. In the first row, the direction of the field is the same of the direction of the frame. In the other rows, since the images were registered, the main field B_0 is not directed as the frame, and so is the $\vec{z}$ axis of the $D(k)$-cone (Sec. 1.2)

4.4 Generalization Skill

We have just verified that the NET3 model is able to reproduce a χ map closer to the gold-standard COSMOS than the TKD method. We also tested it using multiple-orientation ϕ data as input.

We analyzed Z-NET, a 3D susceptibility map with size (160,160,10), obtained by processing the ten axial slices used during the training. To test the generalization property of the model, Z-NET could be expanded by using all of the axial slices. This aims to explore the model reconstruction also outside of the training range.

4.4.1 Outside of Training Range

The Z-NET (Fig. 4.13) was built again, this time using all of the 160 available axial slices. It now has dimensions (160,160,160). Rember that the training range is $Z \in [65, 75]$. $Z = 60$ and $Z = 80$ slices are respectively reported, as examples, in Figs. 4.26 and 4.27.

The model, outside of the training range, does not work as well as inside the range. There are some structures, such as the one containing the putamen, caudate and globus pallidus ($Z = 80$), which are well reconstructed in the slices inside the training range, looking both into the shape and the constrast, but not outside.

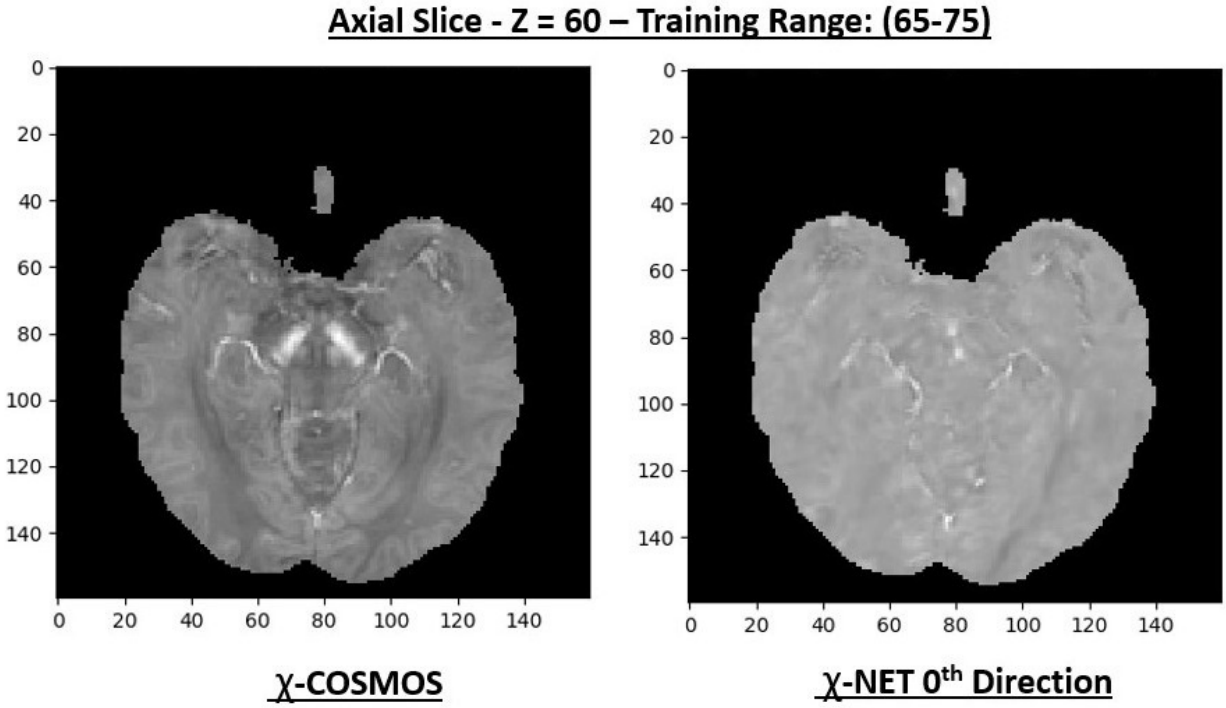


Figure 4.26: Axial slice (Z = 60) from COSMOS (left panel) and Z-NET (right panel) susceptibility map. Axial slices from 65 to 75 ($Z_{cut} \in (65, 75)$ were used during the training stage

4.4.2 Different Slices Orientation

We have tried to consider different slice orientations, to see if they allow an entire 3D map to be constructed.
Three susceptibility maps were built using the trained model, all of them with size (160,160,160). Let them be: **X-NET**, **Y-NET** and **Z-NET**. A scheme of their construction is reported in Figs. 4.28 (X-NET and Y-NET) and 4.13 (Z-NET). Sagittal, coronal and axial slices may be extracted from each of them.
The average map was also evaluated. Let it be **NET**:

$$\mathbf{NET} = \frac{\mathbf{X-NET} + \mathbf{Y-NET} + \mathbf{Z-NET}}{3} \tag{4.2}$$

Fig. 4.29 shows the reconstructions of a sagittal slice from X-NET, Y-NET and Z-NET. Only axial patches were used during the training. X-NET and Y-NET, which used patches respectively sagittally and coronally oriented, do not reproduce a high quality map, compared to the COSMOS, but also to the TKD method. Z-NET and NET, even if they are better than the other two, do not show competitive results.

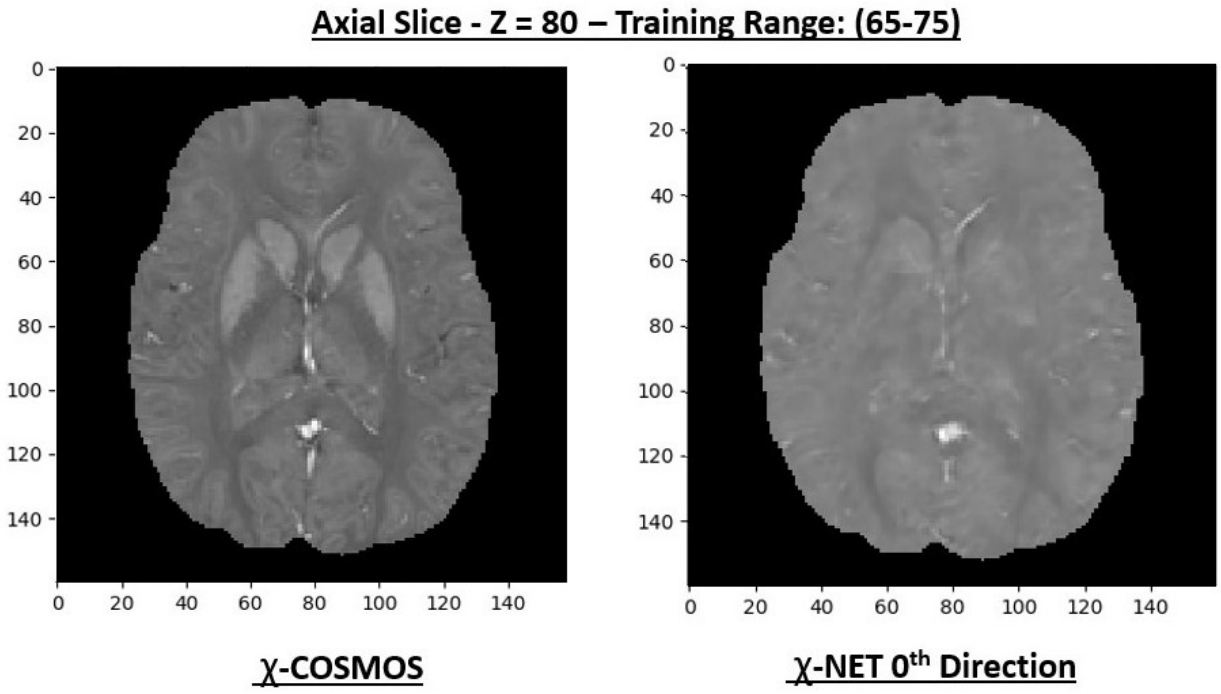


Figure 4.27: Axial slice (Z = 80) from COSMOS (left panel) and Z-NET (right panel) susceptibility map. Axial slices from 65 to 75 ($Z_{cut} \in (65, 75)$ were used during the training stage

To extend the range of training and to use 2D multiple orientation slices may be a good alternative to the 3D training, we are going to return to this point in the next chapter.
We did the same with coronal (Fig. 4.30) and axial slice reconstructions (Fig. 4.31), but we did not obtain satisfactory results.

4.5 Conclusions

- In this chapter we have tested the NET3 (Fig. 3.14) model, training it with 2D data, initially from single-orientation data (Sec. 4.1), then from multiple-orientation data (Sec 4.2).
  The model shows good performance in both these situations: there is a good agreement between its reconstruction and the gold-standard one, and their results are better than other single-orientation methods - e.g. the TKD approach, which we are using as a control map. Remember that, contrary to the TKD and NET3 technique, COSMOS is a multiple-orientation method. Our model achieves similar results, but using only one acquisition as input data.

- We have also looked into the Fourier transform of the χ NET map (Sec. 4.3), and noticed that it is very close to the COSMOS one, especially in the low-frequency region (centre of the k-space).

- We have to try to extend the result, since that the aim is to obtain a complete 3D susceptibility map.

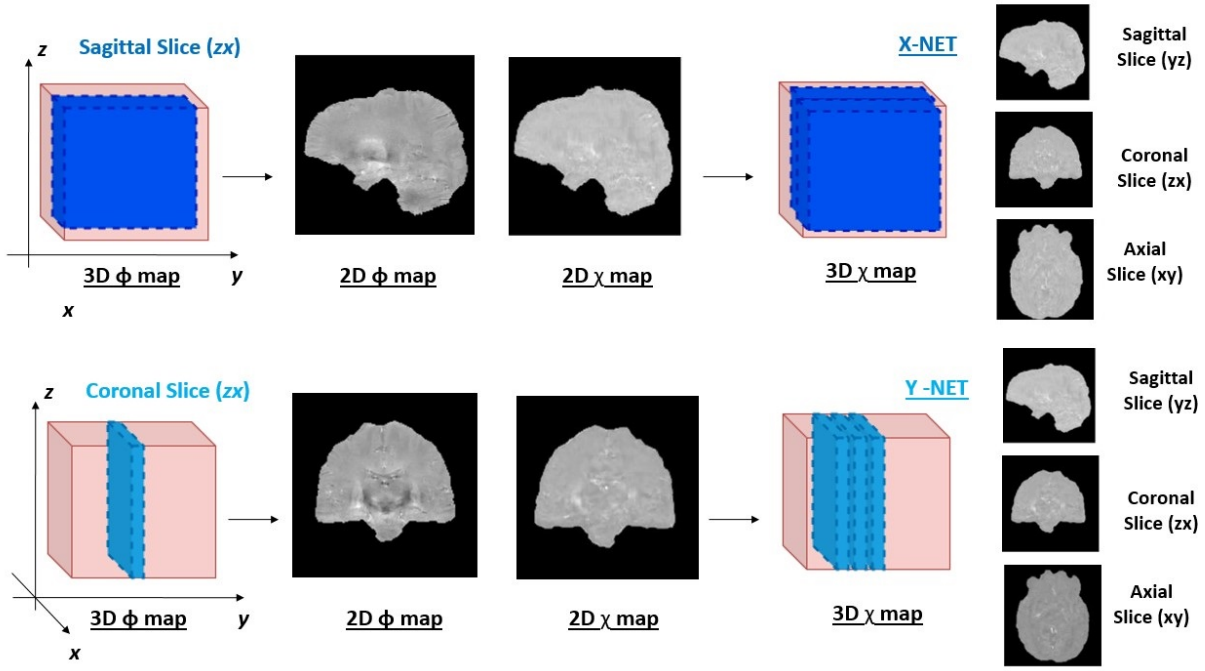


Figure 4.28: Scheme of 3D susceptibility map reconstruction from 2D sagittal (top panel) and coronal (bottom panel) slices processing

First, we have tried to extend the result on axial slices - the training dataset was built only with axial slices - outside of the range used during the training. Unfortunately, in this case the model is not able to reproduce a high-quality susceptibility χ map (Sec. 4.4.1). To have the reconstruction of the entire brain, considering only the axial projection, the training range has to be increased.
To work with 3D images or patches, instead of 2D ones, implies a significant increase in computational effort. We have to verify first whether this is necessary. Therefore, we have tried to reproduce also sagittal and coronal 2D susceptibility maps, even if the model was trained only with axial slices (Sec. 4.2.2). The results were not as good as hoped. In the next chapter we are going to present another experiment, in which NET3 was trained using different orientation slices - i.e. sagittal, coronal and axial ones. The idea is to use 2D data to reproduce a 3D χ map.

- Possible alternative training strategies are:

  1. to train the network with synthetic patches, as in [70].
  2. to invert the registration operation. If the direction of the field $\vec{B_0}$ is fixed in the frame, the model may learn better.

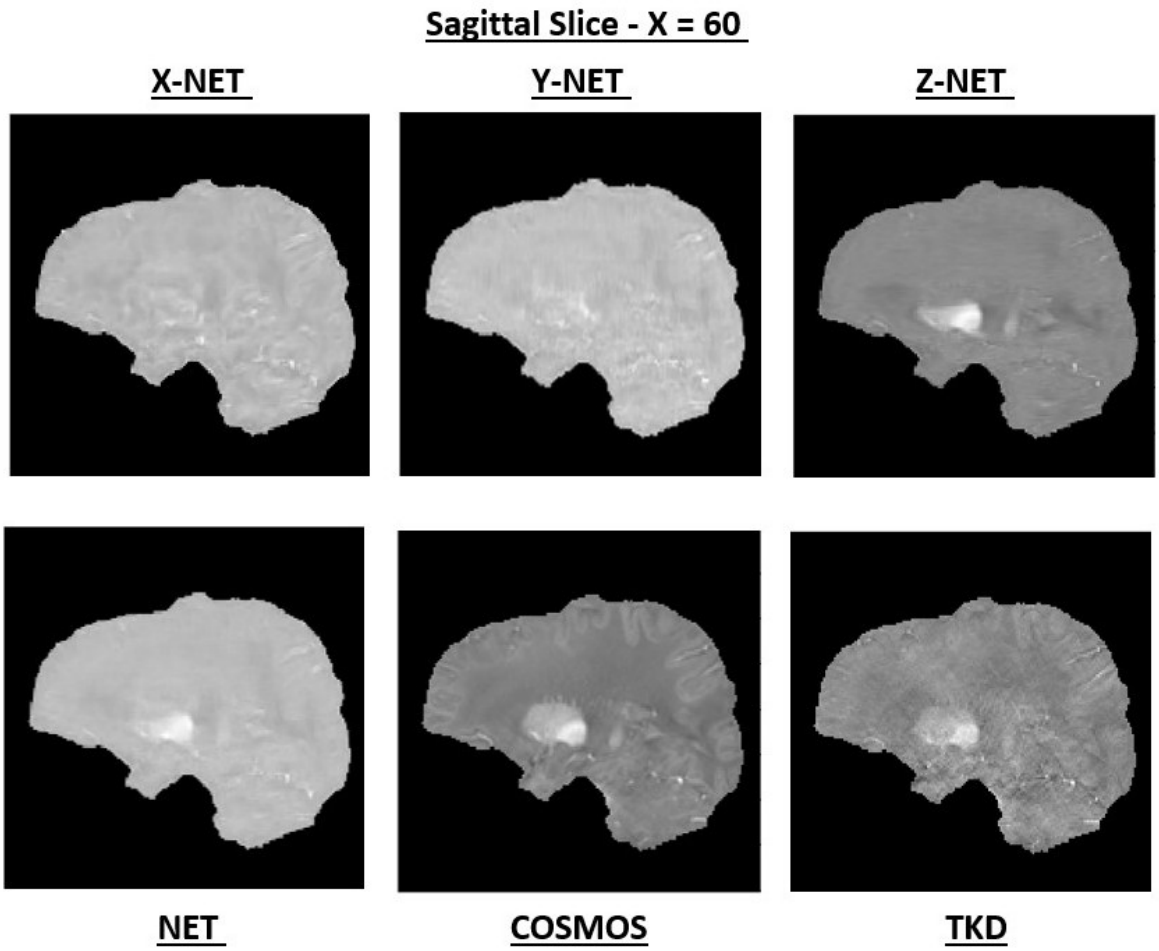


Figure 4.29: 2D sagittal susceptibility map (X=60) from: X-NET map (top-left), Y-NET (top-central), Z-NET (top-right), NET (bottom-left), COSMOS (bottom-central) and TKD (bottom-right) maps. They are maps with size (160,160,160). X-NET, Y-NET and Z-NET maps were obtained repatching (64,64)-patches from sagittal, coronal and axial slices. X-NET, Y-NET, Z-NET and NET were built using the trained NET3 model. NET3 was trained using only axial slices

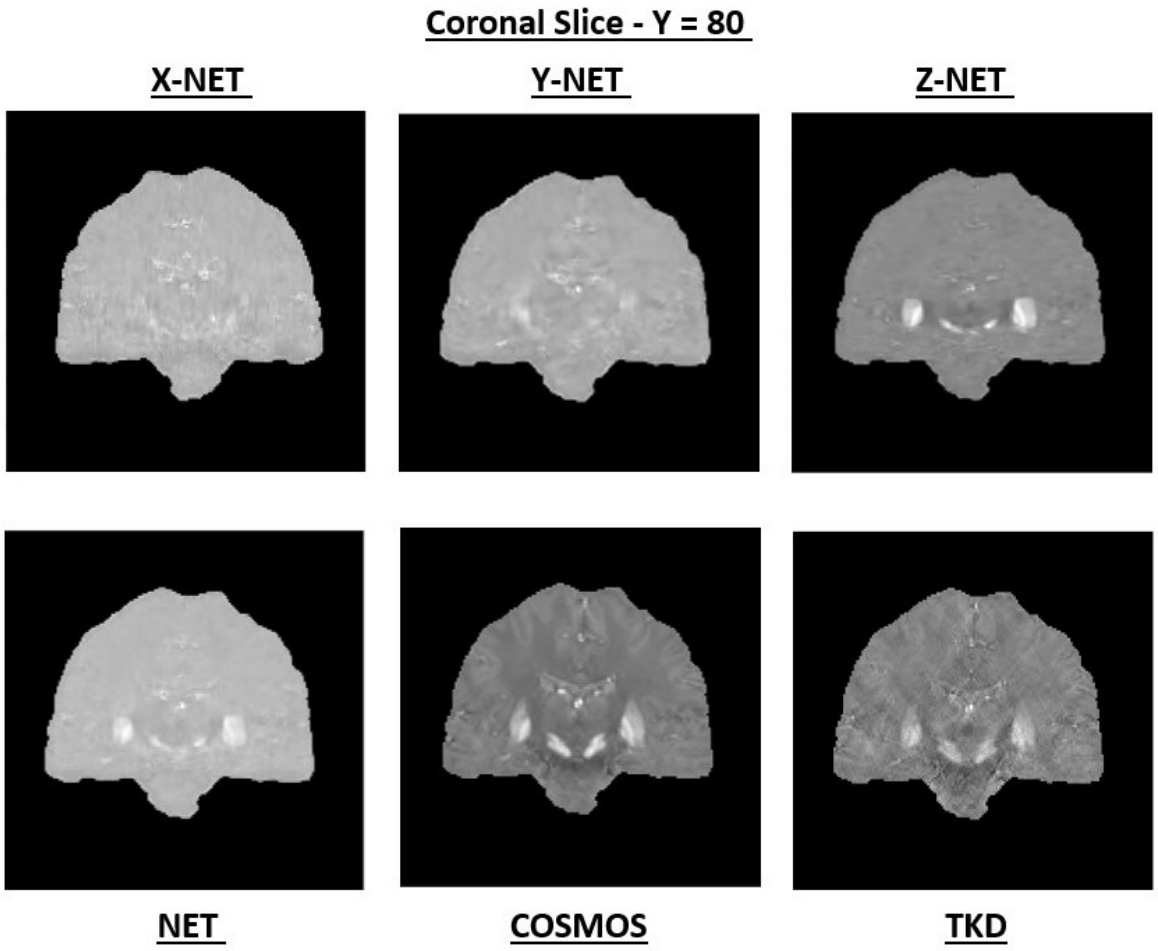


Figure 4.30: 2D coronal susceptibility map (Y=80) from: X-NET map (top-left), Y-NET (top-central), Z-NET (top-right), NET (bottom-left), COSMOS (bottom-central) and TKD (bottom-right) maps. They are maps with size (160,160,160). X-NET, Y-NET and Z-NET maps were obtained repatching (64,64)-patches from sagittal, coronal and axial slices. X-NET, Y-NET, Z-NET and NET were built using the trained NET3 model. NET3 was trained using only axial slices

