Imperial College London

Department of Computing

Spherical CNN for Imaging on Alzheimer's Disease

Author:
Hugo Barbaroux

Supervisor:
Dr. Elsa Angelini

Second Marker:
Dr. Wenjia Bai

Submitted in partial fulfillment of the requirements for the MSc degree in Computing (Machine Learning) of Imperial College London

September 2019

Abstract

Alzheimer's disease, one of the leading causes of dementia, is a very complex condition, in particular in its heterogeneity. We investigate in this project the use of Deep Learning to classify Alzheimer's diseased (AD) subjects against cognitively normal ones (CN), as well as the stability of Mild Cognitive Impairment (MCI), an in-between condition.

In medical imaging, the brain is often model as a 3D surface. Interesting features, such a cortical thickness or curvature, can be extracted from the raw surface, to be further handled as spherical signals. To exploit the underlying spherical symmetries, Feng *et al*. proposed in [1] the use of singular CNNs called Spherical CNNs.

In this project, we extend the work initiated by Feng *et al*., to explore the Spherical CNNs potential further. We consider using brain scans from the non-invasive T1-weighted MRI acquisition technique (1.5 Tesla).

Our noteworthy contribution lies in the use of ensemble methods to combine and improve models' performance. We also make interesting discoveries to improve the quality of the classification, such as combining various type of cortical measures (thickness, volume), or the possibility to make confident predictions even if only partial brain scans are available.

We additionally achieve state-of-art performance, based on previous works found in the literature.

Acknowledgments

I wish to express my sincere gratitude to Dr Elsa Angelini for the opportunity she gave me to work on state-of-the-art research for such an essential application as Medical Imaging can be. By providing me with consistent support throughout this project, she gradually built up my interest in research, especially towards using technology to improve people's lives. I definitely enjoyed working under her supervision, whether it was on a strictly professional level or personal relationship, and would be happy to pursue further research alongside her if the opportunity arises.

I would also want to thank Xinyang Feng, who now holds a doctorate from Columbia University, and the Heffner Biomedical Imaging Laboratory, without whom this project would have never been possible.

Finally, I wish to thank some of my classmates, Alexandre Chenu, Stanislas Hannebelle and Thomas Marchand, for having raised my reflections and brought a fresh outside look at crucial times in this project.

Acronyms

ACC Accuracy.

AD Alzheimer’s Diseased.

ADNI Alzheimers Disease Neuroimaging Initiative.

AUC Area Under a ROC Curve.

BN Batch-Normalization.

CAM Class Activation Map.

CN Cognitively Normal.

CNN Convolutional Neural Network.

CSF Cerebrospinal Fluid.

CT Computerised Tomography.

FFT Fast Fourier Transform.

FN False Negative.

FP False Positive.

GAP Global Average Pooling.

LDA Latent Dirichlet Allocation.

MCI Mild Cognitive Impairment.

MCI-p Mild Cognitive Impairment having progressed to AD.

MCI-s Stable Mild Cognitive Impairment.

MLP Multi-Layer Perceptron.

MRI Magnetic Resonance Imaging.

NAF Neighborhood Approximation Forest.

NLP Natural Language Processing.

PBM Planar Bilateral Multisequential.

PBU Planar Bilateral Unisequential.

PCNN Planar Convolutional Neural Network.

PET Positron Emission Tomography.

PU Planar Unilateral.

ReLU Rectified Linear Unit.

ROC Receiver Operating Characteristic.

RVM Relevance Vector Machines.

SBM Spherical Bilateral Multisequential.

SBU Spherical Bilateral Unisequential.

SCNN Spherical Convolutional Neural Network.

SEN Sensitivity.

SGD Stochastic Gradient Descent.

SPE Specificity.

SU Spherical Unilateral.

SVM Support-Vector Machines.

TN True Negative.

TP True Positive.

UCSF University of California, San Francisco.

wGAP Weighted Global Average Pooling.

Contents

List of Symbols

Binary operators

$*$	Convolution
$\cdot$	Block matrix multiplication
$\star$	Correlation

Elements

$\boldsymbol{M}$	Matrix
$\boldsymbol{x}$	Vector
f	Function

Unitary operators

$\hat{f}$ or $\mathcal{F}(f)$	Fourier transform of f
$\overline{f}$ or $f^{\dagger}$	Complex conjugate of f

Chapter 1

Introduction

1.1 Alzheimer's Disease

The global population is constantly ageing, with a current 72-year life expectancy at birth for the global population, and reaching more than 80 years in the most developed countries, based on the latest reports [2]. In such a context, more and more deaths can be imputed to ageing-associated diseases, especially neurodegenerative ones such as Alzheimer's disease. According to a report from Alzheimer's Disease International in 2018 [3], the estimated number of people who have dementia was about 50 million, and this number is expected to triple by 2050. Such numbers entail many practical challenges, especially regarding the ability to conduct quality diagnosis at a large scale.

Alzheimer's disease is one of the most common factors in dementia. It manifests through the death of brain cells connections and the atrophy of particular brain areas. It generally causes benign memory loss in early stages, but as brain damages increase, symptoms can become as serious as being unable to talk, eat, walk. With no cure at the moment, such symptoms, which involve loss of vital functions, lead inevitably towards an early death.

For these reasons, challenges of Alzheimer's disease diagnostic are many. In general, to get an idea of the disease progression, brain images, such as Magnetic Resonance Imaging (MRI) scans, are analyzed by medical experts. In a context as described above, practitioners might become more and more overwhelmed by the amount of data to treat, so there is a need to find methods to do the job automatically. The idea might not be to substitute totally humans by machines in this field, but rather to help them by reducing the load of analysis tasks, for them to be able to focus on the most serious cases. In addition, being able to analyse MRI scans the best we can is essential to understand the degeneration process in Alzheimer's disease and develop drugs.

1.2 Objectives & Challenges

In this project, we aim to explore automatic analysis tools to study Alzheimer's disease via brain MRI scans. We study state-of-the-art Machine Learning techniques for two classification tasks. The first one has a diagnosis purpose, by trying to differentiate between Alzheimer's diseased patients (AD) and cognitively normal subjects (CN). Apart from these two groups, patients can lie in an intermediary state called Mild Cognitive Impairment (MCI). It describes the phase were cognitive faculties start to decline because of age, but without causing dementia yet. A subject in an MCI state can either stay this way, or progress towards a more severe condition and convert into AD. A second classification task of this project is to predict either stability or conversion of MCI subjects, for example in a 2-year window.

This second application is potentially more decisive than the first one. As a cure for Alzheimer's disease is yet to be found, making an accurate diagnosis can be necessary to assist people in their disease, for instance by attenuating symptoms in the best possible way and provide precise monitoring. On the contrary, the MCI classification task aims to predict a future condition. If it can be done with relatively good precision, work can be done beforehand to postpone the conversion as long as possible. However, this classification task is in practice more difficult because of unclear definition of MCI subjects brain state.

1.3 Contributions

In this project, we continue the work initiated by Feng *et al.* in [1]. The main idea behind their work is to use Convolutional Neural Networks (CNNs), modified in a way that they can take spherical signals as input instead of conventional 2D or 3D images. While we started using their implementation, we optimized it in a way to reduce the overall computation time, by a factor of about 6. We reconducted all of their experiments to validate their choices and modify them when needed. We introduced other architectures in addition to theirs as a comparison basis. In the chapter describing our results (from Section 5.4), we detail the original components of our work. Details about our contribution in curating the ADNI data can be found in Chapter 3.

It should be noted that no practical implementation will be provided alongside the submission of this report. In agreement with the supervisor, we wish to keep any related code private as a research paper is expected to be produced from this project.

1.4 Outline

In Chapter 2, we dive into some theoretical material that we found necessary to exhibit, in order to understand what is really at stake in this project. We first introduce the acquisition process (MRI) used to get the brain images we use in this project, as well as pre-processing approaches which can be done to exploit their full potential. We then come back on Machine Learning fundamentals, from Deep Learning basics up to theoretical concepts for understanding the main architectures used in the project. More practical elements such as models' performance metrics are also mentioned. The last section explores similar works on Alzheimer's disease analysis from the past seven years, to serve as a comparison basis.

Chapter 3 precisely describes the data used for this project (ADNI cohort). It first details the data origin and exposes various statistics. In a second part, we explain the entire pre-processing pipeline used to curate the data as needed by our architectures.

These architectures are described in Chapter 4. We detail in there multiple Deep Learning models based on the theoretical concepts developed in Chapter 2, for finally setting up the entire training process.

In Chapter 5, we describe all the experiments we conducted. Our various performance results can be found, as well as the decisions these results made us take to try to optimize our models the best we could. We particularly explored the interpretability of our models in classifying Alzheimer's disease.

The final chapter summarizes the entire project. We take a critical look at the conducted experiments, and mention some future work to explore, with the aim of producing a research paper.

Chapter 2

Background

2.1 Scanning the brain

Medical images, and especially brain ones, can be acquired via multiple modalities. Among the most common ones are Computerised Tomography (CT), Positron Emission Tomography (PET), or Magnetic Resonance Imaging (MRI). In this project, we only consider brain MRI images.

2.1.1 MRI scans acquisition

MRI is a non-invasive imaging technique involving magnetic field, and does not involve X-rays ionizing radiations.

In order to acquire an MRI scan, a powerful magnetic field is created around the body area which needs to be imaged. It forces body protons to get their spin aligned alongside the established magnetic field lines, creating a constant spin equilibrium. These protons are then excited by radio wave impulses. Physical quantities, such as decay rates, can be measured during the spins relaxation phase back to an equilibrium state. As these measures vary depending on protons' physical and chemical environment, they can be used to build a 3D map of the brain by contrasting multiple tissue substances, as illustrated on Figure 2.1. The magnetization process known as T1-weighted results in images where the darkest regions potentially indicate bones, air or cerebrospinal fluid (CSF); the brightest ones fat and melanin; and brain matter is in between, with grey matter darker than white matter.

The more powerful the magnetic field, the more precise the final image, because of stronger signal and noise reduction in spin equilibrium. MRI scanner field power usually goes from 0.2 to 7 Teslas.

A significant drawback of this method lies in the acquisition time. Scanning a 3D volume with MRI technology can only be performed one 2D slice at a time, because of some physical limitations of the process. For each slice, radiofrequency pulse and atom relaxation take time. In addition, the more we want to increase resolution, the finer we need to sample the acquisition points in K-space.

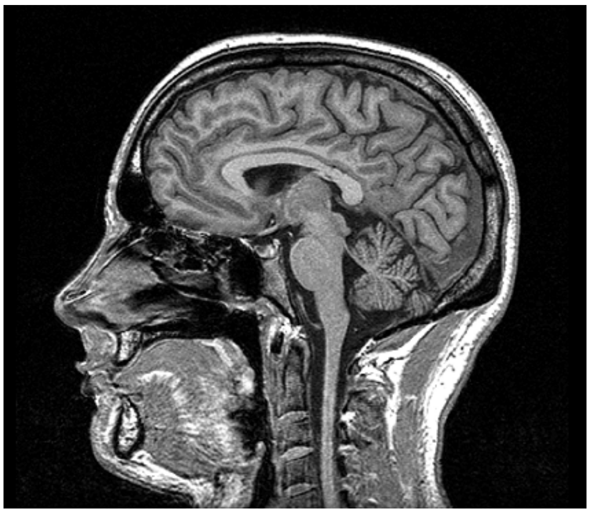


Figure 2.1: Example of a T1-weighted brain MRI scan.

2.1.2 Processing into functional features

Raw MRI scans can be exploited by several standard automated tools. The Athinoula A. Martinos Center for Biomedical Imaging of Harvard Medical School created a piece of software called Freesurfer [4], which implements several useful processing tools. In this section, we dive into some of them in detail. The following subsections mainly describe processing steps originally thought by Dale, Fischl and Sereno [5, 6].

Inter-subject registration

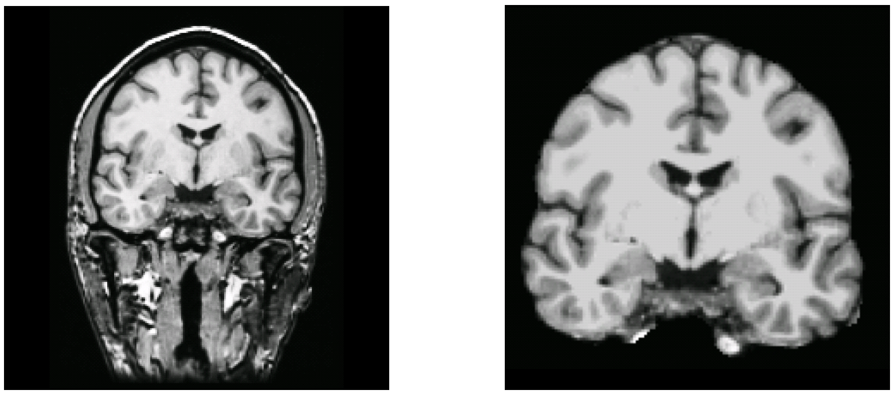
As interpreting images of the human brain is highly facilitated by having them aligned in a common reference frame, registration of MRI scans between subjects might be desired. Talairach and Tournoux [7], and after them Collins *et al.* [8] contributed into building a generic atlas of the brain (made by aggregating data from many subjects) and into establishing a process for estimating the registration between a subject of interest and the atlas.

Skull stripping

MRI brain scans do not exclusively provide a clean brain image. As illustrated on Figure 2.1, every head element is displayed, from nose and mouth mucous membranes, to neck bones and cranium. Apart from the brain matter, these surrounding structures are parasitic information we would want to get rid of.

A boundary-based algorithm was initially designed by Dale *et al.* [5] in the first version of Freesurfer. It involved the local deformation of a tessellated ellipsoidal template. Initially aligned with the brain, two forces can then be applied on its vertices. The first one relies on MRI intensity values on the surface normal from each vertex, to drive the template outwards brighter areas (corresponding to brain matter) and towards darker ones (corresponding to CSF). To avoid the vertices to move too much from one another, a smoothing force is also applied, which imposes continuity in edge directions neighbouring a given vertex.

However, such a method is not flawless [9]. The main drawback comes from the sensitivity of the algorithm to parameters initialization: initial position and size of the template, forces relative weights and threshold intensity boundary between ”darker” and ”brighter” areas. In practice, recurrent errors were also noticed for inferior areas. For these reasons, Segonne *et al.* [9] suggest an improved algorithm. While still relying on deformable surface models to finely shape the brain, they combine it with a watershed approach beforehand. Such region-based methods can be used to segment an image by exploiting region connectivity to clustering ends. While these techniques generally result in oversegmentation, they can be an efficient way of approximating brain boundaries to initialize deformable surface models robustly. An example of skull stripping using Freesurfer can be seen on Figure 2.2.



(a) Initial T1-weighted MRI scan.

(b) Extraction of the brain matter.

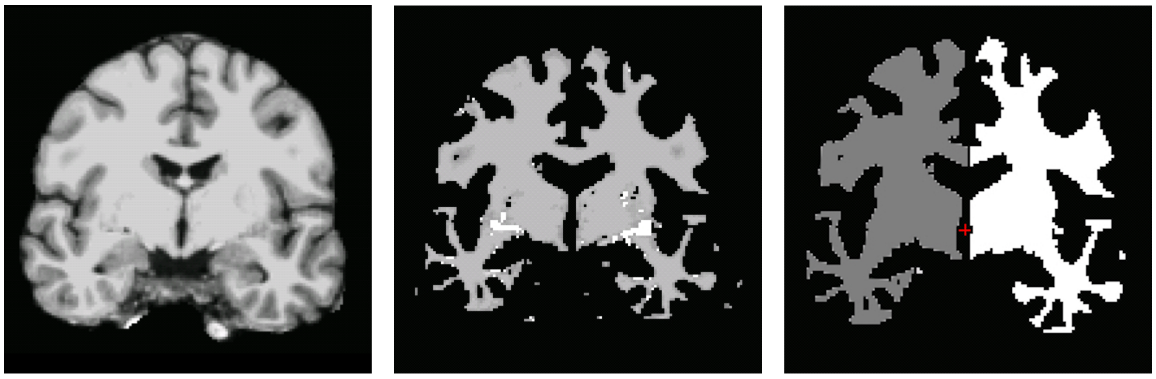
Figure 2.2: Example of skull stripping using Freesurfer.

White matter segmentation

The pial surface (the boundary between grey matter and CSF) is difficult to estimate in itself because of the relatively low intensity of grey matter voxels. Instead, white matter voxels, brighter, can be more easily labelled. We will see in the next Section how it can help to estimate the pial surface.

The main method used by Dale *et al.* for white matter segmentation is a two-step algorithm. First, white matter voxels are estimated based on intensity information only. While inner voxels can be easily classified by this method, uncertainty increases at edges. These areas are further processed in a second algorithm. In short, cutting planes are established by imposing a low-intensity variance in neighbouring voxel intensities for each area delimited by a given plane. It allows to finely classify voxels as white matter or non-white matter on either side of the planes.

A final contour smoothing is applied by computing connected components. By first cutting the brain into left and right hemispheres thanks to alignment on the Talairach atlas, voxels in boundary regions can iteratively be classified (or declassified) as white matter if at least 66% of neighbouring voxels in a $3 \times 3 \times 3$ surrounding box are white matter (or non-white matter). This process results in a unique smooth component for each hemisphere. The complete process is illustrated on Figure 2.3.

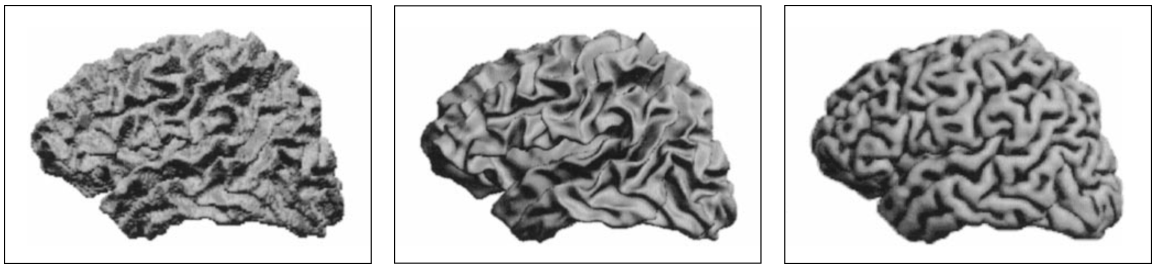


(a) Initial brain matter. **(b)** White matter segmentation. **(c)** Hemisphere split and white matter volume refinement.

Figure 2.3: White matter labelling (from [5]).

Pial surface estimation and cortical measures

White matter segmentation can then be used to estimate better the pial surface than by directly trying to segment it. In a similar manner as for skull stripping, deformable models are used to extend the grey/white matter boundary towards the pial surface. A tessellated surface is initially inferred via the previously segmented white matter volume. Each vertex can then iteratively move, driven by two forces: a smoothing one and an intensity-based one to detect intensity changes. With such a method, the grey/white matter boundary can be refined even further, and the pial surface precisely estimated (Figure 2.4).



(a) Initial grey/white matter boundary. **(b)** Surface refinement. **(c)** Extension towards pial surface.

Figure 2.4: Pial surface reconstruction steps (from [5]).

The pial surface can then be used to compute local measures such as surface curvature, or even cortical thickness [10] and volume, by locally measuring distances between pial surface and grey/white matter interface.

Surface inflation

These cortical measures, locally displayed on the cortical surface, might even be further processed to allow for complete visualization. Indeed, sulci measures (grooves) while potentially being as essential as gyri ones (ridges), can still be partially hidden. Fischl *et al.* suggest in [6] to inflate the cortical surface into a very smooth shape, where neighbouring sulci and gyri measures would be set on the same level. Their method involves sulci inflation while enforcing minimal distortion. Figure 2.5b illustrates the cortical surface after inflation.

While the resulting images are useful for visualization, computer analysis would gain from a more regular signal in terms of spatial distribution. As we will see later in this chapter, some analysis methods are particularly efficient in working with spherical signals. Fischl *et al.* suggest then to inflate even further the surface to map it onto a proper sphere, once again by minimizing local distortions. The full process is summarized on Figure 2.5.

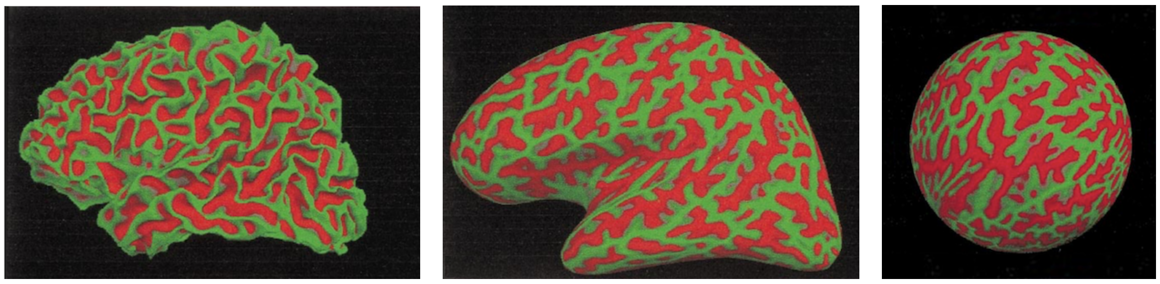

(a) Initial cortical surface. **(b)** Surface inflation. **(c)** Spherical mapping.

Figure 2.5: Cortical surface inflation (surface curvature measures displayed, from [6]).

2.1.3 Structural atlases

While we only looked at the brain as a whole until now, works have been done in the past to extract functional brain areas. We study two different segmentation schemes here: a first one introduced by Desikan *et al.* [11] in 2006, and the second one in 2010 by Destrieux *et al.* [12]. We focus on these two functional atlases here because they are readily available via the Freesurfer software.

Both atlas acquisition methods are actually similar. A cohort of about a few tens of subjects is selected, chosen from a uniform distribution of age and gender, each of them having relatively healthy brains. MRI scans from all subjects are then processed as described in Section 2.1.2 up to inflated brain surfaces as illustrated on Figure 2.5b. Next, experts manually segmented surfaces into predefined functional sections. The aggregation of all the segmented brains after registration is what we call an atlas.

To parcel a new brain scan (assuming it is preprocessed up to an inflated surface), a registration step is first applied to bring the brain surface to the atlas. Then, a probability distribution over the set of functional sections can be assigned to each voxel, given the previously manually labelled samples. Refinement can then be applied by iteratively modifying these probabilities to reduce neighbourhood variance (we can see it as a smoothing step). More details can be found in [13].

The major difference between the two atlases considered here lies in the definition of functional brain sections. While the Desikan atlas segments the brain into 36 regions, 75 different sections form the Destrieux one. Illustrated parcellations and labels can be found in Appendix B (respectively Figure B.1, Table B.1 and Figure C.1, Table C.1 for Desikan and Destrieux atlases).

2.2 Deep Learning fundamentals

After having described the type of images involved in this project and essential pre-processing methods, we can now dive into some practical automatic techniques which can be used for analyzing images and especially brain-related ones.

2.2.1 Motivation for the use of Deep Learning in Medical Imaging

In the early years of image analysis, standard methods involved feature extraction by traditional means such as Shift-Invariant Feature Transform descriptors [14] or Harris corner detector [15]. Such a method aims to describe a given image in a more meaningful way than raw pixels only, in order to understand conveyed information better, for example by extracting knowledge about frequencies and gradients. The main goal for developing these meaningful representations is to feed, in a second phase, powerful machine learning techniques, whether it be Support-Vector Machines or Random Forests for classification, or K-means for clustering (to name but a few), depending on the final task. In the 2000s, this type of approaches achieved state-of-the-art performances for image analysis tasks, as benchmarked by the PASCAL Visual Object Classes challenges [16].

Then, in 2012, a breakthrough in performances happened through the development of Deep Learning. The idea between Deep Learning is to use Machine Learning methods not only for analyzing previously extracted features, but also for letting the algorithms learn relevant features by themselves. In this way, while Deep Learning models tend actually to discover themselves features previously used by humans, they are not constrained by them and can develop their own reasoning to succeed in a particular image analysis task. Its power is proved by the performance scores on the ImageNet Large Scale Visual Recognition Challenge [17, 18] these past years. The error rate in classifying billions of images divided into 1,000 classes was stuck around 25% until 2012. That year, the AlexNet architecture [19], first Deep Learning solution to break performances in this challenge, achieved a 16% classification error. Nowadays, the most advanced Convolutional Neural Networks (CNNs) can be even better than humans in simple image classification tasks. There is no doubt that Deep Learning and especially CNNs are now more than ever needed to improve computer vision performances further.

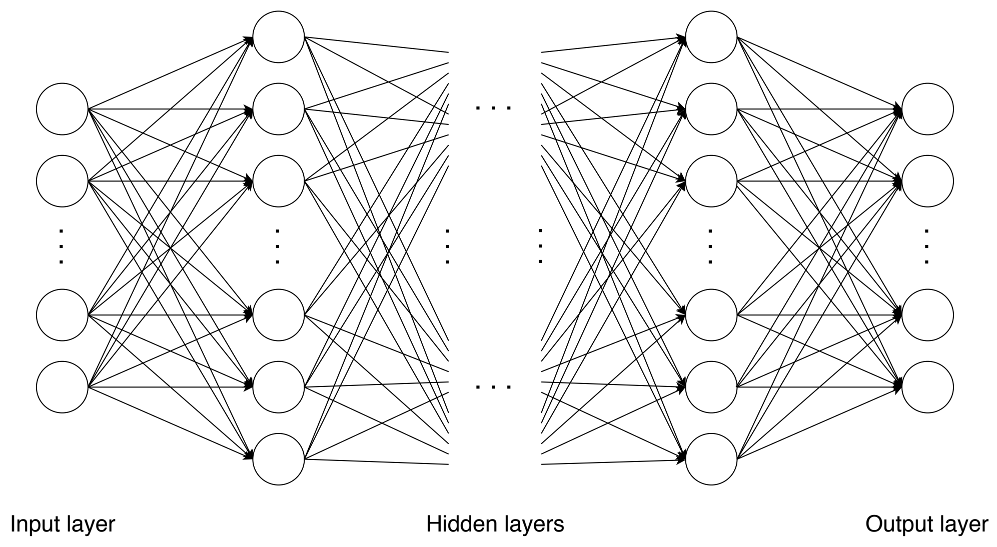


Figure 2.6: Multi-Layer Perceptron

What about medical imaging? Such a context brings its share of difficulties at another level than classical computer vision tasks. Even if human experts can analyze medical images and take relevant decisions, these images are sometimes so complex than the accuracy of the predictions are way lower than any other image analysis situation. Adding Deep Learning into this context makes sense, whether it be to help experts being more precise in their analysis or even to relieve them of this task, which generally takes time and dedication that practitioners can't afford to spend.

2.2.2 Multi-Layer Perceptron (MLP)

Before diving into CNNs and more specifically spherical ones, let us motivate the thought by going back into the basics of Deep Learning: Multi-Layer Perceptrons. Behind these networks lies the extension of linear regression learning. Linear regression models, used for example in SVMs, aim to learn a mapping $f(\boldsymbol{x}) = \boldsymbol{y}$ between inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$, under the assumption that this mapping is linear. The main objective behind such models is then to learn a weight matrix $\boldsymbol{W}$ such that $\boldsymbol{W}\boldsymbol{x} = \boldsymbol{y}$. With real data samples, such a mapping might not always exist, so $\boldsymbol{W}$ is generally inferred by minimizing a loss function $\mathcal{L}(f(\boldsymbol{x})\boldsymbol{y})$, for example the Euclidean distance.

Multi-Layer Perceptrons have been introduced in Machine Learning to extend this idea. These networks are generally fully-connected feedforward networks, composed of multiple linear sequentially (building successive layers and allowing "deep" architectures), as seen on Fig 2.6. Each layer i is defined by a matrix $\boldsymbol{W}_i$, used to compute the layer's output $\boldsymbol{W}_i\boldsymbol{x}_i = \boldsymbol{y}_i$. With this configuration, the mapping learned by the network between inputs $\boldsymbol{x} = \boldsymbol{x}_1$ and outputs $\boldsymbol{y} = \boldsymbol{y}_n$ is simply

$$f(\boldsymbol{x}) = \left(\prod_{i=1}^{n-1} \boldsymbol{W}_i\right) \boldsymbol{x}_1, \tag{2.1}$$

which is nothing more than a basic linear regression model. The power of the Multi-Layer configuration comes from the possibility of using non-linear activation functions at the output of each layer. The learned mapping then becomes

$$f(\boldsymbol{x}) = g_{n-1}\left(\boldsymbol{W}_{n-1} \dots \left(g_1\left(\boldsymbol{W}_i \boldsymbol{x}_i\right)\right)\right) \tag{2.2}$$

By varying the output size (number of ”neurons”) of each layer and the depth of the network, Multi-Layer Perceptrons can learn any continuous function, even non-linear ones.

2.2.3 Convolutional Neural Network (CNN)

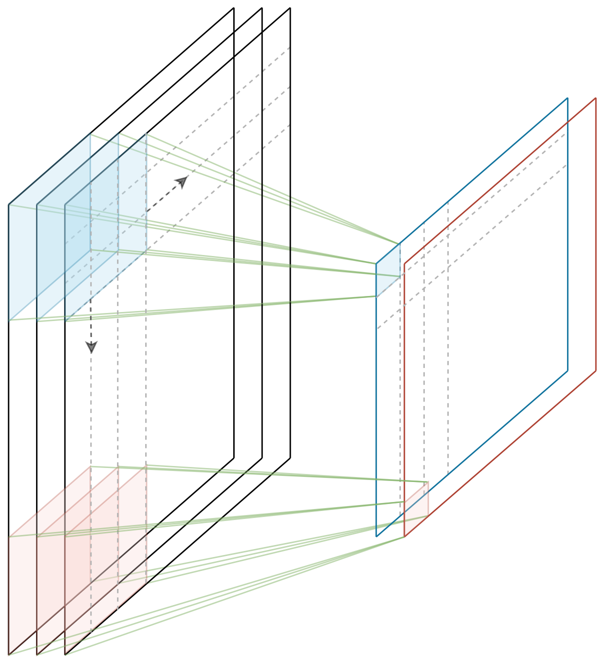


Figure 2.7: Illustration of a convolutional layer. The input data is represented by k channels of $m \times n$ matrices, and each output channel is obtained by the convolution of the input data by a $i \times j \times k$ kernel.

What about Deep Learning for image analysis? One could argue that images are just pixel values, which can therefore be used as input for a Multi-Layer Perceptron. It is a possibility, and for some simple tasks, it might be rather efficient. However, even for something as basic as classifying handwritten digits, MLPs are outperformed by Convolutional Neural Networks. Le Cun provides in [20] a benchmark of Machine Learning solutions for such a task using MNIST images. For an equivalent number of operations for the recognition task, an MLP exhibits an error rate twice as high as one of the very first CNNs ever (which uses a LeNet architecture). So what makes CNNs so particular compared to MLPs? It mainly comes from the fact that 2D images, inputted as so in CNNs, suffer loss of information by being flattened to fit an MLP input. By having learnable weights as small windows (kernel) sliding through

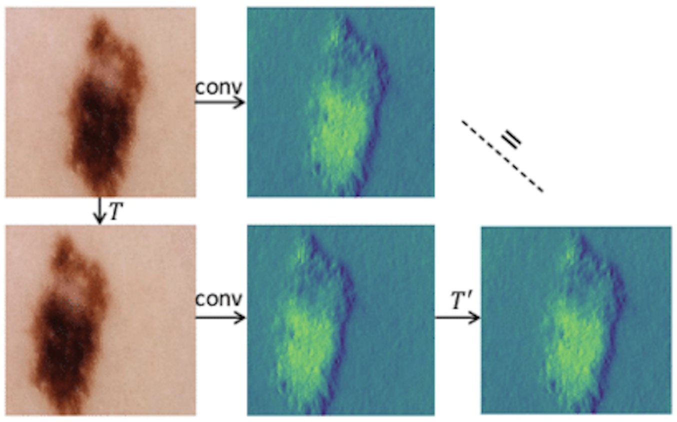


Figure 2.8: Translation equivariance of CNN's feature maps (from [21])

an image (Figure 2.7), CNNs can learn 2D local connectivity information very easily, contrary to MLPs. Moreover, this connectivity can be learned at different scales by varying the kernel size and increasing network depth, offering additional freedom. It is worth noting that in practice, every output neuron (pixel) of a given layer are obtained via the same kernel (in a mono-channel output configuration). It allows global consistency and reduces a lot the number of free parameters to be learned compared to MLPs. For instance, while the two benchmarked networks cited previously exhibit roughly the same number of operations in a forward pass, the CNN has 15 times fewer parameters than the MLP, making the underlying learning space a lot less complex. But the most significant advantage of this weight sharing is in making the output of a CNN translation invariant, or making the feature maps (the name commonly given to convolutional layer outputs) translation equivariant (see Figure 2.8).

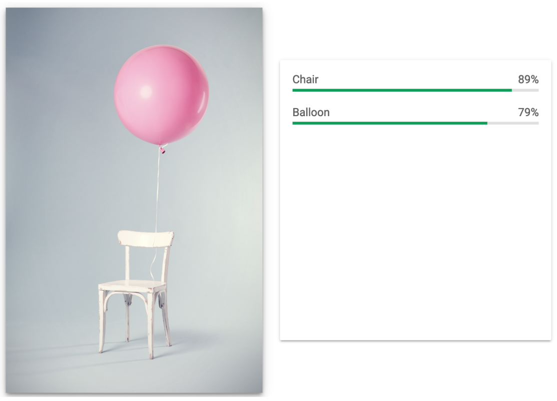
We have noticed that CNNs can easily outperform MLPs even for the simplest image analysis tasks. In a medical context, using CNNs makes even more sense as involved images are a lot more complex and challenging to interpret.

Mathematically speaking, sliding a kernel through an image is a convolution. If we have a kernel $\boldsymbol{\Psi}$ and an image $\boldsymbol{X}$, we can determine the output component $\boldsymbol{Y}(i,j)$ by the convolution

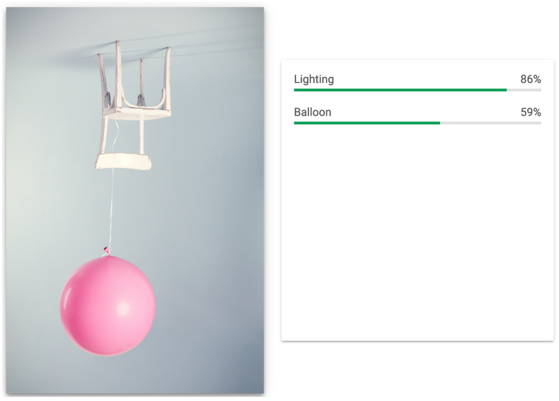
$$(\boldsymbol{X} * \boldsymbol{\Psi})(i,j) = \sum_{i'} \sum_{j'} \boldsymbol{X}(i',j') \boldsymbol{\Psi}(i-i', j-j').^{1} \tag{2.3}$$

In practice, correlation, which is faster to compute than convolution, is implemented

[1]In practice, convolutional layers can have multiple channels k as input, with a different kernel for each one. Equation 2.3 becomes $(\boldsymbol{X} * \boldsymbol{\Psi})(i,j) = \sum_{i'} \sum_{j'} \sum_{k} \boldsymbol{X}_k(i',j') \boldsymbol{\Psi}_k(i-i', j-j')$, where $\boldsymbol{X}$ and $\boldsymbol{\Psi}$ are now 3D tensors. To be even more general, outputs can be also composed of multiple channels l, making $\boldsymbol{\Psi}$ a 4D tensor and $(\boldsymbol{X} * \boldsymbol{\Psi}^l)(i,j) = \sum_{i'} \sum_{j'} \sum_{k} \boldsymbol{X}_k(i',j') \boldsymbol{\Psi}^l_k(i-i', j-j')$ the general equation describing the behaviour of Figure 2.7. For the sake of clarity, we have decided to stick to a mono-channel configuration in this report.



(a) The chair is easily detected on the original picture.



(b) If we rotate the picture by 180, the chair becomes totally undetected.

Figure 2.9: PCNNs lack of rotational robustness (classification results obtained from Google Vision API).

instead:

$$(\boldsymbol{X} \star \boldsymbol{\Psi})(i,j) = \sum_{i'}\sum_{j'} \boldsymbol{X}(i',j')\boldsymbol{\Psi}(i+i',j+j'). \tag{2.4}$$

The results of the second equation are actually the same than the first one if we flipped the kernel in between. As kernels in CNNs are in fact free parameters to be learned, networks are blind to this trick and behave exactly the same if we use correlation instead of convolution.

2.3 Spherical CNNs

2.3.1 Motivations

The limits of CNNs come as other invariances appear. For example, in image classification, an image label does not change if we look at a rotated image. However, classical CNNs (which we will call Planar CNNs or PCNNs from now on) are sensitive to such transformations, as illustrated on Figure 2.9. Some solutions have been proposed to overcome this problem, for instance by augmenting datasets with randomly rotated samples to artificially force networks into learning some kind of rotation invariance [22, 23].

However, if we deal with spherical signals, PCNNs reach their limits. If we want to use convolution architecture with such signals, we would want, in the same way as in PCNNs, having some kind of kernel sliding through the signal. However, to be properly fitted for a PCNN, a spherical signal must be shrunken into a 2D image, like a planisphere. Figure 2.10 shows how this transformation makes kernels unevenly distorted and results in losing any kind of invariances. It actually comes from the fact that a kernel moving on a sphere is subjected to rotations in the standard 3D space and, as we said before, common CNNs do not preserve any rotation equivariance.

In order to address this issue, Cohen & Welling suggest an extension of the convolution operator (actually correlation to be precise) from translations to rotations [24], and more generally to any group of transformations [25]. In the next two sections, we are diving into the theoretical concepts behind it, using some of Cohen & Welling material. The main derived formulae can be directly found in the summary table 2.1.

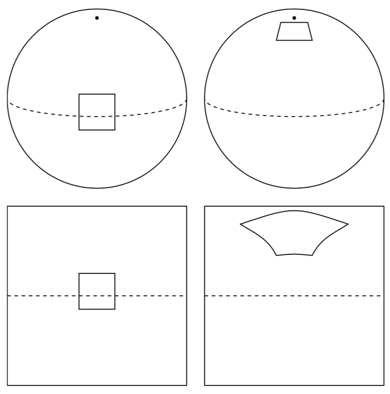


Figure 2.10: Distortion of convolutional windows from the sphere to its planar projection (from [24]).

2.3.2 Theory of Spherical CNNs

If we go back over Equation 2.3 by extending convolution to the continuous case between a signal f and a kernel ψ, we have, for all $\boldsymbol{y}$ in $\mathbb{R}^2$:

$$(f * \psi)(\boldsymbol{y}) = (\psi * f)(\boldsymbol{y}) = \int_{\mathbb{R}^2} \psi(\boldsymbol{x}) f(\boldsymbol{y} - \boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}.^2 \tag{2.5}$$

As we mentioned before, for the use of CNNs, we can equally consider correlation and convolution. In the same way, we can rewrite Equation 2.4:

$$(\psi \star f)(\boldsymbol{y}) = \int_{\mathbb{R}^2} \psi(\boldsymbol{x}) f(\boldsymbol{y} + \boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \tag{2.6}$$

$$= \int_{\mathbb{R}^2} \psi(\boldsymbol{x} - \boldsymbol{y}) f(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \quad \text{(by substituting } \boldsymbol{x} \text{ by } \boldsymbol{x} - \boldsymbol{y}\text{)}. \tag{2.7}$$

Actually here, $\boldsymbol{x}$ and $\boldsymbol{y}$, although both lying in $\mathbb{R}^2$, do exactly not represent the same thing. While $\boldsymbol{x}$ represent the coordinates of a point in the original 2D signal, an image in our situation, $\boldsymbol{y}$ represent a translation of the kernel. Indeed, if we define $t_{\boldsymbol{y}} : \boldsymbol{x} \mapsto \boldsymbol{x} + \boldsymbol{y}$, we can rewrite correlation 2.7 in

$$(\psi \star f)(\boldsymbol{y}) = \int_{\mathbb{R}^2} \psi(t_{\boldsymbol{y}}^{-1}(\boldsymbol{x})) f(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}, \tag{2.8}$$

[2] The possibility of interchanging f with ψ comes from the commutativity of the convolution operator.

or, by using a matrix representation where $\boldsymbol{T}$ is an arbitrary 2D translation matrix:

$$(\psi \star f)(\boldsymbol{T}) = \int_{\mathbb{R}^2} \psi(\boldsymbol{T}^{-1}\boldsymbol{x}) f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \tag{2.9}$$

With the latter, we can clearly see that the codomain is a transformation space rather than the original 2D signal domain. However, in this particular case, as the 2D translation space is $\mathbb{R}^2$ (a 2D translation can be entirely defined by a displacement $\boldsymbol{y}$ in $\mathbb{R}^2$), then the codomain of the above correlation remains $\mathbb{R}^2$, even if theoretically speaking, it is not the same one as the space in which lies a 2D image.

Why have we bothered to derive Equation 2.9 from its original expression? It can now clearly be extended to any input signal and transformation group. In our situation, dealing with spherical signal f and rotating kernel ψ in S^2 becomes straightforward. For any rotation $\boldsymbol{R}$ in the rotation group SO(3) (special orthogonal group), the correlation of ψ and f can be written

$$(\psi \star f)(\boldsymbol{R}) = \int_{S^2} \psi(\boldsymbol{R}^{-1}\boldsymbol{x}) f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}.^3 \tag{2.10}$$

More precisely, SO(3) and S^2 are respectively 3D and 2D manifolds as they can be parameterized by the Euler angles ZYZ $\alpha \in [0, 2\pi]$, $\beta \in [0, \pi]$ and $\gamma \in [0, 2\pi]$ (with γ = 0 for the sphere). This time, the codomain SO(3) is different from the input space S^2. While the latter equation can be used to define the first convolutional layer of what Cohen & Welling call Spherical CNNs (SCNNs), we need an integration on SO(3) to define the behavior of following layers, because the input space is not S^2 anymore:

$$(\psi \star f)(\boldsymbol{R}) = \int_{\mathrm{SO(3)}} \psi(\boldsymbol{R}^{-1}\boldsymbol{Q}) f(\boldsymbol{Q}) \, \mathrm{d}\boldsymbol{Q}. \tag{2.11}$$

From Equation 2.10, we can finally make sure that this extended correlation preserves rotation equivariance. We remind that a function u is said to be equivariant by a group action G if for any transformation v in G, $u \circ v = v \circ u$ (which means the order of function application is of no importance). By defining the operator $L_{\boldsymbol{P}}$ such that $[L_{\boldsymbol{P}} f](\boldsymbol{x}) = f(\boldsymbol{P}^{-1}\boldsymbol{x})$, we have, for any arbitrary rotation $\boldsymbol{P}$:

$$\begin{aligned}(\psi \star [L_{\boldsymbol{P}} f])(\boldsymbol{R}) &= \int_{S^2} \psi(\boldsymbol{R}^{-1}\boldsymbol{x}) f(\boldsymbol{P}^{-1}\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\ &= \int_{S^2} \psi(\boldsymbol{R}^{-1}\boldsymbol{P}\boldsymbol{x}) f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \quad \text{(by substituting } \boldsymbol{x} \text{ by } \boldsymbol{P}\boldsymbol{x}) \\ &= (\psi \star f)(\boldsymbol{P}^{-1}\boldsymbol{R}) \\ &= [L_{\boldsymbol{P}}(\psi \star f)](\boldsymbol{R}),\end{aligned} \tag{2.12}$$

which properly express the fact that rotating an input signal comes down to rotating the output feature map[4].

[3]As mentioned before, Cohen & Welling actually extend even further this equation to any transformation group. If g is a function lying in one of such groups, the general form of Equation 2.10 is $(\psi \star f)(g) = \int \psi(g^{-1}(\boldsymbol{x})) f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$.

[4]We show the rotation equivariance in 2.12 for an input layer from S^2 to SO(3), but it obviously extend to following layers from SO(3) to SO(3).

2.3.3 Practical implementation of the rotational correlation

In practice, this spherical correlation is implemented via a generalized Fourier transform. Indeed, using a Fast Fourier Transform (FFT) algorithm [26] is more efficient than naively convolving signals. This result was originally shown in [27] for signals on $\mathbb{R}$, but can be extended to generalized Fourier transforms (see [28, 29] for more details).

To compute the Fourier transform of the correlation operation, it can be shown [24] that $\widehat{\psi \star f} = \hat{f} \cdot \hat{\psi}^\dagger$, or $\psi \star f = \mathcal{F}^{-1}\left(\hat{f} \cdot \hat{\psi}^\dagger\right)$. We can generalize Fourier transforms on S^2 by using the spherical harmonics $Y^l(\boldsymbol{x})$[5] as unit basis functions, for l upper-bounded by a bandwidth b defining a frequency limit. We can then define the S^2 Fourier transform components as

$$\hat{f}^l = \int_{S^2} f(\boldsymbol{x}) \overline{Y^l(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}.^6 \tag{2.13}$$

In the same way, we can use the Wigner D-functions $D^l(\boldsymbol{R})$[7] to define the SO(3) Fourier transform components as

$$\hat{f}^l = \int_{\mathrm{SO(3)}} f(\boldsymbol{R}) \overline{D^l(\boldsymbol{R})} \, \mathrm{d}\boldsymbol{R}.^8 \tag{2.14}$$

Figure 2.11 sums up the practical implementation of spherical correlation using a generalized FFT algorithm [9]. All the derived equations from the past two sections for computing this FFT are recapped in Table 2.1.

It should be noted that the practical implementation of the S^2 convolutional layer pictured in Figure 2.11 shows a 2D matrix as input while we are originally working with spherical signals. These signals are in practice discretized on a meshgrid (Figure 2.12) and flattened into a planisphere. The non-rotated kernel is defined as an equatorial circle.

[5]To be precise, $Y^l(\boldsymbol{x})$ is a vector of size $2l+1$, indexed by $-l \leq m \leq l$. For this particular case we do not use the defined notation for vectors but rather the common notation for spherical harmonics in literature.

[6]It is to be noted that $\hat{f}^l$ is a vector as $Y^l(\boldsymbol{x})$ is, but to be consistent with Fourier transform notations, we do not use the defined vector notation.

[7]$D^l(\boldsymbol{R})$ is a matrix of size $(2l+1) \times (2l+1)$, indexed by $-l \leq m, n \leq l$. As the S^2 case, we do not use the defined notation for matrices but rather the common notation of Wigner D-functions in literature.

[8]It is to be noted that $\hat{f}^l$ is a matrix as $D^l(\boldsymbol{R})$ is, but to be consistent with Fourier transform notations, we do not use the defined matrix notation.

[9]Figure 2.11 actually describes the general case were inputs are composed of multiple channels, adding to the derived formulae a summation over the number of channels. As in Section 2.2.3, for the sake of clarity, we have derived the entire theory based upon a mono-channel configuration.

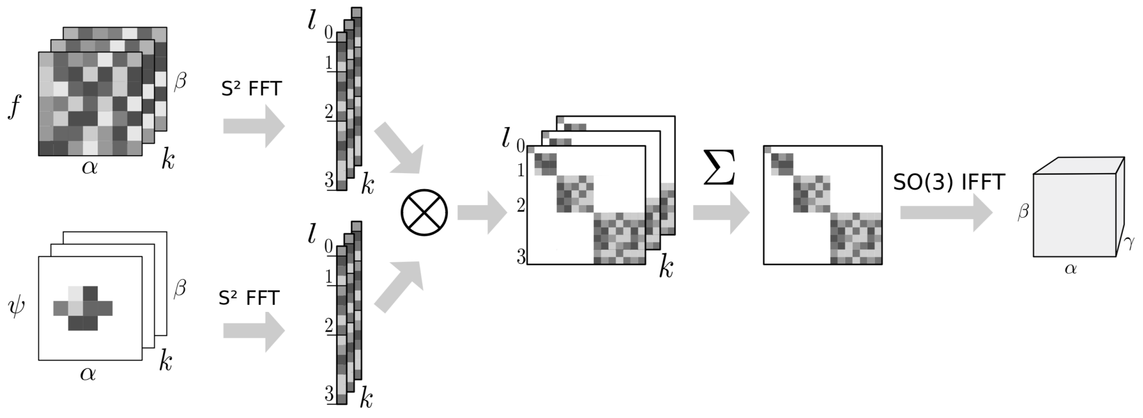


Figure 2.11: Spherical correlation ($S^2 \to \mathrm{SO}(3)$) using Fourier transforms (from [24]). Both the signal f and kernel ψ are Fourier transformed, the outer product of outcomes $\hat{f}^l$ and $\hat{\psi}^{l\dagger}$ is computed (for every l smaller than the output bandwidth b) to form a block in the resultant matrix. A final inverse Fourier Transform is computed to output a 3D tensor indexed in SO(3).

Space	Correlation	Fourier Transform	Inverse Fourier Transform
$\mathbb{R}^2$	$(\psi \star f)(\boldsymbol{y}) = \int_{\mathbb{R}^2} \psi(\boldsymbol{x} - \boldsymbol{y}) f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$	$\hat{f}(\boldsymbol{\xi}) = \int_{\mathbb{R}^2} f(\boldsymbol{x}) \mathrm{e}^{-i\boldsymbol{x}.\boldsymbol{\xi}}\,\mathrm{d}\boldsymbol{x}$	$f(\boldsymbol{x}) = \frac{1}{2\pi} \int_{\mathbb{R}^2} \hat{f}(\boldsymbol{\xi}) \mathrm{e}^{i\boldsymbol{x}.\boldsymbol{\xi}}\,\mathrm{d}\boldsymbol{\xi}$
S^2	$(\psi \star f)(\boldsymbol{R}) = \int_{S^2} \psi(\boldsymbol{R}^{-1}\boldsymbol{x}) f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$	$\hat{f}^l = \int_{S^2} f(\boldsymbol{x}) \overline{Y^l(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x}$	—
SO(3)	$(\psi \star f)(\boldsymbol{R}) = \int_{\mathrm{SO(3)}} \psi(\boldsymbol{R}^{-1}\boldsymbol{Q}) f(\boldsymbol{Q})\,\mathrm{d}\boldsymbol{Q}$	$\hat{f}^l = \int_{\mathrm{SO(3)}} f(\boldsymbol{R}) \overline{D^l(\boldsymbol{R})}\,\mathrm{d}\boldsymbol{R}$	$f(\boldsymbol{R}) = \sum_{l=0}^{b} (2l+1) \sum_{m=-l}^{l} \sum_{n=-l}^{l} \hat{f}^l_{mn} D^l_{mn}(\boldsymbol{R})\,\mathrm{d}\boldsymbol{R}$

Table 2.1: Formulae summary of correlation, Fourier transform and inverse Fourier transform on $\mathbb{R}^2$ and their extensions on S^2 and SO(3). As the S^2 Fourier transform codomain is SO(3), we do not define the inverse Fourier transform on S^2.

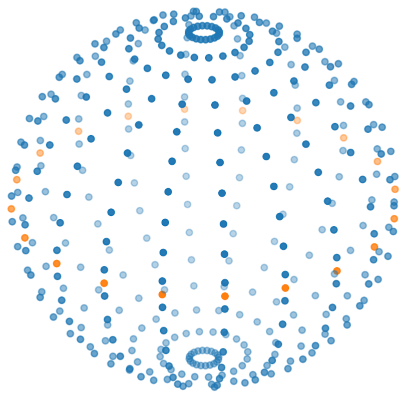


Figure 2.12: Spherical meshgrid used to sample the initial signal (blue) and equatorial kernel (orange).

2.4 Results validation

2.4.1 Class Activation Maps

People often reproach deep neural network models their lack of interpretability. In particular for medical applications, being able to understand a model's decisions and way of thinking might be necessary to validate their behaviour and winning a large-scale acceptance. In an image analysis context, feature maps contain all the information needed to comprehend how a trained network is working.

Zhou *et al.* actually found in [30] that these feature maps can highlight localization information about relevant parts of a given image in the decision process of a neural network. It is particularly true for the deepest feature maps of a CNN. Indeed, feature maps in the first layers are mainly based on global image patterns, and become more and more localized as we go deeper into the network.

Multiple feature maps can be recovered from the last layer of a convolutional neural network, but how can we combine them in a way that reflects the network's decision algorithm? Fortunately, a CNN is generally composed of a fully-connected subnetwork (MLP) after its convolutional layers, to go down to an output vector containing a neuron for each class (in a classification task setting). In general, this vector represents a probability distribution over the set of classes. This configuration in two subnetworks can be seen as an initial feature extraction task by the convolutional layers, followed by an MLP to classify samples based on the previously extracted features.

To answer the question about feature maps combination, the weights of the MLP subnetwork contain valuable information about how the network uses each feature map in its decision process. The most basic ConvNet/MLP configuration is the one depicted on Figure 2.13. In such a situation, the fully-connected subnetwork is composed of a single layer, associating a weight from each final feature map to every output class. Weighting extracted feature maps with the weights associated with a given class can give an idea of the image regions the networks relies on to build its decision process. The resulting combination forms what we call a Class Activation Map (CAM).

While this method fits well for PCNNs, where feature maps lie in the same space than input images ($\mathbb{R}^2$ or $\mathbb{R}^3$), it is less straightforward in an SCNN architecture where feature maps lie in the 3D manifold SO(3) and spherical input signals in a 2D manifold. As we mentioned in Section 2.3.2, SO(3) can be parametrized by the Euler angles $Z(\alpha)Y(\beta)Z(\gamma)$, while S^2 can be similarly described by setting γ to 0. Feng *et al.* then suggest in [1] an extension of the proposed CAM method by considering 2D slices corresponding to $\gamma = 0$ in the original 3D feature maps.

2.4.2 Evaluation measures

To evaluate the performance of the multiple architectures we will be using in this project, we implement several standard measures described below.

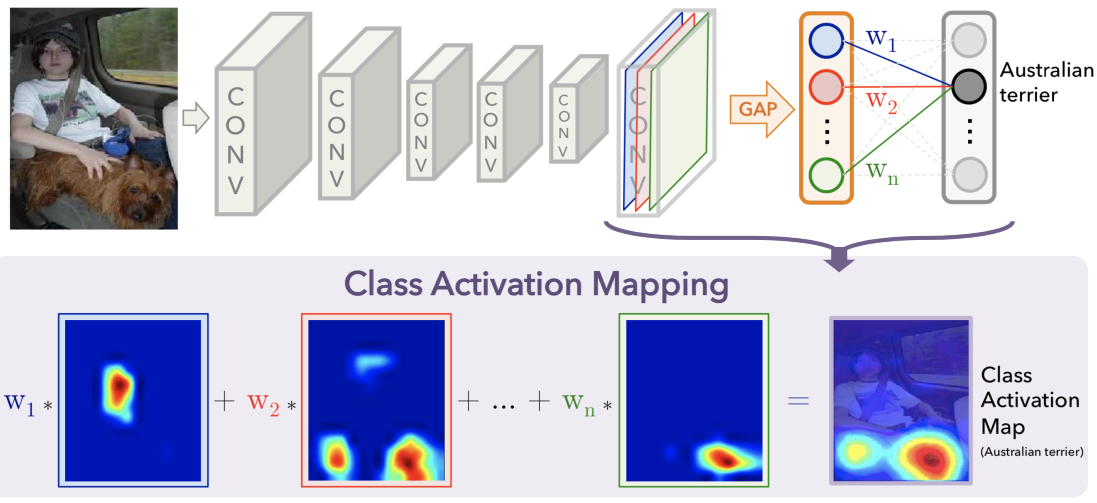

CONV
GAP
w_1
w_2
w_n
Australian terrier
Class Activation Mapping
$w_1 *$ + $w_2 *$ + ... + $w_n *$ =
Class Activation Map (Australian terrier)

Figure 2.13: Class Activation Map estimation process (from [30]).

Accuracy

The very first indicator we will use is the accuracy. In a binary classification task, we can consider one of the two target classes as positive and the other one as negative. We define here TP (for True Positives) the number of positive samples actually being classified as positive, TN (for True Negatives) the number of negative samples actually being classified as negative, FP (for False Positives) the number of negative samples wrongly classified as positive, and FN (for False Negatives) the number of positive samples wrongly classified as negative.

Using these numbers, we can define the accuracy as

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}}. \tag{2.15}$$

The accuracy measure gives an idea of the overall performance of a model, as a number between 0 and 1, 0 meaning that the model makes nothing but mistakes, while a perfect classifier would get an accuracy of 1.

Sensitivity & Specificity

One major drawback of the accuracy measure is that it does not give any information about the model performances on the different target classes. For instance, if a set contains samples equally labelled in both classes, and the accuracy of a given model is 50%, it could mean that it classifies all the samples as being positive, all the samples as being negative, or that it makes mistakes for both classes.

To get a more precise idea about a model's behaviour, we can define its specificity (SPE) and sensitivity (SEN) as

$$\begin{aligned}\mathrm{SEN} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}\\ \mathrm{SPE} &= \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}.\end{aligned} \tag{2.16}$$

A model's sensitivity tells us about its ability to correctly classify positive samples, while its specificity tells us about its ability to classify negative samples correctly. As the accuracy, both values are between 0 and 1.

Area under a Receiver Operating Characteristic curve

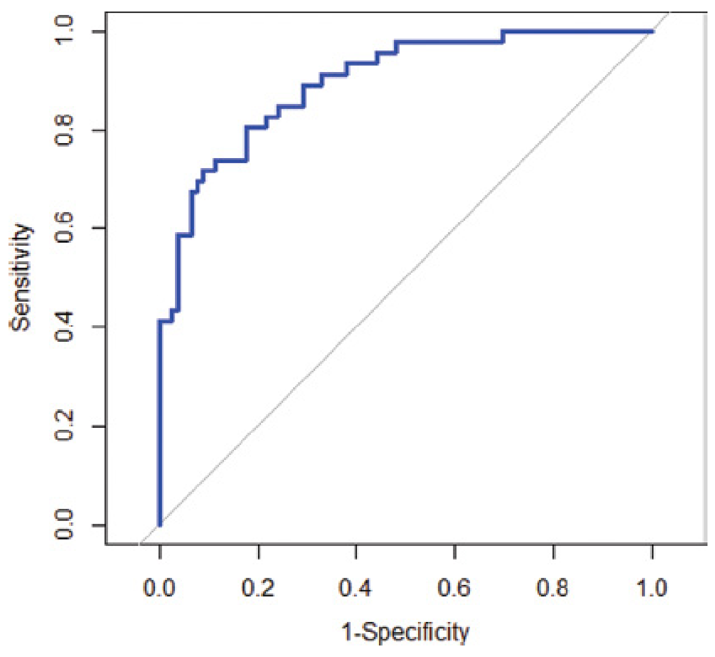


Figure 2.14: Example of a ROC curve.