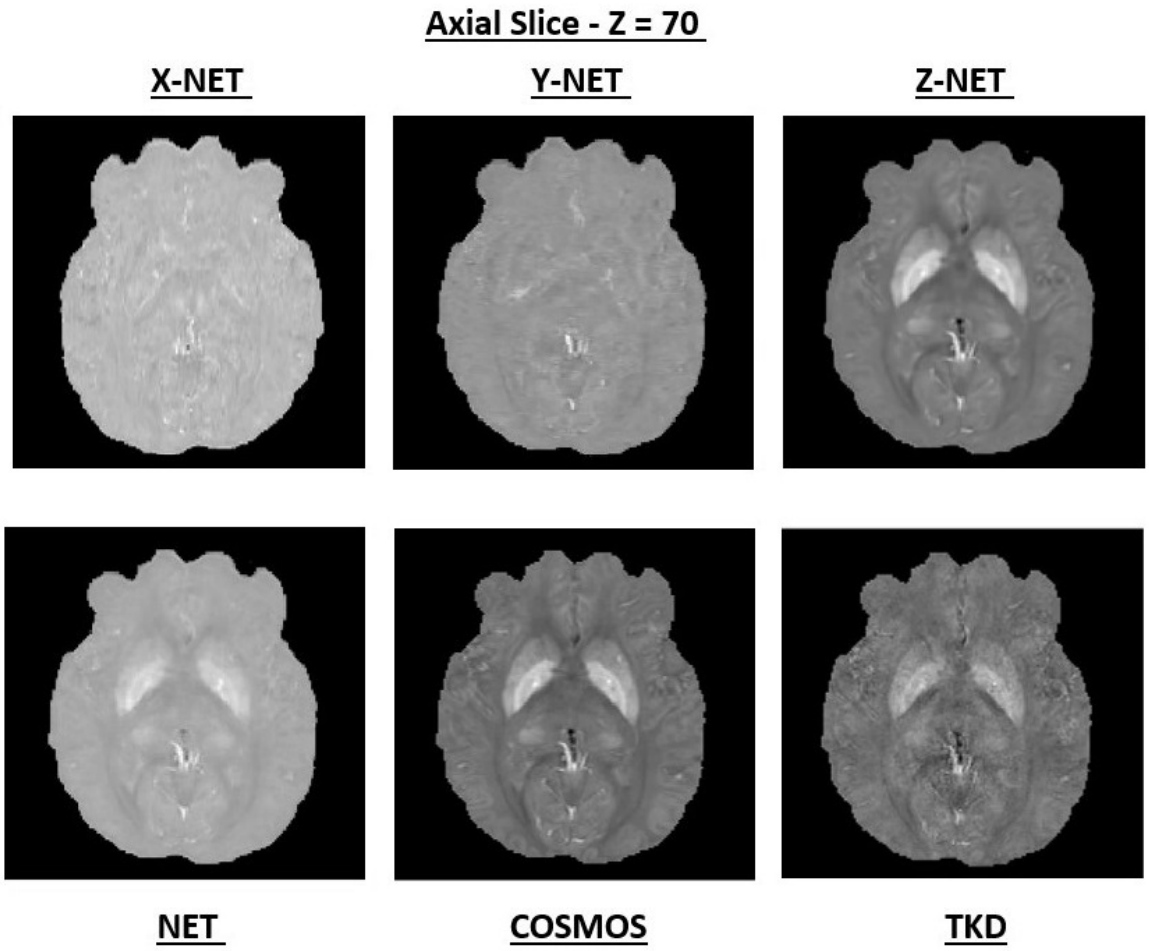


Figure 4.31: 2D axial susceptibility map (Z=75) from: X-NET map (top-left), Y-NET (top-central), Z-NET (top-right), NET (bottom-left), COSMOS (bottom-central) and TKD (bottom-right) maps. They are maps with size (160,160,160). X-NET, Y-NET and Z-NET maps were obtained repatching (64,64)-patches from sagittal, coronal and axial slices. X-NET, Y-NET, Z-NET and NET were built using the trained NET3 model. NET3 was trained using only axial slices

Chapter 5

3D Quantitative Susceptibility Reconstruction

In the previous chapter, we tested the NET3 model (Fig. 3.14) with 2D patches, both from single- (Sec. 4.1) and multiple- (Sec. 4.2) orientation ϕ maps. In those experiments, only axial slices were used. The returned susceptibility maps gave satisfactory results.
At the end of that chapter we have focused on two issues: to observe axial slices outside of the training range and to analyze slices from other orientations - i.e. to be able to reproduce a precise 3D susceptibility map.

In this chapter we propose other two experiments, in order to perform these goals. Chapter 5 is divided into the following sections:

- Section 5.1: **Multiple-Slice-Orientation Data Training** - NET3 is trained using sagittal, coronal and axial slices from a single-orientation ϕ map (0^{th} head-direction). A 12 Gb GPU memory is used. In this way, the training range could be extended. X-NET, Y-NET, Z-NET and NET, resulted from the trained model, are analyzed and compared with the TKD and COSMOS susceptibility maps.

- Section 5.2: **3D Data Training** - NET3 could not be adapted to 3D data experiments by simple changes in the size patches, first of all because of a memory issue. The second reason is that the model hyperparameters have to be evaluated again when the size of the input data is modified. Thus, the architecture of the network is adjusted, and the NET1-3D structure is implemented. It is trained using 3D patches from six head-orientation ϕ maps. The other maps are used during the generalization stage.
  The NET maps are studied and compared with the reconstructions obtained with the TKD technique, another single-orientation method, and the COSMOS reconstruction, the gold-standard susceptibility map.

- Section 5.3: **Conclusions, Outstanding Issues and Future Work** - A summary of the results of this chapter is presented, with a spotlight on the possible follow-up of this work.

5.1 Multiple-Slice-Orientation Data Training

The NET3 model was trained using sagittal, coronal and axial slices from 3D ϕ maps. Referring to Sec. 3.2.4, a 2D dataset, with multiple-orientation slices, was built: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1.
For this training, only single-orientation data were used.
The total training time was $\sim$ 15.6 hours.

RMSProp optimizer, $\eta = 0.001$
BCE loss function
Dropout fraction: 0.02
Batch size: 32
Epochs: 200
Keras 2.2.0, Tensorflow 1.9.0 Backend, Python 3.6
GPU Titanum XP - nVdia 384, Cuda 9.0

Table 5.1: Training details and hyperparameters of the experiment introduced in Sec. 5.1

Not all the available slices were used - i.e. 160 sagittal, 160 coronal and 160 axial ones - because not all of them contain relevant information. Looking into the grey-level histograms, we have realized that there are a lot of zero-value pixels. Focusing on the $\vec{z}$ direction, look into the graphs in Fig. 5.1. The original phase map was divided in $5 \cdot 5 \cdot 5$ patches with size (32,32,32). Then, the grey-level histogram was extracted from each 3D patch. A lot of patches are completely null, others with only few pixels with no-null values.

To work with the GPU memory is necessary in image processing with CNNs. Since the database is not that wide and 2D data are used, the GeForce 940MX graphic card - 4 Gb GPU memory - is an adequate support.
To reproduce a precise 3D susceptibility map, the dataset size has to broaden: the training range has to be extended and slices with multiple orientations have to be considered. In the last experiment we used a 11250-patches dataset, but in this one the number of patches of the training set is 162000, $\sim$ 15 times bigger than before. Consider also that the batch size is still equal to 32. It means that there are $\sim$ 5060 updates for each epochs, and $\sim$ 1012000 for the entire training.

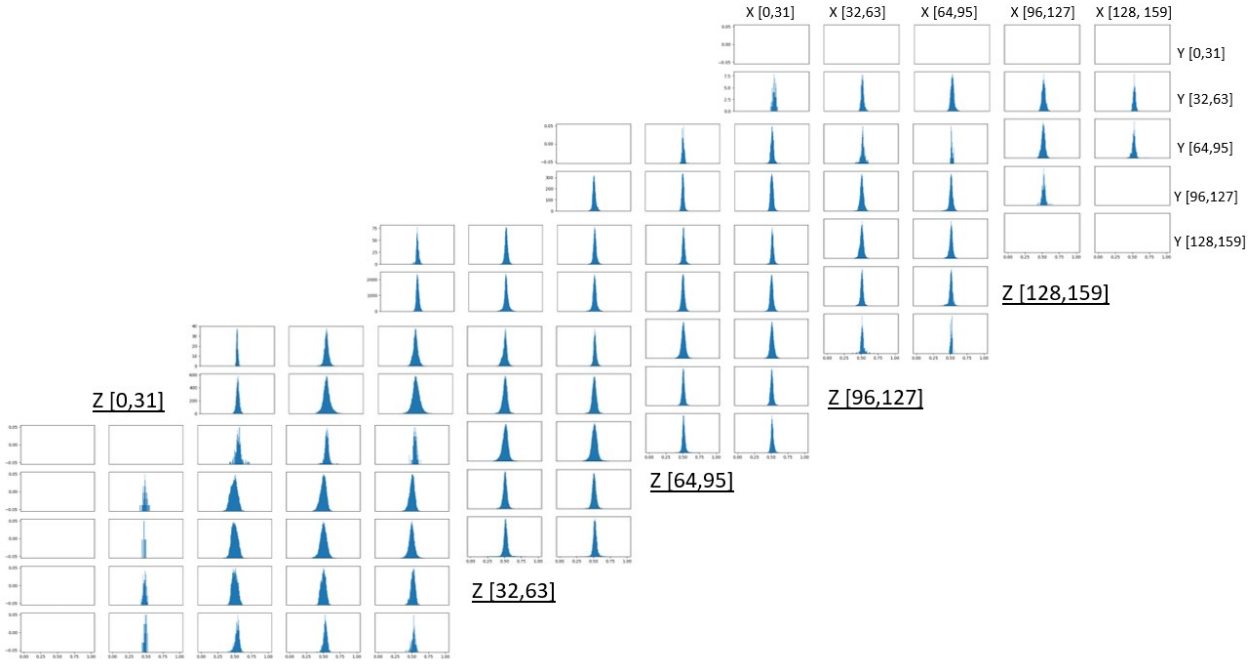


Figure 5.1: The original 0^{th} ϕ map was divided in 125 (32,32,32) patches. The grey-level histograms were extracted

A 4 Gb GPU memory is not enough anymore. We changed the graphic card, using Titanum XP, a 12 Gb GPU memory.

5.1.1 Results: Training

The shape of the training and the validation loss functions is reported in Fig. 5.2. Notice that the error values are reduced than before (cf. Fig. 4.2 and 4.11), even if we did not change the loss function expression (BCE): they rose from $\sim 0.676/0.677$ to 0.483 and to 0.484 respectively for the validation and the training loss.
Note also that, the training function is higher than the validation one; the gap is ~ 0.3 %. Both the error functions easily converge. The fact that the values are smaller means that to use more slices with different orientations helps in finding a minimum in the error gradient space. The model had been trained for 200 epochs, but they could be less.

5.1.2 Results: X-, Y-, Z-NET and NET Reconstructions

The X-NET, Y-NET and Z-NET suceptibility maps were built, following the procedure described in Sec. 4.2.2 (Figs. 4.13 and 4.28). They were obtained processing sagittal (**yz**), coronal (**zx**) and axial (**xy**) slices respectively. Also the average NET map (Eq. 4.2) was calculated. From each of them, sagittal, coronal and axial slices could be extracted.
In Fig. 5.3, a sagittal slice is reported ($X = 60$), from the X-NET, Y-NET, Z-NET, NET, COSMOS and TKD maps. Remember that the sagittal training range is $X \in [50, 110]$. As before, the NET3 model allows a better suceptibility map than the TKD method to be reproduced, closer to the COSMOS one and less noisy.

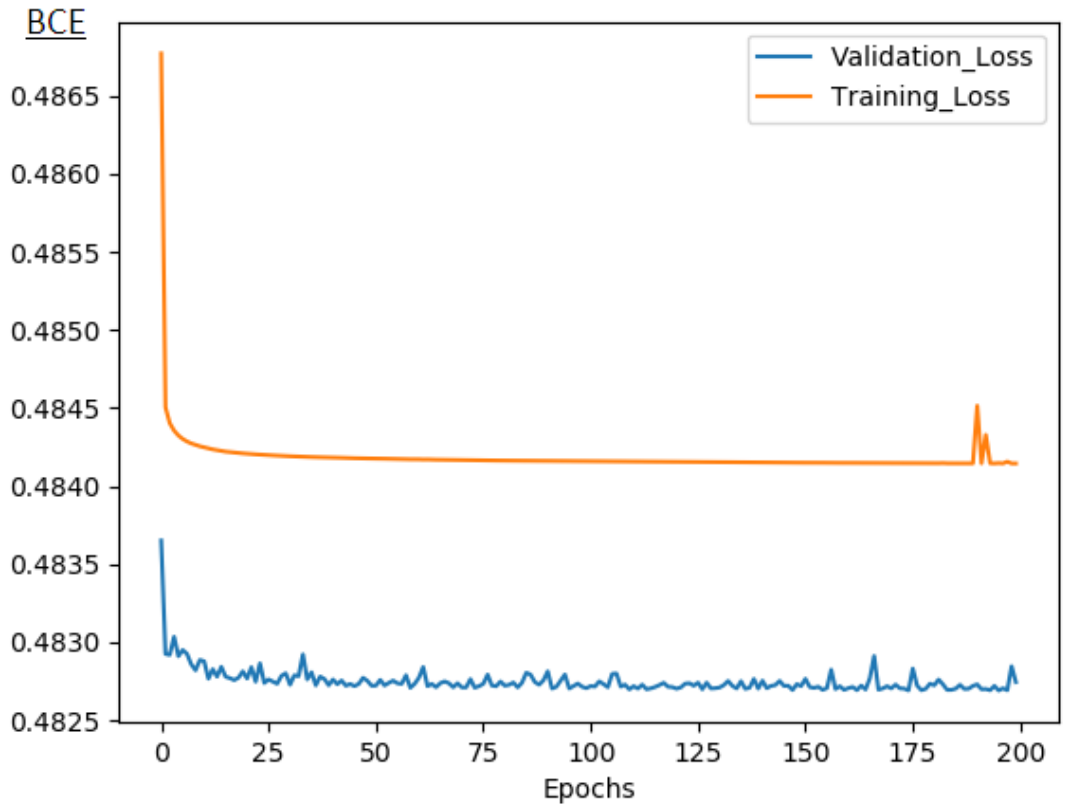


Figure 5.2: Validation and training loss functions of the experiment introduced in Sec. 5.1. Referring to the Sec. 3.2.4, a 2D multiple-orientation dataset, with multiple slice-orientations, was built: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

The X-NET reconstruction is qualitatively better than the Y-NET, Z-NET and NET maps. It means that what the model learns is still related to the orientation of the slices inside the brain, and not to the actual relationship between the phase ϕ and the susceptibility χ. This is not surprising, since the relationship to be learnt is in the 3D space and the model is trained with 2D data.

Two slices, a coronal ($Y = 80$) and an axial ($Z = 70$) one, are respectively reported in Figs. 5.4 and 5.5, from the X-NET, Y-NET, Z-NET, NET, COSMOS and TKD maps. The coronal and axial training range is respectively $Y \in [50, 120]$ and $Z \in [60, 110]$. The result related to the NET3 training is better than the TKD one.
As sagittal patches processing gives better results in order to reproduce a sagittal susceptibility map, coronal and axial patches work better to respectively reconstruct a coronal and an axial map. We are going to confirm that result also studying the metric parameters in the next section.
In Fig. 5.6, sagittal slices from X-NET (first row), coronal slices from Y-NET (second row) and axial slices from Z-NET (third row) are shown.

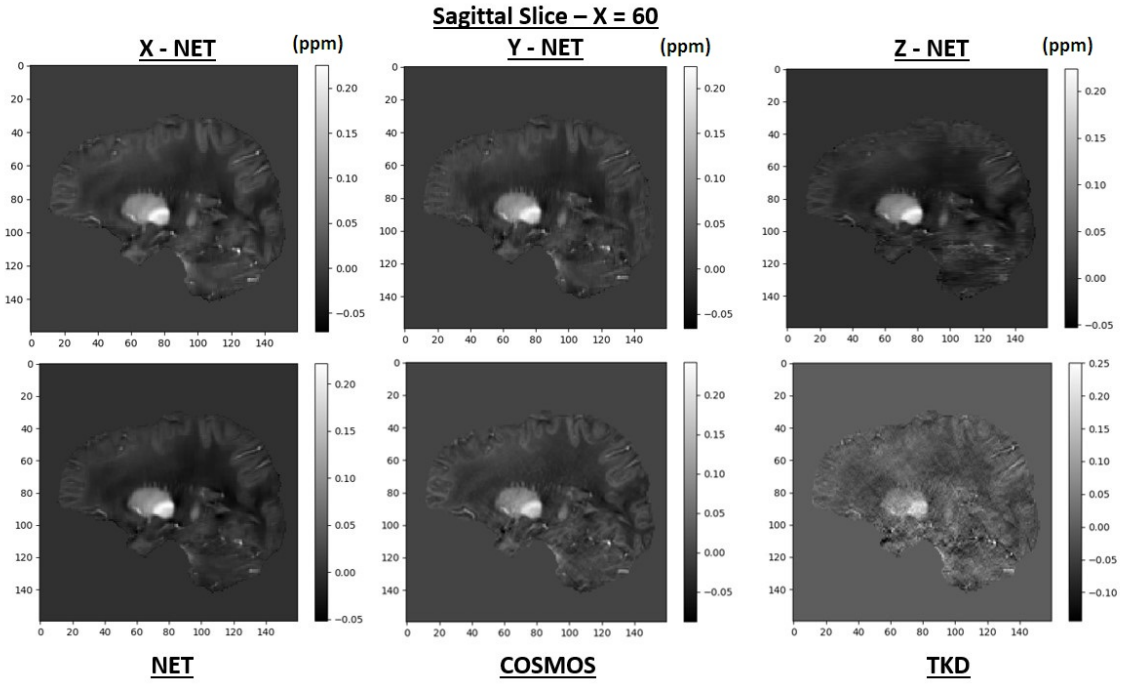


Figure 5.3: Sagittal susceptibility map ($X = 60$) from the three dimensional maps: X-NET, Y-NET, Z-NET, NET, COSMOS and TKD ($\alpha = 0.1$). X-NET, Y-NET, Z-NET and NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

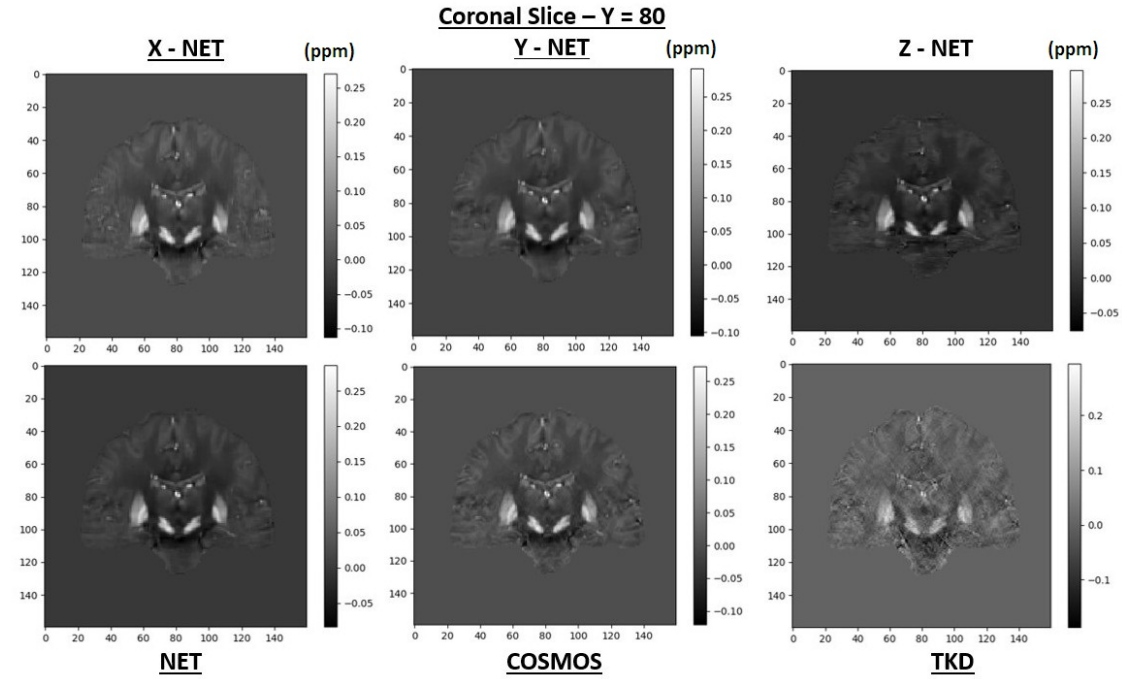


Figure 5.4: Coronal susceptibility map ($Y = 80$) from the three dimensional maps: X-NET, Y-NET, Z-NET, NET, COSMOS and TKD ($\alpha = 0.1$). X-NET, Y-NET, Z-NET and NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