The majority of Machine Learning classifiers actually output a probability distribution over the label space. To get label prediction, in case of binary classification, a probability threshold of 0.5 is set, and the predicted classes are those with a probability greater than this threshold. However, setting a 0.5 threshold is not the only way to obtain classification results. To get a more precise idea about how a given model behaves for each class, we can observe classification results when moving this threshold, as depicted on Figure 2.15. By sampling multiple threshold values, what we call a Receiver Operating Characteristic curve (ROC curve) can be made, drawing a model's sensitivity given its specificity (in practice $1-$specificity) for various thresholds. Figure 2.14 shows an example of such a curve.

From the ROC drawing, we can finally compute the area under the curve (AUC). This value, once again up to 1, can be seen as an indicator of a model's ability to discriminate between classes.

2.5 Related work

Some works have already been done in the past to study Alzheimer's disease. In this section, we review multiple methods which can be found in the literature, specifically on the two classification tasks we focus on here: AD vs CN and MCI progression.

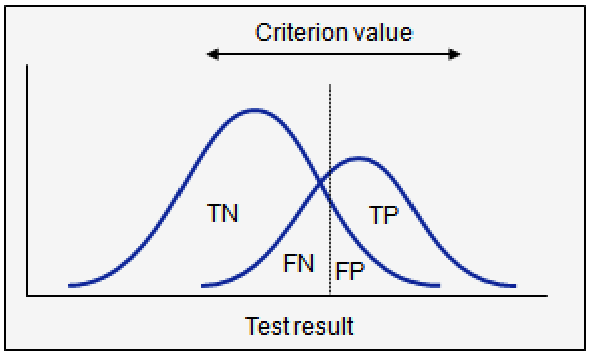


Figure 2.15: Moving the classification threshold (or criterion) provides additional information about a model's behavior.

Table 2.2 and Table 2.3 (respectively for AD vs CN and MCI progression) review all these methods and their performance scores, based on the evaluation methods described in Section 2.4.2.

2.5.1 Spherical CNNs

Feng *et al.* introduce in [1] the idea of using SCNNs for medical imaging and especially for Alzheimer's disease classification. Their architecture exploits spherical signals made of cortical thickness measures, obtained via Freesurfer as described in Section 2.1. The current project is mainly a continuation and improvement of their work, so their results might be the most relevant ones as a comparison basis with ours. Performances can be found in the tables at the end of this section.

2.5.2 Latent Dirichlet Allocation (LDA)

In [31], Yang *et al.* exploit cortical measures obtained from MRI scans on a structural level, i.e. thickness or volume of entire cortical regions (obtained for example via atlas parcellation as described in Section 2.1.3. To study Alzheimer's disease classification with such features, they suggest a Latent Dirichlet Allocation (LDA) algorithm. This method, initially thought for Natural Language Processing (NLP) tasks [32], aim to cluster the data by assigning each sample a probability distribution over a set of topics, where topics are learned from input features (strictly speaking a probability distribution over the set of features). In short, the LDA method can cluster data while having a high level of interpretability (data can be clustered by looking at probability distributions over topics, and each topic ranks features in order of importance in terms of decision impact).

Besides, Yang *et al.* use this LDA model in a supervised fashion, meaning that they can perform classification tasks while being able to understand relative feature importance in the algorithm decision process. For instance, by using information about cortical regions as features, they can extract the ones which are likely to be related to Alzheimer's disease.

While their classification results might not be as high as other methods proposed in the literature, the interpretability of their model can be very useful for understanding Alzheimer's disease progression.

It should be noted that Yang *et al.* also suggest using genomic features in addition to image-based ones in their model, but given the scope of the current project, we only take an interest here in the results they obtain using cortical imaging features.

2.5.3 Conventional Machine Learning methods

From the beginning of the decade, several papers for classifying Alzheimer's disease have emerged in the literature. Almost all of them make use of Support-Vector Machines based algorithms, like Zhang *et al.* [33], Cheng *et al.* [34], Hu *et al.* [35], Sabuncu *et al.* [36] or Beheshti *et al.* [37], after having processed cortical features in various ways. It can go from feature reduction, to clustering, to feature ranking via Genetic Algorithms [37]. Sabuncu *et al.* [36] also make use of other classification methods such as Neighborhood Approximation Forest (NAF), a nearest neighbours approach based on Random Forest architectures, or also Relevance Vector Machines (RVM), a Bayesian algorithm based on Gaussian Processes. We were careful here to only select papers which were developing methods using features from MRI scans only. Some of the papers cited here might also be exploiting other types of images, but the results reported in tables below were the ones obtained exclusively from MRI scans.

In studying Alzheimer's disease, trying to solve classification tasks is not the only subject in the literature. For instance, an interesting paper was published in 2017 by Filho *et al.* [38] on the understanding of Alzheimer's disease progression in the brain. By looking at cortical measures of structural brain regions (from Destrieux and Desikan atlases), they were able to discriminate areas were the disease seems to have a substantial impact. This work, similarly to Yang *et al.* experiments, can help in understanding how the disease operates in the brain.

Table 2.2: Comparison of several methods proposed in the literature for the AD vs CN classification task using T1-weighted MRI.

Paper	ACC (%)	AUC (%)	SEN (%)	SPE (%)
Zhang *et al.*, 2012 [33]	84.80	-	-	-
Westman *et al.*, 2012 [39]	87.00	93.00	83.30	90.10
Eskildsen *et al.*, 2013 [40]	84.50	90.50	79.40	88.90
Sabuncu *et al.*, 2015 [36]	87.00	-	-	-
Hu *et al.*, 2016 [35]	84.13	90.00	82.45	85.63
Beheshti *et al.*, 2017 [37]	93.01	93.51	89.13	96.80
Feng *et al.*, 2018 [1]	90.00	91.50	89.90	90.10
Yang *et al.*, 2019 [31]	88.00	-	-	-

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

Table 2.3: Comparison of several methods proposed in the literature for the MCI progression classification task using T1-weighted MRI. Some of the proposed methods study MCI progression over a 3-year period, while others do it on a 2-year period as we do.

Paper	ACC (%)	AUC (%)	SEN (%)	SPE (%)
Zhang *et al.*, 2012 [33]	62.00	-	56.60	60.20
Westman *et al.*, 2012 [39]	65.40	73.10	65.40	65.40
Eskildsen *et al.*, 2013 [40]	66.70	67.30	59.00	70.20
Cheng *et al.*, 2015 [34]	73.40	76.40	74.30	72.10
Moradi *et al.*, 2015 [41]	74.74	76.61	88.85	51.59
Hu *et al.*, 2016 [35]	76.69	79.00	71.83	82.26
Beheshti *et al.*, 2017 [37]	75.00	75.08	76.92	73.23
Feng *et al.*, 2018 [1]	71.60	70.70	80.20	62.30

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

Chapter 3

Data description

3.1 Data gathering

The data used throughout this project exclusively comes from the Alzheimer's Disease Neuroimaging Initiative (ADNI, [42]). Initiated by Dr Michael W. Weiner in 2004, its goal is to acquire relevant data for understanding Alzheimer's disease by new diagnostic methods.

This study is divided into three phases, and we used data from two of them: the ADNI-1 and ADNI-2 cohorts.

As mentioned before, we study two classification tasks: AD vs CN, and MCI progression in a 2-year period. For the latter purpose, two categories of MCI patients are kept: the ones having stayed stable in the MCI condition for a follow-up period of at least two years (MCI-p), and those having progressed into AD during this period (MCI-s).

In the following subsections, we detail the composition of the ADNI-1 and ADNI-2 cohorts and global statistics on ages and genders, for both classification tasks. Each cohort has been divided into 10 and 5 folds for the use of cross-validation. The division of the ADNI-1 cohort in 10 folds comes from Feng *et al.* and has not been modified in order to keep a consistent comparison basis. We extend this division into 5 folds by pairing existing folds. The division of the ADNI-2 cohort was performed exclusively by us. While Feng *et al.* seemed to focus on balancing the folds in terms of numbers, we refined into the splitting process. We made sure that the number of male AD, female AD, male CN, female CN would be the same in each fold. We went even further by preserving the age distribution of the global cohort in these subcategories. Detailed statistics on the 5-fold and 10-fold splits for each cohort can be found in Appendix.

Table 3.1: Statistics of the ADNI-1 cohort used for the AD vs CN (green) and MCI-p vs MCI-s (blue) classification tasks.

Numbers	M	F	Total
AD	99	89	188
CN	74	77	151
Total	173	166	339
Ages (std)	M	F	Total
AD	75.6 (7.4)	74.7 (7.7)	75.2 (7.5)
CN	75.5 (5.6)	75.8 (5.0)	75.6 (5.3)
Total	75.6 (6.6)	75.2 (6.6)	75.4 (6.6)

Numbers	M	F	Total
MCI-p	85	51	136
MCI-s	72	42	114
Total	157	93	250
Ages (std)	M	F	Total
MCI-p	75.4 (7.1)	73.6 (6.6)	74.7 (6.9)
MCI-s	75.6 (7.5)	73.7 (6.9)	74.9 (7.3)
Total	75.5 (7.3)	73.6 (6.7)	74.8 (7.1)

3.1.1 ADNI-1

We obtained the ADNI-1 data from Feng *et al*. They had already balanced it, and we used it as so. Table 3.1 shows detailed statistical information about gender and age distribution for each subject group in ADNI-1.

Baseline scans are available for all subjects. Some of them also have follow-up visits. Table 3.2 details the number of available follow-up visits for AD and CN subjects. We did not conduct experiments involving follow-up visits for the MCI classification task, so we only include here information on AD and CN subjects.

Table 3.2: Number of AD and CN scans available for various follow-up visits.

Follow-up month	AD	CN
0 (baseline)	188	151
6 (0.5 year)	162	144
12 (1 year)	139	140
18 (1.5 years)	1	0
24 (2 years)	101	119
36 (3 years)	1	96
48 (4 years)	0	29
60 (5 years)	0	1

3.1.2 ADNI-2

Even if Feng *et al.* did not conduct any experiment on the ADNI-2 cohort, they still provided us with the data. While the ADNI-1 data contained follow-up visits, we only had access to baseline scans for the ADNI-2 cohort.

The initial cohort was composed of 347 AD, 540 CN, 311 MCI-p, and 467 MCI-s subjects. We decided to balance AD and CN subjects by decreasing the number of CN subjects to 340. We made sure that removed patients were randomly selected in a way that the original statistics (age and gender) were kept intact. Table 3.3 shows the detailed statistics of the cohort before and after balance.

For MCI subjects, the processing was less straightforward. As we study the conversion of MCI subjects on a 2-year window, we first remove all the MCI stable subjects having a last follow-up visit less than 2 years after the baseline (too much uncertainty). After this removal, the number of MCI-s subjects went down to 378. We had then two choices to process the MCI-p subjects: those having a conversion year greater than 2 could either be removed, or included as being stable on the 2-year window. In the first situation, we end up with 209 MCI-p and 378 MCI-s; in the second one, 209 MCI-p and 480 MCI-s. To balance the latter, we randomly removed MCI-s subjects while being careful to preserve the original statistics. We will call the resulting cohort CC. To balance the former, we thought about two ways. A first cohort, which we can call CS2, was obtained in the same way than CC, by carefully removing MCI-s subjects. A second cohort, which we can call CS4, was obtained by removing MCI-s subjects having a last follow-up visit close to 2 years. While we already removed the ones with a last follow-up visit under 2 years, removing even those with a last follow-up visit close to 2 years assures that the remaining MCI-s subjects are more likely to remain stable for a long time.

To recap:
- CC contains MCI subjects having progressed in AD in at most 2 years (MCI-p), and subjects having either never converted (while being monitored for at least 2 years) or converted only after 2 years (MCI-s).
- CS2 contains MCI subjects having progressed in AD in at most 2 years (MCI-p), and subjects who never converted (while being monitored for at least 2 years, MCI-s).
- CS4 contains MCI subjects having progressed in AD in at most 2 years (MCI-p), and subjects who never converted (while being monitored for at least 4 years, MCI-s).

Statistics on the three cohorts are reported on Table 3.4.

Table 3.3: Statistics of the ADNI-2 cohort on the original AD/CN distribution (left) and the balanced cohort (right).

Numbers	M	F	Total
AD	190	157	347
CN	260	280	540
Total	450	437	887
Ages (std)	M	F	Total
AD	75.9 (7.7)	74.1 (7.9)	75.1 (7.8)
CN	74.8 (6.0)	73.5 (5.6)	74.1 (5.8)
Total	75.2 (6.8)	73.8 (6.5)	74.5 (6.7)

Numbers	M	F	Total
AD	190	157	347
CN	190	150	340
Total	380	307	687
Ages (std)	M	F	Total
AD	75.9 (7.7)	74.1 (7.9)	75.1 (7.8)
CN	74.8 (5.9)	73.5 (5.5)	74.2 (5.7)
Total	75.4 (6.9)	73.8 (6.8)	74.7 (6.9)

Table 3.4: Statistics of the ADNI-2 MCI cohorts. Top left: CC; Top right: CS2; Bottom: CS4.

Numbers	M	F	Total
MCI-p	126	83	209
MCI-s	126	84	210
Total	252	167	419
Ages (std)	M	F	Total
MCI-p	74.6 (7.0)	72.7 (6.9)	73.9 (7.0)
MCI-s	73.5 (6.9)	71.8 (7.4)	72.8 (7.1)
Total	74.1 (7.0)	72.3 (7.1)	73.3 (7.1)

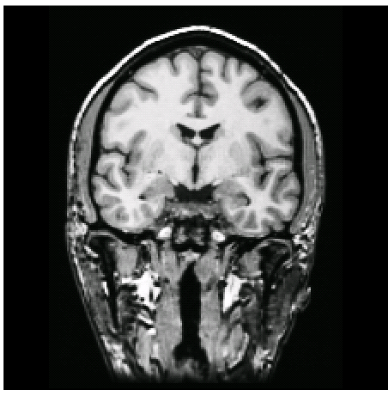
Numbers	M	F	Total
MCI-p	126	83	209
MCI-s	124	84	208
Total	250	167	417
Ages (std)	M	F	Total
MCI-p	74.6 (7.0)	72.7 (6.9)	73.9 (7.0)
MCI-s	72.8 (7.2)	71.8 (7.2)	72.4 (7.2)
Total	73.7 (7.1)	72.3 (7.1)	73.1 (7.1)

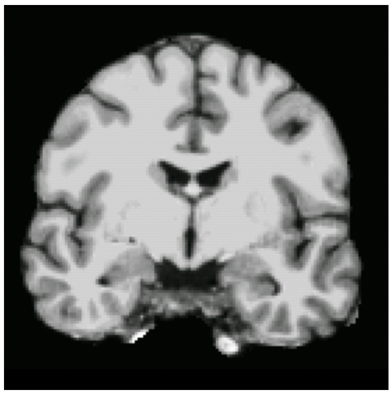
Numbers	M	F	Total
MCI-p	126	83	209
MCI-s	121	82	203
Total	247	165	412
Ages (std)	M	F	Total
MCI-p	74.6 (7.0)	72.7 (6.9)	73.9 (7.0)
MCI-s	71.5 (6.7)	71.8 (8.0)	71.6 (7.2)
Total	73.1 (7.0)	72.2 (7.5)	72.7 (7.2)

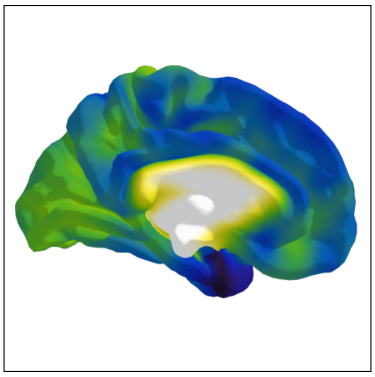
3.2 Data processing

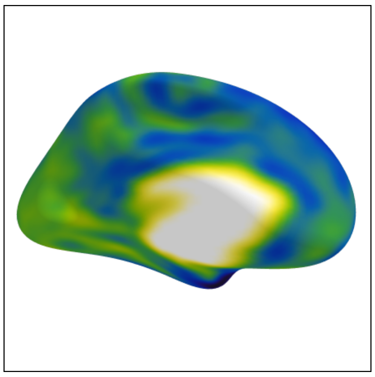
Raw data consists of T1-weighted MRI brain scans, obtained by 1.5 T MRI scanners, and pre-processed with the standard Mayo clinic pipeline. This processing essentially applies intensity normalization and gradient unwrapping (bias-field non-linearity correction).

The MRI scans were then post-processed by UCSF using Freesurfer [4]. All the processing steps being applied to the scans are described in Section 2.1. The brain region was segmented, and from this point the grey/white matter and pial surfaces inferred. The former can help build a 3D surface of the brain. Cortical measures were then computed and mapped onto the 3D pial surface. This volume was finally inflated to exhibit hidden sulci measures and mapped onto a sphere. All these steps are illustrated in Figure 3.1. We mainly focus on the exploitation of cortical thickness measures in this project, but we also, at some point, take an interest in cortical volume measures.


(a) The initial MRI scans are registered into a common coordinate basis.


(b) The brain region is segmented from registered MRI scans.


(c) The grey/white matter surface is inferred to create a 3D surface of the brain. Cortical thickness measures can be extracted from local distance between pial surface and grey/white matter boundary, and mapped onto the 3D surface.


(d) To exhibit hidden sulci measures, the previous surface is inflated into a smoother version.

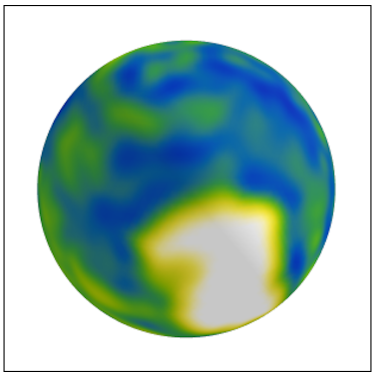
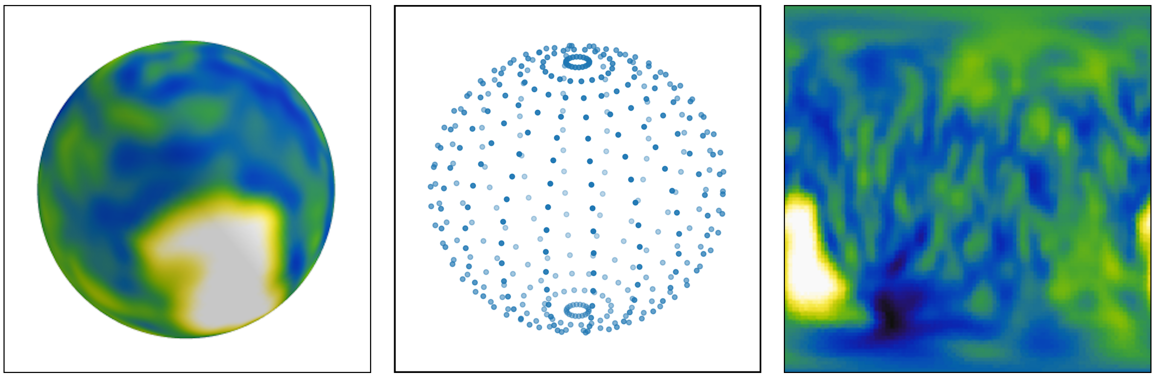

(e) The surface is mapped onto a sphere with minimal deformation to derive a proper spherical signal to feed spherical CNNs.

Figure 3.1: Processing steps from T1-weighted structural MRI brain scans to spherical measures of cortical thickness.

We further processed spherical signals to fit the standard input of Spherical CNNs (see Figure 2.11). To this purpose, spherical images were flattened onto a planisphere of size $2b_0 \times 2b_0$ determined by an input bandwidth parameter b_0 which sets up the input image resolution. This process is illustrated in Figure 3.2. Fully-processed brain scans become 2D images of size $2b_0 \times 2b_0$ (for a given b_0), depicting spherical cortical measures acquired originally from raw MRI images.



(a) The original image is a sphere made of cortical thickness measures as obtained by the process described on Figure 3.1.

(b) A spherical meshgrid of bandwidth b_0 is generated.

(c) The meshgrid is flattened onto a 2D regular grid to input the original image mapped on a planisphere.

Figure 3.2: Processing steps for mapping spherical measures of cortical thickness into a proper 2D input for SCNNs.

Chapter 4

Models

In this chapter, we detail the CNNs architectures used in this project, whether it be SCNN or PCNN. The last part of the chapter elaborates on the protocol we follow to train models and test their performance.

4.1 Spherical CNN

4.1.1 Layers

The following subsections hold all necessary information about the different types of layers used to build the SCNN architectures. The practical implementation of these layers is to be granted to Cohen *et al.* [11] using the theoretical aspects developed in Section 2.2.

S^2 convolutional layer

The input convolutional layer of the SCNN architecture is a S^2 convolution, as illustrated in Figure 2.11. As mentioned in Section 2.3.3, the actual input of such a layer is a 2D matrix, indexed in rows by $\beta \in [0, \pi]$ and in columns by $\alpha \in [0, 2\pi]$, respectively corresponding to rotation angles around the Y axis and the Z axis (Euler rotation $Z(\alpha)Y(\beta)Z(\gamma)$ with $\gamma = 0$). A pixel at indexes (i, j) in the 2D input matrix is then associated to the angles:

$$\begin{aligned} \alpha &= \frac{\pi j}{b_0} \\ \beta &= \frac{(2i+1)\pi}{4b_0}. \end{aligned} \tag{4.1}$$

The S^2 convolutional layer is defined by an output bandwidth and number of channels. To be consistent with Feng *et al.* notations, we note S2Conv(b, c) a S^2 convolutional layer of output bandwidth b and number of channels c. The c outputs are feature maps stored as 3D tensors (lying in the 3D manifold SO(3)) of size $2b \times 2b \times 2b$, indexed in rows by $\beta \in [0, \pi]$, in columns by $\alpha \in [0, 2\pi]$, and in the third dimension by $\gamma \in [0, 2\pi]$, corresponding to the Euler rotation $Z(\alpha)Y(\beta)Z(\gamma)$. A pixel at indexes (i, j, k) in one of these feature maps is then associated to the angles:

$$\begin{aligned}\alpha &= \frac{\pi j}{b_0} \\ \beta &= \frac{(2i+1)\pi}{4b_0} \\ \gamma &= \frac{\pi k}{b_0}.\end{aligned} \tag{4.2}$$

SO(3) convolutional layer

After a first S^2 layer, all the following convolutional layers are SO(3) layers. In the same way as the S^2 one, such a layer is defined by an output bandwidth and a number of channels. We note SO3Conv(b, c) an SO(3) convolutional layer of output bandwidth b and number of channels c. Both the input and output are 3D tensors lying in SO(3), with the same indexes conventions as described in the S^2 case.

Batch-Normalization layer

During the training process of a neural network, the distribution of every hidden layers' input is likely to change, as layers' weights are updated at each training step. As raised by Ioffe & Szegedy in [43], it makes the training more difficult. Indeed, let us consider a subnetwork computing

$$F_2(F_1(\boldsymbol{u}, \boldsymbol{\Theta_1}), \boldsymbol{\Theta_2}) \tag{4.3}$$

where F_1 and F_2 are arbitrary transformations performed respectively by one layer and the next one, Θ_1 and $\boldsymbol{\Theta_2}$ their weights, and $\boldsymbol{u}$ the input of the first layer. The learning scheme for the second layer is to update weights Θ_2 given the distribution of inputs $\boldsymbol{x} = F_1(\boldsymbol{u}, \boldsymbol{\Theta_1})$. However, the distribution of these inputs is likely to change as the training process goes on, which forces $\boldsymbol{\Theta_2}$ to adapt to these changes.

To overcome this issue, and therefore speeding up the training process, Ioffe & Szegedy thought about normalizing the input data of a given layer by adding a Batch-Normalization (BN) layer before the next one. In a MLP, if we consider that features might be decorrelated, the mean and variance of each feature x_i of the BN layer input $\boldsymbol{x}$ are computed over the entire batch of training samples. The normalized features are defined as:

$$x_i' = \frac{x_i - \mathrm{E}[x_i]}{\sqrt{\mathrm{Var}[x_i]}} \tag{4.4}$$

In a CNN, we would want all pixels of the same feature map to be normalized the same way, to stick to the idea of sharable weights in the convolution operation making all pixels similarly treated. To this purpose, the mean and variance are computed not only over the training batch but also over all the feature map dimensions (three for the 3D manifold SO(3) in our situation). If the BN layer input is composed of multiple channels, we would consider different mean and variance for each of them as channels should be seen as multiple different ways of analyzing the data.

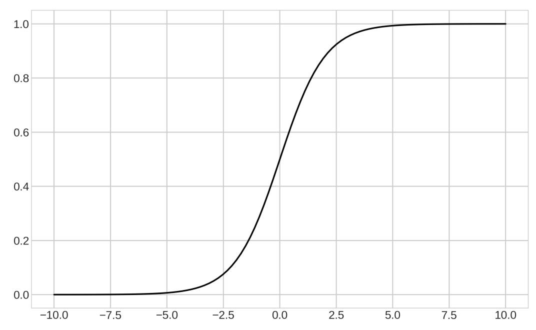
Ioffe & Szegedy suggest an additional step by adding two learnable parameters γ and β [1] such that the actual output of the BN layer would be $\boldsymbol{y} = (y_1, y_2, \dots)$ defined by

$$y_i = \gamma x_i' + \beta. \tag{4.5}$$

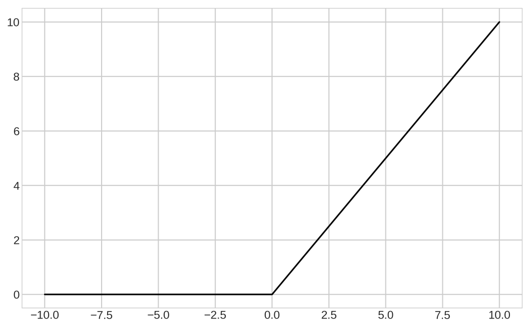
As the normalization step might modify what the layer can represent, this additional trick gives more freedom to the way the BN layer normalizes inputs. It especially allows it to learn the identity function. This way, we make sure that adding such a layer inside a neural network has little risks to worsen its performances.

Rectified Linear Unit activation function

As mentioned in Section 2.2.2, the power of Deep Learning comes, among other things, from the possibility to use non-linear activation functions between the output of a given layer and the input of the next one. The first function to be popularised in neural networks was the sigmoid (Figure 4.1a). However, this function has a significant issue when used in neural network training. The update of learnable parameters in these networks is done by what we call backpropagation. We will not go into the details of this process here, but the important thing to know is that it implies successive multiplications of backward chained gradients over the network layers, activation functions included. For this reason, if the input of a given sigmoid activation has a high absolute value, the corresponding gradient will be close to zero, and the weights of upstream layers will not update, stopping the learning process. To partly overcome this issue, a Rectified Linear Unit (ReLU) activation function can be used instead (Figure 4.1b), as it does not encounter such a problem for positive inputs. Besides, using ReLU speeds up the backpropagation process as its gradient is faster to compute [19].



(a) Sigmoid function.



(b) Rectified Linear Unit.

Figure 4.1: Non-linear activation functions.

[1]It should be noted that these parameters are distinct from the angles defined in the previous sections. We nonetheless use γ and β here to described the parameters to stick to Ioffe & Szegedy notations in [43].

SO(3) integration

For classification tasks as the ones we study here (labelling an entire image), we generally want, at a certain point of a CNN, to go down to output feature maps of one pixel, to then apply some fully-connected layers to exploit the resulting features in an MLP architecture. To this purpose, an integration over the SO(3) space can be performed to the outputs of a given SO(3) convolutional layer. In practice, in a discrete configuration, it goes down to applying a global average pooling (GAP), averaging values of each feature map over its three dimensions. However, to compensate for the non-uniformity of the SO(3) grid in the Y axis (α and γ are in $[0, 2\pi]$ while β is in $[0, \pi]$), a weighted GAP (wGAP) is used. The weight compensation along the Y axis is:

$$w_i = \frac{2}{b} \sin\left(\frac{(2i+1)\pi}{4b}\right) \sum_{k=0}^{b-1} \frac{1}{2k+1} \sin\left(\frac{(2i+1)(2k+1)\pi}{4b}\right), \tag{4.6}$$

applied to all elements of indices (i, j, k) (corresponding to β, α and γ respectively as described before), and where b is the bandwidth associated to the feature maps on which integration is performed. The derivation of this equation is provided by Kostelec & Rockmore in [44].