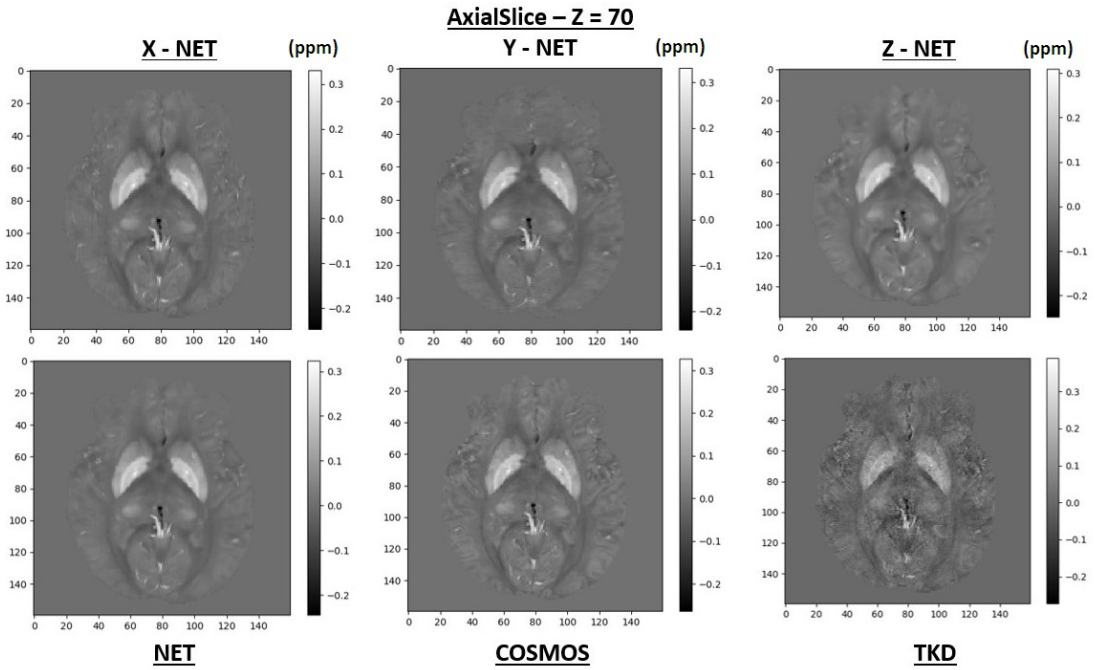


Figure 5.5: Axial susceptibility map ($Z = 70$) from the three dimensional maps: X-NET, Y-NET, Z-NET, NET, COSMOS and TKD ($\alpha = 0.1$). X-NET, Y-NET, Z-NET and NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

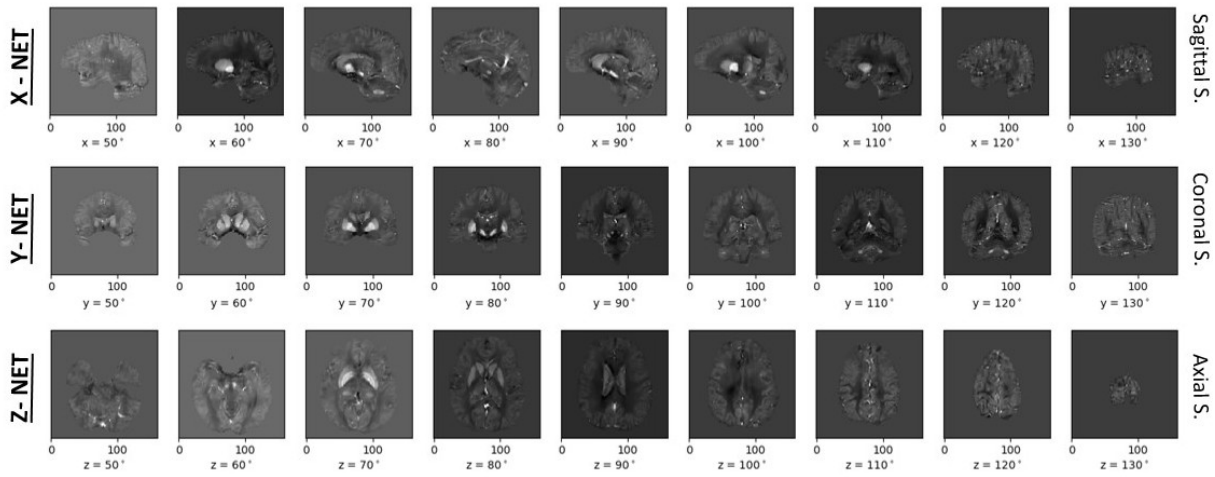


Figure 5.6: First row: sagittal slices from the X-NET map. Second row: coronal slices from the Y-NET map. Third row: axial slices from the Z-NET map. X-NET, Y-NET, Z-NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

5.1.3 Results: Metrics

Similarity parameters were studied for sagittal, coronal and axial slices from the X-NET, Y-NET, Z-NET, NET and TKD maps. The COSMOS map was used as reference for all the examined susceptibility reconstructions.
The first reported results are related to the sagittal slices. From each 3D susceptibility maps, the slices from $X = 50$ to $X = 110$ were analyzed. They are the same slices used during the training stage, which cover all the significant information of the original ϕ image. In Tab. 5.2, the average values of RMSE, pSNR, SSIM and HFEN are reported.

- The X-NET map shows better performance with respect to all the criteria. The NET reconstruction is rank second.

- The relative errors of RMSE and HFEN in the X-NET map improve. They are respectively 20%, from 25%, and 17%, from 27%. The ones related to pSNR and SSIM did not change and are $\sim$ 3% and 0.3%.

- The X-NET and Y-NET maps are more similar each other than the Z-NET reconstruction.

- The RMSE and SSIM values are better than before (cf. Sec. 4.1 and 4.2), while pSNR and HFEN are almost the same.

- The TKD perfomance changes. The RMSE and the HFEN values worse, while pSNR and SSIM are almost the same.

Sagittal S.	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
X-NET	0.00005	0.00001	31.7	1.0	0.972	0.003	0.35	0.06
Y-NET	0.00009	0.00003	29.2	1.0	0.945	0.008	0.47	0.06
Z-NET	0.00014	0.00004	27.0	1.0	0.939	0.008	0.63	0.11
NET	0.00006	0.00002	30.8	1.0	0.967	0.003	0.41	0.06
TKD	0.00030	0.00013	24.2	1.2	0.90	0.02	0.85	0.07

Table 5.2: RMSE, pSNR, SSIM and HFEN considering sagittal slices from X=50 to X=110. This range corresponds with the training range. The X-NET, Y-NET, Z-NET, NET and TKD ($\alpha = 0.1$) maps were examined, using the COSMOS map as reference. X-NET, Y-NET, Z-NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

The results associated with coronal and axial slices are reported in Tabs. 5.3 and 5.4. Slices in $Y \in [50, 120]$ and $Z \in [60, 110]$ were respectively taken into account.
Looking into the Y-results it may be notice that:

Coronal S.	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
X-NET	0.00006	0.00001	31.3	1.7	0.962	0.008	0.44	0.06
Y-NET	0.00004	0.00001	32.9	1.7	0.972	0.006	0.32	0.04
Z-NET	0.00010	0.00005	29.1	1.8	0.95	0.03	0.61	0.11
NET	0.00004	0.00001	32.6	1.6	0.973	0.008	0.38	0.05
TKD	0.00022	0.00003	25.4	1.9	0.91	0.01	0.81	0.09

Table 5.3: RMSE, pSNR, SSIM and HFEN considering coronal slices from Y=50 to Y=120. This range corresponds with the training range. The X-NET, Y-NET, Z-NET, NET and TKD ($\alpha = 0.1$) maps were examined, using the COSMOS map as reference. X-NET, Y-NET, Z-NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

Axial S.	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
X-NET	0.000049	0.000004	30.5	1.7	0.95	0.01	0.44	0.07
Y-NET	0.000063	0.000004	29.8	1.8	0.938	0.009	0.46	0.09
Z-NET	0.000037	0.000005	32.2	1.7	0.978	0.002	0.40	0.10
NET	0.000038	0.000003	32.1	1.7	0.965	0.006	0.36	0.08
TKD	0.00016	0.00003	25.2	1.8	0.90	0.03	0.79	0.11

Table 5.4: RMSE, pSNR, SSIM and HFEN considering axial slices from Z=60 to Z=110. This range corresponds with the training range. The X-NET, Y-NET, Z-NET, NET and TKD ($\alpha = 0.1$) maps were examined, using the COSMOS map as reference. X-NET, Y-NET, Z-NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

1. the Y-NET and the NET maps show results consistent each other. The X-NET and Z-NET reconstructions are worse; as before, Z-NET shows more differences with respect to Y-NET than X-NET.

2. The HFEN relative error, in Y-NET, increases, rising from 17% to 12%. The pSNR and the SSIM ones worse, but not significantly - they are repectively 5% and 0.6%. The RMSE relative error is the same as before - i.e. 20%.

3. Y-NET results in analyzing coronal slices are consistent with X-NET in analyzing sagittal slices.

4. The TKD parameters are consistent with the one in Tab. 5.2, except for the RMSE parameter, which increases.

Looking into the Z-results it may be notice that:

1. Z-NET and NET perfomances are consistent each other, and they are better than the other reconstructions. Z-NET results in analyzing axial slices are consistent with Y-NET ones in analyzing coronal slices and with X-NET ones in analyzing sagittal slices.

2. The RMSE and the SSIM relative errors increase, rising to 13% and 2%. The HFEN one is worse than before - i.e. 22%.

3. The TKD results do not change, except for the RMSE parameters, which is better.

Remember that to take a look into the relative errors of the metric parameters is important in order to verify the uniform response of the model. Also, consider that it is not surprising that the four parameters, aimed to evaluate the similarity between two 2D images, are not always in agreement.
From Tabs. 5.2, 5.3 and 5.4, we have observed that, with a view to reproduce a 2D susceptibility map with a certain orientation - i.e. sagittal, coronal or axial one -, patches with the same orientation are better to be processed. To confirm this, we analyzed the outline of the four parameters along the sagittal, coronal and axial slices. Results are shown in Figs. 5.7 (RMSE), 5.8 (pSNR), 5.9 (SSIM) and 5.10 (HFEN), respectively in the left, central and right panels.

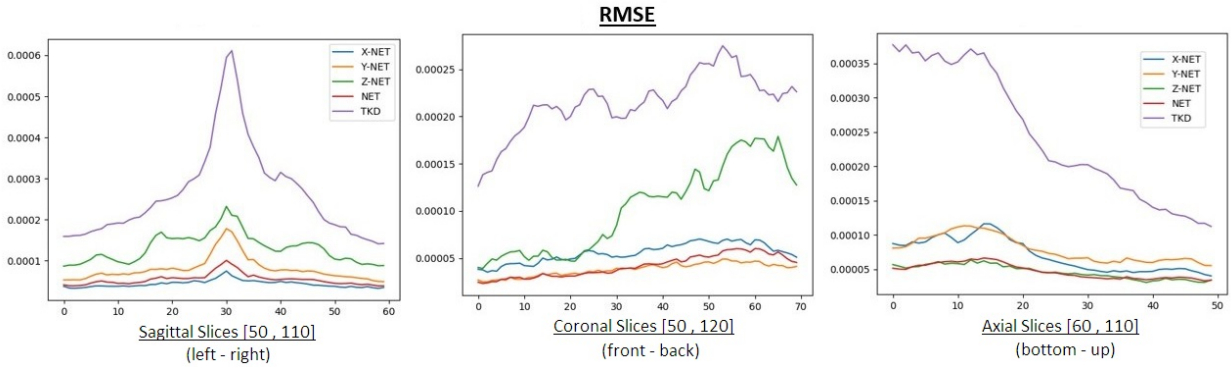


Figure 5.7: The RMSE outline analyzing sagittal (left panel, $X \in [50, 110]$), coronal (central panel, $Y \in [50, 120]$) and axial (right panel, $Z \in [60, 110]$) slices. The X-NET, Y-NET, Z-NET, NET and TKD ($\alpha = 0.1$) reconstructions were analyzed. X-NET, Y-NET, Z-NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

That analysis confirms the result obtained before: the X-NET (blue line) map shows the best result for sagittal slices, Y-NET (orange line) for coronal slices and Z-NET (green line) for axial slices. The NET reconstruction, because it is an average, is close to the best map whatever is the considered slice orientation.
It could be noticed also that there is no a uniform response of the model moving through the brain, in all the three directions. Some parameters, such as RMSE and SSIM, are more sensitive than others. The same shape is observed both for our model and the TKD reconstructions.

- RMSE (Fig. 5.7) - It shows a net trend for all the directions. Sagittal slices are worse reconstructed in the middle region, around $X = 30$ - considering the range $X \in [50, 110]$) -, coronal slices worse increasing Y, axial slices increase increasing Z. The same trends are followed by all the techniques.

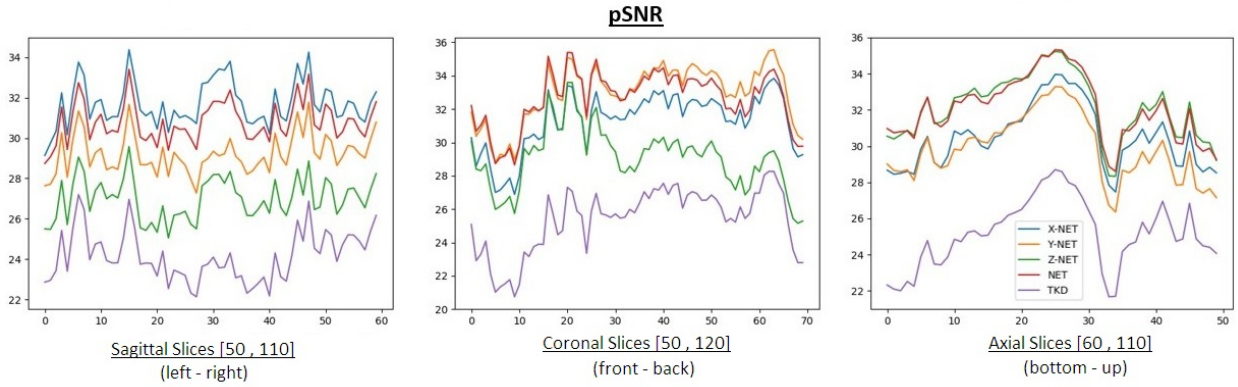


Figure 5.8: The pSNR outline analyzing sagittal (left panel, $X \in [50, 110]$), coronal (central panel, $Y \in [50, 120]$) and axial (right panel, $Z \in [60, 110]$) slices. The X-NET, Y-NET, Z-NET, NET and TKD ($\alpha = 0.1$) reconstructions were analyzed. X-NET, Y-NET, Z-NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

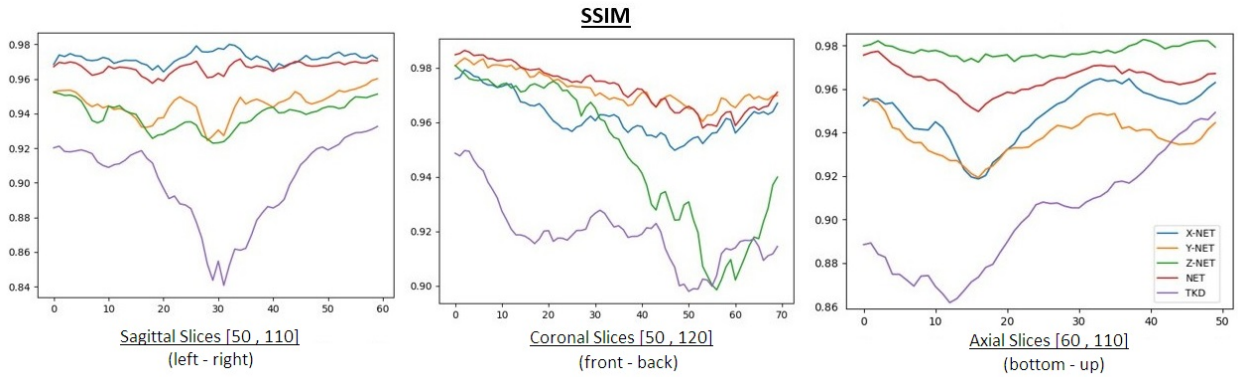


Figure 5.9: The SSIM outline analyzing sagittal (left panel, $X \in [50, 110]$), coronal (central panel, $Y \in [50, 120]$) and axial (right panel, $Z \in [60, 110]$) slices. The X-NET, Y-NET, Z-NET, NET and TKD ($\alpha = 0.1$) reconstructions were analyzed. X-NET, Y-NET, Z-NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

- pSNR (5.8) - It shows a more uniform response. It confirms the fact that its relative error is smaller than the RMSE one. Only for the axial slices a gap between two region of the brain seems to occur: in the firs part ($Z \leq 25$) pSNR values are high - i.e. the reconstructions are better -, and they are lower in the second part. This fact agrees with the shape of the RMSE along the axial slices.

- SSIM (Fig. 5.9) - Its shape along the sagittal, coronal and axial orientations is agree with the RMSE one: there is a central area in the sagittal range in which the reconstructions are worse, they are better in the first part of the coronal range than in the secon one, and the opposite happens in the axial range.

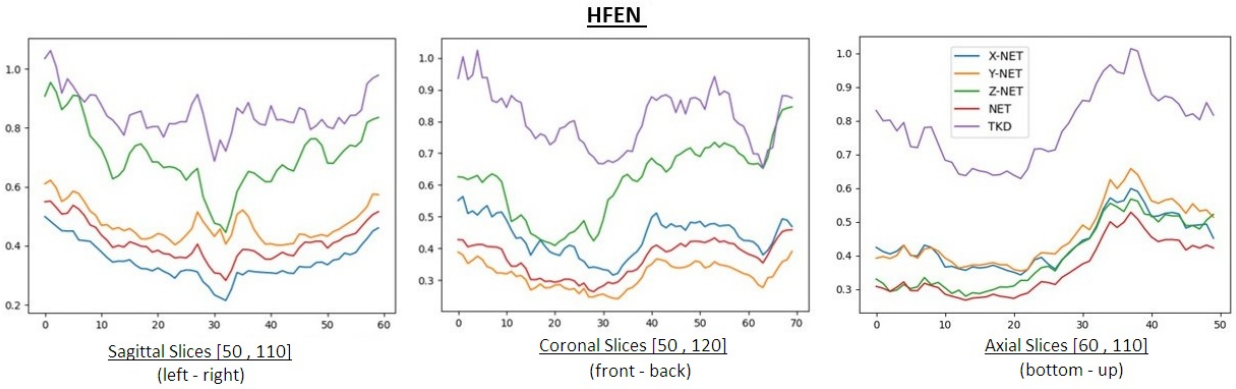


Figure 5.10: The HFEN outline analyzing sagittal (left panel, $X \in [50, 110]$), coronal (central panel, $Y \in [50, 120]$) and axial (right panel, $Z \in [60, 110]$) slices. The X-NET, Y-NET, Z-NET, NET and TKD ($\alpha = 0.1$) reconstructions were analyzed. X-NET, Y-NET, Z-NET ware obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

- HFEN (Fig. 5.10) - The outline along the sagittal and the axial range is agree with the SSIM and RMSE results. For the coronal range, a decline is not observed.

5.1.4 Results: Intensity and Contrast Analysis

The intensity and contrast analysis was focused on the ROI1 and ROI2 (Fig. 4.1). They are from axial slices, so the analysis was carried out only for the Z-NET and NET maps. The model perfomance was compared with the TKD and the COSMOS ones.

Mean intensity values of the putamen, globus pallidus, caudate, red nucleus, substantia nigra and background are reported in Fig. 5.11. Rember that PUT, GP, CAU, RN and SN are grey-matter-composed structures, while B(1) and B(2) are mainly composed of white matter, respectively of general white matter and white matter tracts. The standard deviation was also extracted, and it was considered as an error.
As before, the image processing software ImageJ was used to extract the intensity and the standard deviation values. The maps were convertend in 8-bit type images, with dynamic range [0,255]. The values on the $\vec{y}$ axis are in arbitrary units and they represent the intensities of the examined areas.
The correspondent values are reported in Tab. 5.5. The COSMOS, Z-NET and NET values are consistent each other everywhere - they are a bit far in the RN and SN points. At contrary, the TKD reconstruction shows different behaviour in many points. The error values of the TKD maps also show that the reconstruction is noisier than the others.

The intensity and the standard deviation values were used to evaluate the CNR parameter. Its error was calculated with the propagation of uncertainty. Fig. 5.12 and Tab. 5.6 show the result.
For the couples in the ROI1, there is a good agreement between the COSMOS and NET values. The Z-NET map shows higher contrast in the WM-background points, while it is

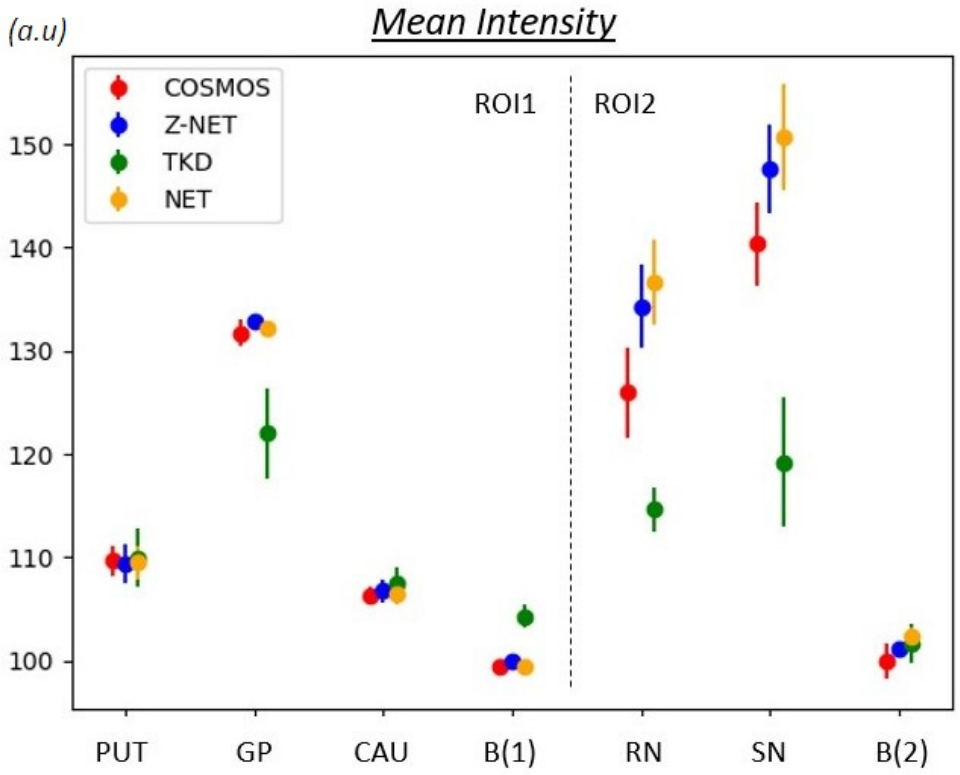


Figure 5.11: Mean intensity values in COSMOS, Z-NET, NET and TKD ($\alpha = 0.1$) susceptibility maps, from ROI1 and ROI2. The correspondent values are in Tab. 5.5

	$(a.u.)$	PUT	GP	CAU	B(1)	RN	SN	B(2)
COSMOS	μ	109.7	131.7	106.3	99.5	125.9	140.3	99.9
	σ	1.5	1.3	0.8	0.6	4.4	4.0	1.8
TKD	μ	109.9	122.0	107.5	104.3	114.7	119.2	101.7
	σ	2.9	4.3	1.5	1.1	2.1	6.3	1.9
Z-NET	μ	109.3	132.9	106.7	100.0	134.3	147.7	101.1
	σ	1.8	0.8	1.1	0.1	4.0	4.3	0.8
NET	μ	109.5	132.2	106.5	99.5	136.7	150.7	102.3
	σ	1.6	0.5	1.0	0.5	4.1	5.1	0.9

Table 5.5: Mean intensity values in COSMOS, Z-NET, NET and TKD ($\alpha = 0.1$) susceptibility maps, from ROI1 and ROI2. The Z-NET and NET maps were obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

consistent in the PUT-background points. In particular, there is a spot which assumes a CNR value higher than 300 (GP-B). In the right panel of the figure, we can observe the same results without that point, in order to give a better visualization of the other values.
In ROI2, both the NET and the Z-NET maps shows higher contrast values than the COSMOS map, consistent each other.
The values assumed by the TKD map are almost everywhere inconsistent with respect to the other reconstructions.

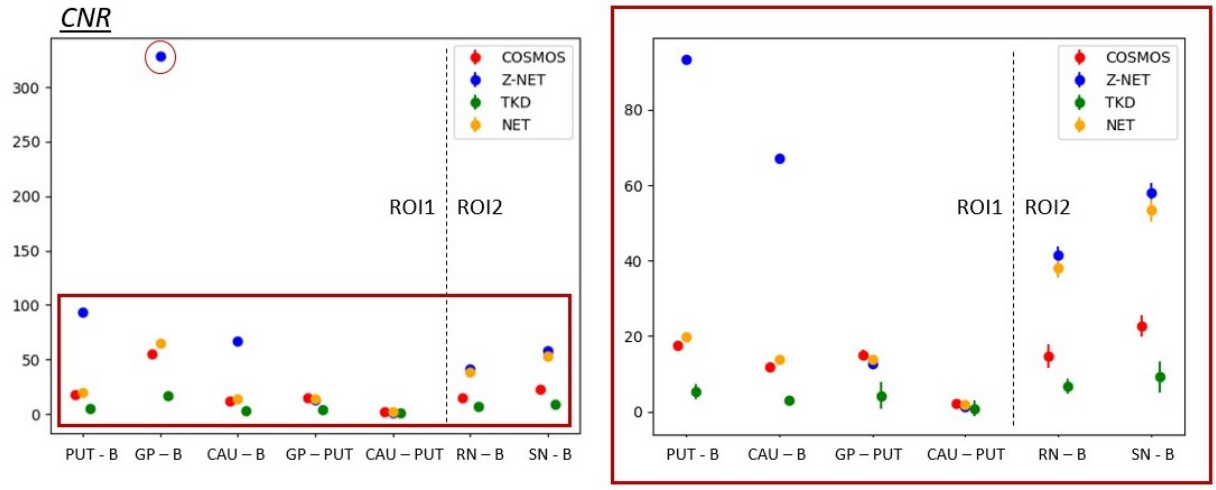


Figure 5.12: Left panel: CNR values in COSMOS, Z-NET, NET and TKD ($\alpha = 0.1$) susceptibility maps, from ROI1 and ROI2. The correspondent values are in Tab. 5.6. Right panel: same values, without the GP-B point

		PUT-B	GP-B	CAU-B	GP-PUT	CAU-PUT	RN-B	SN-B
COSMOS	CNR	17.5	55.5	11.8	15.1	2.3	14.6	22.7
	ϵ	1.0	0.9	0.7	1.4	1.1	3.1	2.9
TKD	CNR	5.2	16.4	3.0	4.2	0.8	6.8	9.2
	ϵ	2.0	2.7	1.3	3.6	2.2	2.0	4.1
Z-NET	CNR	93.4	328.9	67.2	12.8	1.4	41.4	58.1
	ϵ	1.0	0.4	0.6	1.3	1.5	2.4	2.5
NET	CNR	19.9	65.0	14.0	14.0	1.8	38.0	53.4
	ϵ	1.1	0.5	0.8	1.1	1.3	2.5	3.0

Table 5.6: CNR values in COSMOS, Z-NET, NET and TKD ($\alpha = 0.1$) susceptibility maps, from ROI1 and ROI2. The Z-NET and NET maps were obtained thanks to the NET3 model training (Sec. 4.2.2 - Figs. 4.13 and 4.28). NET3 was trained, referring to the Sec. 3.2.4, setting: D=1 (0^{th}-direction, Fig. 3.6), $X_{cut} = 60$ ($X \in [50, 110]$), Y_{cut}= 70 ($Y \in [50, 120]$), Z_{cut}=50 ($Z \in [60, 110]$), K=30 (random angles in $[-20°, +20°]$), N=30. Patches have size (64,64). The dataset size is 162000, with $f_{train} = 0.8$ and $f_{test} = f_{val} = 0.1$. The other set details and hyperparameters are reported in Tab. 5.1. The total training time was $\sim$ 15.6 h

The CNR values in a scatter-plot form could be observed in Fig. 5.13. Notice that the NET and Z-NET values are higher or at least the same as the COSMOS ones, represented by the red line, while the TKD values are always lower.

5.1.5 Conclusions

- Sagittal, coronal and axial slices were used to train the NET3 model. Only single-orientation data were used (0^{th} head direction). The size of the dataset was increased, and to change the graphic card was necessary - from 4 to 12 Gb GPU memory. Remember that it is almost impossible working only with CPU memory since we are using convolutional neural networks aimed to process 2D and 3D images.

- The X-NET, Y-NET, Z-NET and NET 3D maps were built. Some 2D slices are

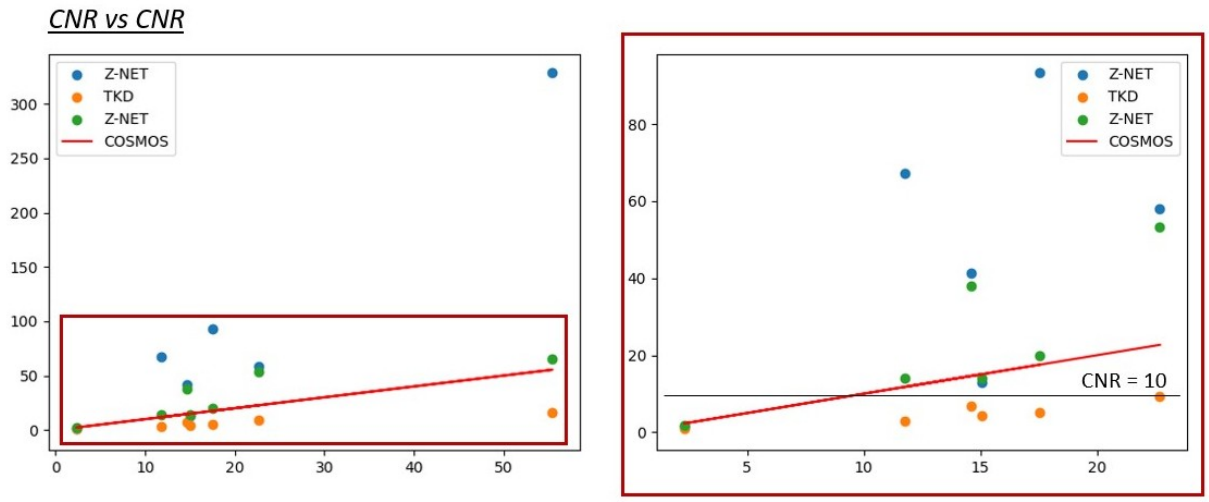


Figure 5.13: Left panel: CNR-values scatter plot, comparing Z-NET, NET and TKD reconstructions with the COSMOS one. Right panel: same values considering the CNR range < 100

reported in Sec. 5.1.1. Observing the reconstructions, it could be notice that the perfomances are better if the orientation of the patches used to build the map is the same of the considered slice. So, X-NET, Y-NET and Z-NET are better in order to observing respectively sagittal, coronal and axial slices.

- The metric parameter evaluation (Sec. 5.1.2) confirms that result. Also, we have analyzed the non uniform response of the model moving along all the three direction of the brain. In the third chapter, we have already noticed that. In according to the parameters chosen to evaluate the similarity, there are regions of the brain harder to be reproduced. This does not occur only for our model, but also for the TKD susceptibility map.
  The maps obtained using the trained model are always better than the TKD one. The COSMOS map were used as reference both for the NET3 and the TKD technique.

- Intensity and contrast analysis was also carried out (Sec. 5.1.3). There is not a complete agreement between out model and the COSMOS results - but NET and Z-NET values are higher or at least the same than the COSMOS ones. At contrary, the TKD map shows minor agreement and worse performance.

5.2 3D Data Training

In the previous section we verified that, extending the training range and using slices with different orientations, the NET3 allows a well reconstructed 3D susceptibility map to be obtained. From sagittal, coronal and axial slices processing, three (160,160,160) χ maps - i.e. X-NET, Y-NET, Z-NET - were returned. Respectively, they were used to analyze sagittal, coronal and axial slices.
The results were good, even if we had to build three, and not a single one, susceptibility maps

in order to observe the brain from different orientations. This is necessary, since 2D data are used to train the model and the network shows a convolutional autoencoder architecture.

This section is instead dedicated to a 3D data experiment. The main issue in moving from 2D to 3D data use is related to the memory. The computational effort substantially increases, because a significant amount of extra data has to be managed. Let l be the number of grey levels of the examined image. If its size is (x_{dim}, y_{dim}), then the required number of bytes is equal to $n_b = (x_{dim} \cdot y_{dim} \cdot l)$. Instead, if it is a three dimensional image with size $(x_{dim}, y_{dim}, z_{dim})$, $n_b = (x_{dim} \cdot y_{dim} \cdot z_{dim} \cdot l)$. Thus, to use (64,64,64) patches instead (64,64) ones means to have an available space 64 times bigger than before for each patch.
The number of feature maps and the batch size have also to be taken into account. In fact, the available memory has to be big enough to manage all the patches in same batch through the different convolutional layers.
For these reasons, even if the GPU memory in use was increased, the patch size and the number of feature maps have to be reduced. Also other changes in the network architecture are going to be presented. Secs. 5.2.1 and 5.2.2 are dedicated to the adjustment of the structure. Then, the proper experimental part is proposed, from Sec. 5.2.3 on out.

5.2.1 Reduction of the Number of Kernels

As we have just mentioned, we have to reduce the number of kernels in the convolutional layers of the network, in order to perfom a 3D data training.
Looking into the 32 feature maps of the first 2D convolutional layer, we did not think it would have been an issue. All the 32 feature maps obtained from that layer, using the ϕ-ROI1 as input, are reported in Fig. 5.14. A lot of them are almost flat, no carrying any relevant information. The idea to initially reduce the number of kernels comes from this consideration.
Therefore, a 2D data training was performed using the NET3 model, reducing the kernel configuration. The number of feature maps rose from (32, 64, 128, 256, 128, 64, 32) to (16, 32, 64, 128, 64, 32, 16). The ϕ-ROI1 was processed by the first convolutional layer, and the 16 feature maps are reported in Fig. 5.15.

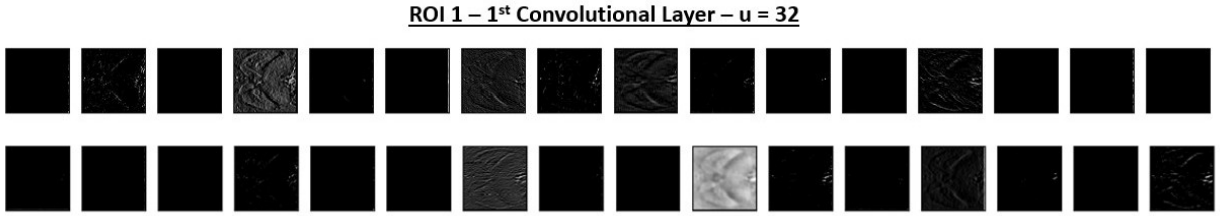


Figure 5.14: ϕ-ROI1 was processed by the first 2D convolutional layer of NET3. The model was trained with the kernel configuration (32,64,128,256,128,64,32)

But, a precise and analytical consideration could not be made only qualitatively looking into the feature maps. In fact, they assume are very small since the first layers and the fit

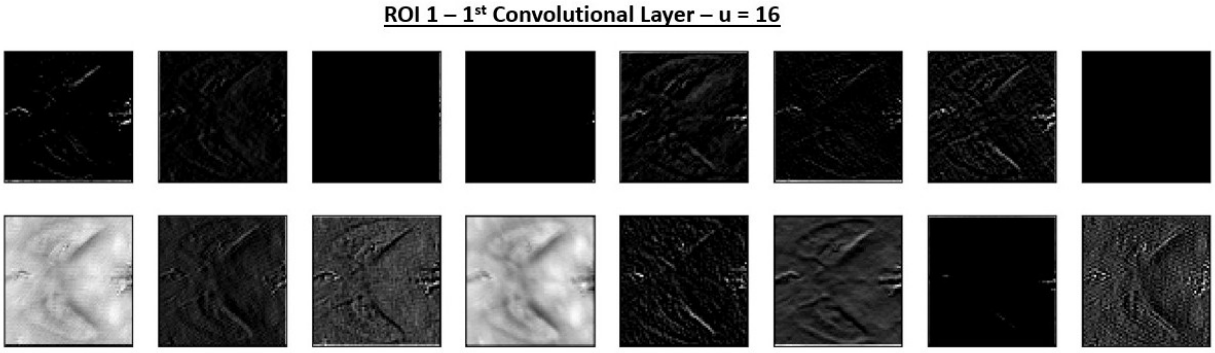


Figure 5.15: ϕ-ROI1 was processed by the first 2D convolutional layer of NET3. The model was trained with the kernel configuration (16,32,64,128,64,32,16)

function is complex ($f : \mathbb{R}^{64\cdot64} \rightarrow \mathbb{R}^{64\cdot64}$). So, despite the fact that the 32 feature maps in the first convolutional layer seem to be too many, the result using less kernels is not good as before. The model was trained using both the largest and the smallest kernel configuration, and the $Z = 75$ axial slice is proposed in Fig. 5.16. In the blue box, there is the susceptibility map coming from the first training, while in the red box there is the other one. Reducing the number of feature maps, a loss of details occurs, especially looking into the grey matter structures around the brain sides.
Anyway, this operation was necessary to perfom a 3D data training.

5.2.2 Architecture Adjustment

The kernel configuration adjustment was not the only needed change. Also, we had to set the size of the patches to (40,40,40) instead of (64,64,64), to reduce the batch size from 32 to 16 and to delete one convolutional layer in each block. It means also to eliminate the short skip connections.
The network structure is reported in Fig. 5.17. Let call it **NET1-3D**, because it is the adaptation of the NET1 model (Sec. 3.3, Fig. 3.7) to the 3D data experiment.

5.2.3 Results: Training

A dataset of 10000 (40,40,40) patches was used to train the model, 1000 patches compose the validation dataset. Six head-orientation ϕ maps were used to build the training and validation sets, the other six were used during the test stage. Each map was rotated by a random angle - with respect to the $\vec{x}$, $\vec{y}$ or $\vec{z}$ axis. From each rotated map, (x_p, y_p, z_p) patches were extracted. Rembering the consideration about the null-value pixels (Fig. 5.1), the random coordinates respectively range: $x_p \in [40, 130]$, $y_p \in [40, 140]$ and $z_p \in [40, 120]$. The database construction lasted $\sim$ 30 min, the training time $\sim$ 3 hours. The other datails are reported in Tab. 5.7.

The shape of the training and validation loss functions is reported in Fig. 5.18. As we have verified in other multiple-orientation data experiments (Sec. 4.2), the validation loss is

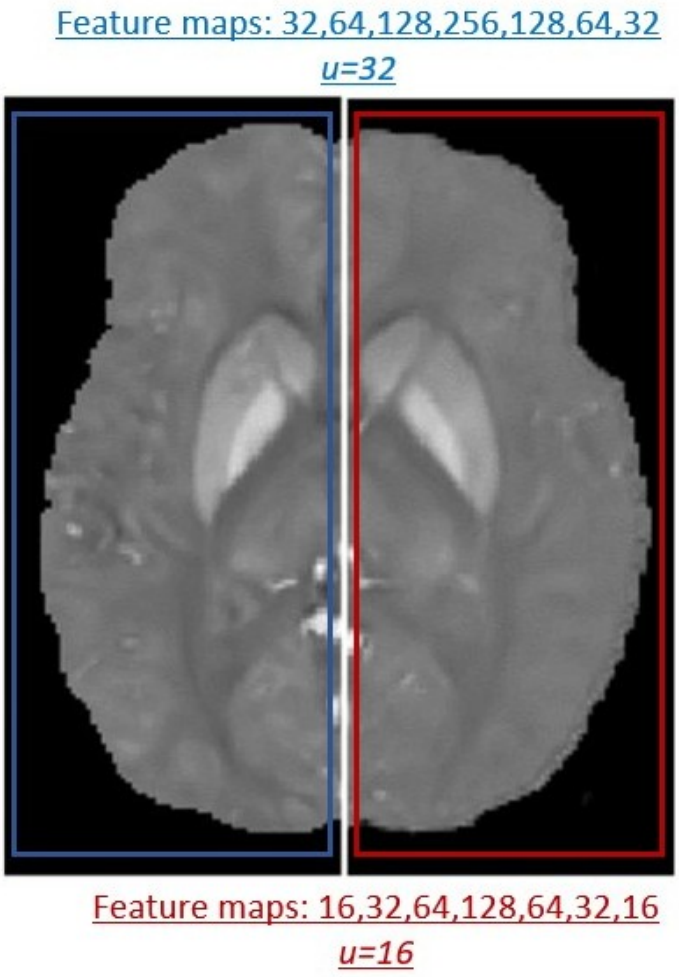


Figure 5.16: The $Z = 75$ axial slice from a susceptibility map, obtained training the NET3 model. Red box: (16,32,64,128,64,32,16) was used as a kernel configuration. Blue box: (32,64,128,256,128,64,32) was used as kernel configuration. Reducing the number of feature maps, a loss of information could be notice, especially looking into the grey matter structures around the brain sides

RMSProp optimizer, $\eta = 0.001$
BCE loss function
Batch size: 16
Epochs:100
Keras 2.2.0, Tensorflow 1.9.0 Backend, Python 3.6
GPU Titanum XP - nVdia 384, Cuda 9.0

Table 5.7: Training details and hyperparameters of the experiment introduced in Sec. 5.2 - 3D training

slightly lower than the training one. Conceptually, this is something that it is not expected. The gap is ~ 0.1 %.