4.1.2 SCNN architectures

Feature extraction module

The feature extraction module used by Feng *et al.* (illustrated in Figure 4.2) is composed of an input S^2 layer followed by two SO(3) layers. In between, Batch-Normalization and ReLU activation are inserted. They start with an image of bandwidth 64, dividing it successively by 2 across the convolutional layers and multiplying the number of channels by 2 at the same time. The feature extraction module can be summarized as S2Conv(32, 32) - BN - ReLU - SO3Conv(16, 64) - BN - ReLU - SO3Conv(8, 128) - BN - ReLU - wGAP. To extend it to any arbitrary input bandwidth b, we have generalized its implementation into S2Conv(b/2, b/2) - BN - ReLU - SO3Conv(b/4, b) - BN - ReLU - SO3Conv(b/8, 2b) - BN - ReLU - wGAP. We compare computation time and performance of several input bandwidths in Section 5.3. In this project, we tested several variants of the global architecture.

Spherical Bilateral Unisequential architecture (SBU)

The output of the feature extraction module is a vector of size 2b, after wGAP integration of the last layer. Feng *et al.* chose to feed the feature extraction module with both left and right hemispheres, and to concatenate the resulting vectors to feed a final fully connected subnetwork, as shown in Figure 4.2. This subnetwork comprises a single layer with two output neurons. The resulting vector can be passed into a sigmoid function to generate a probability vector over the two classes: AD vs CN or MCI-p vs MCI-s). The effective classification decision of the network is obtained by considering the class with maximum output probability. We call this architecture SBU, for Spherical Bilateral (two hemispheres) Unisequential (weight sharing).

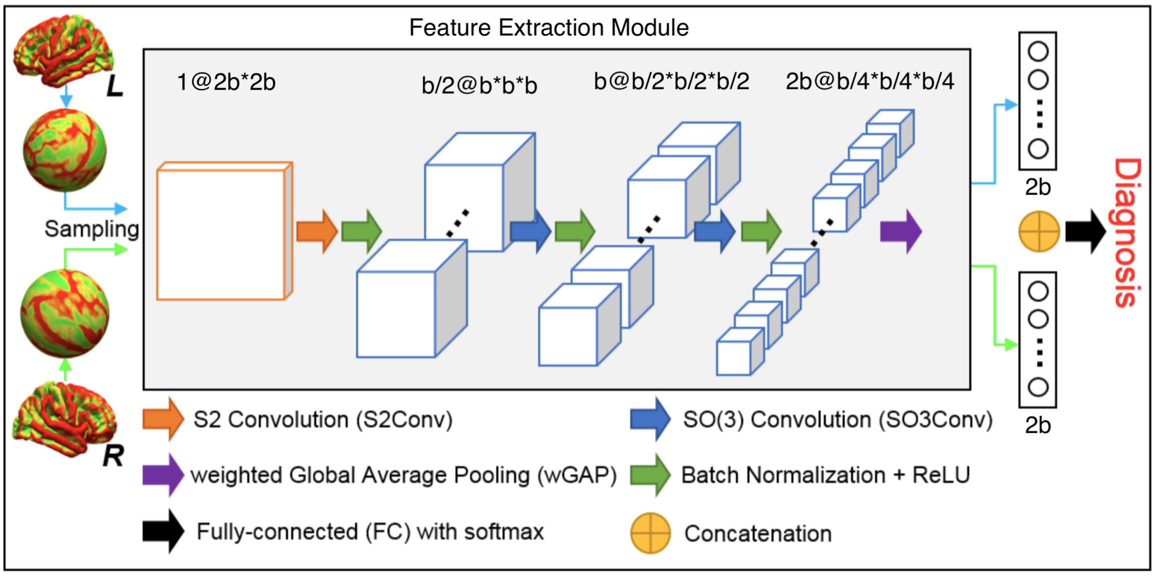

Feature Extraction Module
L
R
Sampling
1@2b*2b
b/2@b*b*b
b@b/2*b/2*b/2
2b@b/4*b/4*b/4
2b
2b
Diagnosis
S2 Convolution (S2Conv)
SO(3) Convolution (SO3Conv)
weighted Global Average Pooling (wGAP)
Batch Normalization + ReLU
Fully-connected (FC) with softmax
Concatenation

Figure 4.2: Illustration of the Spherical Bilateral Unisequential architecture (inspired from [1]). The notation a@b*c*d means the corresponding layer contains a feature maps of shape $b \times c \times d$.

Spherical Unilateral architecture (SU)

As sharing weights between both hemisphere might be problematic, we introduce an architecture taking only one of the two hemispheres as input (could be left or right), which we call SU for Spherical Unilateral (one hemisphere). The only difference between this architecture and the previous lies in the shape of the final fully-connected layer. As an SU model only takes one hemisphere as input, the input size of the fully-connected layer will be 2b, with no concatenation from a second hemisphere.

Spherical Bilateral Multisequential architecture (SBM)

The drawback of the previous architecture is the loss of information from the removal of one hemisphere. To stick to the idea of avoiding weight sharing but still using both hemispheres, we introduce a third architecture noted SBM, for Spherical Bilateral (two hemispheres) Multisequential (independent weights). In this model, two networks, made of the feature extraction module described above, are trained in parallel, one for each hemisphere. In practice, it does not change much: right and left hemispheres are going through their own convolutional subnetwork, and the resulting vectors after wGAP integration are concatenated to feed a final fully-connected layer. The difference will mainly come from the weights update being totally independent for each hemisphere.

4.2 Planar CNN

We now introduce a Planar CNN model (PCNN) as a comparison basis. To stick with Feng *et al.* architecture, we use 2D convolutional layers of stride 2 and kernel size 3, denoted Conv(c). As a consequence, to build a model with about the same number of parameters as its spherical equivalent, we slightly modify the architecture used by Feng *et al.*, and define the main PCNN block as Conv(b') - BN - ReLU - Conv(2b') - BN - ReLU - Conv(4b') - BN - ReLU (with $b' = 24$ for $b = 32$). As for the spherical architectures, this feature extraction module is followed by a fully-connected sub-network ending up in a binary probability vector. We use PCNN models with various architectures, as SCNNs (PU, PBU, PBM).

4.3 Training setup

4.3.1 Preparation of the data for cross-validation

Almost all of our experiments were made on the ADNI-1 cohort, as the data from the ADNI-2 cohort came later. We still dedicate a section in the next chapter to explore the possibilities offered by ADNI-2 (Section 5.8).

To evaluate their model's performances, Feng *et al.* decided to use a 10-fold cross-validation setting. They considered 10 models, each of them using 8 folds as training data, 1 as validation data to perform early-stopping during the training and avoid over-fitting, and the last one as a test set to get the final performance results. We have decided to instead go for 5-fold cross-validation in this project. Indeed, as we saw in the previous chapter, the ADNI-1 cohort is composed of 339 subjects for the AD-vs-CN task, and 250 for the MCI one. As a result, in a 10-fold setting, each fold is composed of respectively 34 (or 33) and 25 subjects. We consider that these numbers are too low to be able to make robust decisions. Indeed, a change in only one class prediction can make the classification accuracy change by respectively 3% and 4%. The impact is even twice as the validation accuracy on one fold is used to select a trained architecture by early stopping and the final performance is evaluated on another fold. To be able to compare ourselves to Feng *et al.*, we use roughly the same fold distribution. Detailed statistics of the splitting of ADNI-1 can be found in Appendix D.

4.3.2 Bagging

In addition, we actually do not consider cross-validation as a way of computing a model's performance for averaging the results of each trained model. We rather decided to combine our trained models and make them part of a "super" model. This model is obtained via an ensemble method called bagging. In short, to classify a sample, we aggregate the output probabilities of the trained models. In practice, the very final output probabilities are obtained by summing up the ones of every single model, and normalizing them back as a probability distribution. The idea behind a bagging method is that every single model is likely to make classification errors, but as they are not trained based on the exact same sets, the classification errors might be on different samples. In combining them, some of these errors will be ignored by the aggregation, and the overall performance increased. As we will see in the next chapter, this method allows a major breakthrough in performance results.

Besides, for each bagged model we trained, we decided to train 14 similar ones to explore multiple weight initializations. The idea was to decrease the bias that fixing a random seed can bring, while performing enough experiments to reduce the overall variance brought by not fixing the seed, in a reasonable computation time. The metric values given in Chapter 5 are obtained by averaging the 15 experiment performances.

4.3.3 Parameter initialization

Regarding the network parameters, we initialized the values with the one of Feng *et al.*, as our goal is first to replicate their findings. Thus, we started with a 0.1 learning rate, a learning rate decay of factor 10 at mid-experiment, and SGD with momentum as an optimizer, for a 200 epochs training and batches of size 8. The number of epochs is not of great importance as early stopping is performed. We only need to be sure not to underfit, but 200 epochs are sufficient as an training accuracy of 100% is always obtained at the end of the training.

4.3.4 Learning rate

We tested several strategies to tune the learning rate. We first tried multiple decay rates: a constant learning rate; a single-step decay, dividing the learning rate by 10 at mid-experiment; a multi-step decay, dividing the learning rate by 2 once every 40 epochs; a continuous exponential decay. We also tried several learning rate values: 0.3, 0.1, 0.05.

4.3.5 Bandwidth

We considered, in our experiments, changing the input bandwidth (originally 64 in Feng *et al.* settings) by decreasing it to 32 for computation reasons (see Section 5.3 for more details).

Chapter 5

Evaluation of performances & successive improvements

In this final chapter, we come back on all the experiments we made in throughout this project. In particular, in the first three sections, we come back on Feng *et al.* choices, either by validating them or slightly modifying them. From Section 5.4, we go further by investigating potential improvements.

5.1 Validating the use of SCNNs

5.1.1 Architecture comparison

From a theoretical point of view, using SCNNs is more suitable for handling spherical signals than using PCNNs. In this part, we look at this on a practical level by comparing various performance measures. Besides, as mentioned in Chapter 4, the weight sharing modality for both hemispheres used in the Bilateral Unisequential architecture might not seem evident, so we compare it to the Unilateral and Bilateral Multisequential ones.

Table 5.1 gathers the performances of every architecture in terms of accuracy, area under the ROC curve, sensitivity and specificity. These first results come from a simple cross-validation setting (no bagging).

AD vs CN cross-validation

Let us detail results for the AD-vs-CN classification task. In a Unilateral setup, PCNNs seem to perform a little better than SCNNs in exploiting left hemispheres (of about 1.5% in term of accuracy). However, PCNNs struggle with right hemispheres, with performances decreasing of about 4 points compared to left hemispheres. It could mean that right hemisphere relevant information might be more complicated to read, and especially that it might have more spherical dependencies. The latter idea might be confirmed by the SU performances on the right hemisphere, increased compared to left hemisphere use (unlike PU). As a result, combining both hemispheres in a Bilateral way is not that efficient with PCNNs, but makes the most of

Table 5.1: Performance measures (%) of multiple SCNN and PCNN architectures on a 5-fold cross-validation setting (green part for AD vs CN, blue part for MCI progression).

Architecture	ACC (std)	AUC (std)	SEN (std)	SPE (std)	ACC (std)	AUC (std)	SEN (std)	SPE (std)
PU (left)	80.81 (1.35)	88.06 (0.69)	76.98 (2.80)	85.38 (2.71)	62.12 (2.43)	66.34 (2.86)	69.88 (6.27)	52.68 (8.84)
PU (right)	76.49 (1.14)	84.52 (0.82)	75.32 (3.09)	77.90 (3.98)	58.76 (2.12)	63.04 (3.08)	69.58 (5.29)	45.58 (5.89)
PBU	80.44 (1.52)	88.41 (0.65)	73.06 (3.39)	**89.25 (3.25)**	61.70 (2.30)	66.14 (1.45)	72.20 (6.65)	48.91 (7.86)
PBM	80.73 (1.25)	88.86 (0.83)	75.90 (3.35)	86.51 (3.10)	61.08 (2.48)	65.33 (2.14)	70.54 (5.95)	49.57 (6.76)
SU (left)	79.29 (1.49)	88.06 (1.01)	76.44 (3.73)	82.69 (2.53)	**68.40 (2.09)**	**72.61 (1.52)**	73.04 (4.58)	**62.75 (5.38)**
SU (right)	82.08 (1.79)	89.71 (0.93)	80.32 (4.80)	84.19 (5.38)	61.77 (2.59)	66.78 (2.22)	70.71 (5.98)	50.87 (5.67)
SBU	**85.68 (1.22)**	**92.28 (0.99)**	**83.87 (3.89)**	87.85 (4.04)	64.67 (2.76)	69.11 (2.27)	70.48 (6.42)	57.61 (5.13)
SBM	85.00 (2.10)	92.08 (0.67)	82.61 (2.57)	**87.85 (3.60)**	67.32 (1.81)	71.69 (1.56)	**75.83 (5.30)**	56.96 (6.75)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

both hemisphere information for SCNN models.

SBU models reach an 85.68% accuracy, 92.28% AUC, 83.87% sensitivity and 87.85% specificity[1] on average. These numbers are slightly lower than those obtained by Feng *et al*. There might be two reasons for that: we are using a 5-fold cross-validation setting instead of a 10-fold one, and we average results over multiple runs while they decided to go with a fixed random seed. We are nonetheless confident that these two changes in the test setup produce more accurate results.

SBM models seem to perform slightly worse. While they potentially better capture differences between left and right hemispheres, SBU models make the most of left and right hemisphere similarities through weight sharing. Both architectures have their strengths, and it seems it does not change much in terms of performances.

MCI progression cross-validation

In the MCI progression task, we notice the same phenomenon about right hemispheres. It confirms that right hemisphere information is more difficult to interpret than left hemisphere's. However, distinguishing between MCI stable and progressive subjects seems too complicated for even the SCNN models to be able to handle right hemisphere data efficiently. As a consequence, SBU models, like PBU ones, are not able to efficiently combine both hemispheres. Top performances are obtained for a SU setup using left hemisphere, with a 68.40 accuracy, 72.61 AUC, 73.04 sensitivity and 62.75 specificity. Once again, the difference between our results and those of Feng *et al*. surely comes from the differences in the experiment setup.

The performances of SBM models further validate our readings on left/right hemispheres. Here, this architecture seems to perform significantly better than SBU ones, increasing accuracy and AUC by about 2.5%. While SBU struggles in combining left and right hemisphere by weight sharing, SBM brings a certain degree of freedom which improves the overall performance. Still, the improvement is not enough to reach the SU results on left hemispheres.

[1]Specificity measures are slightly lower than in a PBU setting. However, sensitivity is much better, so it seems that PBU classification tends to be biased towards CN. SBU is then definitely prefered to PBU.

Table 5.2: Performance measures (%) of multiple SCNN and PCNN architectures on a bagging setting (green part for AD vs CN, blue part for MCI progression).

Architecture	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
PU (left)	86.76	84.02 (2.14)	89.50 (1.22)	79.46 (3.51)	89.46 (3.75)	72.55	65.49 (4.85)	67.70 (3.99)	71.90 (10.18)	57.68 (13.19)
PU (right)	80.88	77.75 (1.56)	86.66 (0.99)	75.59 (3.36)	81.51 (3.10)	68.63	60.39 (5.04)	64.46 (3.48)	71.67 (8.15)	46.67 (11.79)
PBU	86.76	83.63 (2.21)	89.57 (0.98)	74.95 (3.46)	**93.98 (2.40)**	68.63	62.75 (4.06)	66.61 (2.77)	75.48 (5.03)	47.25 (9.70)
PBM	86.76	84.41 (1.99)	90.12 (1.23)	77.12 (3.21)	93.12 (3.42)	66.67	61.31 (3.10)	66.47 (3.30)	73.81 (7.23)	46.09 (10.62)
SU (left)	83.82	82.25 (1.71)	89.39 (1.12)	77.48 (2.64)	87.96 (2.85)	76.47	**72.29 (3.22)**	**75.33 (2.57)**	74.52 (7.13)	**69.57 (4.03)**
SU (right)	88.24	85.29 (2.42)	90.95 (1.04)	82.88 (5.56)	88.17 (5.27)	72.55	65.10 (4.21)	69.19 (2.91)	72.14 (9.18)	50.87 (5.67)
SBU	**94.12**	90.49 (1.66)	**94.08 (1.08)**	**88.47 (3.61)**	92.90 (4.43)	**78.43**	67.97 (4.95)	72.14 (2.35)	71.19 (6.94)	64.06 (7.25)
SBM	92.65	**90.88 (1.49)**	93.73 (0.86)	**88.47 (1.90)**	93.76 (3.33)	76.47	69.80 (3.54)	75.02 (2.28)	**80.48 (9.04)**	56.81 (11.56)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

Bagging

Table 5.2 gathers results obtained when combining in a bagging way the 5-fold trained models. As one can see, this method induces a significant performance increase. In average, best architectures for the AD-vs-CN task see their results increase of about 5% in accuracy, 2% in AUC, 4.5% in sensitivity, 5% in specificity, while the increase for the MCI tasks is about 4% in accuracy, 3% in AUC, 1.5% in sensitivity, and 7% in specificity. In addition, accuracy for the best models among 15 performed runs can go up to 94.12% for the first task and 76.47% for the second one[2].

In the AC-vs-CN setting, the tiny gap between SBM and SBU vanishes. SBM performs even slightly better in terms of accuracy, but its AUC stays lower. As the maximum accuracy is reached by an SBU model (94.12% against 92.65%), we decided to stick to this architecture. To recap, given what we saw in this Section, we decided to go with the SBU architecture for AD-vs-CN, and SU (left hemisphere) for MCI-progression.

5.1.2 Rotational equivariance

We have validated the use of spherical models in terms of performances. However, we need to remember that these architectures were initially introduced for preserving rotational equivariance. It is well known that theory and practice might not always perfectly match. In Chapter 2, we proved the theoretical rotational equivariance; let us see if it can be validated in practice.

We randomly selected two trained models, a PCNN and an SCNN (on the AD vs CN task only, no need to consider both tasks to try rotational equivariance). Lines 1 and 3 of Table 5.3 give the performance measures of each model. We then computed 15 times the test results by using the original test cohort, except we applied to each spherical image a rotation $Z(\alpha)Y(\beta)Z(\gamma)$, with α, β and γ all randomly chosen in $[0, \frac{\pi}{10}, \frac{\pi}{8}, \frac{\pi}{6}]$. As shown by Table 5.3, the spherical model's performances are almost unaffected by the performed rotations, with a less than 0.5% gap in accuracy and

[2]One can notice that the SBU architecture can outperform the SU one in the MCI task. We attribute it to chance, and still consider SU architectures to be better (besides, a model with a 78.43% accuracy rather than 76.47% classifies only one extra subject correctly).

Table 5.3: Evaluation of rotational equivariance in Planar and Spherical models.

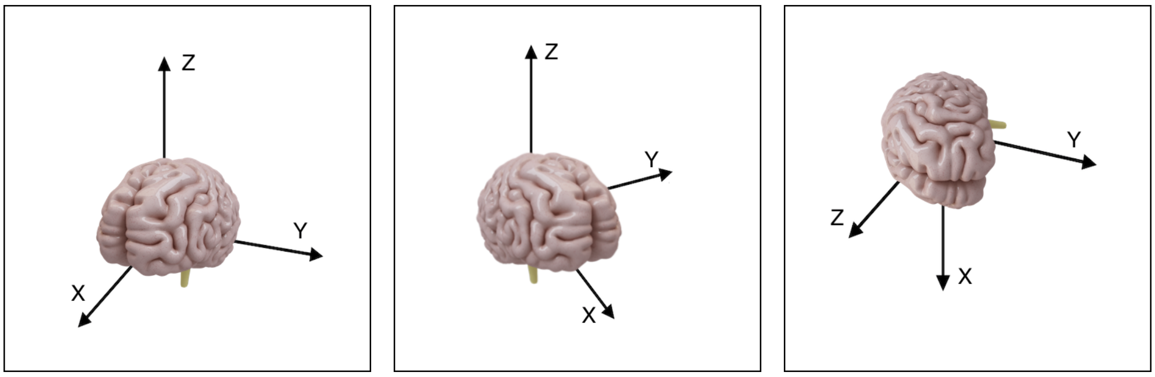
Architecture	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
Planar - Non rotated Test	-	85.29	90.67	78.38	93.55
Planar - Rotated Test	76.47	68.53 (5.06)	75.07 (4.44)	75.14 (6.15)	60.65 (6.92)
Spherical - Non rotated Test	-	91.18	95.64	89.19	93.55
Spherical - Rotated Test	92.65	90.78 (1.41)	95.42 (0.35)	91.35 (1.12)	90.11 (2.27)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

AUC, while these scores decrease by about 15% for the planar model.

The average difference in output probabilities between an original sample and a rotated one is about 12% for spherical models, while being about 37% for planar ones. Given these numbers, we get that predictions in a planar architecture tend to switch a lot more than in a spherical one. In a way, spherical models are 3 times more robust to rotations than planar ones.

Besides, we initially thought that the slight lack of rotational equivariance of the SCNN might be imputed to the last fully-connected layer. However, we looked at the feature maps at the end of the convolutional block, and found out that values in these feature maps differed in average by 13% from an original sample to a rotated one[3]. Thus, it seems than in practice, even the convolutional block in itself is not perfectly rotational equivariant. Still, given the relatively low decrease in performance using rotated samples, we consider the rotational equivariance of SCNNs to be proven.



(a) Original reference frame. **(b)** Z-axis rotation. **(c)** Y-axis rotation.

Figure 5.1: Rotating a brain around a horizontal axis modifies its spacial structure more than a rotation around a vertical axis.

As an aside, we also noticed that SCNNs are more robust against rotations on the Z axis than on the Y axis. This property might be due to brain spacial symmetries. Indeed, a rotation around the Y axis profoundly modifies the spacial structure of the brain compared to a Z-axis rotation (see Figure 5.2).

[3]As one would expect errors to propagate throughout a network, it might seem surprising that this number is higher than errors for the final output. Actually, 12% was an absolute percentage, while 13% was a relative one. Therefore, these numbers do not relate and no analogy should be made.

Table 5.4: Performance measures (%) of various learning rate decay settings (green for AD-vs-CN on SBU, blue for MCI-progression on SU with left hemisphere).

Decay pattern	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
Constant	**94.12**	**91.67 (1.54)**	**94.61 (1.22)**	**88.47 (2.39)**	**95.48 (2.04)**	74.51	70.72 (2.51)	73.51 (2.06)	73.57 (6.45)	65.58 (3.74)
Single-step	**94.12**	90.49 (1.66)	94.08 (1.08)	**88.47 (3.61)**	92.90 (4.43)	76.47	**72.29 (3.22)**	**75.33 (2.57)**	**74.52 (7.13)**	**69.57 (4.03)**
Multi-step	92.65	89.80 (1.52)	93.53 (1.27)	87.39 (3.34)	92.69 (3.33)	76.47	71.63 (3.22)	74.64 (2.04)	75.48 (7.74)	66.96 (5.88)
Exponential	91.18	89.22 (1.44)	93.26 (1.08)	86.13 (2.01)	92.90 (2.78)	**78.43**	72.16 (2.98)	74.62 (2.41)	74.29 (6.91)	**69.57 (5.20)**

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

5.2 Learning rate

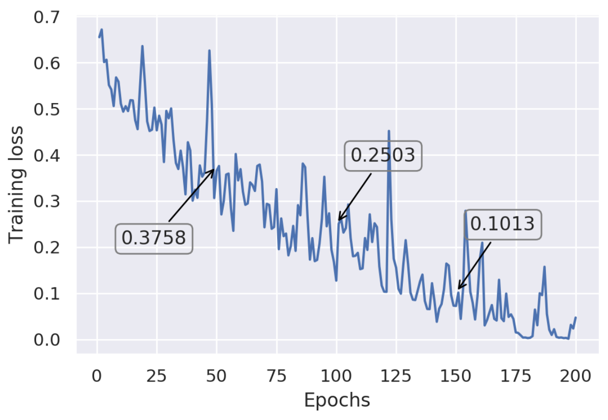
Throughout this project, we studied multiple decay rates, as described in Section 4.3.4. As illustrated in Figure 5.2b, the training loss of the initial training setup (single-step decay at mid-experiment) is really noisy. This is generally due to a too high learning rate, so we decided to extend the initial one-step decay to multiple steps, and even further to an exponential decay ("infinite" number of steps). The idea behind this is to impose lower learning rates earlier in the training process, rather than a single significant decrease relatively late. As depicted in Figures 5.2c-5.2d, the more decay steps we use, the smoother is the training loss.

AD vs CN

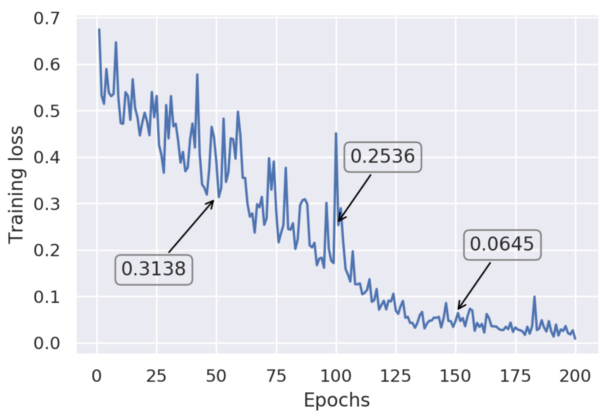
However, if we compare the genuine performances of each setting, we notice that the less steps we use, the better is the test performance (see Table 5.4), as least for the AD vs CN task.

Given this trend, we therefore thought about decreasing the decay rather than adding steps, by using a constant learning rate. As shown on Table 5.4 and Figure 5.2a, the trend is confirmed. As we generally prefer a smooth learning curve, it can be surprising that the performance metrics show an opposite trend. A possible explanation might be that brain images are so complex that the loss function is highly non-convex, with a lot of local minima. Under this assumption, a noisier learning curve signifies a higher number of visited local minima. One might argue that high learning rates (in the case of a constant learning rate for example) cannot finely dive into tight minima. Actually here, it might not be that necessary to finely achieve a low loss, as a 100% training accuracy is always obtained and early-stopping chooses models that do not achieve a particularly low training loss. For these reasons, it does not seem necessary to decrease the learning rate during training; rather, we chose to use a constant learning rate given classification results.

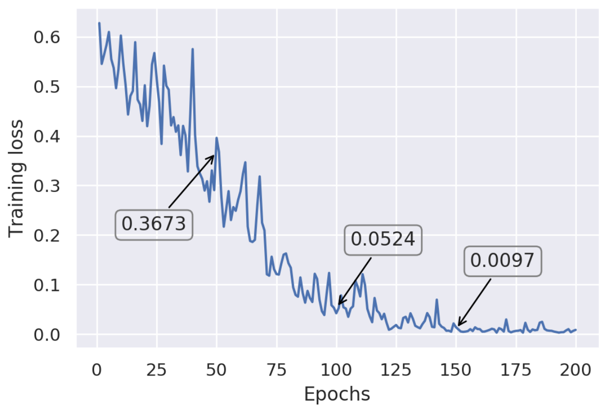
Besides, we tried several learning rates. Apart from 0.1, we tested 0.3 and 0.05. A 0.1 learning rate seems a good compromise, as performance deteriorates when we either increase or decrease the learning rate. Detailed metrics can be found in Appendix (Table G.1).



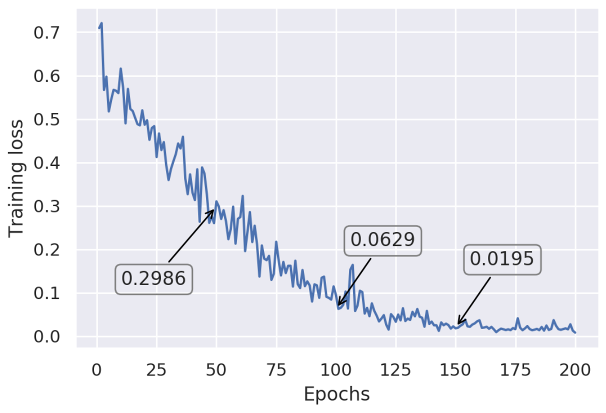
(a) Constant learning rate.



(b) Single-step decay.



(c) Multi-step decay.



(d) Exponential decay (continuous).

Figure 5.2: Effect of various decay rates on the training loss; pointers at epochs 50, 100 and 150. Each time, the curve comes from the first trained model out of the four in the 5-fold training setup (see Appendix **??** for additional figures on every model).

MCI Progression

While the various learning decay settings behave the same in terms of training loss curve, performance metrics comparison is not as straightforward as in the AD-vs-CN case. As depicted by Table 5.4, single-step and exponential rate decays globally perform better than other decay modalities, with single-step achieving slightly better performance . Therefore, there does not seem to have any linear relationship between decay rates and performances for the MCI task. It can partly be explained by the low number of samples per fold (around 50) and the relative complexity of the MCI task compared to the AD-vs-CN one. Thus, it is difficult to come to any robust conclusion here. For this reason, we decided to keep to both decay settings.