Notice also that the error values are reduced with respect to the ones found in Sec 4.2: it means that, even if the structure is simpler, working with 3D data is better than using 2D data in order to perform a susceptibility map. It makes sense, because the relationship between the phase and the susceptibility is in the three dimensional space. But, the values are higher than the ones obtained in the previous section. It could worth to train the complex model (kernel configuration: (32,64,128,256,128,64,32)) with the 3D data, in order to obtain

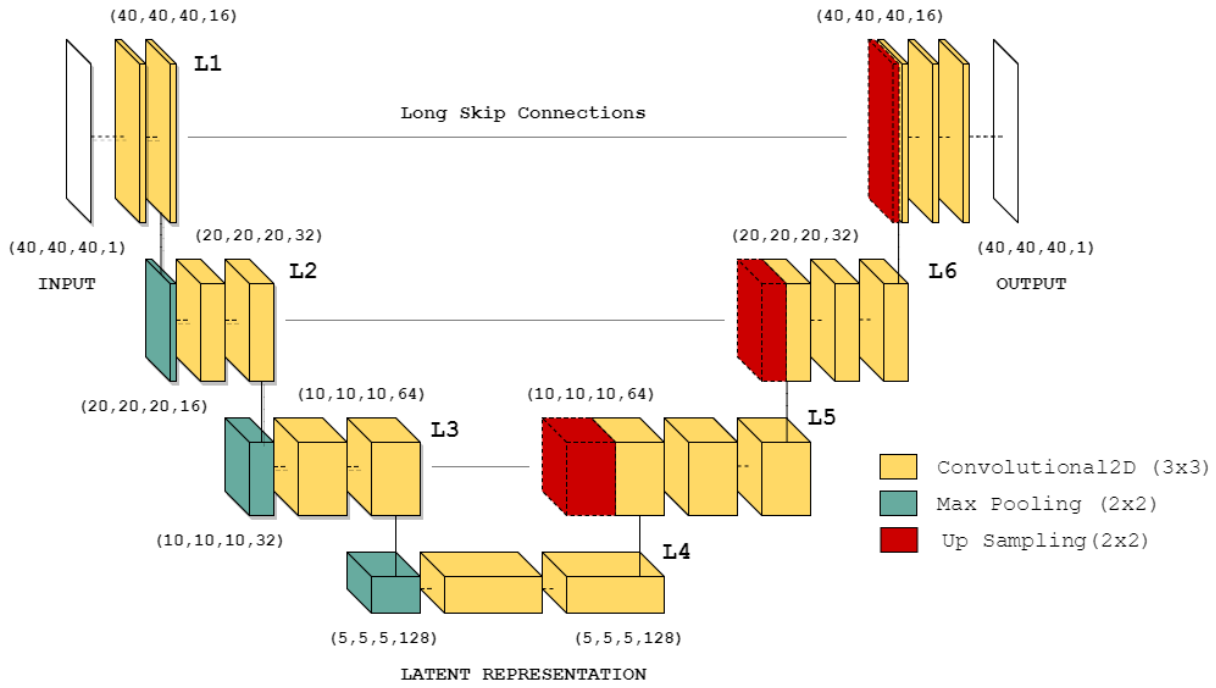


Figure 5.17: The NET1-3D model. That model was trained to perform 3D data experiments

the best result. The dimension of the patches may be also changed, in order to find the optimal size.

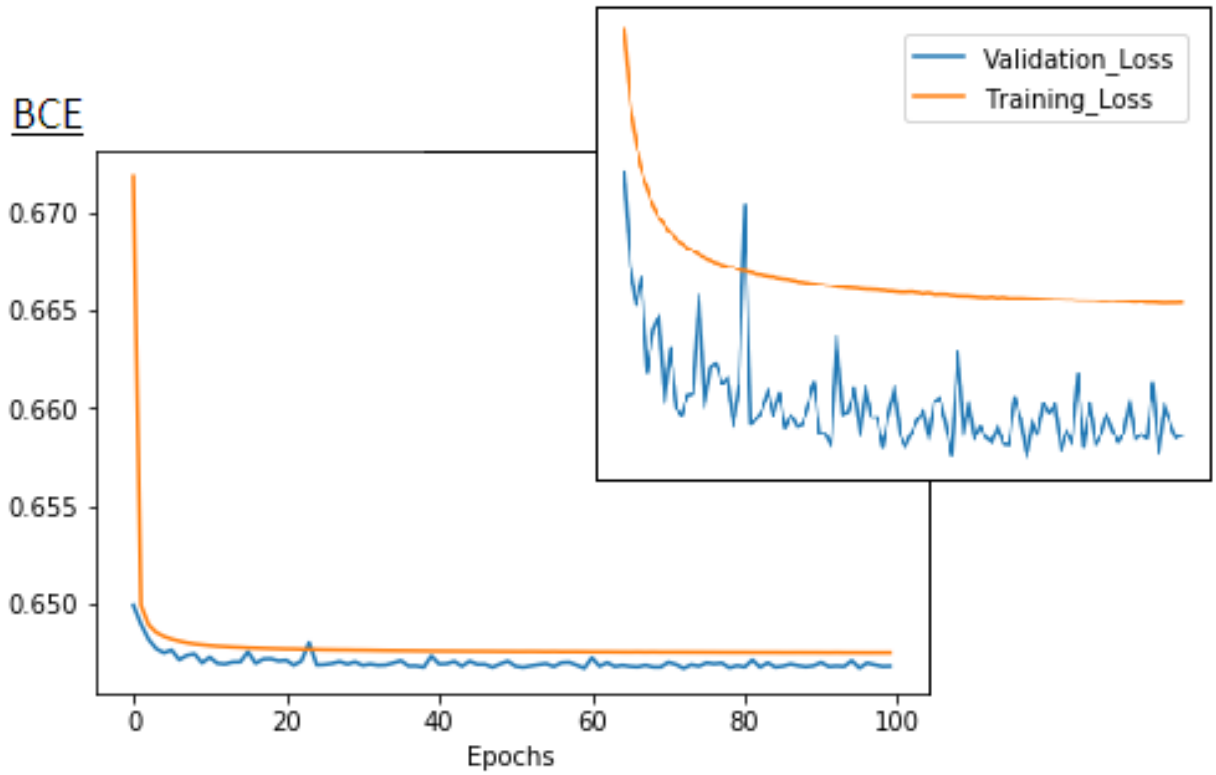


Figure 5.18: Validation and training loss functions of the experiment introduced in Sec. 5.2. 3D patches from six-head orientation maps were used to train the NET1-3D model. Details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

5.2.4 Results: Metrics, 12 Head-Orientations

We used only six head-orientation ϕ maps to train the model. The similarity parameters (RMSE, pSNR, SSIM and HFEN) were evaluated both on them and on the other six, unseen during the training stage.

The parameters refer to 3D (160,160,160) maps, and not anymore to 2D (64,64) patches. Let call **NET-i** - with $i = 0, 1, 2, ..11$ - the susceptibility maps resulting from the trained model. Their perfomance was compared to the **TKD-i** ones. The COSMOS map was used as reference.

In Fig. 5.19, the outline of the metric parameters is shown, for all the 12 head orientations. During the training, the $0^{th}, 2^{nd}, 4^{th}, 6^{th}$ and 8^{th} ϕ maps were used. RMSE, pSNR, SSIM and HFEN are respectively in the top-left, top-right, bottom-left and bottom-right panel. In red, the results related to the TKD reconstructions could be found, in blue the ones of the NET maps. The average values and the errors - evaluated as standard deviation in the 12-map group - are reported in Tab. 5.8.

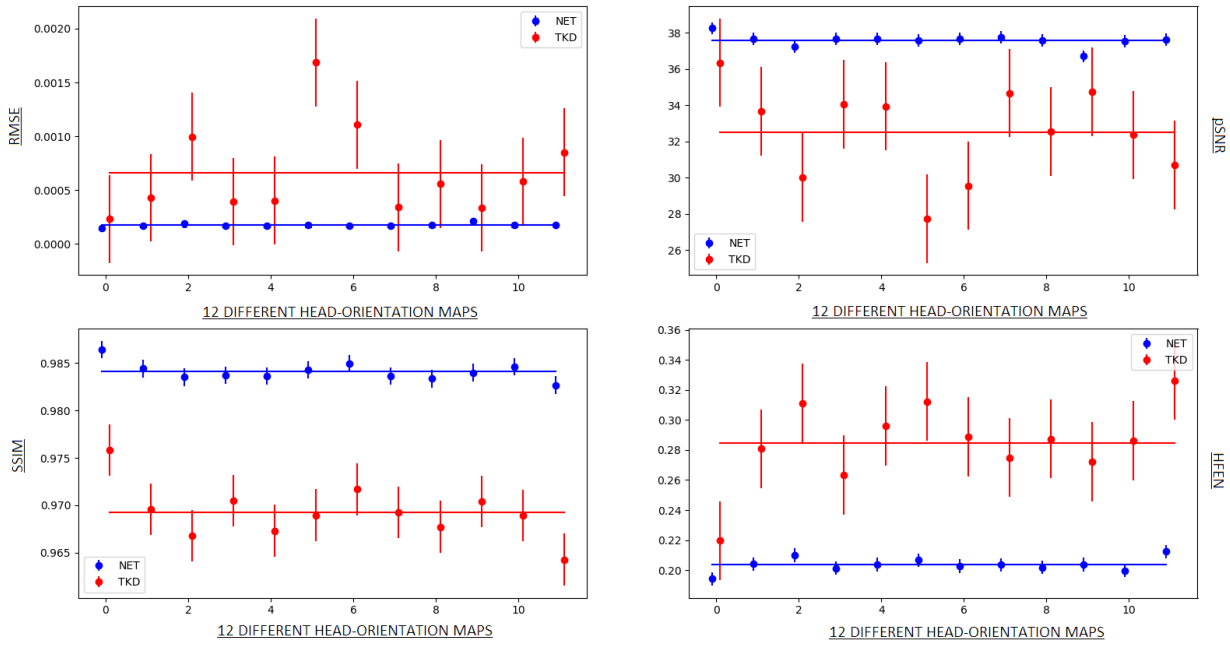


Figure 5.19: RMSE (top-left panel), pSNR (top-right panel), SSIM (bottom-left panel) and HFEN (bottom-right panel) using the NET1-3D model (Fig. 5.17) and the TKD technique ($\alpha = 0.1$). 3D patches from six-head orientation maps were used to train the model, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3. 12 3D maps with size (160,160,160) were built, for both the techniques, starting from the twelve ϕ maps available in the QSM_{2016} ([10]). Average values are in Tab. 5.8

The model response is sufficiently uniform. The relative errors are, associated with RMSE, pSNR, SSIM and HFEN, respectively equal to 5%, 0.7%, 0.1% and 2%, lower with respect to the ones of the TKD reconstructions (57%, 7%, 0.3% and 10%).

The NET maps shows better perfomance than the TKD ones, acccording to the chosen metric parameters.

	RSME	σ	pSNR	σ	SSIM	σ	HFEN	σ
NET	0.00018	0.00001	37.6	0.3	0.984	0.001	0.203	0.004
TKD	0.0007	0.0004	32.5	2.4	0.969	0.003	0.28	0.03

Table 5.8: Average values of the metric parameters along the 12 susceptibility maps, obtained using the NET1-3D and the TKD ($\alpha = 0.1$) techniques. 3D patches from six-head orientation maps were used to train the network, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1 and 5.2.2.12 3D maps with size (160,160,160) were built, for both the techniques, starting from the twelve ϕ maps available in the QSM_{2016} ([10]). In Fig. 5.19, the outline of RMSE, pSNR, SSIM and HFEN along the considered directions

5.2.5 Results: Susceptibility Reconstruction

We have just verified that the return of the model is, according to the metric parameters, uniform, evaluating them for all the 12 NET-i maps. Remind that only six head-orientation maps were used during the training - i.e. $i = 0, 2, 4, 6, 8$ -. It means that the model is able to generalize on the direction of the head in the scan.
Because of the fact that the reconstructions are similar each other, only the NET-0 map is shown as an example. NET-0, TKD-0 and the COSMOS maps were compared in Figs. 5.20 (sagittal slices), 5.21 (coronal slices), 5.22 (axial slices), 5.23 ($X = 60$ sagittal slice), 5.24 ($Y = 80$ coronal slice) and 5.25 ($Z = 70$ axial slice).

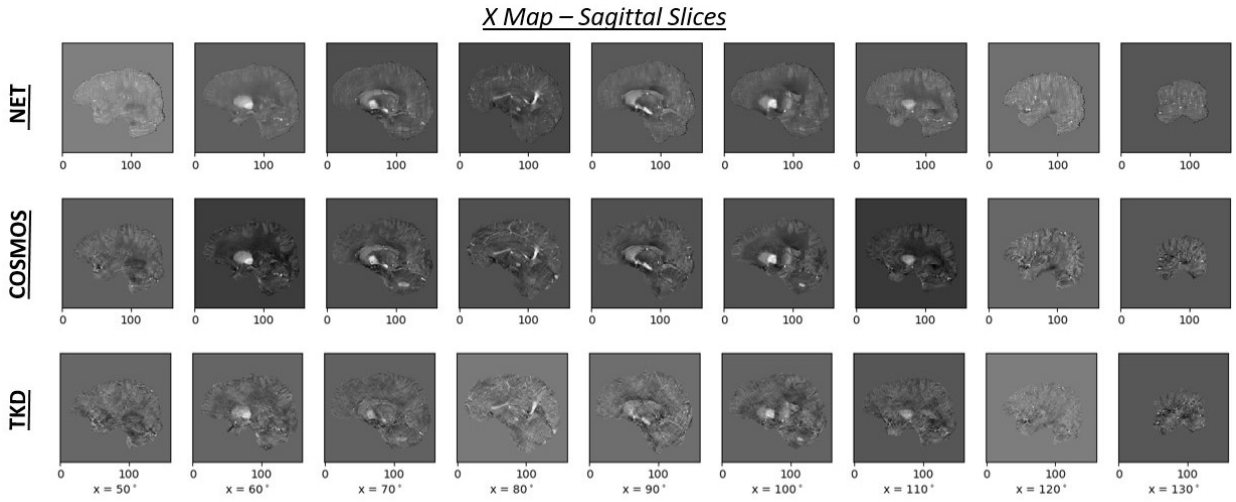


Figure 5.20: Sagittal slices from NET-0 (first row), COSMOS (second row) and TKD-0 ($\alpha = 0.1$, third row) susceptibility maps. NET-0 and TKD-0 were built using the 0^{th} ϕ map from [10]. NET-0 was obtained with the trained NET1-3D model. It was trained using 3D patches from six head-orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

The NET-0 reconstruction is better than the TKD-0 one, less noisy and closer to the COSMOS map, used as a gold-standard reference.
The reported results are not as good as the ones obtained with the 2D experiments (Sec. 4.1, 4.2, 5.1). This is not due to the use of 3D data, but to the semplification of the network structure, necessary considering the available memory.

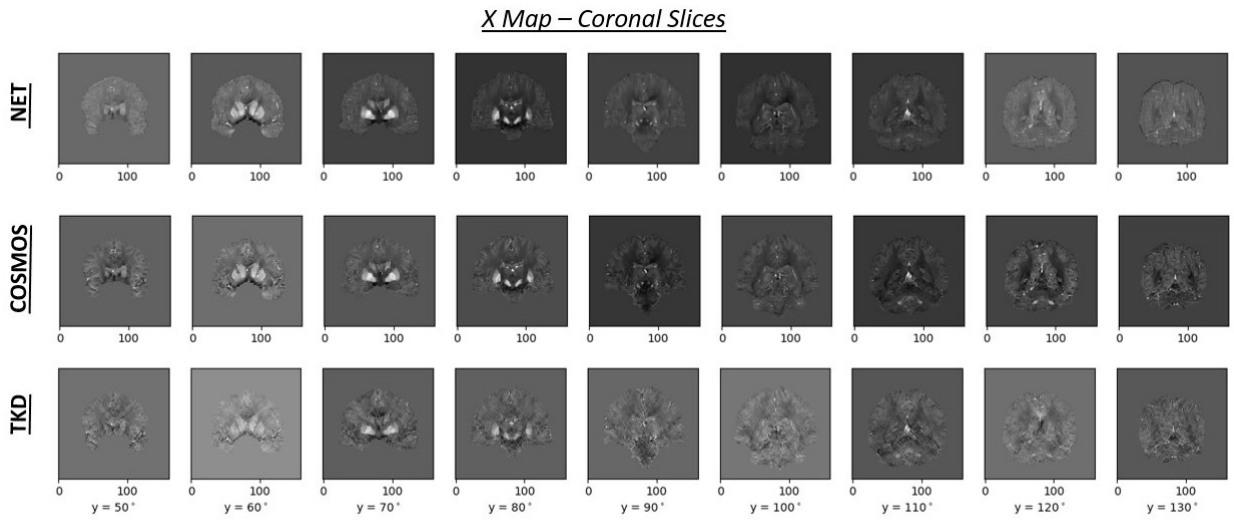


Figure 5.21: Coronal slices from NET-0 (first row), COSMOS (second row) and TKD-0 ($\alpha = 0.1$, third row) susceptibility maps. NET-0 and TKD-0 were built using the 0^{th} ϕ map from [10]. NET-0 was obtained with the trained NET1-3D model. It was trained using 3D patches from six head-orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

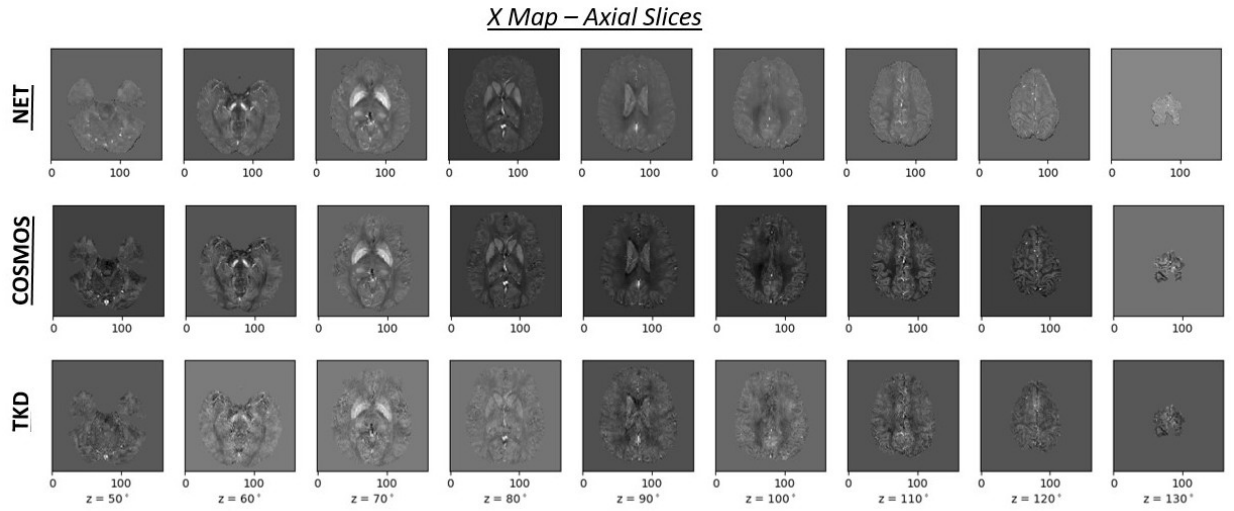


Figure 5.22: Axial slices from NET-0 (first row), COSMOS (second row) and TKD-0 ($\alpha = 0.1$, third row) susceptibility maps. NET-0 and TKD-0 were built using the 0^{th} ϕ map from [10]. NET-0 was obtained with the trained NET1-3D model. It was trained using 3D patches from six head-orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

5.2.6 Results: Intensity and Contrast Analysis

The outcomes related to the intensity and contrast analysis are also reported. The ones of the NET-0 and TKD-0 maps are introduced first, before the NET-i and TKD-i reconstructions - $i = 0, 1, .., 11$. The software ImageJ was used. The images were transformed in 8-bit type maps, so the dynamic range is [0,255]. The mean intensity values are in arbitrary units.

Single-Head Orientation: NET and TKD

In Fig. 5.26, the results regarding the 0^{th} direction are summarized.
The left panel shows the mean intensity values of the PUT, GP, and CAU in ROI1, the

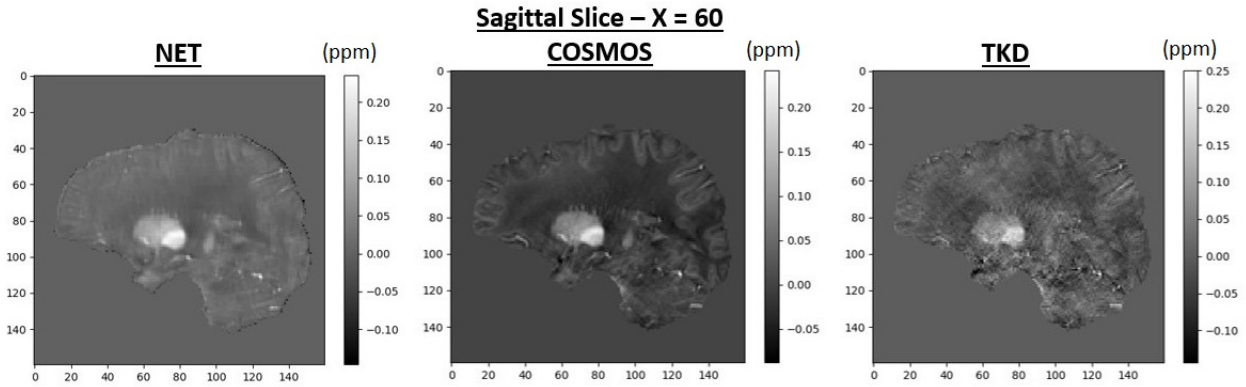


Figure 5.23: Sagittal slice ($X = 60$) from NET-0 (first row), COSMOS (second row) and TKD-0 ($\alpha = 0.1$, third row) susceptibility maps. NET-0 and TKD-0 were built using the 0^{th} ϕ map from [10]. NET-0 was obtained with the trained NET1-3D model. It was trained using 3D patches from six head-orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

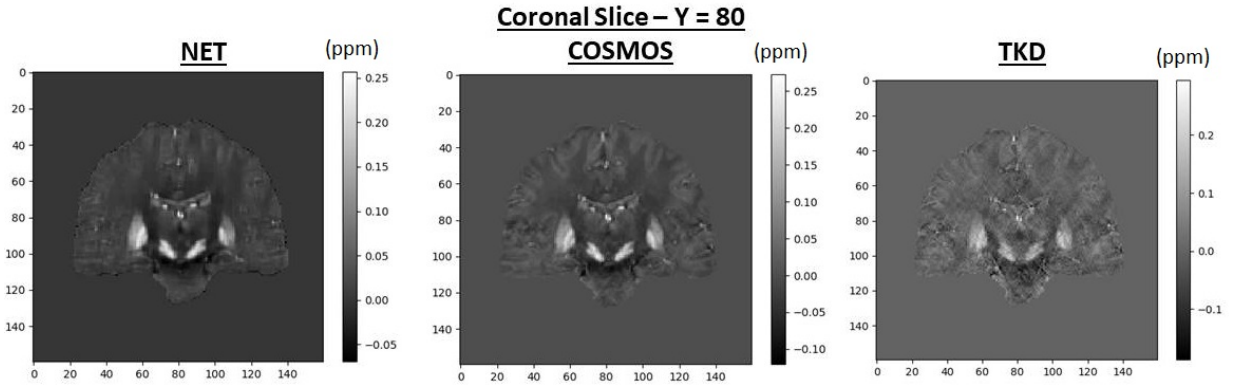


Figure 5.24: Coronal slice ($Y = 80$) from NET-0 (first row), COSMOS (second row) and TKD-0 ($\alpha = 0.1$, third row) susceptibility maps. NET-0 and TKD-0 were built using the 0^{th} ϕ map from [10]. NET-0 was obtained with the trained NET1-3D model. It was trained using 3D patches from six head-orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

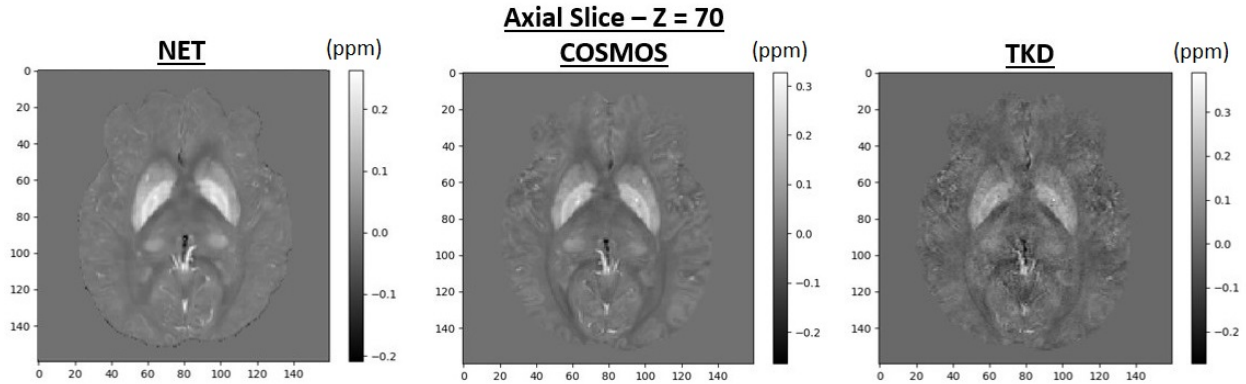


Figure 5.25: Axial slice ($Z = 70$) from NET-0 (first row), COSMOS (second row) and TKD-0 ($\alpha = 0.1$, third row) susceptibility maps. NET-0 and TKD-0 were built using the 0^{th} ϕ map from [10]. NET-0 was obtained with the trained NET1-3D model. It was trained using 3D patches from six head-orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

RN and SN in ROI2, and the background (B(1) and B(2)). The perfomances of the three methods are compared. The same values may be found in Tab. 5.9, both mean and standard deviation ones. The agreement between the NET-0 map and the COSMOS one is not perfect: they are not consistent each other in four points. But, in three of them, the intensity of the

model is higher than the COSMOS one.
The TKD reconstruction shows instead significant differences with respect to the other two, and it reports higher error values.

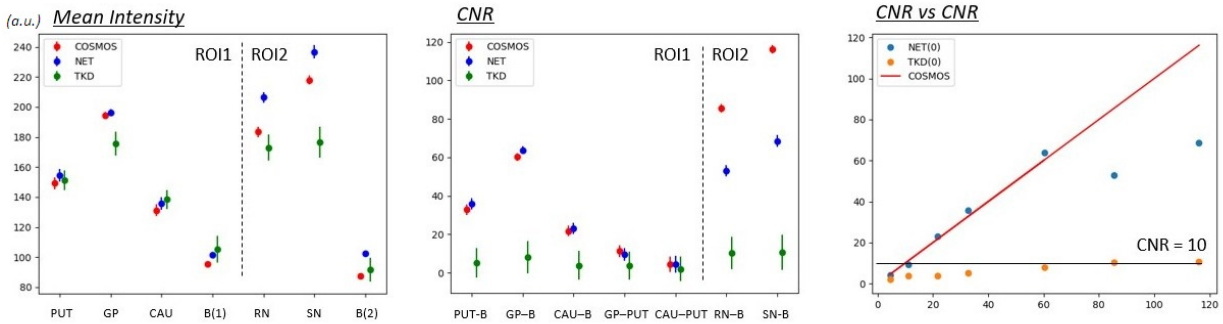


Figure 5.26: Left panel: mean intensity values of the COSMOS, TKD-0 ($\alpha = 0.1$) and NET-0 three dimensional susceptibility maps. The areas in the ROI1 and ROI2 were examined. The TKD-0 and NET-0 maps were built using the 0^{th} ϕ map. The correspondent values are reported in Tab. 5.9. Central and right panels: CNR of the COSMOS, TKD-0 ($\alpha = 0.1$) and NET-0 three dimensional susceptibility maps. The correspondent values are in Tab. 5.10

	$(a.u.)$	PUT	GP	CAU	B(1)	RN	SN	B(2)
COSMOS	μ	149.2	194.2	131.2	95.6	183.5	218.0	87.3
	σ	4.1	2.4	3.9	1.6	3.5	3.1	1.1
TKD(0)	μ	151.0	175.7	138.4	105.2	173.0	176.4	91.5
	σ	6.5	8.1	6.4	8.8	8.8	10.2	8.0
NET(0)	μ	154.5	196.1	135.7	101.3	206.3	236.7	102.3
	σ	4.4	2.3	4.3	1.5	3.7	4.3	2.0

Table 5.9: Mean intensity and standard deviation values of the COSMOS, TKD-0 ($\alpha = 0.1$) and NET-0 three dimensional susceptibility maps (Fig. 5.26, left panel). The areas in the ROI1 and ROI2 were examined. The TKD-0 and NET-0 maps were built using the 0^{th} ϕ map. NET-0 was built with the NET1-3D trained model. It was trained using 3D patches from six-head orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

In Fig. 5.26, central and right panels, the CNR results are reported. The correspondent values are in Tab. 5.10. There is a good agreement between the NET-0 and the COSMOS susceptibility maps in all the analyzed points in ROI1. The TKD map assumes lower values than the other two. In the ROI2, the NET-0 map assumes contrast values lower than the COSMOS one, but higher than the TKD reconstruction.

Multiple-Head Orientations: NET

In Fig. 5.27, the intensity analysis related to the multiple-orientation NET-i maps is reported. The correspondent values are in Tab. 5.11.
In ROI1, the NET-i maps values are consistent with the COSMOS ones. In ROI2, the NET-i values are consistent each other, while the COSMOS maps assumes lower values.

		PUT-B	GP-B	CAU-B	GP-PUT	CAU-PUT	RN-B	SN-B
COSMOS	CNR	32.7	60.3	21.8	11.2	4.5	85.5	116.2
	ϵ	2.8	2.0	2.7	3.2	3.9	2.3	2.1
TKD-0	CNR	5.2	8.0	3.8	3.8	1.9	10.2	10.6
	ϵ	7.7	8.4	7.6	7.3	6.5	8.4	9.1
NET-0	CNR	35.8	63.8	23.2	9.5	4.3	53.0	68.5
	ϵ	2.9	1.9	2.9	3.4	4.4	2.8	3.1

Table 5.10: CNR values of the COSMOS, TKD-0 ($\alpha = 0.1$) and NET-0 three dimensional susceptibility maps (Fig. 5.26, central and right panels). The areas in the ROI1 and ROI2 were examined. The TKD-0 and NET-0 maps were built using the 0^{th} ϕ map. NET-0 was built with the NET1-3D trained model. It was trained using 3D patches from six-head orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

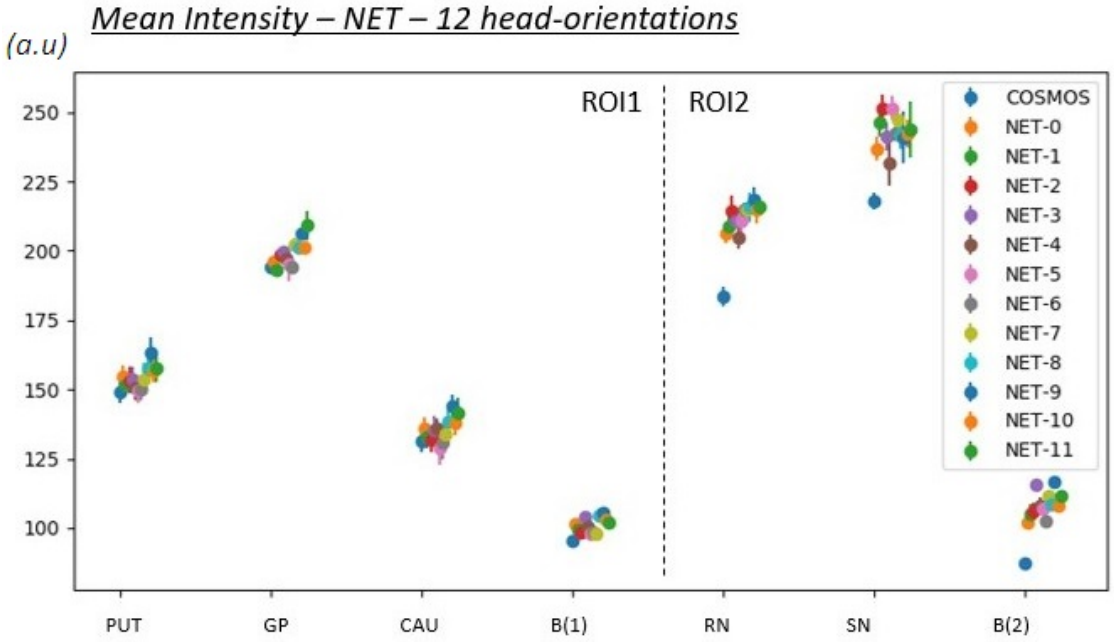


Figure 5.27: Mean intensity values of the NET$-i$ maps, obtained changing the head orientation of the input ϕ map. The areas in the ROI1 and ROI2 were examined. The correspondent values are reported in Tab. 5.11

The contrast analysis is reported in Figs. 5.28, 5.31 and in Tab. 5.12. The values from the different reconstructions are not consistent each other when the white matter is considered as background. The dispersion seems to be bigger than before. We have already explained that the worse perfomance in this 3D data experiment are probably due to the semplification of the architecture of the used model.

Multiple-Head Orientations: TKD

The same analysis was carried out for the susceptibility maps obtained using the TKD technique. Each TKD$-i$ map was obtained using the i^{th} ϕ map as input.
The intensity analysis is reported in Fig. 5.29 and in Tab. 5.13. The values assumed by the different reconstructions are more spread out with respect to the NET-i-related results (cf. Fig. 5.27). Notice also that the error values are higher.

	$(a.u.)$	PUT	GP	CAU	B(1)	RN	SN	B(2)
NET-1	μ	151.8	193.2	132.7	99.6	208.8	246.3	105.3
	σ	3.3	1.1	3.9	3.2	4.9	5.1	3.3
NET-2	μ	153.4	198.8	131.9	98.7	214.2	251.2	106.0
	σ	4.9	2.2	4.5	2.9	5.9	5.1	3.0
NET-3	μ	153.5	199.6	135.6	104.0	211.0	241.3	115.5
	σ	4.5	2.2	4.9	1.3	2.4	5.2	1.9
NET-4	μ	150.4	197.4	135.8	100.5	204.7	231.5	108.3
	σ	4.6	4.0	3.9	1.5	3.8	8.2	3.0
NET-5	μ	149.6	195.0	128.8	98.1	210.7	251.5	107.3
	σ	4.3	5.7	5.9	1.9	3.8	4.4	2.6
NET-6	μ	150.3	194.0	130.9	98.3	214.8	242.2	102.5
	σ	4.3	2.2	5.8	1.8	2.6	2.7	1.9
NET-7	μ	153.8	202.4	134.1	98.1	214.7	247.3	111.5
	σ	6.1	2.5	5.2	1.8	2.8	3.9	1.7
NET-8	μ	157.7	201.2	138.7	104.8	215.7	241.7	108.8
	σ	4.9	2.0	3.8	2.5	5.3	5.0	1.5
NET-9	μ	163.1	206.3	144.2	105.8	218.3	240.9	116.8
	σ	5.8	1.9	3.8	1.2	4.6	9.5	0.9
NET-10	μ	156.7	201.2	138.2	103.0	214.8	242.1	108.3
	σ	4.1	1.4	4.9	1.5	5.0	5.1	2.1
NET-11	μ	158.0	209.1	141.3	102.1	215.7	243.7	111.8
	σ	5.5	5.4	6.0	1.1	2.1	10.1	0.9

Table 5.11: Mean intensity and standard deviation values of the NET$-i$ maps, obtained changing the head orientation of the input ϕ map (Fig. 5.27). The areas in the ROI1 and ROI2 were examined. The NET-i maps were built with the NET1-3D trained model. It was trained using 3D patches from six-head orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

The contrast analysis is reported in Fig. 5.30 and in Tab. 5.14. The CNR values of the different reconstructions are consistent each other because of the high error, and they are lower than the COSMOS ones. A clearer visualization of it could be found in the scatter plot in Fig. 5.31, bottom panel.

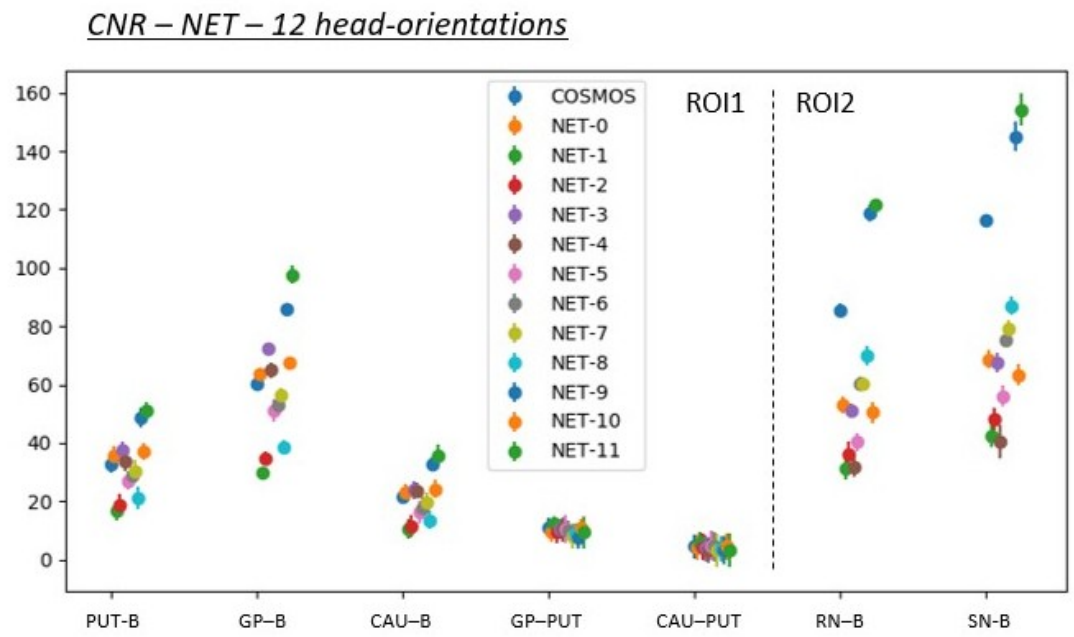


Figure 5.28: CNR values of the NET$-i$ maps, obtained changing the head orientation of the input ϕ map. The areas in the ROI1 and ROI2 were examined. The correspondent values are reported in Tab. 5.12

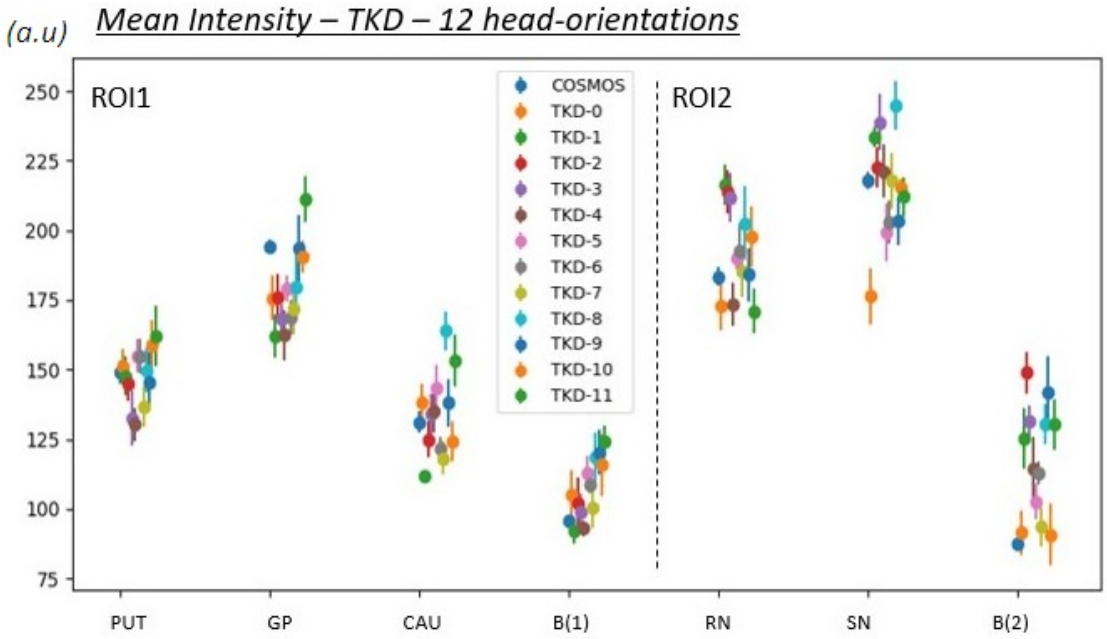


Figure 5.29: Mean intensity values of the TKD$-i$ maps ($\alpha = 0.1$), obtained changing the head orientation of the input ϕ map. The areas in the ROI1 and ROI2 were examined. The correspondent values are reported in Tab. 5.13

		PUT-B	GP-B	CAU-B	GP-PUT	CAU-PUT	RN-B	SN-B
NET-1	CNR	16.6	29.7	10.5	12.5	5.8	31.3	42.6
	ϵ	3.2	2.1	3.5	2.2	3.6	4.1	4.2
NET-2	CNR	18.9	34.6	11.5	9.3	4.4	35.9	48.2
	ϵ	3.9	2.5	3.7	3.5	4.7	4.7	4.0
NET-3	CNR	37.6	72.5	24.0	10.3	4.0	51.3	67.5
	ϵ	2.9	1.7	3.1	3.3	4.7	2.1	3.5
NET-4	CNR	33.5	65.0	23.7	10.3	3.2	31.7	40.5
	ϵ	3.0	2.7	2.7	4.3	4.3	3.4	5.6
NET-5	CNR	27.1	50.9	16.1	10.5	4.8	40.3	56.1
	ϵ	3.1	3.8	3.9	5.0	5.1	3.2	3.5
NET-6	CNR	28.7	52.9	18.0	10.2	4.5	60.3	75.0
	ϵ	3.1	2.0	3.8	3.2	5.1	2.2	2.3
NET-7	CNR	30.2	56.6	19.5	8.0	3.2	60.2	79.3
	ϵ	4.0	2.2	3.5	4.3	5.7	2.2	2.8
NET-8	CNR	21.1	38.5	13.5	8.9	3.9	70.0	87.0
	ϵ	3.7	2.3	3.1	3.5	4.3	3.4	3.3
NET-9	CNR	48.8	85.6	32.7	7.5	3.3	118.7	145.1
	ϵ	3.5	1.6	2.5	3.9	4.8	2.7	5.2
NET-10	CNR	37.0	67.6	24.2	10.9	4.5	50.4	63.4
	ϵ	2.8	1.4	3.2	2.7	4.5	3.6	3.6
NET-11	CNR	50.9	97.6	35.7	9.3	3.0	121.4	154.1
	ϵ	3.3	3.2	3.6	5.4	5.8	1.5	5.5

Table 5.12: CNR values of the NET$-i$ maps, obtained changing the head orientation of the input ϕ map (Fig. 5.28). The areas in the ROI1 and ROI2 were examined. The NET-i maps were built with the NET1-3D trained model. It was trained using 3D patches from six-head orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