We tried various learning rate values: 0.3, 0.1, 0.05, for both decay settings. As depicted on Table G.2, the results are one again not unequivocal. We do not necessarily expect some sort of trend as for the decay settings. But, we cannot

[3]One can notice that the exponential rate decay setting can outperform the single-step one in the MCI task. As notified earlier, given the number of test samples, this difference is actually slim.

even tell with confidence which learning rate is really better. A learning rate of 0.05 seems slightly better for an exponential rate decay setting, while a learning rate of 0.3 seems better for a single-step one. Given the results instability, we therefore decided to stop the MCI-progression experiments. SU models (left hemisphere) with a 0.3 learning rate, single-step decay, achieve for the MCI task a 73.99% accuracy, 74.73% AUC, 80.95% sensitivity, 65.51% specificity in average over 15 runs, and a maximum 76.47% accuracy. SU models (left hemisphere) with a 0.05 learning rate, exponential rate decay, achieve for the MCI task a 72.81% accuracy, 73.14% AUC, 74.76% sensitivity, 70.43% specificity in average over 15 runs, and a maximum 78.43% accuracy. We come back on the MCI task in Section 5.8, where the ADNI-2 cohort is introduced.

5.3 Input bandwidth

The experiments of the past sections were made with an input bandwidth parameter set to 32, for computational reasons. We decided to also investigate bandwidth parameters 16, 64 and 128. Table 5.6 gathers information about computation times. While models in a 16 or 32 bandwidth setting can be trained in a reasonable amount of time (respectively 2.5 hours and 5 hours), computation times for bandwidth parameters 64 and 128 were too large for the number of experiments we performed during the project, even for just a single one in a 128 setting.

We nonetheless compared the performance results of each bandwidth setting (Table 5.5), except for a bandwidth of 128. The higher the bandwidth, the better the performances. Such a result is expected, as a higher bandwidth means a better input resolution in practice. While we consider the trade-off computation time / performance to be better in a 32 bandwidth setting than a 16 one, we cannot say the same for 64 vs 32. As notified before, the overall computation time of a single experiment in a 64 setting would have prevented us from conducting all the experiments of this project. Nevertheless, in the parameter setting defined in the past sections (SBU with a constant 0.1 learning rate), increasing the bandwidth parameter from 32 to 64 improves the performance. In particular, the average accuracy increases by 0.5% in average (to 92.16%), and some models achieve a 95.59% accuracy.

We pursued further experiments with a 32 bandwidth setting, but we need to keep in mind that input images with bandwidth 64 (and potentially 128) improve classification results.

Table 5.5: Performance metrics (%) of various bandwidth settings in the AD-vs-CN task (SBU architecture, 0.1 learning rate, no learning rate decay).

Bandwidth	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
16	92.65	90.00 (1.69)	**94.69 (0.79)**	88.29 (2.21)	92.04 (4.02)
32	94.12	91.67 (1.54)	94.61 (1.22)	88.47 (2.39)	**95.48 (2.04)**
64	**95.59**	**92.16 (1.90)**	94.59 (1.02)	**89.73 (2.93)**	95.05 (2.69)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

Table 5.6: Computation time of various bandwidth settings on a NVIDIA TITAN RTX 24GB GPU. ”Iteration” means each batch training step, second column exhibits overall training time over 15 runs.

Bandwidth	Time per iteration	Overall 15 bagging runs’ time
16	0.02 seconds	2.5 hours
32	0.02 seconds	5 hours
64	0.2 seconds	15 hours
128	2 seconds	31 days

5.4 Cortical features

In addition to cortical thickness measures, we investigated the use of volume measures. The first two lines of Table 5.7 exhibit the performance metrics of architectures trained on cortical thickness images, or cortical volume ones. Both achieve similar results, which is not surprising given the high correlation between thickness and volume (see Figure 5.3).

We thought also about combining them, by inputting two channels, one for each image feature. In addition, we examined enhanced architectures. As combining features makes the amount of information double, we increased the number of channels in the layers of the feature extraction module. We can see on Table 5.7 that using the initial architecture and inputting both features achieves lower results (a loss of about 0.8% in accuracy, sensitivity, specificity, and 0.4% in AUC, while no model reaches a 94.12% accuracy). However, the performance improve as the number of channels in hidden layers increases, up to three times more channels than the initial setup. It improves metrics values up to 1.3%.

What can we learn about this? First, the network’s initial complexity might not be sufficient to correctly learn about the data of two input channels. However, if we increase its complexity too much, it starts to have difficulty in training.

We were able to find configurations were adding cortical volume data improves performances, while its correlation with cortical thickness is high. Therefore, we think that adding complementary features, such that surface curvature, might be even more efficient.

Table 5.7: Performance metrics (%) of various cortical feature combinations in the AD-vs-CN task (SBU architecture, 0.1 learning rate, no learning rate decay).

Features	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
Thickness	**94.12**	91.67 (1.54)	94.61 (1.22)	88.47 (2.39)	95.48 (2.04)
Volume	**94.12**	91.08 (3.22)	94.58 (1.38)	87.39 (5.07)	95.48 (2.38)
Thick + Vol	92.65	90.88 (1.59)	94.18 (0.88)	87.75 (2.87)	94.62 (1.99)
Thick + Vol (x2 channels)	**94.12**	91.57 (1.88)	94.85 (1.13)	88.11 (3.79)	95.70 (1.99)
Thick + Vol (x3 channels)	**94.12**	**92.65 (1.24)**	**95.11 (0.64)**	**89.73 (1.83)**	**96.13 (1.34)**
Thick + Vol (x4 channels)	**94.12**	91.08 (1.88)	94.70 (1.15)	87.21 (3.30)	95.70 (1.57)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

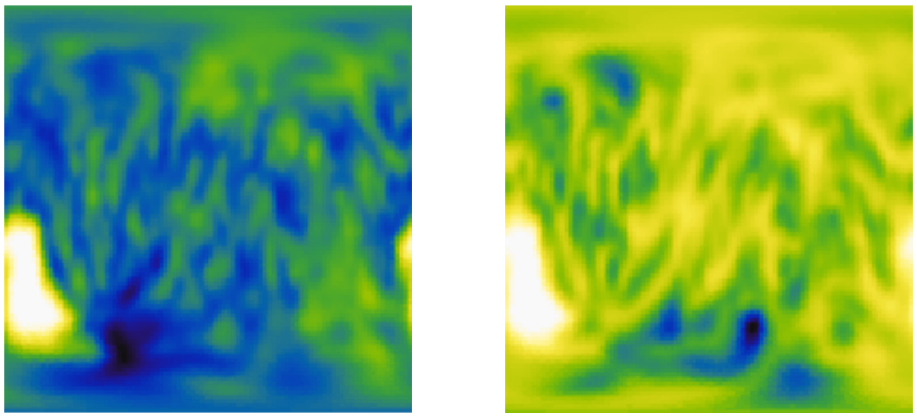


Figure 5.3: Illustration of cortical thickness (left) and volume (right) measures.

5.5 Hemispheres combination

In the brain, left and right hemispheres are mirrored. The pre-processing pipeline preserves this symmetry, so that hemispheres inputted in the SBU architecture are not registered, as illustrated in Figure 5.4.

Intuitively, it seems surprising that weight sharing with unregistered images performs well. Therefore, we mirrored the right hemisphere to get it in the same reference frame as the left one (and similarly we investigated registration on the right hemisphere). The first part of Table 5.8 shows that registration worsen the performance (of about 0.8% to 1.7% depending on the metric). A possible explanation might be that registered images are too similar, in that SBU models see these images as representing the same entity, and do not understand the slight difference which remains between the two hemispheres.

We also tried to combine hemisphere in a more readable way. We decided, instead of inputting left and right hemisphere images as so, to combine them into their sum and difference: left+right, left-right. We followed the intuition that both hemispheres are similar, but slight differences might show spacial heterogeneity in Alzheimer's disease. We were careful to register hemispheres before combining

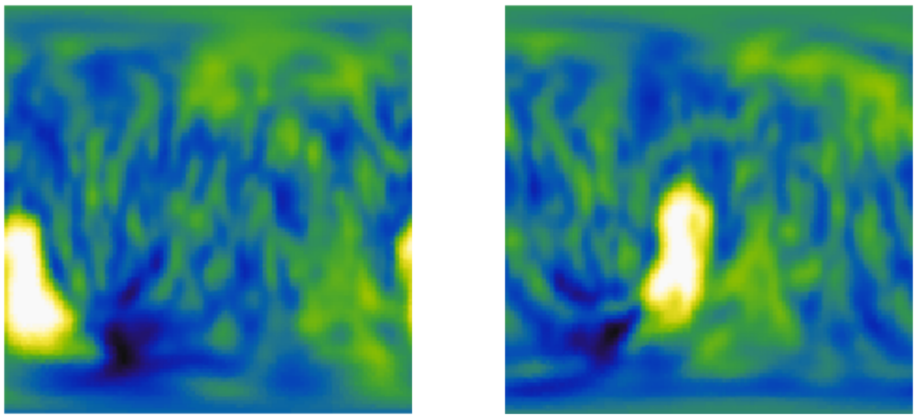


Figure 5.4: Cortical thickness measures of left and right hemispheres.

Table 5.8: Performance metrics (%) of various hemisphere combinations in the AD-vs-CN task (SBU architecture, 0.1 learning rate, no learning rate decay).

Combination	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
None (original)	**94.12**	**91.67 (1.54)**	**94.61 (1.22)**	**88.47 (2.39)**	**95.48 (2.04)**
Registration on left hemisphere (RL)	**94.12**	90.00 (1.86)	93.79 (0.85)	86.31 (3.61)	94.41 (2.85)
Registration on right hemisphere (RR)	92.65	90.20 (1.64)	93.77 (0.96)	86.85 (2.68)	94.19 (2.50)
Combination left+right, left-right (C)	85.29	76.76 (4.24)	86.50 (2.42)	69.55 (10.29)	85.38 (8.08)
C + RL	85.29	76.76 (4.13)	87.18 (2.60)	69.73 (10.17)	85.16 (6.88)
C + RR	82.35	77.35 (3.84)	85.99 (2.52)	74.59 (9.01)	80.65 (7.32)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

Table 5.9: Performance metrics (%) of various hemisphere combinations in the AD-vs-CN task (SBM architecture, 0.1 learning rate, no learning rate decay).

Combination	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
None (original)	**94.12**	90.59 (1.55)	94.35 (0.76)	**87.39 (3.78)**	94.41 (4.30)
Registration on left hemisphere (RL)	**94.12**	**91.37 (2.07)**	94.01 (0.87)	87.21 (4.27)	**96.34 (1.67)**
Registration on right hemisphere (RR)	**94.12**	90.78 (3.17)	**94.41 (1.05)**	**87.39 (5.56)**	94.84 (2.38)
Combination left+right, left-right (C)	91.18	88.24 (1.76)	93.77 (0.76)	83.24 (3.10)	94.19 (2.78)
C + RL	91.18	88.73 (1.73)	93.91 (1.01)	85.05 (3.94)	93.12 (3.19)
C + RR	92.65	89.22 (1.54)	93.44 (0.97)	84.50 (4.02)	94.84 (2.94)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

them. We investigated three situations: leaving the combinations registered onto the left hemisphere, onto the right one, or having them unregistered back. Combinations are illustrated in Appendix, additional Figure F.5.

As show on the second part of Table 5.8, such combinations highly deteriorate the SBU architecture performance. The first explanation we see is that the combination produces images that are too different (Figure F.5) for being correctly exploited by a SBU architecture with weight sharing. In addition, the registration might not actually be as straightforward as a simple mirroring. Figure 5.5 shows a possible transformation to register hemispheres with precision.

SBM architecture

We then switched to a SBM architecture to prevent the flaws of weight sharing. The same combination experiments have been performed and are gathered in Table 5.9. With such an architecture, results on registration are improved (about 1%-2%), to even outperform the original SBM configuration (but still do not outperform the original SBU configuration). We thought that SBU performance is deteriorated by registration because images are too similar for the network to understand that they represent two different hemispheres. It seems this hypothesis holds given the results on SBM. While the network exploits efficiently the registered image similarities, the

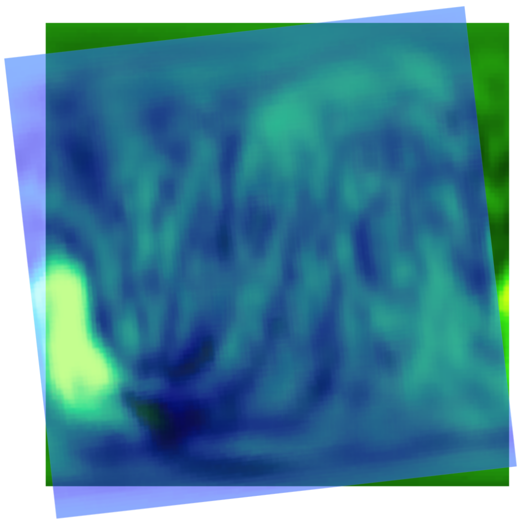


Figure 5.5: Possible registration between left and right hemispheres.

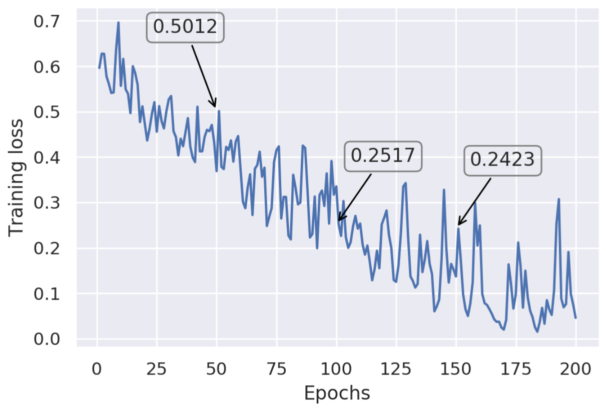
multisequential setting offers a certain degree of freedom to understand the slight remaining variations.

Improvements on combinations are even better: accuracy increases by 12% compared to combinations used in a SBU architecture. Overall, they remain lower than in the unregistered uncombined configuration, but it validates the theory on combined images being too unalike for weight sharing to work properly.

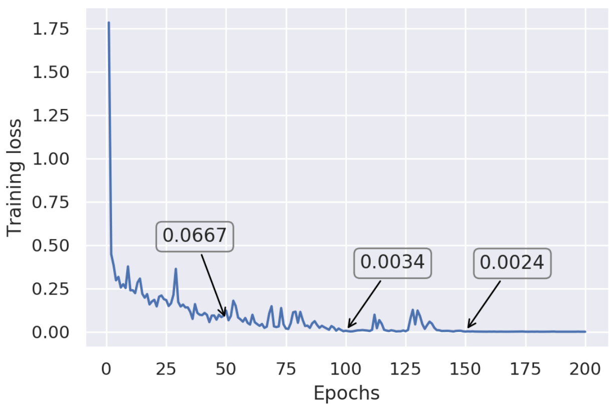
In short, SBU performs better than SBM in the original configuration, but SBM possibly have more potential. Therefore, rather than training SBM models from scratch, we used SBU pretrained weights for using SBM as a finely tuning model. We tried multiple configurations: SBU with unregistered input images, registered, combined, or registered and combined..., and similarly for SBM. We could not find a setting achieving outstanding performances. The effect of pretraining is nonetheless visible; as illustrated on Figure 5.6, the training loss decreases a lot faster when SBM network's weights are initialized with pretrained SBU weights. However, as exposed before, this kind of learning curve might not be suitable for the highly non-convex training loss that medical imaging entails. It might explain why pretraining does not seem to induce performance improvement.

5.6 Longitudinal features

After that, we investigated the use of longitudinal features. As shown on Table 3.2, follow-up visits at 6, 12 and 24 months after baseline can be exploited. We considered every combinations of these follow-ups: 6 months, 12 months, 24 months, 6 & 12, 6 & 24, 12 & 24, 6 & 12 & 24. Images from each follow-up visit are inputted as supplementary channels, in addition to baseline. Table G.3 in Appendix gathers the results for every combination. Every one of them reaches lower performance than baseline only. We impute these results to two factors. On the one hand,



(a) Training from scratch.



(b) Pretrained weights from SBU.

Figure 5.6: Effect of initializing SBM models with pretrained SBU weights.

the more we consider follow-up visits, the fewer are the available subjects. Cohorts decrease, and do not decrease evenly throughout the split folds. Fold statistics are deteriorated, which decreases the training quality. On the other hand, we saw earlier (Section 5.4) that the original feature selection module architecture might not be complex enough to handle multiple input channels.

5.7 Exploiting atlases

To better understand the decisions of our models, we decided to exploit the cortical atlases introduced in Section 2.1.3. As a reminder, the Desikan atlas defines a 36-parcel segmentation of the brain, while the Destrieux atlas suggests 75 parcels. To understand what lies behind our models' decision process, we trained multiple models (always in 15-run setting), one for each structure (by masking the input signal for every given parcel). This way, we obtained detailed performances for each individual structure. In addition, we crossed our results with those obtained by Yang *et al.* in [31] and Filho *et al.* in [38]. Classification accuracy for each individual structure of the Destrieux atlas are displayed on Figure 5.7 and those of the Desikan atlas on Figure 5.8.

We can see that both atlases include structures which individually achieve more than 80% accuracy, up to 85% for the best ones. The structure achieving higher performance in the Destrieux atlas accounts for 3.5% of the total 2D original image, and respectively 3.8% for the best one in the Desikan atlas (see Figure F.6 in Appendix). These results open up interesting possibilities. If the structural segmentation cannot be done properly, because of a deteriorated MRI scan for instance, we could potentially still be able to make predictions with high confidence.

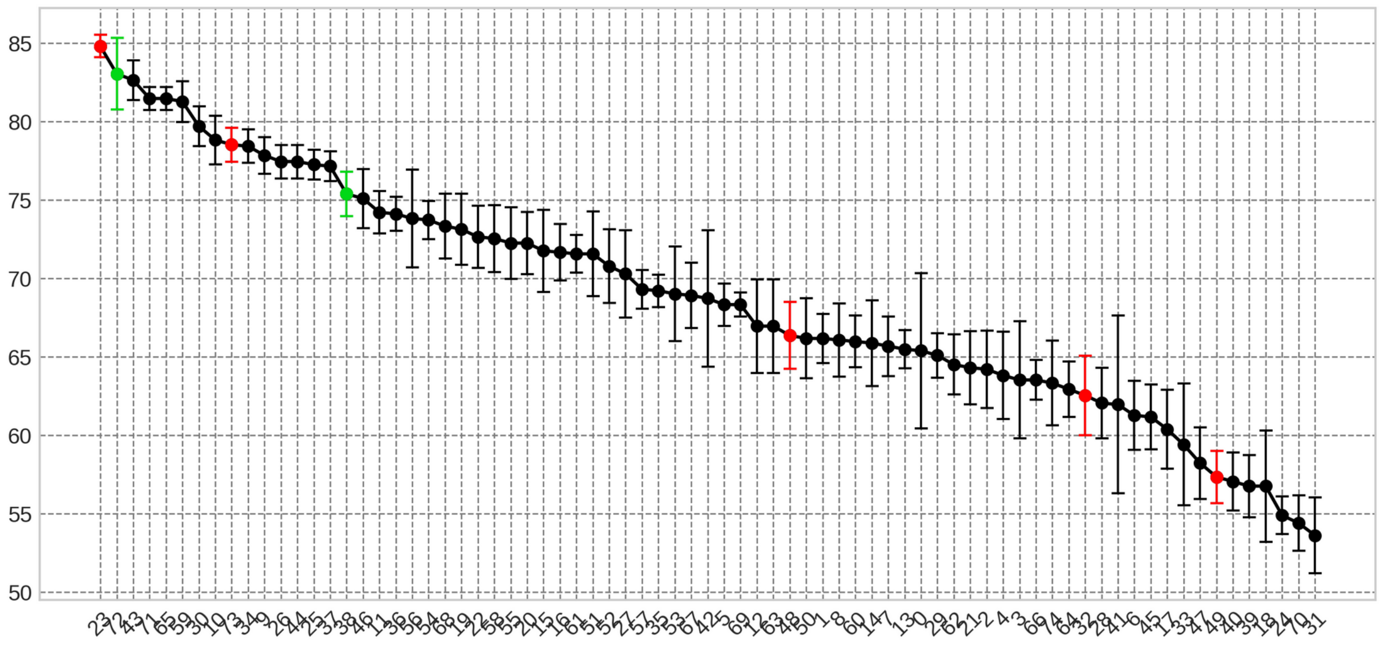


Figure 5.7: Individual accuracy of Destrieux structures. Circles correspond to mean test accuracy after having trained a SBU model (over 15 runs), with a constant 0.1 learning rate, while inputting masked versions of the original images to only keep a given structure. Error bars denote the standard deviation. In green: structures highlighted by Yang *et al.* as having high correlation with Alzheimer's disease classification. In red: By Filho *et al.* See Appendix for structure names.

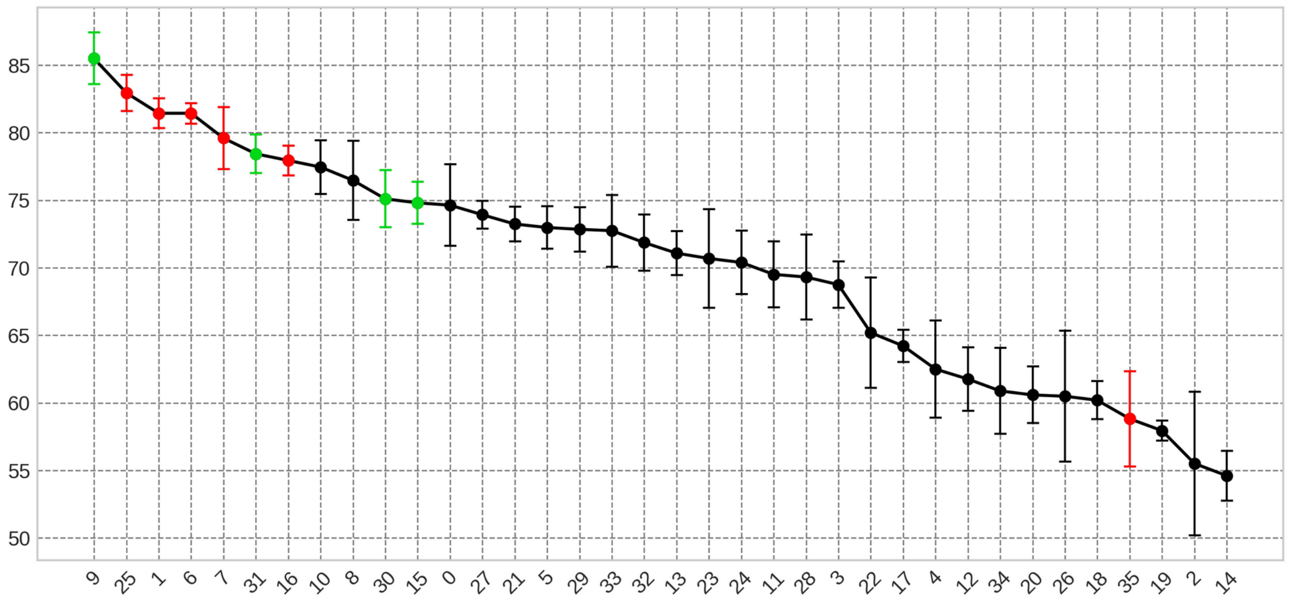


Figure 5.8: Individual accuracy of Desikan structures. Circles correspond to mean test accuracy after having trained a SBU model (over 15 runs), with a constant 0.1 learning rate, while inputting masked versions of the original images to only keep a given structure. Error bars denote the standard deviation. In green: structures highlighted by Yang *et al.* as having high correlation with Alzheimer's disease classification. In red: By Filho *et al.* See Appendix for structure names.

One can notice that previously identified structures (being linked to Alzheimer's disease by the two cited papers) do not particularly relate to those we can find for the Destrieux atlas. On the contrary, the best Desikan parcels, in terms of accuracy, match well. It probably comes from the 32 bandwidth setting. The Destrieux atlas is made of small and fine structures; as a 32 bandwidth deteriorates the input resolution (and therefore the atlas resolution), the Destrieux atlas might be relatively impacted, unlike Desikan's. This phenomenon can be seen on the classification accuracies heat map in Appendix (Figure F.7).

Therefore, this experiment would gain from increasing the bandwidth. Still, results remain interesting. We could thought for example to use our solution on bad-quality MRI scans, by masking pre-processed 2D images with some of the best structures we highlight here, and expect a confident prediction.

5.8 Adding data: ADNI-2 cohort

Table 5.10: Performance metrics (%) of ADNI-1 vs ADNI-2 in the AD- vs-CN task (SBU architecture, 0.1 learning rate, no learning rate decay).

Cohort	ACC (std)	AUC (std)	SEN (std)	SPE (std)
ADNI-1 (cross-validation)	86.10 (2.15)	92.79 (1.17)	82.84 (3.33)	90.00 (3.08)
ADNI-1 (bagging)	**91.67 (1.54)**	94.61 (1.22)	**88.47 (2.39)**	**95.48 (2.04)**
ADNI-2 (cross-validation)	87.29 (1.12)	94.55 (0.76)	86.01 (2.87)	88.58 (2.81)
ADNI-2 (bagging)	90.36 (0.88)	**95.80 (0.61)**	88.31 (1.85)	92.45 (2.41)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

Table 5.11: Performance metrics (%) of various cohorts in the MCI-progression task (0.1 learning rate, no learning rate decay).

Combination	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
CC - SU (left)	72.62	67.30 (2.46)	72.00 (2.10)	73.81 (7.48)	60.79 (7.36)
CC - SBU	72.62	66.67 (2.74)	69.85 (1.70)	70.95 (7.70)	62.38 (7.91)
CS2 - SU (left)	75.00	70.16 (2.27)	78.29 (1.34)	**75.08 (6.79)**	65.24 (7.83)
CS2 - SBU	78.57	**74.84 (2.88)**	**80.80 (2.47)**	70.16 (6.90)	**79.52 (6.22)**
CS4 - SU (left)	75.00	69.20 (2.51)	76.89 (2.18)	73.43 (6.33)	64.79 (5.98)
CS4 - SBU	**79.52**	72.69 (3.27)	79.75 (2.62)	68.25 (5.45)	77.24 (4.09)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

One of the main reasons for which we went for a 5-fold setting instead of a 10 one was the small size of the ADNI-1 cohort, which lower the confidence we have in the obtained results. We can even go further by switching cohorts to go for a larger one. We acquired baseline scans for the ADNI-2 cohort, which includes more subjects than ADNI-1.

AD vs CN

Table 5.10 gathers performances for the AD vs CN task on ADNI-1 and ADNI-2 cohorts. Performances of the latter are lower than those of the former on a bagging setup, but higher by using cross-validation. It means that the ensemble method, on the ADNI-2 cohort, loses efficiency. Actually, the ADNI-2 cohort is bigger than ADNI-1. As data augmentation reduces variance, a part of it is already removed when bagging is applied, so that the performance increase is not as good as for the ADNI-1 cohort.

If the remaining error is not due to variance, it must be a bias error. Therefore, using a boosting ensemble method rather than a bagging one should pay off. We will not linger over boosting as we did not implement it during the project, but we keep it in mind for future work.

MCI progression

In this part, we come back on the MCI-progression task we dropped from Section 5.3. As the ADNI-2 cohort is larger than ADNI-1, variance is necessarily reduced and stability is improved. As described in Section 3.1.2, we consider 3 balanced cohorts, CC, CS2, CS4. As reminder:
- CC contains MCI subjects having progressed in AD in at most 2 years (MCI-p), and subjects having either never converted (while being monitored for at least 2 years) or converted only after 2 years (MCI-s).
- CS2 contains MCI subjects having progressed in AD in at most 2 years (MCI-p), and subjects who never converted (while being monitored for at least 2 years, MCI-s).
- CS4 contains MCI subjects having progressed in AD in at most 2 years (MCI-p), and subjects who never converted (while being monitored for at least 4 years, MCI-s).

Table 5.11 gathers some performance results for the three cohorts, on both SU (left hemisphere) and SBU architecture. The experiments have been conducted with a learning rate of 0.1 without decay. We chose this parameter setting as it is the one maximizing the performances. We do not display the others here, as relative performances along the table are similar.

The intuition about the MCI-progression task on the ADNI-1 cohort being too complex to properly exploit both hemispheres seems to be right. Indeed, we can see that CS2 and CS4 cohorts achieve relatively high performances, with low variance thanks to the data augmentation brought by ADNI-2. For these cohorts, a Bilateral architecture outperform a Unilateral one. In addition, the CC cohort does not reach the same performances, and goes back to SBU being outperformed by SU.

CC achieving rather low performances means that including MCI subjects having converted after 2 years as stables might not be the best of ideas. It is reasonable, in a sense that models see these subjects on the same level as ”actual” stable ones, while the probability is high that these switched subjects are in reality closer to converted ones (the kept ones, converted before 2 years).

Chapter 6

Conclusion

Alzheimer's is a complex disease for which we know little; a cure is yet to be found, and its causes are still unknown. The disease could potentially be understood via brain scans analysis, but they turn out to be rather complex to interpret.