	$(a.u.)$	PUT	GP	CAU	B(1)	RN	SN	B(2)
TKD-1	μ	147.8	162.0	111.7	92.2	216.3	233.5	125.3
	σ	7.0	1.9	2.3	4.8	7.2	3.6	10.7
TKD-2	μ	145.0	176.1	124.9	101.9	213.7	222.5	149.0
	σ	6.2	7.9	6.6	9.4	7.7	7.4	7.4
TKD-3	μ	132.6	168.1	133.9	98.7	211.7	238.8	131.5
	σ	10.0	5.3	7.3	6.8	8.9	10.1	5.9
TKD-4	μ	130.3	162.3	135.1	33.0	173.4	221.2	114.5
	σ	5.8	8.9	7.8	3.0	7.5	9.6	11.4
TKD-5	μ	155.0	179.2	143.4	122.8	190.0	199.1	102.8
	σ	6.0	4.7	8.3	6.4	4.0	10.3	6.5
TKD-6	μ	154.7	168.5	121.9	108.5	192.8	202.8	113.0
	σ	6.1	6.2	4.0	2.9	9.6	7.7	4.1
TKD-7	μ	136.6	172.0	118.2	100.2	185.2	217.9	93.5
	σ	7.1	9.6	6.0	7.1	9.3	10.1	7.0
TKD-8	μ	149.8	179.9	163.9	118.3	202.5	244.9	130.5
	σ	8.1	11.5	6.9	8.9	13.1	8.6	7.3
TKD-9	μ	145.7	193.6	138.0	120.7	184.1	203.5	142.0
	σ	10.7	12.0	8.4	8.1	9.8	9.0	12.8
TKD-10	μ	159.2	190.4	124.3	115.9	197.9	215.6	90.8
	σ	8.7	5.7	7.2	11.1	10.8	3.0	11.2
TKD-11	μ	161.9	211.1	153.5	124.4	171.1	212.2	130.3
	σ	10.9	8.1	9.3	5.8	8.0	7.1	9.1

Table 5.13: Mean intensity and standard deviation values of the TKD$-i$ maps ($\alpha = 0.1$), obtained changing the head orientation of the input ϕ map (Fig. 5.29). The areas in the ROI1 and ROI2 were examined

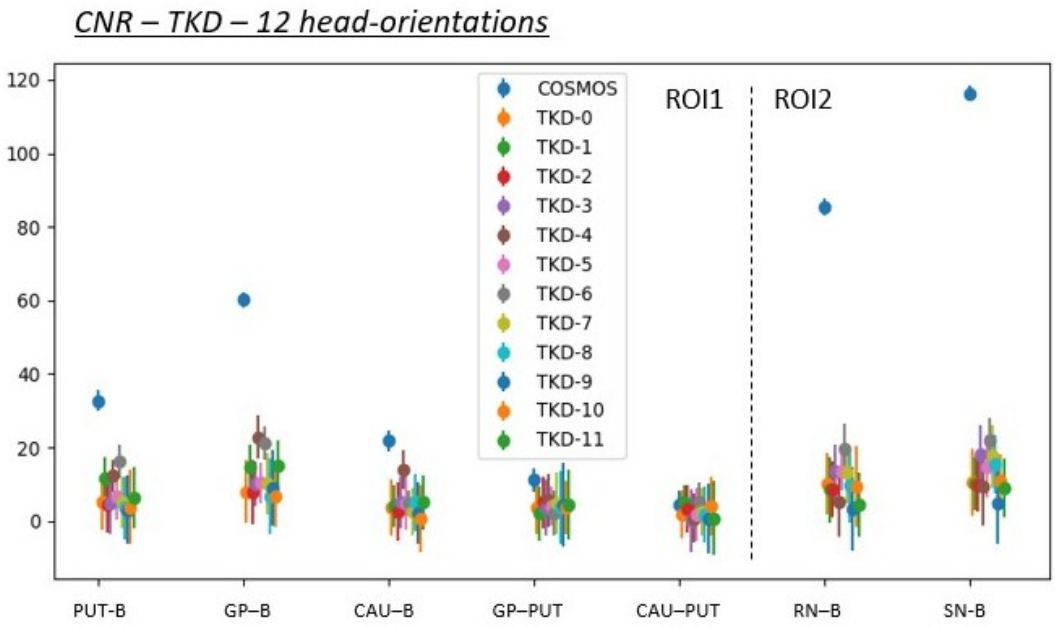


Figure 5.30: CNR values of the TKD$-i$ maps ($\alpha = 0.1$), obtained changing the head orientation of the input ϕ map. The areas in the ROI1 and ROI2 were examined. The correspondent values are reported in Tab. 5.14

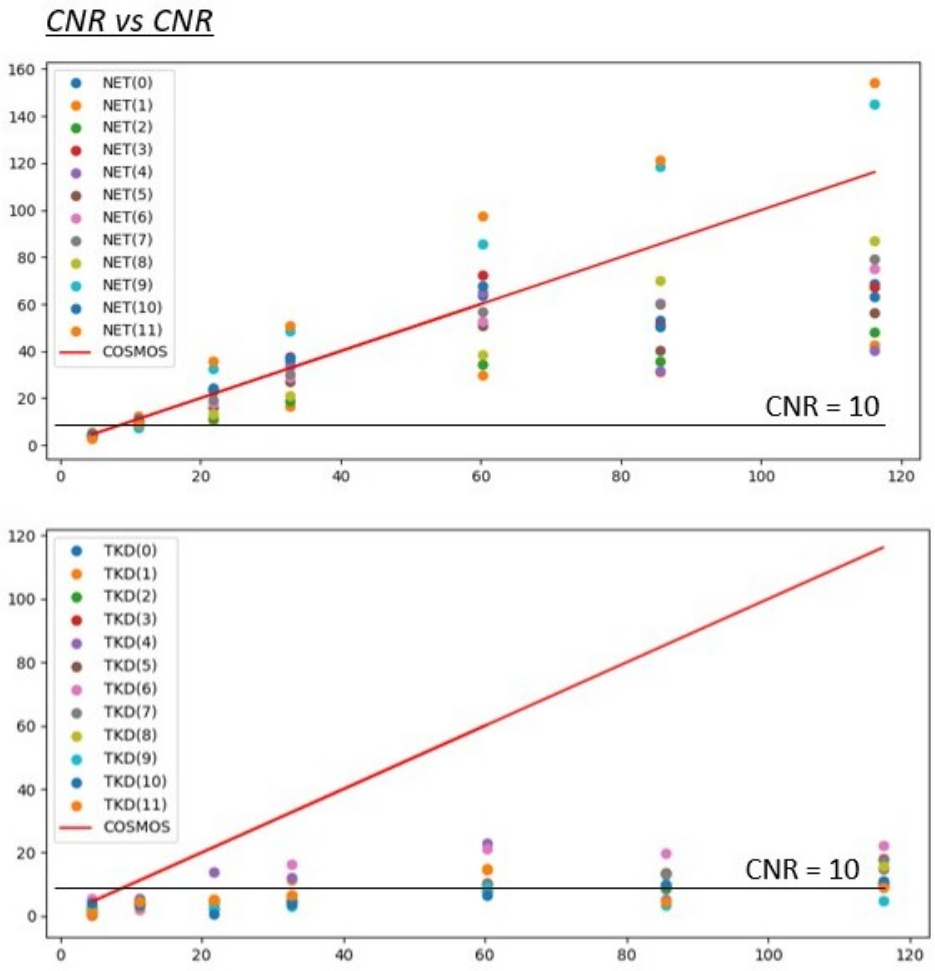


Figure 5.31: Scatter plots CNR-NET values (top panel) and CNR-TKD values (bottom panel) vs CNR-COSMOS values. The same results are reported respectively in Fig. 5.28 and 5.30

		PUT-B	GP-B	CAU-B	GP-PUT	CAU-PUT	RN-B	SN-B
TKD-1	CNR	11.6	14.6	4.1	2.0	5.1	8.5	10.1
	ϵ	5.9	6.3	5.6	7.4	4.7	8.9	7.2
TKD-2	CNR	4.6	7.9	2.5	5.1	3.2	8.7	10.0
	ϵ	7.8	8.7	8.0	7.1	6.4	7.5	7.4
TKD-3	CNR	4.9	10.1	5.1	3.6	0.1	13.6	18.2
	ϵ	8.4	6.1	7.1	7.6	8.6	7.4	8.0
TKD-4	CNR	12.3	22.9	13.9	5.5	0.8	5.2	9.3
	ϵ	4.4	6.0	5.4	7.3	6.8	9.5	10.5
TKD-5	CNR	6.6	10.4	4.8	4.0	1.9	13.5	14.9
	ϵ	6.2	5.5	7.3	5.4	7.2	5.3	8.4
TKD-6	CNR	16.2	21.1	4.7	2.3	5.4	19.6	22.1
	ϵ	4.5	4.5	3.4	6.1	5.0	6.8	5.9
TKD-7	CNR	5.1	10.1	2.5	5.0	2.6	13.0	17.7
	ϵ	7.1	8.3	6.6	8.3	6.6	8.2	8.5
TKD-8	CNR	3.5	6.9	5.1	3.7	1.7	9.8	15.6
	ϵ	8.5	10.2	7.9	9.8	7.5	10.2	8.0
TKD-9	CNR	3.1	9.0	2.2	4.5	0.7	3.3	4.8
	ϵ	9.4	10.1	8.3	11.4	9.5	11.3	10.9
TKD-10	CNR	3.9	6.7	0.8	3.6	4.0	9.5	11.1
	ϵ	9.9	8.4	9.1	7.2	7.9	11.0	7.1
TKD-11	CNR	6.5	15.1	5.0	4.5	0.8	4.5	9.0
	ϵ	8.3	6.9	7.5	9.5	10.1	8.6	8.1

Table 5.14: CNR values of the TKD$-i$ maps ($\alpha = 0.1$), obtained changing the head orientation of the input ϕ map (Fig. 5.30). The areas in the ROI1 and ROI2 were examined

5.2.7 Results: FFT and K-space

We analyzed the k-space of the COSMOS, TKD-0 and NET-0 susceptibility maps. The NET-0 and TKD-0 reconstructions use the 0^{th} ϕ map as input.
The results are reported in Figs. 5.32, 5.33 and 5.34. Sagittal, coronal and axial sections respectively figure. In Fig. 5.35, the difference matrices between the COSMOS and NET-0 reconstructions (first block) and between the COSMOS and TKD-0 reconstructions (second block) are shown, with respect to the three k_x, k_y and k_z directions.

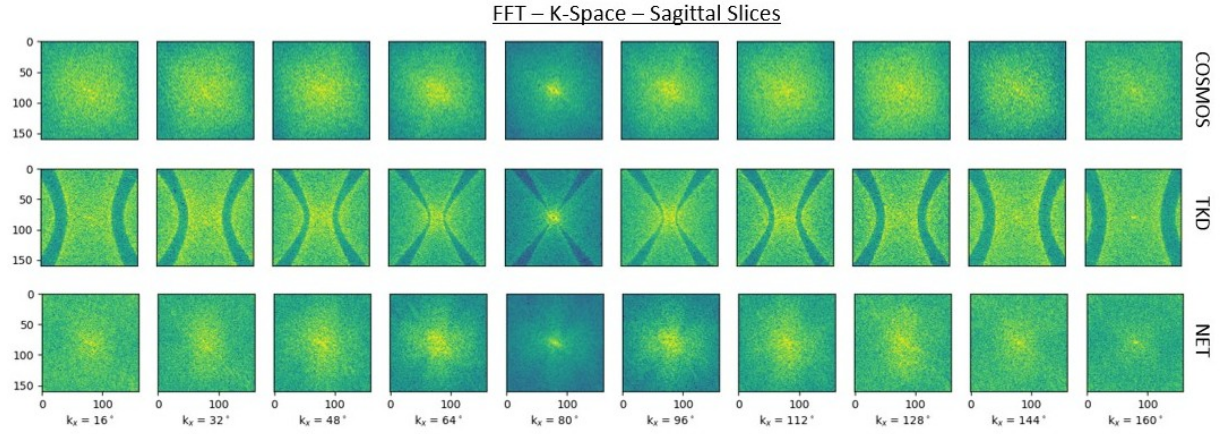


Figure 5.32: Sagittal slices from the 3D FFT of the COSMOS (first row), TKD-0 ($\alpha = 0.1$, second row) and NET-0 (third row) susceptibility maps. The NET-0 and TKD-0 reconstructions use the 0^{th} ϕ map as input. The NET-0 map was built with the trained NET1-3D model. It was trained using 3D patches from six-head orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

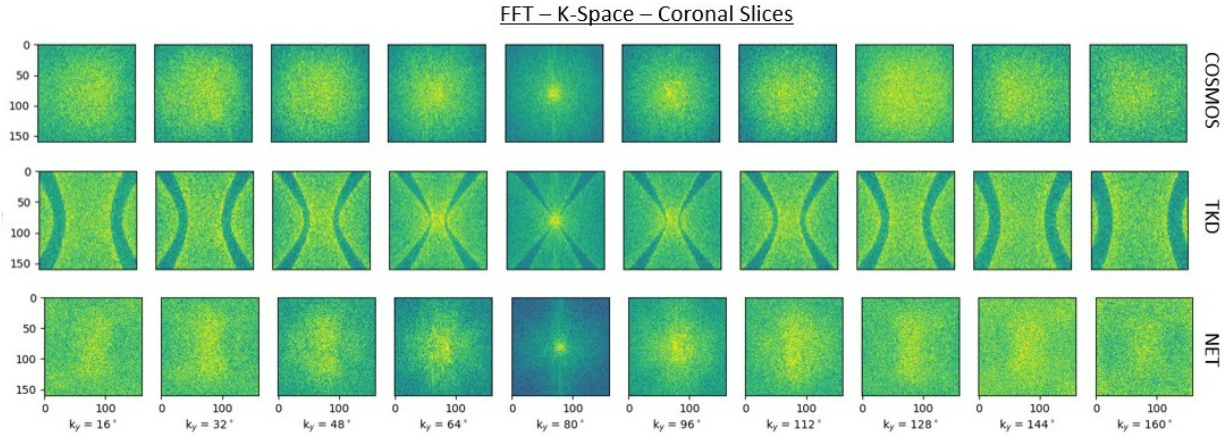


Figure 5.33: Coronal slices from the 3D FFT of the COSMOS (first row), TKD-0 ($\alpha = 0.1$, second row) and NET-0 (third row) susceptibility maps. The NET-0 and TKD-0 reconstructions use the 0^{th} ϕ map as input. The NET-0 map was built with the trained NET1-3D model. It was trained using 3D patches from six-head orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

The double cone in the TKD k-space is clearly visible, while it is not in the COSMOS and NET k-spaces. However, comparing to the previous results (Sec. 4.3), there is not anymore a good agreement between the NET and COSMOS frequency spectrum, especially into the low-frequency region - i.e. central area of the k-space. It is clearly visible looking into the difference matrices (Fig. 5.35).

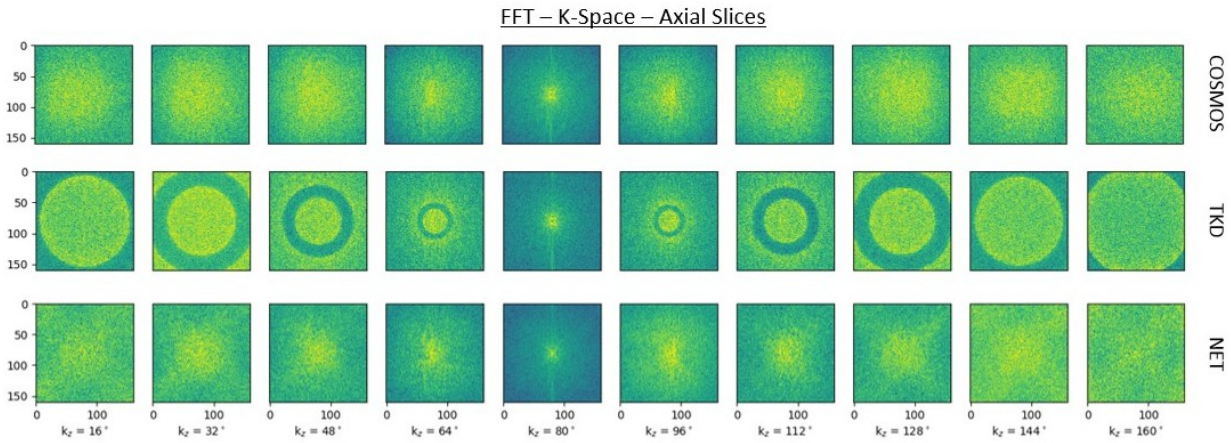


Figure 5.34: Axial slices from the 3D FFT of the COSMOS (first row), TKD-0 ($\alpha = 0.1$, second row) and NET-0 (third row) susceptibility maps. The NET-0 and TKD-0 reconstructions use the 0^{th} ϕ map as input. The NET-0 map was built with the trained NET1-3D model. It was trained using 3D patches from six-head orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

5.2.8 Conclusions

- We decided to perfom a new training, this time using 3D patches, from the spatial domain of multiple-orientation ϕ map. Because of the computational effort implied working with 3D images, we had to semplify the architecture of the network, to reduce the size of the patches and to change some hyperparameters setting. The NET1-3D model was then trained (Sec. 5.2.2).

- Six ϕ maps, with different head-orientation, were used to train the model. The remaining six were used to test the generalization capability of the network. Metric parameters were evaluated for all the twelve phase maps, and the results are reported in Sec. 5.2.4. In according to them, the model shows a uniform response whatever is the head direction of the input map. The relative errors of all the parameters are reduced than the previous 2D data experiments.
  The similarity parameters values are better than the ones obtained with the TKD method, which shows worse perfomance and bigger dispersion changing the orientation of the head.
  For both the TKD and NET reconstructions, the COSMOS suceptibility map was used as reference.

- Intensity and contrast values were observed for both the 0^{th} head-direction map and the multiple-orientation maps. The areas in ROI1 and ROI2 were analyzed - i.e. the putamen, globus pallidus, caudate, red nucleus, substantia nigra and white-matter background (general white matter in B(1) and white matter tracts in B(2)). The results are presented in Sec. 5.2.6.
  Observing the 0^{th} head-direction, we have noticed that there is, also in 3D data experiments, a good agreement between the COSMOS and NET reconstructions, that is not that good for the TKD one.

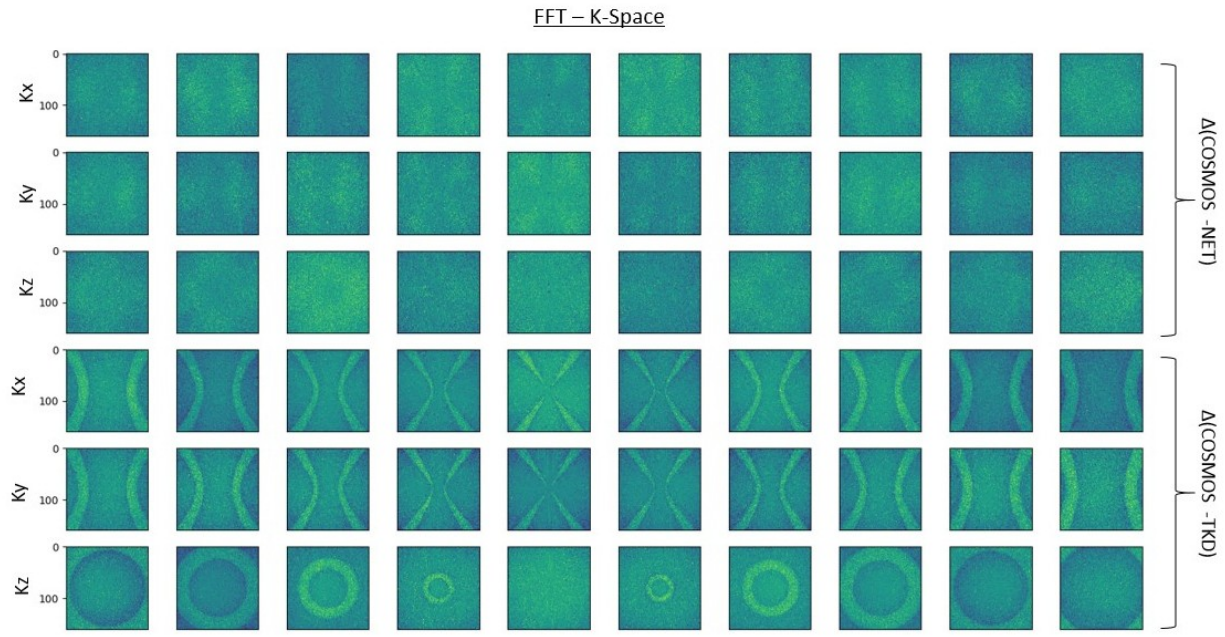


Figure 5.35: First block: difference matrices between the 3D FFT of the COSMOS and NET-0 maps. Sagittal (X), coronal (Y) and axial (Z) sections are reported. Second block: difference matrices between the 3D FFT of the COSMOS and TKD-0 ($\alpha = 0.1$) maps. Sagittal (X), coronal (Y) and axial (Z) sections are reported. The NET-0 and TKD-0 reconstructions use the $0th$ ϕ map as input. The NET-0 map was built with the NET1-3D trained model. It was trained using 3D patches from six-head orientation maps, details and hyperparameters of the training and validation stages are reported in Secs. 5.2.1, 5.2.2 and 5.2.3

After, we analyzed the results of the NET1-3D model changing the head direction of the input data. There is a good agreement with COSMOS according to the intensity values. But, there is not with respect to the contrast analysis, especially in the white-matter-background areas. The dispersion of the contrast values, changing the head orientation, increases increasing the intensity of the contrast.
About the TKD reconstructions, we have noticed a bigger dispersion also in the intensity values. The CNR values are smaller than the COSMOS and NET ones in all the considered areas.