In this project, we showed that rotational equivariant convolutional neural networks could achieve state-of-the-art performance on distinguishing Alzheimer's diseased subjects from cognitively normal ones, as well as on predicting the stability or progression towards Alzheimer's disease of mild cognitive impairment subjects. Indeed, while we worked during this project in a low input resolution configuration, our models achieved an average accuracy and AUC of about 92.7% and 95.1% in the AD vs CN classification task, and even 94.1% accuracy for the best models among them. We even went up to 96.6% for higher input resolutions.

These numbers go down to 75% and 80% on average with at best 79.7% accuracy for the MCI-progression task. However, this task is much more challenging than the first one. Firstly, it implies distinguishing between two nuances of the same condition (MCI). Secondly, this task is predictive, while the first one has diagnosis purposes. While offering a confident diagnosis is essential, the prediction is always a more significant concern, as it might help delay symptoms arrival. These difficulties induce a deterioration of results, but our models turned out to achieve top state-of-the-art performance on classification through T1-weighted 1.5 T MRI scans.

We investigated possible improvement lines in this project, with varying degrees of success. While we struggled to exploit hemispheres combinations and longitudinal features, the combination of multiple cortical features turned out to be efficient.

Future work

A significant issue of this project was about the variance brought by the ADNI-1 cohort. It obliged us to increase the computation time by multiplying runs for decreasing the variance. As we saw, working with larger cohorts increases the confidence we have in the obtained results. The next clear step will be to go entirely for a larger cohort such as ADNI-2 or ADNI-3, and switch to a bandwidth parameter 64. The efficiency of ensemble methods have been proven during this project, and a low variance setting might benefit from boosting methods rather than bagging.

While using cortical volume turned out to improve our models' performances, other measures, such as surface curvature, less correlated with the initial cortical thickness, might further help the classification task.

Finer tuning can also be considered, via smart techniques such as Bayesian Optimisation, to perfectly optimize the studied architectures, especially promising ones such as SBMs.

Finally, the MCI-progression classification task seemed rather difficult to handle. We think about switching to a regression task, where the final goal is to predict the conversion year of an MCI subject. Such a task might even be harder to control, but the use of longitudinal features should help. If it were to work, the community would definitely gain from it.

Appendix A

Ethics Checklist

Section 3: This project does not involve human participant directly, but uses data collected via a large-scale study in North-America (ADNI).
Section 4: The ADNI data used in this project have been recovered directly from Feng et al. It includes pre-processed brain scans, that we further processed, and information about gender, Alzheimer's condition, age of subjects is possessed. However, the information was previously anonymised in referencing each subject by a random ID. Moreover, even if we did not recover the data from ADNI directly, we still went through the entire ADNI process to be accepted for data collection. Even if this data is officially public, we made sure that we had all the necessary authorizations to detain and use it.

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		X
Does your project involve the use of human embryos?		X
Does your project involve the use of human foetal tissues / cells?		X
Section 2: HUMANS		
Does your project involve human participants?	X	
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from “Human Embryos/Foetuses” i.e. Section 1)?		X
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?	X	
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	X	
Does it involve processing of genetic information?		X
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		X
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?	X	
Section 5: ANIMALS		
Does your project involve animals?		X
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		X
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		X
Could the situation in the country put the individuals taking part in the project at risk?		X
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		X
Does your project deal with endangered fauna and/or flora /protected areas?		X
Does your project involve the use of elements that may cause harm to humans, including project staff?		X
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		X
Section 8: DUAL USE		
Does your project have the potential for military applications?		X
Does your project have an exclusive civilian application focus?		X
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		X
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		X
Section 9: MISUSE		

Does your project have the potential for malevolent/criminal/terrorist abuse?		X
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		X
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		X
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		X
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		X
Will your project use or produce goods or information for which there are data protection, or other legal implications?		X
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		X

Appendix B

Desikan atlas

Table B.1: Desikan atlas description: label values and names of structures (inspired from [11]).

Index	Description
0	Unlabelled region
1	Banks superior temporal sulcus
2	Caudal anterior-cingulate cortex
3	Caudal middle frontal gyrus
4	Corpus callosum
5	Cuneus cortex
6	Entorhinal cortex
7	Fusiform gyrus
8	Inferior parietal cortex
9	Inferior temporal gyrus
10	Isthmuscingulate cortex
11	Lateral occipital cortex
12	Lateral orbital frontal cortex
13	Lingual gyrus
14	Medial orbital frontal cortex
15	Middle temporal gyrus
16	Parahippocampal gyrus
17	Paracentral lobule
18	Pars opercularis
19	Pars orbitalis
20	Pars triangularis
21	Pericalcarine cortex
22	Postcentral gyrus

23	Posterior-cingulate cortex
24	Precentral gyrus
25	Precuneus cortex
26	Rostral anterior cingulate cortex
27	Rostral middle frontal gyrus
28	Superior frontal gyrus
29	Superior parietal cortex
30	Superior temporal gyrus
31	Supramarginal gyrus
32	Frontal pole
33	Temporal pole
34	Transverse temporal cortex
35	Insula

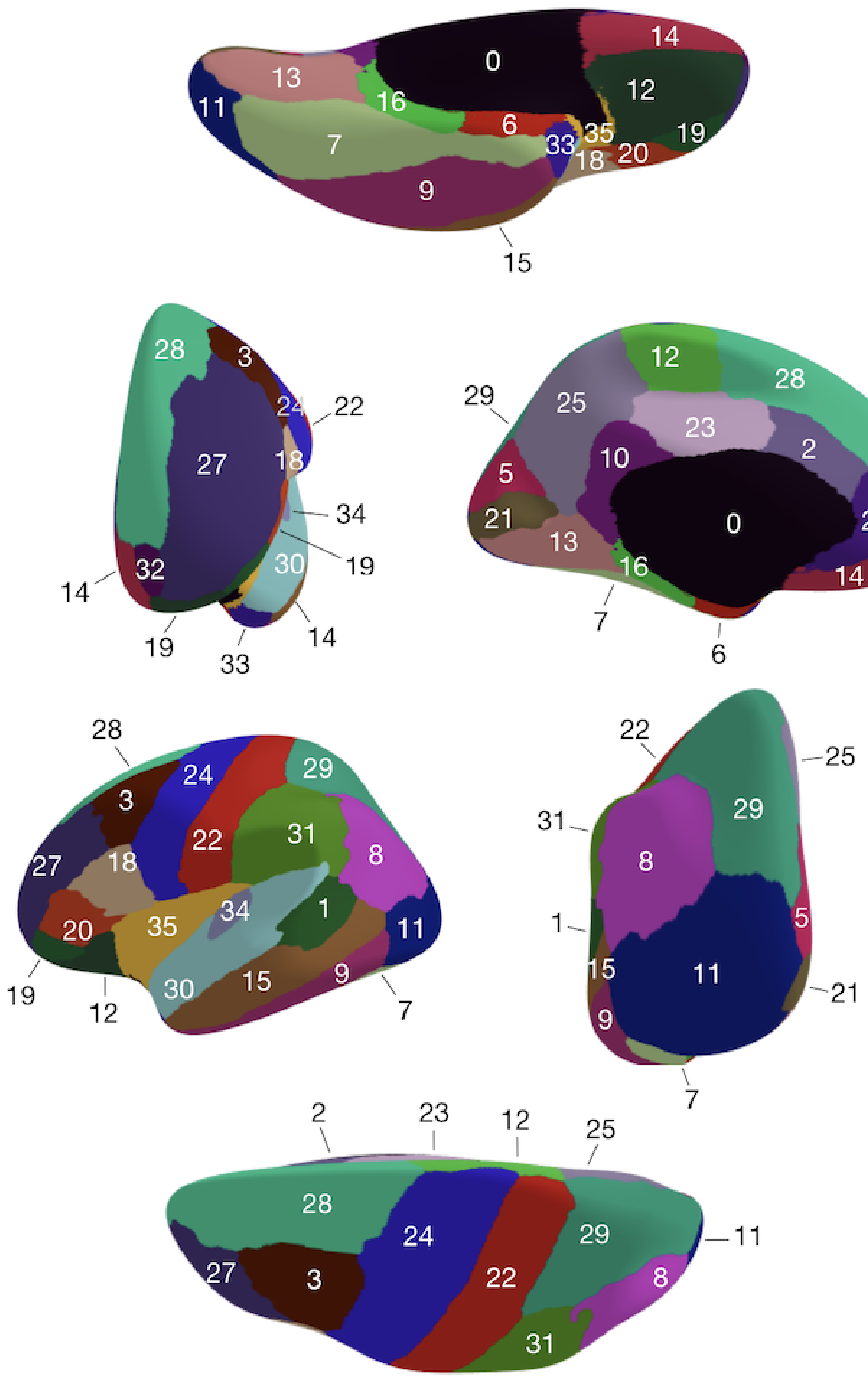


Figure B.1: Spatial distribution of the labelled structural regions in the Desikan atlas. Views from top to bottom and left to right: inferior, anterior, medial, lateral, posterior, superior.

Appendix C

Destrieux atlas

Table C.1: Destrieux atlas description: label values and names of structures (inspired from [12]).

Index	Short Name	Description
0	-	Unlabelled region
1	G_and_S_frontomargin	Fronto-marginal gyrus and sulcus
2	G_and_S_occipital_inf	Inferior occipital gyrus and sulcus
3	G_and_S_paracentral	Paracentral lobule and sulcus
4	G_and_S_subcentral	Subcentral gyrus and sulci
5	G_and_S_transv_frontopol	Transverse frontopolar gyri and sulci
6	G_and_S_cingul-Ant	Anterior part of the cingulate gyrus and sulcus
7	G_and_S_cingul-Mid-Ant	Middle-anterior part of the cingulate gyrus and sulcus
8	G_and_S_cingul-Mid-Post	Middle-posterior part of the cingulate gyrus and sulcus
9	G_cingul-Post-dorsal	Posterior-dorsal part of the cingulate gyrus
10	G_cingul-Post-ventral	Posterior-ventral part of the cingulate gyrus
11	G_cuneus	Cuneus
12	G_front_inf-Opercular	Opercular part of the inferior frontal gyrus
13	G_front_inf-Orbital	Orbital part of the inferior frontal gyrus
14	G_front_inf-Triangul	Triangular part of the inferior frontal gyrus
15	G_front_middle	Middle frontal gyrus
16	G_front_sup	Superior frontal gyrus
17	G_Ins_lg_and_S_cent_ins	Long insular gyrus and central sulcus of the insula
18	G_insular_short	Short insular gyri
19	G_occipital_middle	Middle occipital gyrus
20	G_occipital_sup	Superior occipital gyrus
21	G_oc-temp_lat-fusifor	Lateral occipito-temporal gyrus
22	G_oc-temp_med-Lingual	Ligual part of the medial occipito-temporal gyrus

23	G_oc-temp_med-Parahip	Parahippocampal part of the medial occipito-temporal gyrus
24	G_orbital	Orbital gyri
25	G_pariet_inf-Angular	Angular gyrus
26	G_pariet_inf-Supramar	Supramarginal gyrus
27	G_parietal_sup	Superior parietal lobule
28	G_postcentral	Postcentral gyrus
29	G_precentral	Precentral gyrus
30	G_precuneus	Precuneus
31	G_rectus	Straight gyrus, Gyrus rectus
32	G_subcallosal	Subcallosal area, subcallosal gyrus
33	G_temp_sup-G_T_transv	Anterior transverse temporal gyrus
34	G_temp_sup-Lateral	Lateral aspect of the superior temporal gyrus
35	G_temp_sup-Plan_polar	Planum polare of the superior temporal gyrus
36	G_temp_sup-Plan_tempo	Planum temporale of the superior temporal gyrus
37	G_temporal_inf	Inferior temporal gyrus
38	G_temporal_middle	Middle temporal gyrus
39	Lat_Fis-ant-Horizont	Horizontal ramus of the anterior segment of the lateral sulcus
40	Lat_Fis-ant-Vertical	Vertical ramus of the anterior segment of the lateral sulcus
41	Lat_Fis-post	Posterior ramus of the lateral sulcus
42	Pole_occipital	Occipital pole
43	Pole_temporal	Temporal pole
44	S_calcarine	Calcarine sulcus
45	S_central	Central sulcus
46	S_cingul-Marginalis	Marginal branch of the cingulate sulcus
47	S_circular_insula_ant	Anterior segment of the circular sulcus of the insula
48	S_circular_insula_inf	Inferior segment of the circular sulcus of the insula
49	S_circular_insula_sup	Superior segment of the circular sulcus of the insula
50	S_collat_transv_ant	Anterior transverse collateral sulcus
51	S_collat_transv_post	Posterior transverse collateral sulcus
52	S_front_inf	Inferior frontal sulcus
53	S_front_middle	Middle frontal sulcus
54	S_front_sup	Superior frontal sulcus
55	S_interm_prim-Jensen	Sulcus intermedius primus
56	S_intrapariet_and_P_trans	Intraparietal sulcus and transverse parietal sulci
57	S_oc_middle_and_Lunatus	Middle occipital sulcus and lunatus sulcus
58	S_oc_sup_and_transversal	Superior occipital sulcus and transverse occipital sulcus
59	S_occipital_ant	Anterior occipital sulcus and preoccipital notch

60	S_oc-temp_lat	Lateral occipito-temporal sulcus
61	S_oc-temp_med_and_Lingual	Medial occipito-temporal sulcus and lingual sulcus
62	S_orbital_lateral	Lateral orbital sulcus
63	S_orbital_med-olfact	Medial orbital sulcus
64	S_orbital-H_Shaped	Orbital sulci
65	S_parieto_occipital	Parieto-occipital sulcus
66	S_pericallosal	Pericallosal sulcus
67	S_postcentral	Postcentral sulcus
68	S_precentral-inf-part	Inferior part of the precentral sulcus
69	S_precentral-sup-part	Superior part of the precentral sulcus
70	S_suborbital	Suborbital sulcus
71	S_subparietal	Subparietal sulcus
72	S_temporal_inf	Inferior temporal sulcus
73	S_temporal_sup	Superior temporal sulcus
74	S_temporal_transverse	Transverse temporal sulcus

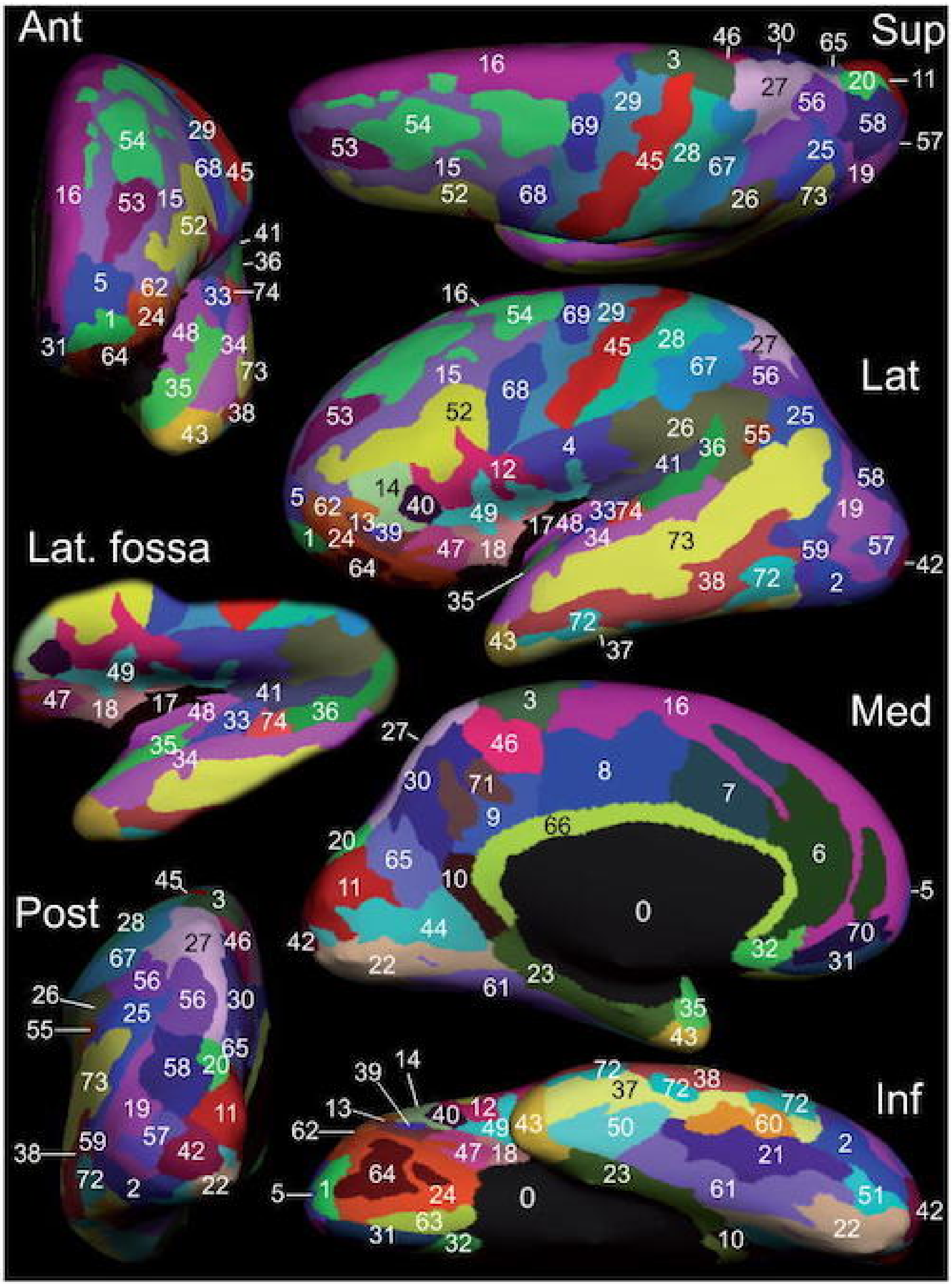


Figure C.1: Spatial distribution of the labelled structural regions in the Destrieux atlas (from [12]).

Appendix D

ADNI-1 additional statistics

D.1 AD vs CN cohorts

FOLD 1			
Numbers	M	F	Total
AD	18	19	37
CN	16	15	31
Total	34	34	68
Ages (std)	M	F	Total
AD	76.4 (7.0)	73.3 (8.0)	74.8 (7.6)
CN	75.7 (5.9)	76.1 (4.2)	75.9 (5.0)
Total	76.1 (6.4)	74.5 (6.7)	75.3 (6.5)

FOLD 2			
Numbers	M	F	Total
AD	22	16	38
CN	15	15	30
Total	37	31	68
Ages (std)	M	F	Total
AD	76.2 (6.1)	73.9 (6.3)	75.2 (6.2)
CN	74.4 (5.6)	75.3 (5.8)	74.8 (5.6)
Total	75.5 (5.9)	74.6 (6.0)	75.1 (5.9)

FOLD 3			
Numbers	M	F	Total
AD	20	18	38
CN	16	14	30
Total	36	32	68
Ages (std)	M	F	Total
AD	77.2 (7.0)	73.8 (6.6)	75.6 (6.9)
CN	76.6 (5.9)	74.4 (6.2)	75.6 (6.0)
Total	76.9 (6.4)	74.1 (6.3)	75.6 (6.5)

FOLD 4			
Numbers	M	F	Total
AD	20	17	37
CN	13	17	30
Total	33	34	67
Ages (std)	M	F	Total
AD	74.5 (7.9)	75.7 (7.0)	75.1 (7.4)
CN	74.9 (5.1)	76.3 (4.3)	75.7 (4.6)
Total	74.7 (6.8)	76.0 (5.8)	75.4 (6.3)

FOLD 5			
Numbers	M	F	Total
AD	19	19	38
CN	14	16	30
Total	33	35	68
Ages (std)	M	F	Total
AD	73.7 (8.9)	76.6 (9.8)	75.2 (9.4)
CN	75.8 (5.8)	76.5 (4.6)	76.2 (5.1)
Total	74.6 (7.7)	76.6 (7.8)	75.6 (7.7)

Table D.1: Statistics of the 5-fold splitting in ADNI-1 (AD vs CN).

FOLD 1			
Numbers	M	F	Total
AD	9	9	18
CN	9	7	16
Total	18	16	34
Ages (std)	M	F	Total
AD	77.1 (7.0)	73.5 (8.5)	75.3 (7.8)
CN	73.0 (5.6)	76.6 (5.0)	74.6 (5.5)
Total	75.1 (6.5)	74.9 (7.1)	75.0 (6.7)

FOLD 2			
Numbers	M	F	Total
AD	9	10	19
CN	7	8	15
Total	16	18	34
Ages (std)	M	F	Total
AD	75.7 (7.2)	73.1 (8.0)	74.3 (7.6)
CN	79.3 (4.2)	75.7 (3.6)	77.4 (4.2
Total	77.2 (6.2)	74.3 (6.4)	75.7 (6.4)

FOLD 3			
Numbers	M	F	Total
AD	11	8	19
CN	8	7	15
Total	19	15	34
Ages (std)	M	F	Total
AD	78.1 (6.0)	72.8 (6.9)	75.9 (6.8)
CN	74.2 (7.5)	76.9 (6.5)	75.4 (6.9)
Total	76.5 (6.8)	74.7 (6.8)	75.7 (6.7)

FOLD 4			
Numbers	M	F	Total
AD	11	8	19
CN	7	8	15
Total	18	16	34
Ages (std)	M	F	Total
AD	74.2 (5.9)	75.1 (5.9)	74.6 (5.7)
CN	74.7 (2.8)	73.9 (5.1)	74.3 (4.1)
Total	74.4 (4.8)	74.5 (5.4)	74.4 (5.0)

FOLD 5			
Numbers	M	F	Total
AD	9	10	19
CN	8	7	15
Total	17	17	34
Ages (std)	M	F	Total
AD	78.0 (8.1)	72.5 (7.6)	75.1 (8.1)
CN	77.7 (6.4)	72.1 (5.8)	75.1 (6.6)
Total	77.8 (7.1)	72.4 (6.7)	75.1 (7.4)

FOLD 6			
Numbers	M	F	Total
AD	11	8	19
CN	8	7	15
Total	19	15	34
Ages (std)	M	F	Total
AD	76.6 (6.3)	75.5 (5.2)	76.1 (5.7)
CN	75.4 (5.5)	76.8 (6.2)	76.1 (5.6)
Total	76.1 (5.8)	76.1 (5.5)	76.1 (5.6)

FOLD 7			
Numbers	M	F	Total
AD	12	7	19
CN	6	9	15
Total	18	16	34
Ages (std)	M	F	Total
AD	74.8 (9.5)	73.7 (6.2)	74.4 (8.2)
CN	74.9 (5.7)	76.8 (4.6)	76.0 (5.0)
Total	74.8 (8.2)	75.4 (5.4)	75.1 (6.9)

FOLD 8			
Numbers	M	F	Total
AD	8	10	18
CN	7	8	15
Total	15	18	33
Ages (std)	M	F	Total
AD	74.2 (5.2)	77.2 (7.6)	75.8 (6.6)
CN	74.8 (4.9)	75.8 (4.3)	75.4 (4.4)
Total	74.5 (5.9)	76.6 (6.2)	75.6 (5.7)

FOLD 9			
Numbers	M	F	Total
AD	12	7	19
CN	7	8	15
Total	19	15	34
Ages (std)	M	F	Total
AD	72.9 (8.0)	76.7 (12.2)	74.3 (9.6)
CN	76.2 (7.1)	77.9 (2.6)	77.1 (5.1)
Total	74.2 (7.6)	77.3 (8.2)	75.6 (7.9)

FOLD 10			
Numbers	M	F	Total
AD	7	12	19
CN	7	8	15
Total	14	20	34
Ages (std)	M	F	Total
AD	74.9 (10.8)	76.6 (8.7)	76.0 (9.3)
CN	75.4 (4.6)	75.0 (5.7)	75.2 (5.1)
Total	75.2 (8.0)	76.0 (7.5)	75.6 (7.6)

Table D.2: Statistics of the 10-fold splitting in ADNI-1 (AD vs CN).

D.2 MCI cohort: stable vs progr.

FOLD 1			
Numbers	M	F	Total
MCI-p	18	10	28
MCI-s	16	7	23
Total	34	17	51
Ages (std)	M	F	Total
MCI-p	76.4 (7.6)	73.8 (6.3)	75.5 (7.2)
MCI-s	74.8 (9.0)	74.7 (7.9)	74.8 (8.5)
Total	75.7 (8.2)	74.2 (6.7)	75.2 (7.7)

FOLD 2			
Numbers	M	F	Total
MCI-p	17	10	27
MCI-s	10	12	22
Total	27	22	49
Ages (std)	M	F	Total
MCI-p	74.4 (7.7)	73.3 (7.5)	74.0 (7.5)
MCI-s	76.2 (6.9)	72.6 (7.3)	74.2 (7.2)
Total	75.1 (7.3)	72.9 (7.3)	74.1 (7.3)

FOLD 3			
Numbers	M	F	Total
MCI-p	17	11	28
MCI-s	14	9	23
Total	31	20	51
Ages (std)	M	F	Total
MCI-p	75.4 (6.8)	76.7 (4.0)	75.9 (5.8)
MCI-s	78.1 (6.4)	72.3 (8.8)	75.9 (7.8)
Total	76.6 (6.7)	74.7 (6.8)	75.9 (6.7)

FOLD 4			
Numbers	M	F	Total
MCI-p	18	9	27
MCI-s	13	9	22
Total	31	18	49
Ages (std)	M	F	Total
MCI-p	73.8 (7.4)	74.9 (5.6)	74.1 (6.8)
MCI-s	76.1 (6.8)	75.8 (4.7)	76.0 (5.9)
Total	74.8 (7.1)	75.3 (5.1)	75.0 (6.4)

FOLD 5			
Numbers	M	F	Total
MCI-p	15	11	26
MCI-s	19	5	24
Total	34	16	50
Ages (std)	M	F	Total
MCI-p	77.0 (5.9)	69.3 (7.7)	73.7 (7.6)
MCI-s	73.8 (7.8)	73.6 (5.9)	73.7 (7.3)
Total	75.2 (7.1)	70.7 (7.3)	73.7 (7.4)

Table D.3: Statistics of the 5-fold splitting in ADNI-1 (MCI-p vs MCI-s).

FOLD 1			
Numbers	M	F	Total
MCI-p	10	4	14
MCI-s	7	5	12
Total	17	9	26
Ages (std)	M	F	Total
MCI-p	79.8 (6.2)	71.8 (8.1)	77.5 (7.5)
MCI-s	76.3 (10.6)	73.2 (9.0)	75.0 (9.7)
Total	78.3 (8.2)	72.6 (8.1)	76.3 (8.5)

FOLD 2			
Numbers	M	F	Total
MCI-p	8	6	14
MCI-s	9	2	11
Total	17	8	25
Ages (std)	M	F	Total
MCI-p	72.3 (7.4)	75.1 (5.1)	73.5 (6.4)
MCI-s	73.7 (8.0)	78.5 (2.7)	74.6 (7.5)
Total	73.0 (7.5)	76.0 (4.7)	74.0 (6.8)

FOLD 3			
Numbers	M	F	Total
MCI-p	8	5	13
MCI-s	5	6	11
Total	13	11	24
Ages (std)	M	F	Total
MCI-p	72.2 (6.2)	71.0 (8.8)	71.7 (7.0)
MCI-s	74.8 (9.5)	70.2 (8.4)	72.3 (8.8)
Total	73.2 (7.3)	70.6 (8.2)	72.0 (7.7)

FOLD 4			
Numbers	M	F	Total
MCI-p	9	5	14
MCI-s	5	6	11
Total	14	11	25
Ages (std)	M	F	Total
MCI-p	76.5 (8.7)	75.7 (5.9)	76.2 (7.6)
MCI-s	77.6 (3.3)	74.9 (5.9)	76.1 (4.9)
Total	76.9 (7.1)	75.3 (5.6)	76.2 (6.4)

FOLD 5			
Numbers	M	F	Total
MCI-p	10	4	14
MCI-s	7	4	11
Total	17	8	25
Ages (std)	M	F	Total
MCI-p	76.3 (6.0)	75.9 (4.3)	76.2 (5.4)
MCI-s	76.3 (7.1)	80.6 (3.6)	77.9 (6.2)
Total	76.3 (6.3)	78.2 (4.5)	76.9 (5.7)

FOLD 6			
Numbers	M	F	Total
MCI-p	7	7	14
MCI-s	7	5	12
Total	14	12	26
Ages (std)	M	F	Total
MCI-p	74.1 (8.2)	77.2 (4.1)	75.6 (6.4)
MCI-s	80.0 (5.5)	65.7 (4.7)	74.0 (8.9)
Total	77.0 (7.4)	72.4 (7.2)	74.9 (7.5)

FOLD 7			
Numbers	M	F	Total
MCI-p	7	6	13
MCI-s	6	5	11
Total	13	11	24
Ages (std)	M	F	Total
MCI-p	73.1 (8.8)	74.3 (5.5)	73.7 (7.2)
MCI-s	79.7 (3.3)	77.0 (5.9)	78.5 (4.6)
Total	76.2 (7.4)	75.5 (5.6)	75.9 (6.5)

FOLD 8			
Numbers	M	F	Total
MCI-p	11	3	14
MCI-s	7	4	11
Total	18	7	25
Ages (std)	M	F	Total
MCI-p	74.2 (6.8)	76.0 (6.9)	74.6 (6.6)
MCI-s	73.1 (7.8)	74.2 (2.8)	73.5 (6.3)
Total	73.8 (7.0)	75.0 (4.6)	74.1 (6.4)

FOLD 9			
Numbers	M	F	Total
MCI-p	7	6	13
MCI-s	9	3	12
Total	16	9	25
Ages (std)	M	F	Total
MCI-p	79.9 (5.1)	64.9 (5.8)	73.0 (9.3)
MCI-s	73.8 (7.8)	72.0 (7.4)	73.3 (7.4)
Total	76.5 (7.3)	67.3 (6.9)	73.2 (8.3)

FOLD 10			
Numbers	M	F	Total
MCI-p	8	5	13
MCI-s	10	2	12
Total	18	7	25
Ages (std)	M	F	Total
MCI-p	74.5 (5.6)	74.6 (6.6)	74.5 (5.7)
MCI-s	73.7 (8.1)	76.1 (3.1)	74.1 (7.4)
Total	74.1 (6.9)	75.0 (5.6)	74.3 (6.5)

Table D.4: Statistics of the 10-fold splitting in ADNI-1 (MCI-p vs MCI-s).

Appendix E

ADNI-2 additional statistics

E.1 AD vs CN cohorts

FOLD 1			
Numbers	M	F	Total
AD	38	31	69
CN	38	30	68
Total	76	61	137
Ages (std)	M	F	Total
AD	76.0 (8.2)	74.1 (7.8)	75.2 (8.0)
CN	74.8 (5.8)	73.6 (5.6)	74.2 (5.7)
Total	75.4 (7.1)	73.8 (6.7)	74.7 (6.9)

FOLD 2			
Numbers	M	F	Total
AD	38	32	70
CN	38	30	68
Total	76	62	138
Ages (std)	M	F	Total
AD	76.0 (7.6)	74.1 (7.7)	75.1 (7.6)
CN	74.8 (5.9)	73.4 (5.7)	74.2 (5.8)
Total	75.4 (6.8)	73.8 (6.7)	74.7 (6.8)

FOLD 3			
Numbers	M	F	Total
AD	38	31	69
CN	38	30	68
Total	76	61	137
Ages (std)	M	F	Total
AD	75.8 (7.4)	74.1 (8.0)	75.0 (7.7)
CN	74.8 (6.0)	73.5 (5.6)	74.2 (5.8)
Total	75.3 (6.7)	73.8 (6.9)	74.6 (6.8)

FOLD 4			
Numbers	M	F	Total
AD	38	32	70
CN	38	30	68
Total	76	62	138
Ages (std)	M	F	Total
AD	75.9 (8.1)	74.2 (8.2)	75.1 (8.1)
CN	74.9 (6.0)	73.6 (5.4)	74.3 (5.7)
Total	75.4 (7.1)	73.9 (7.0)	74.7 (7.0)

FOLD 5			
Numbers	M	F	Total
AD	38	31	69
CN	38	30	68
Total	76	61	137
Ages (std)	M	F	Total
AD	75.8 (7.6)	74.2 (8.5)	75.1 (8.0)
CN	74.8 (6.1)	73.5 (5.5)	74.2 (5.8)
Total	75.3 (6.9)	73.9 (7.1)	74.7 (7.0)

Table E.1: Statistics of the 5-fold splitting in ADNI-2 (AD vs CN).

FOLD 1			
Numbers	M	F	Total
AD	19	16	35
CN	19	15	34
Total	38	31	69
Ages (std)	M	F	Total
AD	75.9 (7.6)	74.1 (7.8)	75.1 (7.7)
CN	74.8 (5.8)	73.6 (5.7)	74.3 (5.7)
Total	75.4 (6.7)	73.9 (6.8)	74.7 (6.7)

FOLD 2			
Numbers	M	F	Total
AD	19	15	34
CN	19	15	34
Total	38	30	68
Ages (std)	M	F	Total
AD	76.1 (8.9)	74.1 (7.9)	75.2 (8.4)
CN	74.7 (5.9)	73.5 (5.8)	74.2 (5.8)
Total	75.4 (7.5)	73.8 (6.8)	74.7 (7.2)

FOLD 3			
Numbers	M	F	Total
AD	19	16	35
CN	19	15	34
Total	38	31	69
Ages (std)	M	F	Total
AD	76.1 (7.7)	74.1 (7.7)	75.2 (7.7)
CN	74.9 (6.0)	73.5 (5.9)	74.3 (5.9)
Total	75.5 (6.8)	73.8 (6.8)	74.7 (6.8)

FOLD 4			
Numbers	M	F	Total
AD	19	16	35
CN	19	15	34
Total	38	31	69
Ages (std)	M	F	Total
AD	75.9 (7.7)	74.1 (7.8)	75.1 (7.7)
CN	74.7 (5.9)	73.4 (5.7)	74.1 (5.8)
Total	75.3 (6.8)	73.7 (6.8)	74.6 (6.8)

FOLD 5			
Numbers	M	F	Total
AD	19	16	35
CN	19	15	34
Total	38	31	69
Ages (std)	M	F	Total
AD	75.8 (7.5)	74.1 (7.8)	75.0 (7.5)
CN	74.8 (5.8)	73.6 (5.7)	74.3 (5.7)
Total	75.3 (6.6)	73.9 (6.7)	74.6 (6.7)

FOLD 6			
Numbers	M	F	Total
AD	19	15	34
CN	19	15	34
Total	38	30	68
Ages (std)	M	F	Total
AD	75.8 (7.6)	74.1 (8.5)	75.0 (7.9)
CN	74.8 (6.4)	73.3 (5.6)	74.2 (6.0)
Total	75.3 (6.9)	73.7 (7.1)	74.6 (7.0)

FOLD 7			
Numbers	M	F	Total
AD	19	16	35
CN	19	15	34
Total	38	31	69
Ages (std)	M	F	Total
AD	75.8 (7.9)	74.2 (8.9)	75.1 (8.3)
CN	74.9 (6.0)	73.7 (5.5)	74.4 (5.8)
Total	75.4 (6.9)	74.0 (7.4)	74.7 (7.1)

FOLD 8			
Numbers	M	F	Total
AD	19	16	35
CN	19	15	34
Total	38	31	69
Ages (std)	M	F	Total
AD	75.9 (8.5)	74.1 (7.8)	75.1 (8.1)
CN	74.8 (6.1)	73.4 (5.5)	74.2 (5.8)
Total	75.4 (7.3)	73.8 (6.7)	74.7 (7.0)

FOLD 9			
Numbers	M	F	Total
AD	19	15	34
CN	19	15	34
Total	38	30	68
Ages (std)	M	F	Total
AD	75.9 (7.8)	74.4 (9.1)	75.2 (8.3)
CN	74.8 (6.3)	73.5 (5.4)	74.3 (5.9)
Total	75.4 (7.1)	73.9 (7.4)	74.7 (7.2)

FOLD 10			
Numbers	M	F	Total
AD	19	16	35
CN	19	15	34
Total	38	31	69
Ages (std)	M	F	Total
AD	75.8 (7.6)	74.1 (8.1)	75.0 (7.8)
CN	74.8 (6.0)	73.5 (5.7)	74.2 (5.8)
Total	75.3 (6.8)	73.8 (7.0)	74.6 (6.9)

Table E.2: Statistics of the 10-fold splitting in ADNI-2 (AD vs CN).

E.2 MCI - Cohort CC

FOLD 1			
Numbers	M	F	Total
MCI-p	25	17	42
MCI-s	25	17	42
Total	50	34	84
Ages (std)	M	F	Total
MCI-p	74.6 (7.1)	72.8 (6.9)	73.9 (7.0)
MCI-s	73.5 (7.1)	71.8 (7.6)	72.8 (7.2)
Total	74.0 (7.0)	72.3 (7.2)	73.3 (7.1)

FOLD 2			
Numbers	M	F	Total
MCI-p	25	16	41
MCI-s	25	17	42
Total	50	33	83
Ages (std)	M	F	Total
MCI-p	74.5 (6.8)	72.8 (7.1)	73.8 (6.9)
MCI-s	73.5 (7.1)	71.8 (7.5)	72.8 (7.2)
Total	74.0 (6.9)	72.3 (7.2)	73.3 (7.0)

FOLD 3			
Numbers	M	F	Total
MCI-p	26	17	43
MCI-s	25	16	41
Total	51	33	84
Ages (std)	M	F	Total
MCI-p	74.6 (7.5)	72.7 (7.5)	73.9 (7.5)
MCI-s	73.5 (7.1)	71.8 (7.5)	72.8 (7.2)
Total	74.1 (7.2)	72.3 (7.4)	73.4 (7.3)

FOLD 4			
Numbers	M	F	Total
MCI-p	25	17	42
MCI-s	26	16	42
Total	51	33	84
Ages (std)	M	F	Total
MCI-p	74.7 (6.8)	72.7 (6.9)	73.9 (6.8)
MCI-s	73.5 (7.1)	71.8 (7.5)	72.9 (7.2)
Total	74.1 (6.9)	72.3 (7.1)	73.4 (7.0)

FOLD 5			
Numbers	M	F	Total
MCI-p	25	16	41
MCI-s	25	18	43
Total	50	34	84
Ages (std)	M	F	Total
MCI-p	74.6 (7.2)	72.6 (6.8)	73.8 (7.0)
MCI-s	73.5 (7.0)	71.8 (7.6)	72.8 (7.2)
Total	74.1 (7.1)	72.2 (7.1)	73.3 (7.1)

Table E.3: Statistics of the 5-fold splitting in ADNI-2 (MCI-p vs MCI-s, cohort CC).

FOLD 1			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	9	21
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.6 (7.1)	72.8 (6.9)	73.9 (6.9)
MCI-s	73.5 (7.2)	71.8 (7.8)	72.8 (7.3)
Total	74.0 (7.0)	72.3 (7.2)	73.3 (7.1)

FOLD 2			
Numbers	M	F	Total
MCI-p	12	9	21
MCI-s	13	8	21
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.5 (7.4)	72.8 (7.3)	73.8 (7.2)
MCI-s	73.5 (7.2)	71.8 (7.8)	72.8 (7.3)
Total	74.0 (7.2)	72.3 (7.4)	73.3 (7.2)

FOLD 3			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	8	20
Total	25	16	41
Ages (std)	M	F	Total
MCI-p	74.6 (6.9)	72.9 (7.8)	73.9 (7.1)
MCI-s	73.5 (7.2)	71.9 (7.8)	72.9 (7.3)
Total	74.1 (6.9)	72.4 (7.5)	73.4 (7.1)

FOLD 4			
Numbers	M	F	Total
MCI-p	12	8	20
MCI-s	13	9	22
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.5 (7.0)	72.7 (6.9)	73.7 (6.8)
MCI-s	73.5 (7.2)	71.8 (7.8)	72.8 (7.3)
Total	74.0 (7.0)	72.2 (7.2)	73.3 (7.0)

FOLD 5			
Numbers	M	F	Total
MCI-p	13	9	22
MCI-s	12	8	20
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.6 (6.8)	72.6 (6.8)	73.8 (6.7)
MCI-s	73.5 (7.2)	71.7 (7.7)	72.8 (7.3)
Total	74.1 (6.9)	72.2 (7.0)	73.3 (6.9)

FOLD 6			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	13	8	21
Total	26	16	42
Ages (std)	M	F	Total
MCI-p	74.7 (8.3)	72.9 (8.7)	74.0 (8.3)
MCI-s	73.5 (7.2)	71.8 (7.8)	72.9 (7.3)
Total	74.1 (7.6)	72.4 (8.0)	73.4 (7.7)

FOLD 7			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	13	8	21
Total	26	16	42
Ages (std)	M	F	Total
MCI-p	74.7 (7.0)	72.7 (7.1)	73.9 (6.9)
MCI-s	73.5 (7.2)	71.8 (7.7)	72.9 (7.3)
Total	74.1 (7.0)	72.2 (7.2)	73.4 (7.0)

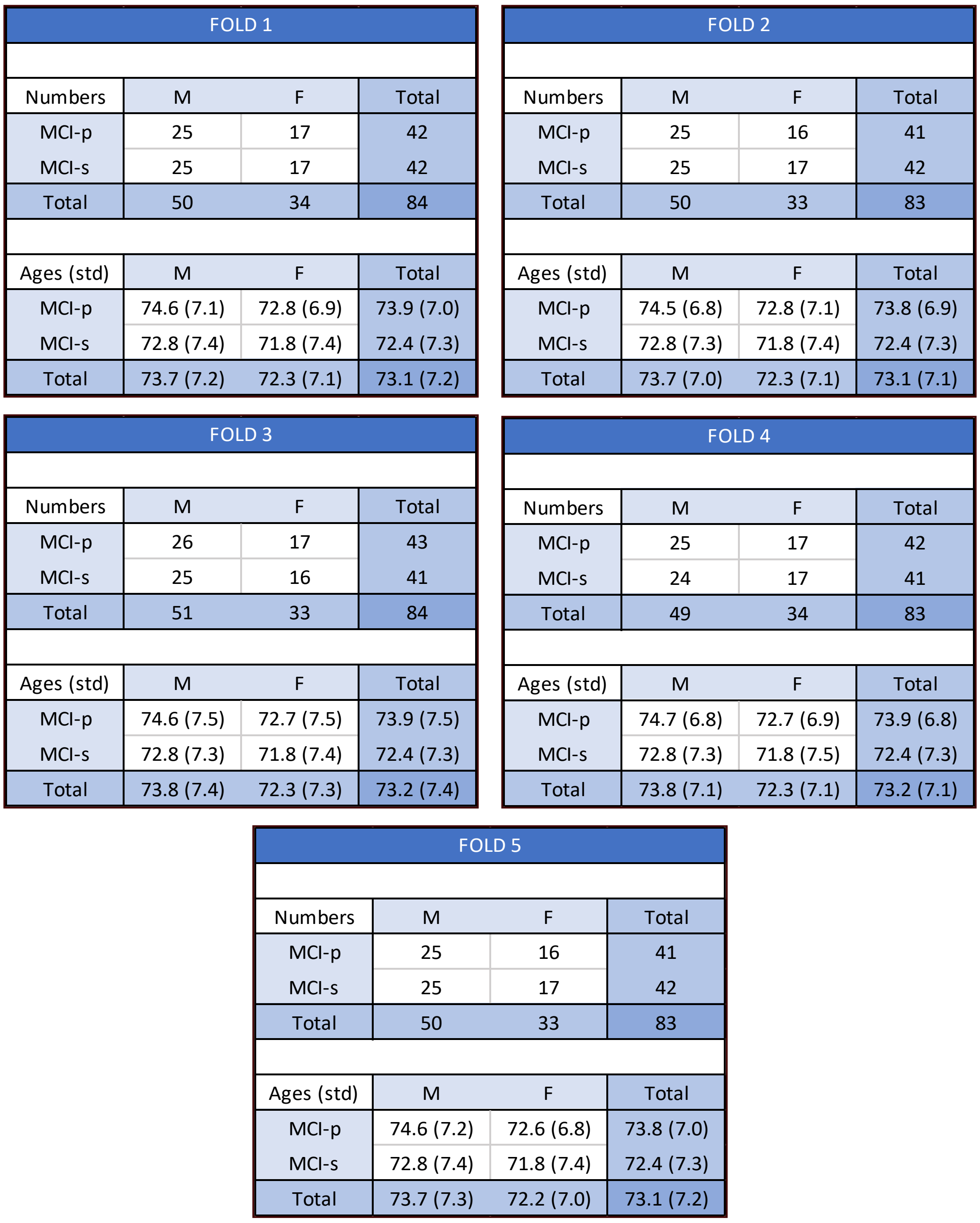
FOLD 8			
Numbers	M	F	Total
MCI-p	12	9	21
MCI-s	13	8	21
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.8 (6.9)	72.7 (7.2)	73.9 (6.9)
MCI-s	73.5 (7.2)	71.8 (7.8)	72.9 (7.3)
Total	74.1 (6.9)	72.3 (7.2)	73.4 (7.0)

FOLD 9			
Numbers	M	F	Total
MCI-p	12	8	20
MCI-s	13	9	22
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.7 (7.3)	72.6 (7.3)	73.8 (7.2)
MCI-s	73.5 (7.2)	71.8 (7.8)	72.8 (7.3)
Total	74.1 (7.1)	72.2 (7.3)	73.4 (7.2)

FOLD 10			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	9	21
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.6 (7.4)	72.6 (6.8)	73.9 (7.1)
MCI-s	73.5 (7.2)	71.8 (7.8)	72.7 (7.3)
Total	74.1 (7.2)	72.2 (7.1)	73.3 (7.1)

Table E.4: Statistics of the 10-fold splitting in ADNI-2 (MCI-p vs MCI-s, cohort CC).

E.3 MCI - Cohort CS2



FOLD 1			
Numbers	M	F	Total
MCI-p	25	17	42
MCI-s	25	17	42
Total	50	34	84
Ages (std)	M	F	Total
MCI-p	74.6 (7.1)	72.8 (6.9)	73.9 (7.0)
MCI-s	72.8 (7.4)	71.8 (7.4)	72.4 (7.3)
Total	73.7 (7.2)	72.3 (7.1)	73.1 (7.2)

FOLD 2			
Numbers	M	F	Total
MCI-p	25	16	41
MCI-s	25	17	42
Total	50	33	83
Ages (std)	M	F	Total
MCI-p	74.5 (6.8)	72.8 (7.1)	73.8 (6.9)
MCI-s	72.8 (7.3)	71.8 (7.4)	72.4 (7.3)
Total	73.7 (7.0)	72.3 (7.1)	73.1 (7.1)

FOLD 3			
Numbers	M	F	Total
MCI-p	26	17	43
MCI-s	25	16	41
Total	51	33	84
Ages (std)	M	F	Total
MCI-p	74.6 (7.5)	72.7 (7.5)	73.9 (7.5)
MCI-s	72.8 (7.3)	71.8 (7.4)	72.4 (7.3)
Total	73.8 (7.4)	72.3 (7.3)	73.2 (7.4)

FOLD 4			
Numbers	M	F	Total
MCI-p	25	17	42
MCI-s	24	17	41
Total	49	34	83
Ages (std)	M	F	Total
MCI-p	74.7 (6.8)	72.7 (6.9)	73.9 (6.8)
MCI-s	72.8 (7.3)	71.8 (7.5)	72.4 (7.3)
Total	73.8 (7.1)	72.3 (7.1)	73.2 (7.1)

FOLD 5			
Numbers	M	F	Total
MCI-p	25	16	41
MCI-s	25	17	42
Total	50	33	83
Ages (std)	M	F	Total
MCI-p	74.6 (7.2)	72.6 (6.8)	73.8 (7.0)
MCI-s	72.8 (7.4)	71.8 (7.4)	72.4 (7.3)
Total	73.7 (7.3)	72.2 (7.0)	73.1 (7.2)

Table E.5: Statistics of the 5-fold splitting in ADNI-2 (MCI-p vs MCI-s, cohort CS2).

FOLD 1			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	9	21
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.6 (7.1)	72.8 (6.9)	73.9 (6.9)
MCI-s	72.8 (7.5)	71.7 (7.7)	72.3 (7.4)
Total	73.7 (7.2)	72.2 (7.1)	73.1 (7.1)

FOLD 2			
Numbers	M	F	Total
MCI-p	12	9	21
MCI-s	13	8	21
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.5 (7.4)	72.8 (7.3)	73.8 (7.2)
MCI-s	72.8 (7.5)	71.8 (7.6)	72.4 (7.4)
Total	73.6 (7.4)	72.4 (7.3)	73.1 (7.3)

FOLD 3			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	9	21
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.6 (6.9)	72.9 (7.8)	73.9 (7.1)
MCI-s	72.8 (7.4)	71.8 (7.6)	72.4 (7.3)
Total	73.7 (7.0)	72.3 (7.4)	73.2 (7.1)

FOLD 4			
Numbers	M	F	Total
MCI-p	12	8	20
MCI-s	13	8	21
Total	25	16	41
Ages (std)	M	F	Total
MCI-p	74.5 (7.0)	72.7 (6.9)	73.7 (6.8)
MCI-s	72.8 (7.5)	71.8 (7.7)	72.4 (7.4)
Total	73.6 (7.2)	72.2 (7.1)	73.1 (7.1)

FOLD 5			
Numbers	M	F	Total
MCI-p	13	9	22
MCI-s	12	8	20
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.6 (6.8)	72.6 (6.8)	73.8 (6.7)
MCI-s	72.9 (7.5)	71.9 (7.6)	72.5 (7.3)
Total	73.8 (7.1)	72.2 (7.0)	73.1 (7.0)

FOLD 6			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	13	8	21
Total	26	16	42
Ages (std)	M	F	Total
MCI-p	74.7 (8.3)	72.9 (8.7)	74.0 (8.3)
MCI-s	72.8 (7.5)	71.8 (7.7)	72.4 (7.4)
Total	73.8 (7.8)	72.3 (8.0)	73.2 (7.8)

FOLD 7			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	9	21
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.7 (7.0)	72.7 (7.1)	73.9 (6.9)
MCI-s	72.8 (7.5)	71.8 (7.7)	72.4 (7.4)
Total	73.8 (7.2)	72.2 (7.2)	73.1 (7.1)

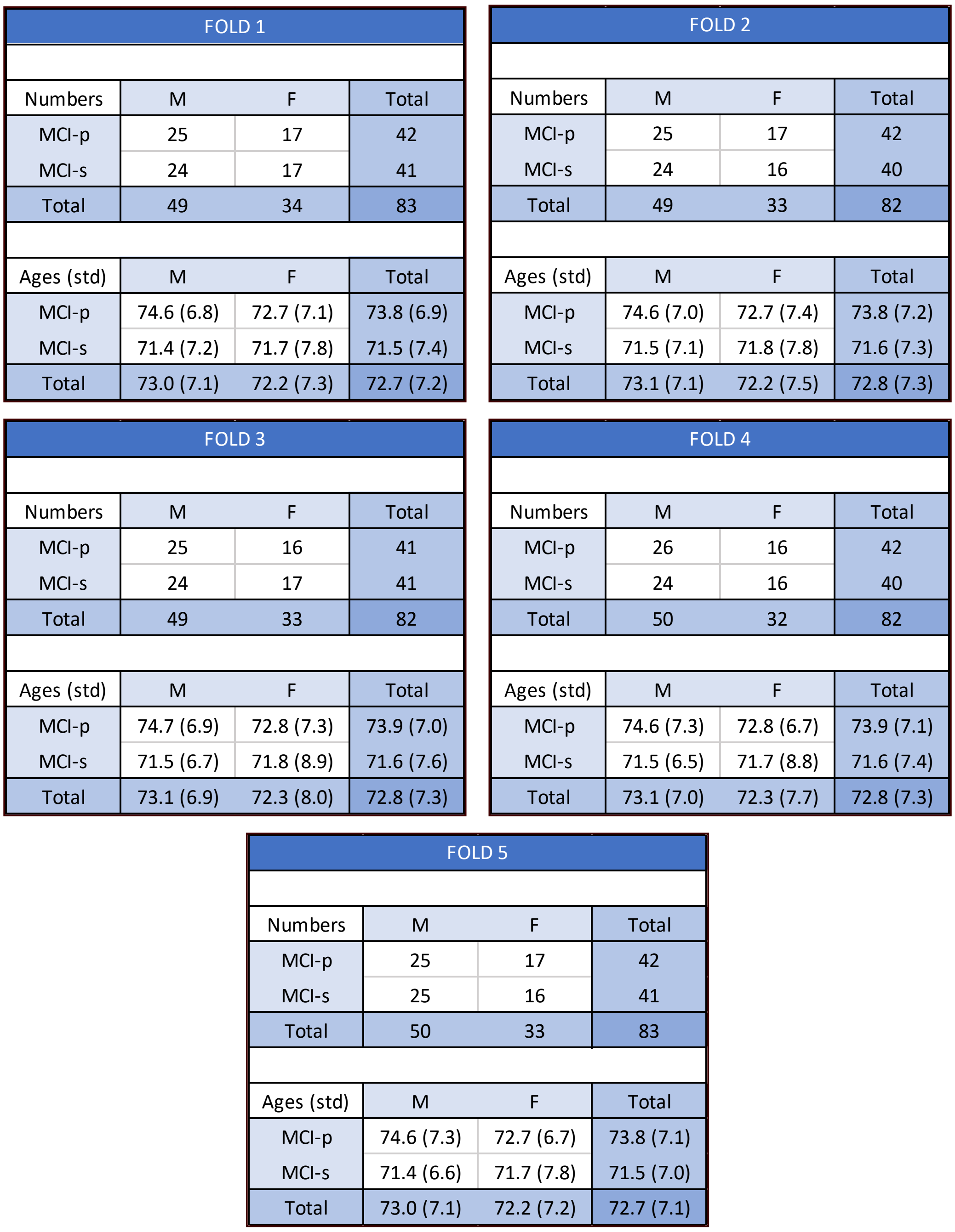
FOLD 8			
Numbers	M	F	Total
MCI-p	12	9	21
MCI-s	12	8	20
Total	24	17	41
Ages (std)	M	F	Total
MCI-p	74.8 (6.9)	72.7 (7.2)	73.9 (6.9)
MCI-s	72.8 (7.5)	71.8 (7.7)	72.4 (7.4)
Total	73.8 (7.1)	72.3 (7.2)	73.2 (7.1)

FOLD 9			
Numbers	M	F	Total
MCI-p	12	8	20
MCI-s	13	8	21
Total	25	16	41
Ages (std)	M	F	Total
MCI-p	74.7 (7.3)	72.6 (7.3)	73.8 (7.2)
MCI-s	72.8 (7.5)	71.8 (7.7)	72.4 (7.4)
Total	73.7 (7.3)	72.2 (7.2)	73.1 (7.2)

FOLD 10			
Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	9	21
Total	25	17	42
Ages (std)	M	F	Total
MCI-p	74.6 (7.4)	72.6 (6.8)	73.9 (7.1)
MCI-s	72.8 (7.5)	71.8 (7.6)	72.4 (7.4)
Total	73.8 (7.4)	72.2 (7.0)	73.1 (7.2)

Table E.6: Statistics of the 10-fold splitting in ADNI-2 (MCI-p vs MCI-s, cohort CS2).

E.4 MCI - Cohort CS4



FOLD 1

Numbers	M	F	Total
MCI-p	25	17	42
MCI-s	24	17	41
Total	49	34	83

Ages (std)	M	F	Total
MCI-p	74.6 (6.8)	72.7 (7.1)	73.8 (6.9)
MCI-s	71.4 (7.2)	71.7 (7.8)	71.5 (7.4)
Total	73.0 (7.1)	72.2 (7.3)	72.7 (7.2)

FOLD 2

Numbers	M	F	Total
MCI-p	25	17	42
MCI-s	24	16	40
Total	49	33	82

Ages (std)	M	F	Total
MCI-p	74.6 (7.0)	72.7 (7.4)	73.8 (7.2)
MCI-s	71.5 (7.1)	71.8 (7.8)	71.6 (7.3)
Total	73.1 (7.1)	72.2 (7.5)	72.8 (7.3)

FOLD 3

Numbers	M	F	Total
MCI-p	25	16	41
MCI-s	24	17	41
Total	49	33	82

Ages (std)	M	F	Total
MCI-p	74.7 (6.9)	72.8 (7.3)	73.9 (7.0)
MCI-s	71.5 (6.7)	71.8 (8.9)	71.6 (7.6)
Total	73.1 (6.9)	72.3 (8.0)	72.8 (7.3)

FOLD 4

Numbers	M	F	Total
MCI-p	26	16	42
MCI-s	24	16	40
Total	50	32	82

Ages (std)	M	F	Total
MCI-p	74.6 (7.3)	72.8 (6.7)	73.9 (7.1)
MCI-s	71.5 (6.5)	71.7 (8.8)	71.6 (7.4)
Total	73.1 (7.0)	72.3 (7.7)	72.8 (7.3)

FOLD 5

Numbers	M	F	Total
MCI-p	25	17	42
MCI-s	25	16	41
Total	50	33	83

Ages (std)	M	F	Total
MCI-p	74.6 (7.3)	72.7 (6.7)	73.8 (7.1)
MCI-s	71.4 (6.6)	71.7 (7.8)	71.5 (7.0)
Total	73.0 (7.1)	72.2 (7.2)	72.7 (7.1)

Table E.7: Statistics of the 5-fold splitting in ADNI-2 (MCI-p vs MCI-s, cohort CS4).