- The semplification of the structure in general has affected the quality of the reconstructions. We need a bigger GPU memory to perfom a 3D training with the complete network. By the way, we are satisfied that, training the model with 3D patches, the returned maps are consistent with the gold-standard COSMOS and better than other from single-orientation methods such as the TKD technique.

5.3 Conclusions, Outstanding Issues and Future Work

- The experiments introduced in this chapter are aimed to reproduce a well reconstructed 3D χ map, looking into different brain sections.

- Two methods are proposed: a multiple-orientation 2D- and a 3D- patches training. For both of them, an increase in the GPU memory was required.

- In Sec. 5.1, the first experiment is introduced. Sagittal, coronal and axial slices were used to train the NET3 model. A single-orientation ϕ map was used to build the input database. Three 3D χ maps were obtained from the training: X-NET, Y-NET and Z-NET. Their performances, compared with the TKD ones, are good, and there is agreement with the COSMOS susceptibility map.

- In Sec 5.2, the 3D data experiment is presented. The network structure had to be adjusted, and the NET1-3D model was trained, using six ϕ maps with multiple head-orientations.
  The generalization skill of the model were tested using the six ϕ maps unseen during the training. In according to the metric parameters, the model response is pretty uniform.
  The reduction of the number of kernel, somewhere, negatively affects the machine learning approach, returning results which are worse than the 2D data training. But we are satisfied, because they are still better than the TKD ones, looking into the similarity parameters, the intensity and the contrast analysis.

This is the last chapter of this work. Below, a list of the possible developments is given.

- **The QSM problem - model extension** - The aim of this work is to reproduce a 3D χ map, starting from ϕ data, in order to perform a fast and precise method for the ill-posed QSM problem. In this way, the QSM tool may be applied to the clinical area. We tested the machine learning approach with a fully convolutional autoencoder and using spatial domain data. It achieved the desidered results, both with 2D and 3D data.
  The next step is the extension of the model to other data, to see if it could actually be applied on different input.

  1. **7T data** - One big challenge is to verify if the model can hold changes in the acquisition system ([70], [71]). First, we would like to test it using 7T ϕ data as input. Then, other settings may be changed - e.g. the spatial resolution or the model of the scanner.
  2. **Pre-processing stage** - It could be interesting to observe how the pre-processing stages affect the result of the model. The data used in all these experiments were pre-processed with the LBV ([25]) algorithm, but there are alternative techniques to process the raw phase data.
  3. **UK Biobank Database** - It is a prospective epidemiological resource, containing questionnaires, biological samples, physical and cognitive measures, ([78]). A first part of the database is still closed and contains around 500000 participants. Another part has been collected since 2016, counting to this day around 100000. The machine learning approach may be applied on a that wide database, in order to a) test the model ability and b) include in it QSM maps.

- **The machine learning approach** - In parallel with the model extension, the kind of data and structures used may be modified with the view to observe changes in the machine learning approach.

    1. **K-space data** - Because of the fact that the QSM problem is ill-posed in the k-space, a first change may regard the kind of data. FFTs from 3D patches may be used as input. The complex information of the Fourier transform has to be kept together, both the real and the immaginary part, during the training.
    2. **2.5D approach** ([79]) - It is a middle-ground between the 2D and the 3D data training. The second one is the best option, but not always an enough wide GPU memory is available. A 2.5D method means to start from a 3D volume (x, y, z) and to use as input z slices (x, y) at the same moment. It should reduce the computational effort and give results consistent with the 3D experiments.
    3. **Recurrent Neural Networks - RNNs** ([77]) - The model structure would be completly modified. This kind of network has been already used to solve inversion problem issues, and good results were obtained.

Conclusion

This is a machine learning project, aimed at applying a neural network algorithm to an image reconstruction task. The main topic is the QSM (Quantitative Susceptibility Mapping ([1])) tool, which is a recent MRI application that allows a susceptibility χ map to be obtained starting from MRI ϕ data. Multiple- (e.g. COSMOS ([5])) and single-orientation (e.g. TKD [7])) approaches have already been implemented to achieve this purpose, but they do not offer a good compromise between the accuracy and the speed of the reconstruction. A deep learning method could be a solution ([9]), allowing this tool to be applied in the clinical area.

The data used in the experiments are from the QSM_{2016} Challenge ([10]) database. It contains: ϕ maps of the brain from one individual from 12 different orientations, 12 χ TKD maps, a χ COSMOS map. The TKD approach requires a single phase image to reproduce a susceptibility map; the result is fast to produce but noisy. The COSMOS method instead uses all the available ϕ maps - 12 in this case - for the reconstruction. It provides a precise and accurate image. All the maps in the database are 3D brain images.
A fully convolutional autoencoder was implemented and optimized. A supervised learning technique was adopted: ϕ patches were used as input data and the corresponding patches from the COSMOS map were used as labels. The TKD reconstructions were used to compare our method with another single-orientation one.
The investigated similarity parameters were: RMSE, pSNR, SSIM and HFEN. Also, the susceptibility values of the resulting maps, from all the three methods, were compared to each other. Using these, a contrast analysis was carried out. The examined regions of interest include: the putamen, globus pallidus, caudate, substantia nigra and red nucleus, all of them being sub-cortical grey matter structures. Some white-matter-composed areas were also considered.
The results are summarised in the list below.

- In the first experiments two-dimensional data were processed. The axial slab orientation was chosen for this study. First of all, a single head-orientation ϕ map was used to train the network. The machine learning approach provides a good reconstruction, closer to the COSMOS one than the TKD method. The returned susceptibility map is accurate and noise-free. Intensity and contrast analysis gives satisfactory results.

After that, multiple head-orientations were taken into account. The similarity parameters and the intensity analysis confirm the results obtained from the single head-orientation experiment: the model susceptibility reconstruction is fast and produces results that are close to the gold-standard ones, also considering the changing in the head direction with respect to the magnetic field orientation. Performing the contrast analysis, a non uniform response of the network with respect to the head direction occurs. This does not happen in all the examined regions, but mainly in the high-contrast areas involving white matter areas. It could be due to the χ modelling: in fact, even if the susceptibility shows both isotropic and anistropic behaviours, the COSMOS method models it as a scalar property, and the network will have learnt this behaviour. By the way, the recognition ability of the model is never compromised, and it shows better perfomance than the control map.

- Another 2D data experiment was also carried out, this time considering not only axial slices, but also sagittal and coronal ones. For this experiment, single head-orientation data were used.
  Three susceptibility maps were built and examined, processing respectively sagittal, coronal and axial patches. Observing the reconstructions, even if the training set was composed of mixed patches, it was found that the perfomance is better if the orientation of the patches used to build the map is the same as that of the considered slice. This is not surprising since two-dimensional data are used to train the network and the actual relatioship between χ and ϕ takes place in the three-dimensional space.
  The quality of the reconstruction, according to the similarity parameters, is different in relation to the examined regions: moving along the three directions, the agreement between the COSMOS and our model susceptibility map is better in some areas than in others.
  The obtained reconstructions, however, are better than before, compared with the COSMOS susceptibility map; intensity and contrast analysis shows good performance.

- Finally, a 3D data experiment is proposed. The algorithm had to be adapted because of the change in the size of the patches. The structure had to be simplified, because the computational efforts increases a lot changing from 2D to 3D data processing.
  Multiple head-orientation data were used during the training stage, and the model shows a good generalization skill in changing the direction of the input data. Its perfomance is better than that of the TKD method according to all the similarity parameters. The intensity and the contrast analysis also confirms the accuracy of the reconstructed map and its similarity to the gold-standard. As before, a spreading out in the contrast values, increasing the contrast intensities, shows up. By the way, this does not affect the recongnition ability and however provides results consistent with the COSMOS method and better than the TKD one.

The implemented method achieves the goal of the work. Processing both two- and three-dimensional data, from a single head-orientation ϕ map, it provides a susceptibility map close to the gold-standard COSMOS, even if the latter is a multiple-orientation method. The machine learning approach reveals better perfomance than other single-orientation methods, such as the TKD one.

This work could be extended to working both on the processed data and on the model structure. To test the generalization skill is challenging, changing the acquisition system details - e.g. intensity of the static field or the spatial resolution - and the pre-processing stages of the data, to evaluate how much they affect the result. Then, the model could be applied on a big database (e.g. the UK Biobank Dataset), to include QSM maps and to find correlation between susceptibility and other tissue properties.
Other machine learning methods may also be implemented. First, the 2.5D approach, which may allow to obtain results close to the 3D ones but with less computational effort. Another is the use of Recurrent Neural Networks, which have already showed good outcomes in solving image reconstruction problems.

Bibliography

[1] A. Deistung, F. Schweser, and J.R. Reichenbach. Overview of quantitative susceptibility mapping. *NMR in Biomedicine*, 30, 2017.

[2] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016.

[4] E.M. Haacke, Y. Xu, Y.N. Cheng, and R. Reichenbach. Susceptibility Weighted Imaging (SWI). *Magnetic Resonance in Medicine*, 52:612–618, 2004.

[5] T. Liu, P. Spincemaille, L. de Rochefort, B. Kressler, and Y. Wang. Calculation of Susceptibility Through Multiple Orientation Sampling (COSMOS): A Method for Conditioning the Inverse PRoblem From Measured Magnetic Field Map to Susceptibility Source Image in MRI. *Magnetic Resonance in Medicine*, 61:196–204, 2009.

[6] C. Liu. Susceptibility Tensor Imaging. *Magnetic Resonance in Medicine*, 63:1471–1477, 2010.

[7] S. Wharton and R. Bowtell. Whole-brain susceptibility mapping at high field: A comparison of multiple- and single-orientation methods. *NeuroImage*, 53:515–525, 2010.

[8] P. Picton. *Neural Networks*. 2^{nd} edition, 2000.

[9] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521, 2015.

[10] C. Langkammer, F. Scheweser, K. Shmueli, C. Kames, X. Li, L. Guo anc C. Milovic, J. Kim, H. Wei, K. Bredies, S. Buch, Y. Guo, Z. Liu, J. Meineke, A. Rauscher, J.P. Marques, and B. Bilgic. Quantitative Susceptibility Mapping: Report from the 2016 Reconstruction Challenge. *Magnetic Resonance in Medicine*, 79:1661–1673, 2018.

[11] Richard Bowtell. Magnetic susceptibility of biological tissues . Sir Peter Mansfield Magnetic Resonance Centre, School of Physics and Astronomy, University of Nottingham, UK. Lecture Slides.

[12] Richard Bowtell. Susceptibility mapping with MRI . Sir Peter Mansfield Magnetic Resonance Centre, School of Physics and Astronomy, University of Nottingham, UK.

[13] https://www.scribd.com/document/315221178/Swi. Marìa J. Otero Dìaz, Magnetic Susceptibility in MRI. Lecture Slides.

[14] C. Liu, W. Li, K.A. Tong, K.W. Yeom, and S. Kuzminski. Susceptibility-Weighted Imaging and Quantitative Susceptibility Mapping in the Brain. *Journal of Magnetic Resonance Imaging*, 42:23–41, 2015.

[15] Y. Wang and T. Liu. Quantitative Susceptibility Mapping (QSM): Decoding MRI Data for a Tissue Magnetic Biomarker. *Magnetic Resonance in Medicine*, 73:82–101, 2015.

[16] C.M. Collins, B. Yang, Q.X. Yang, and M.B.Smith. Numerical calculations of the static magnetic field in three-dimensional multi-tissue models of the human head. *Magnetic Resonance Imaging*, 20:413–424, 2002.

[17] J.F. Schenck. Safety of Strong, Static Magnetic Fields. *Journal of Magnetic Resonance Imaging*, 12:2–19, 2000.

[18] Alessandro Palombit. Metodi per la quantificazione in-vivo della suscettivita' magnetica dei tessuti cerebrali da immagini di fase. Laurea magistrale in ingegneria dell'informazione, Alma Mater Sturdiorum, Università di Bologna, 2014/2015.

[19] S. Wharton and R.Bowtell. Effects of White Matter Microstructure on Phase and Susceptibily Maps. *Magnetic Resonance in Medicine*, 73:1258–1269, 2015.

[20] C. Laule, I.M. Vavasour, S.H. Kolind, D.K.B. Li, T.L. Traboulsee, G.R.W. Moore, and A.L. MacKey. Magnetic Resonance Imaging of Myelin. *Neurotherapeutics*, 4(3):460–484, 2007.

[21] W. Lu, B. Wu, and C. Liu. Quantitative susceptibility mapping of human brain reflects spatial variations in tissue composition. *Neuroimage*, 59:2625–2635, 2012.

[22] R.W. Brown, Y.C. Norman Cheng, E.M. Haacke, M.R. Thompson, and R. Venkatesan. *Magnetic Resonance Imaging: Physical Principles and Sequence Design.* 2^{nd} edition, 2014.

[23] Samuel Worthon. Susceptibility Mapping in High Field MRI. Phd thesis, Sir Peter Mansfield Imaging Centre, School of Physics and Astronomy, University of Nottingham, 2011.

[24] M. Schofield and Y. Zhu. Fast phase unwrapping algorithm for interferometric applications. *Optics Letters*, 28:1194–1196, 2003.

[25] D. Zhou adn T. Liu, P. Spincemaille, and Y. Wang. Background field removal by solving Laplacian boundary value problem. *NMR in Biomedicine*, 27:312–319, 2014.

[26] L.C. Evens. *Partial Differential Equations*. American Mathematical Society: Providence, Rhode Island, 2^{nd} edition, 2010.

[27] T. Liu, I. Khalidov, L. de Rochefort, P. Spincemaille, J. Liu, A.J. Tsiouris, and Y. Wang. A novel background field removal method for MRI using projection onto dipole fields (PDF). *NMR in Biomedicine*, 24:1129–1136, 2011.

[28] T.K. Moon and W.C.Stirling. *Pseudoinverses and the SVD. Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall: Upper Saddle River, New Jersey, pp. 116-117, 2000.

[29] H. Sun and A.H. Wilman. Background Field Removal Using Spherical Mean Value Filtering and Tikhonov Regularization. *Magnetic Resonance in Medicine*, 71:1151–1157, 2014.

[30] http://mriquestions.com/magic-angle.html.

[31] J.P.Marques and R. Bowtell. Application of a Fourier-Based Method for Rapid Calculation of Field Inhomogeneity Due to Spatial Variation of Magnetic Susceptibility. *Concepts in Magnetic Resonance*, 25B, 2005.

[32] G. Deville, M. Bernier, and J. Delrieux. NMR multiple echoes observed in solid ^{3}He. *Physical Review B*, 19, 1979.

[33] J.H. Duyn and T.M. Barbara. Sphere of Lorentz and Demagnetization Factors in White Matter. *Magnetic Resonance in Medicine*, 72:1–3, 2014.

[34] C. Wisnieff, T. Liu, P. Spincemaille, S. Wang, D. Zhou, and Y. Wang. Magnetic susceptibility anisotropy: cylindrical symmetry from macroscopically ordered anisotropic molecules and accuracy of MRI measurements using few orientations. *Neuroimage*, 70:363–376, 2013.

[35] C. Milovic, B. Bilgic, B. Zhao, J. Acosta-Cabronero, and C. Tejos. Fast Nonlinear Susceptibility Inversion With Variational Regularization. *Magnetic Resonance in Medicine*, 80:814 – 821, 2018.

[36] F. Schweser, K. Sommer, A. Deistung, and J.R.Reichenbach. Quantitative susceptibility mapping for investigating subtle susceptibility variations in human brain. *NeuroImage*, 62:2083–2100, 2012.

[37] Y. Wang and T. Liu. Quantitative susceptibility mapping (QSM): decoding MRI data for a tissue magnetic biomarker. *Magnetic Resonance in Medicine*, 73(1):82–101, 2015.

[38] Bohdann Macukow. Neural Networks – State of Art, Brief History, Basic Models and Architecture. In Khalid Saeed and Władysław Homenda, editors, *Computer Information Systems and Industrial Management*, pages 3–14. Springer International Publishing, 2016.

[39] Tom M. Mitchell. *Machine Learning*. 1997.

[40] D. Kotzias, M. Denil, P. Blunsom, and N. de Freitas. Deep Multi-Instance Transfer Learning. *CoRR*, abs/1411.3128, 2014.

[41] D. Kotzias, M. Denil, N. de Freitas, and P. Smyth. From Group to Individual Labels using Deep Features. 2015.

[42] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning Representation by back-propagating errors. *Nature*, 323:533–536, 1986.

[43] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. . *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambrige*, 1:318–362, 1986.

[44] Prashant Gupta. Cross-Validation in Machine Learning, 2017. Towards Data Science.

[45] W. Pitt and W.S. McCulloch. A Logical Calculus of the Ideas Immanent in Nervous Acitivity. *Bullettin of Mathematical Biology*, 5:115–133, 1943.

[46] Barnas Poczos. Introduction to Machine Learning - Perceptron.

[47] Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, pages 96–104, New York, 1960. IRE.

[48] D.O. Hebb. *Organization of Behavior*. New York: Wiley and Sons, 1949.

[49] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feed-forward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[50] https://en.wikipedia.org/wiki/Activation_function.

[51] http://yeephycho.github.io/2017/09/16/Loss-Functions-In-Deep-Learning/.

[52] Rob Di Pietro. A Friendly Introduction to Cross-Entropy Loss, 2016.

[53] M. Drozdal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pall. The Importance of Skip Connections in Biomedical Image Segmentation. *Deep Learning and Data Labeling for Medical Applications - Lecture note in Computer Science*, 10008:179–187, 2016.

[54] Sebastian Ruder. An overview of gradient descent optimization algorithms. Insight Centre for Data Analytics, NUI Galway, Aylien Ltd., Dublin.

[55] https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10.

[56] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *NIPS*, 2014.

[57] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[58] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[59] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 153–160, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[60] http://yann.lecun.com/ex/research/index.html.

[61] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. 3^{rd} edition, 2006.

[62] O. Ronneberger, P. Fisher, and T.Bronx. U-net : Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention, PT III - Lecture Notes in Computer Science*, 9351:243–241, 2015.

[63] B. Zhu, J.Z. Liu, S.F. Cauley, B.R. Rosen, and M. S. Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555:487–492, 2018.

[64] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[66] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, Dec 2017.

[67] Mattia Neri. Segmentazione di Immagini Mammografiche con Convolutional Neural Networks. Laurea magistrale in fisica, Alma Mater Sturdiorum, Università di Bologna, 2012/2013.

[68] https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035.

[69] C.A.Glasbey and K.V.Mardia. A review of image warping methods. *Journal of Applied Statistics*, 25, 1998.

[70] K.G.B. Rasmussen, M. Kristensen, R.G. Blendal, L.R. Ostergaard, M. Plocharski, K. O'Brien, C. Langkammer, A. Janke, M. Barth, and S. Bollmann. DeepQSM - Using Deep Learning to Solve the Dipole Inversion for MRI Susceptibility Mapping.

[71] J. Yoon, E. Gong, I. Chatnuntawech, B. Bilgic, J. Lee, W. Jung, J. Ko, H. Jung, K. Setsompop, G. Zaharchuck, E.Y. Kim, J. Pauly, and J. Leel. Quantitative Susceptibility Mapping using Deep Neural Network: QSMnet. *Neuroimage*, 179:199–206, 2018.

[72] S.M. Smith. Fast robust automated brain extraction. *Human Brain Mapping*, 2002.

[73] J. Ashburner and K.J. Friston. Chapter 2: Rigid Body Registration. 2003.

[74] Yusra Al-Najjar and Soong Der Chen. Comparison of image quality assessment: Psnr, hvs, ssim, uiqi. *International Journal of Scientific & Engineering Research*, 3:1–5, 01 2012.

[75] J. Wang and L. Perez. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. 2017.

[76] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Neural Network Training by Reducing Internal Covariate Shift. *ICML'15 Proceedings of the 32nd International Conference on International Conferene on Machine Learning - Volume 37*, 37, 2015.

[77] Patrick Putzky and Max Welling. Recurrent inference machines for solving inverse problems. June 2017.

[78] K.L. Miller, F. Alfaro-Almagro, N.K. Bangerter, D.L. Thomas, E. Yacoub, J. Xu, A.J. Bartsch, S. Jbabdi, S.N. Sotiropoulos, J.L.R. Andresson, L. Griffanti, G. Douaud, T.W. Okell, P. Weale, I. Dragonu, S. Garratt, S. Hudson, R. Collins, M. Jenkinson, P.M. Matthwes, and Stephen M.Smith. Multimodal population brain imaging in the uk biobank prospective epidemiological study. *Nature Neuroscience*, 19(11):1–5, November 2016.

[79] Vijay Mankar and M S. Khan. A review on 2d, 2.5d and 3d image visualization techniques. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 5:620–627, March 2016.