FOLD 1

Numbers	M	F	Total
MCI-p	13	9	22
MCI-s	12	8	20
Total	25	17	42
Ages (std)	**M**	**F**	**Total**
MCI-p	74.6 (7.0)	72.6 (7.8)	73.8 (7.2)
MCI-s	71.5 (6.7)	71.7 (8.0)	71.5 (7.0)
Total	73.1 (6.9)	72.2 (7.6)	72.7 (7.1)

FOLD 2

Numbers	M	F	Total
MCI-p	12	8	20
MCI-s	12	9	21
Total	24	17	41
Ages (std)	**M**	**F**	**Total**
MCI-p	74.6 (6.9)	72.8 (6.9)	73.9 (6.8)
MCI-s	71.3 (8.0)	71.7 (8.0)	71.5 (7.8)
Total	73.0 (7.5)	72.2 (7.3)	72.7 (7.4)

FOLD 3

Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	8	20
Total	25	16	41
Ages (std)	**M**	**F**	**Total**
MCI-p	74.6 (7.3)	72.7 (8.4)	73.9 (7.6)
MCI-s	71.5 (7.8)	71.8 (8.1)	71.7 (7.7)
Total	73.1 (7.5)	72.3 (8.0)	72.8 (7.6)

FOLD 4

Numbers	M	F	Total
MCI-p	12	9	21
MCI-s	12	8	20
Total	24	17	41
Ages (std)	**M**	**F**	**Total**
MCI-p	74.6 (7.0)	72.6 (7.0)	73.8 (6.9)
MCI-s	71.5 (6.7)	71.7 (8.0)	71.6 (7.0)
Total	73.1 (6.9)	72.2 (7.3)	72.7 (7.0)

FOLD 5

Numbers	M	F	Total
MCI-p	12	8	20
MCI-s	12	9	21
Total	24	17	41
Ages (std)	**M**	**F**	**Total**
MCI-p	74.6 (7.1)	72.7 (6.9)	73.8 (6.9)
MCI-s	71.5 (7.0)	71.8 (8.1)	71.6 (7.3)
Total	73.0 (7.1)	72.2 (7.3)	72.7 (7.1)

FOLD 6

Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	8	20
Total	25	16	41
Ages (std)	**M**	**F**	**Total**
MCI-p	74.7 (7.0)	72.8 (8.1)	74.0 (7.3)
MCI-s	71.5 (6.7)	71.9 (10.3)	71.7 (8.0)
Total	73.1 (6.9)	72.4 (9.0)	72.8 (7.7)

FOLD 7

Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	8	20
Total	25	16	41
Ages (std)	**M**	**F**	**Total**
MCI-p	74.7 (7.0)	72.8 (6.9)	74.0 (6.9)
MCI-s	71.5 (6.6)	71.7 (10.0)	71.6 (7.9)
Total	73.2 (6.9)	72.2 (8.3)	72.8 (7.4)

FOLD 8

Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	8	20
Total	25	16	41
Ages (std)	**M**	**F**	**Total**
MCI-p	74.6 (7.9)	72.8 (6.9)	73.9 (7.4)
MCI-s	71.5 (6.7)	71.8 (8.1)	71.6 (7.1)
Total	73.1 (7.4)	72.3 (7.3)	72.8 (7.3)

FOLD 9

Numbers	M	F	Total
MCI-p	12	9	21
MCI-s	13	8	21
Total	25	17	42
Ages (std)	**M**	**F**	**Total**
MCI-p	74.5 (8.0)	72.7 (7.0)	73.7 (7.4)
MCI-s	71.4 (6.7)	71.7 (8.0)	71.5 (7.0)
Total	72.9 (7.4)	72.2 (7.2)	72.6 (7.2)

FOLD 10

Numbers	M	F	Total
MCI-p	13	8	21
MCI-s	12	8	20
Total	25	16	41
Ages (std)	**M**	**F**	**Total**
MCI-p	74.6 (7.0)	72.8 (6.9)	73.9 (6.9)
MCI-s	71.4 (6.8)	71.7 (8.2)	71.5 (7.2)
Total	73.1 (7.0)	72.2 (7.4)	72.7 (7.0)

Table E.8: Statistics of the 10-fold splitting in ADNI-2 (MCI-p vs MCI-s, cohort CS4).

Appendix F

Additional figures

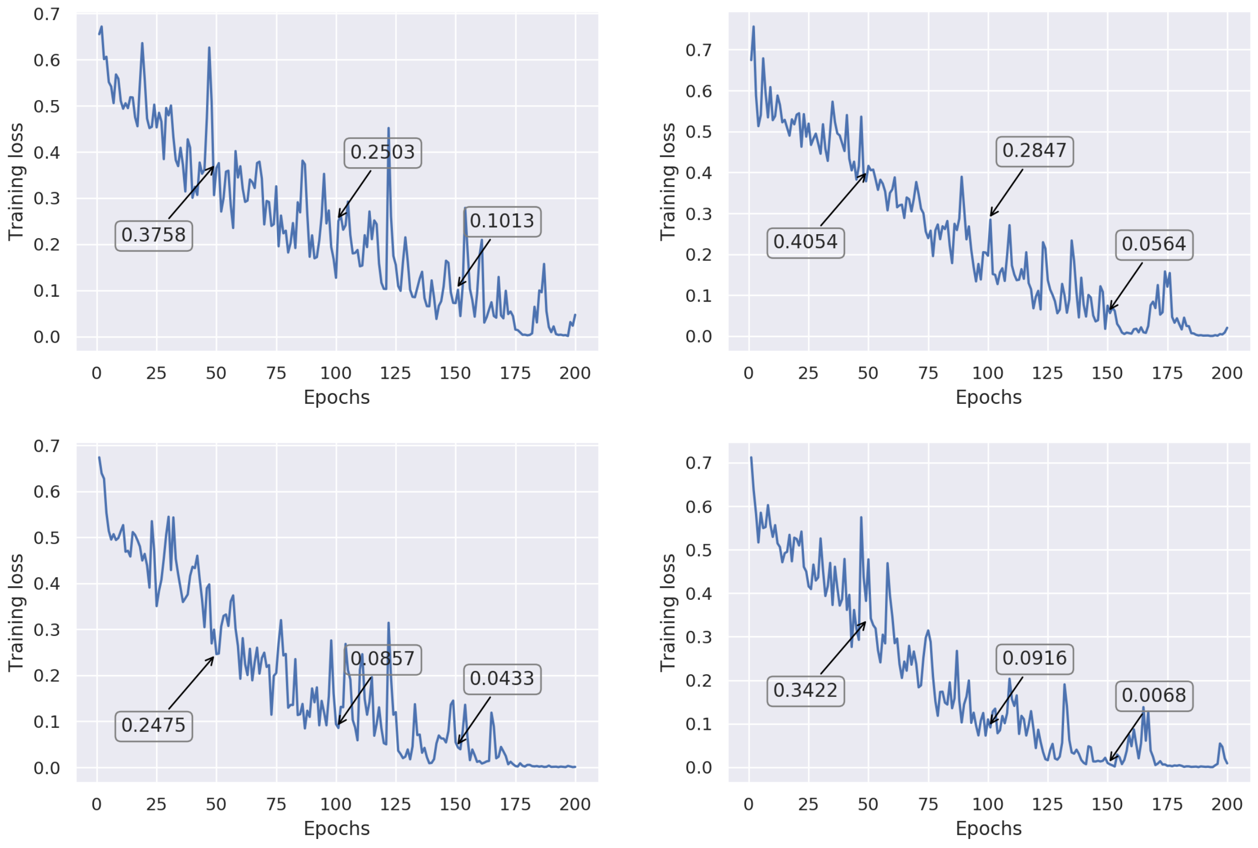


Figure F.1: Training losses on the SBU architecture for the AD-vs-CN task, with constant learning rate of 0.1 (pointers at epochs 50, 100 and 150).

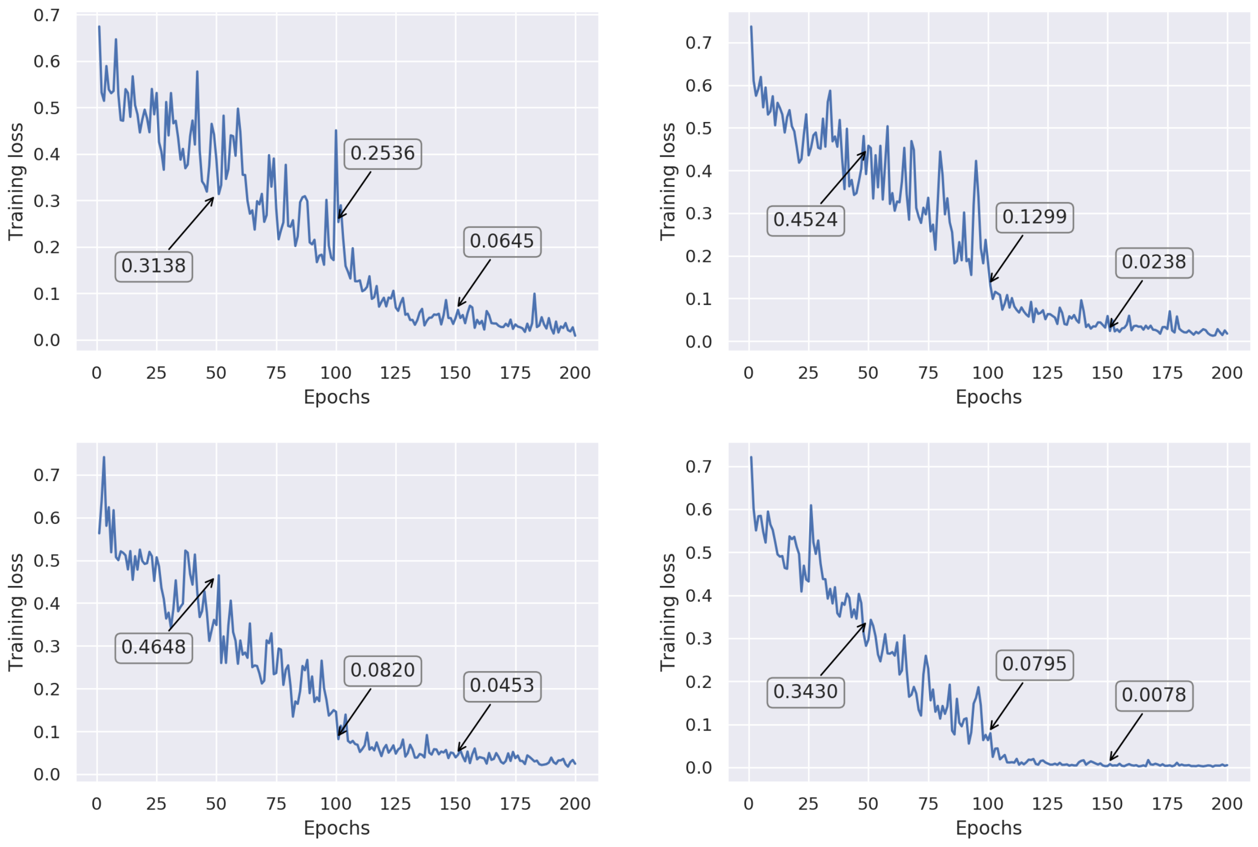


Figure F.2: Training losses on the SBU architecture for the AD-vs-CN task, with an initial learning rate of 0.1 decreased to 0.01 at epoch 100 (pointers at epochs 50, 100 and 150).

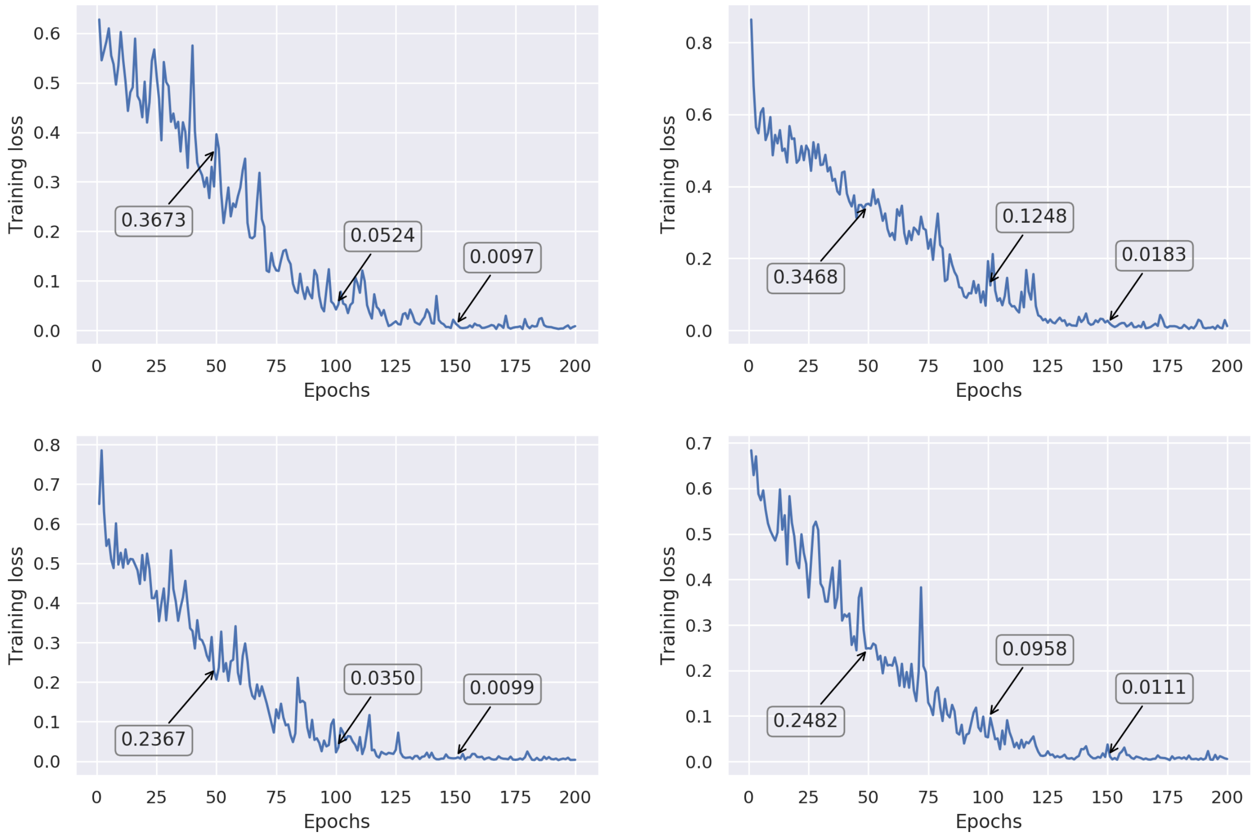


Figure F.3: Training losses on the SBU architecture for the AD-vs-CN task, with an initial learning rate of 0.1 decreased by 2 once every 40 epochs (pointers at epochs 50, 100 and 150).

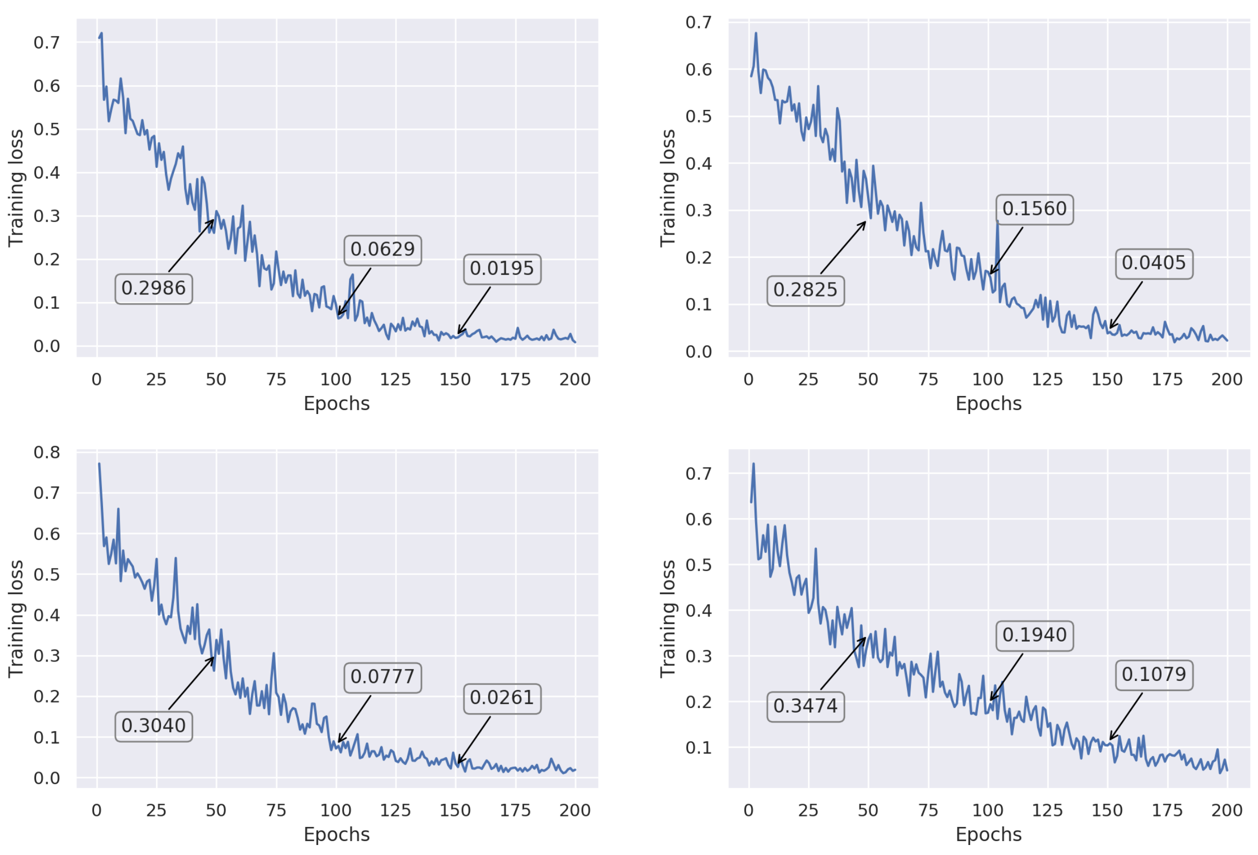


Figure F.4: Training losses on the SBU architecture for the AD-vs-CN task, with an initial learning rate of 0.1 and continuous exponential decay (pointers at epochs 50, 100 and 150).

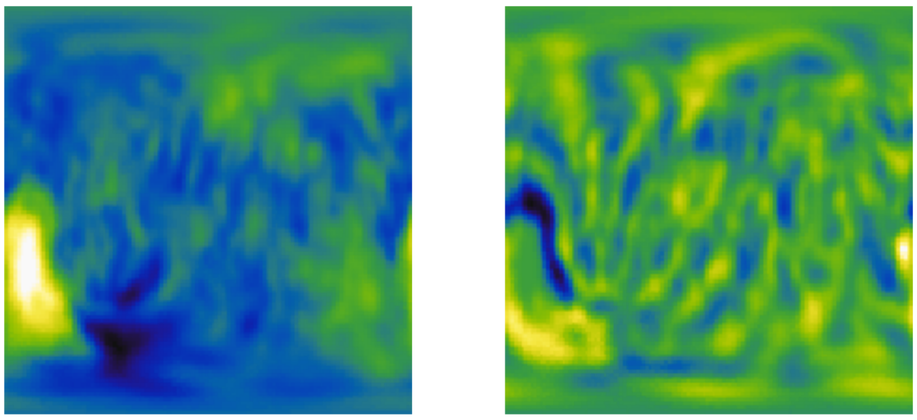


Figure F.5: Hemisphere combinations left+right (on the left) and left-right (on the right).

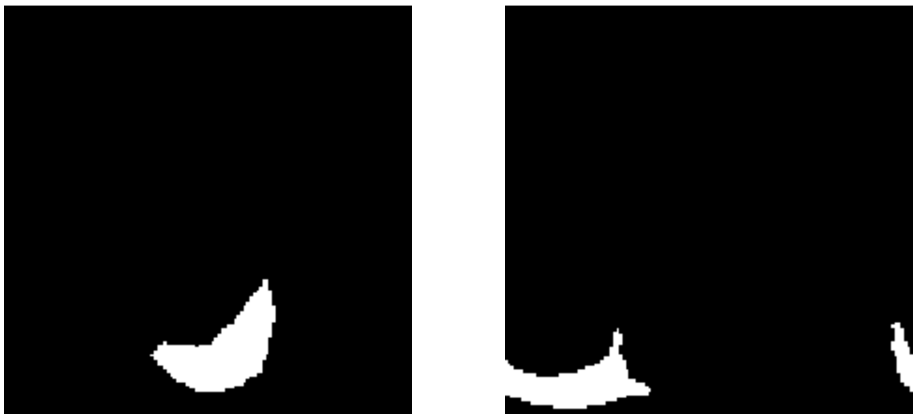


Figure F.6: Masked top performing Desikan (left, n9: inferior temporal gyrus) and Destrieux (right, n23: Parahippocampal part of the medial occipito-temporal gyrus) structures.

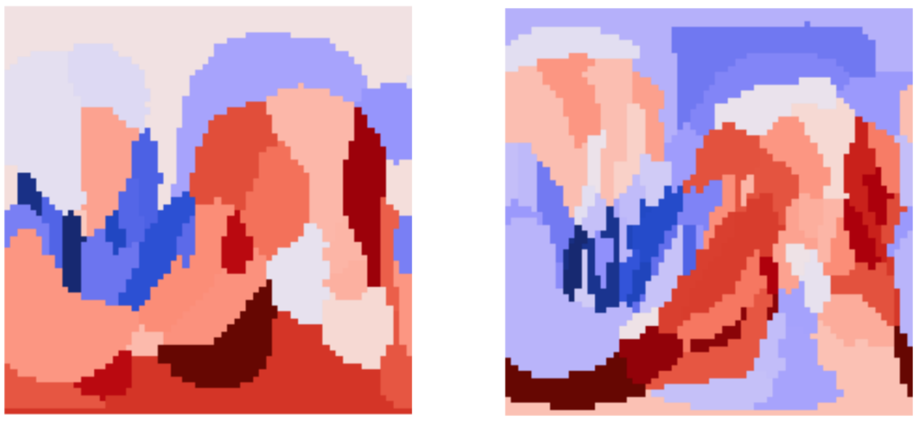


Figure F.7: Heat map of classification accuracies for each individual structure in Desikan (left) and Destrieux (right) atlases → blue to red from low to high accuracy.

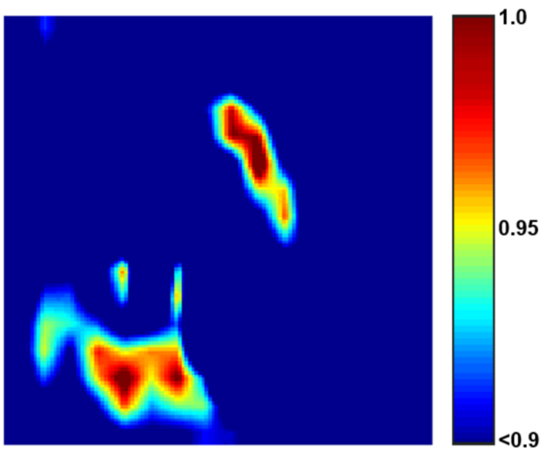


Figure F.8: Average Class Activation Map for Spherical CNNs (from [1]).

Appendix G

Additional tables

Table G.1: Performance metrics (%) of various learning rates in the AD-vs-CN task (SBU architecture, no learning rate decay).

Learning rate	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
0.3	**94.12**	91.27 (2.19)	**94.89 (1.15)**	88.11 (3.51)	95.05 (2.95)
0.1	**94.12**	**91.67 (1.54)**	94.61 (1.22)	**88.47 (2.39)**	95.48 (2.04)
0.05	92.65	89.61 (1.80)	93.43 (0.61)	84.32 (3.42)	**95.91 (1.48)**

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

Table G.2: Performance metrics (%) of various learning rates in the MCI-progression task (SU architecture with left hemisphere; left: exponential decay rate, right: single-step decay).

Learning rate	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
0.3	76.47	70.59 (4.32)	73.79 (1.93)	74.05 (10.84)	66.38 (8.29)	**76.47**	**73.99 (2.51)**	74.73 (1.45)	**80.95 (6.57)**	65.51 (7.79)
0.1	**78.43**	72.16 (2.98)	**74.62 (2.41)**	74.29 (6.91)	69.57 (5.20)	**76.47**	72.29 (3.22)	**75.33 (2.57)**	74.52 (7.13)	69.57 (4.03)
0.05	**78.43**	**72.81 (2.55)**	73.14 (1.29)	**74.76 (5.48)**	**70.43 (5.50)**	**76.47**	72.29 (3.77)	74.28 (1.55)	72.62 (9.51)	**71.88 (4.89)**

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

Table G.3: Performance metrics (%) of multiple longitudinal combinations in the AD-vs-CN task (SBU architecture, constant 0.1 learning rate).

Longitudinal features	Max ACC	ACC (std)	AUC (std)	SEN (std)	SPE (std)
Baseline	**94.12**	**91.67 (1.54)**	94.61 (1.22)	**88.47 (2.39)**	**95.48 (2.04)**
B + 6-month visit	90.77	87.28 (2.99)	92.36 (0.91)	80.57 (6.13)	95.11 (2.48)
B + 12-month visit	90.16	85.36 (2.66)	93.38 (0.76)	75.11 (5.02)	95.27 (2.06)
B + 24-month visit	92.16	88.37 (2.28)	94.76 (1.36)	85.87 (3.96)	90.77 (2.83)
B + 6M + 12M	91.53	84.86 (3.30)	91.67 (1.40)	74.71 (6.60)	94.67 (1.69)
B + 6M + 24M	92.00	87.87 (3.34)	94.13 (1.05)	84.72 (6.80)	90.77 (2.83)
B + 12M + 24M	91.84	88.03 (2.54)	94.75 (1.36)	84.35 (5.14)	91.28 (1.76)
B + 6M + 12M + 24M	93.75	88.33 (3.03)	**95.00 (1.49)**	83.03 (6.76)	92.82 (2.46)

(ACC: Accuracy / AUC: Area under a ROC curve / SEN: Sensitivity / SPE: Specificity)

Bibliography

[1] X. Feng, J. Yang, A. F. Laine, and E. D. Angelini. Discriminative analysis of the human cortex using spherical CNNs - a study on Alzheimer's disease diagnosis. `https://arxiv.org/abs/1812.07749`, 2018. Accessed: 2019-09-06.

[2] World Health Organization. World health statistics 2016: Monitoring health for the SDGs. `https://www.who.int/gho/publications/world_health_statistics/2016/Annex_B/en/`. Accessed: 2019-09-06.

[3] Alzheimer's Disease International. World Alzheimer Report, 2018.

[4] Freesurfer. `https://surfer.nmr.mgh.harvard.edu/`. Accessed: 2019-09-06.

[5] A. M. Dale, B. Fischl, and M. I. Sereno. Cortical surface-based analysis I: Segmentation and surface reconstruction. *NeuroImage*, 9:179–194, 1999.

[6] B. Fischl, M. I. Sereno, and A. M. Dale. Cortical surface-based analysis II: Inflation, flattening, and a surface-based coordinate system. *NeuroImage*, 9:195–207, 1999.

[7] J. Talairach and P. Tournoux. *Co-Planar Stereotaxic Atlas of the Human Brain*. Thieme Medical Publishers, 1988.

[8] D. L. Collins, P. Neelin, T. M. Peters, and A. C. Evans. Automatic 3D intersubject registration of MR volumetric data in standardized Talairach space. *Journal of Computer Assisted Tomography*, 18(2):192–205, 1994.

[9] F. Segonne, A. M. Dale, E. Busa, M. Glessner, D. Salat, H. K. Hahn, and B. Fischl. A hybrid approach to the skull stripping problem in MRI. *NeuroImage*, 22(3):1060–1075, 2004.

[10] B. Fischl and A. M. Dale. Measuring the thickness of the human cerebral cortex from magnetic resonance images. *Proceedings of the National Academy of Sciences of the United States of America*, 97(20):11050–11055, 2000.

[11] R. S. Desikan, F. Segonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman, M. S. Albert, and R. J. Killiany. An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest. *NeuroImage*, 31(3):968–980, 2006.

[12] C. Destrieux, B. Fischl, A. M. Dale, and E. Halgren. Automatic parcellation of human cortical gyri and sulci using standard anatomical nomenclature. *NeuroImage*, 53(1):1–15, 2010.

[13] B. Fischl, A. van der Kouwe, C. Destrieux, E. Halgren, F. Segonne, D. H. Salat, E. Busa, L. J. Seidman, J. Goldstein, D. Kennedy, V. Caviness, N. Makris, B. Rosen, and A. M. Dale. Automatically parcellating the human cerebral cortex. *Cerebral Cortex*, 14(1):11–22, 2004.

[14] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1150–, 1999.

[15] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[16] PASCAL Visual Object Classes. `http://host.robots.ox.ac.uk/pascal/VOC/`. Accessed: 2019-09-06.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[18] ImageNet Large Scale Visual Recognition Challenge. `http://image-net.org/challenges/LSVRC/`. Accessed: 2019-09-06.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, volume 1, pages 1097–1105, 2012.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[21] X. Li, L. Yu, C. Fu, and P. Heng. Deeply supervised rotation equivariant network for lesion segmentation in dermoscopy images. In *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, pages 235–243, 2018.

[22] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the 7th International Conference on Document Analysis and Recognition*, volume 2, pages 958–962, 2003.

[23] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. `https://arxiv.org/abs/1712.04621`, 2017. Accessed: 2019-09-06.

[24] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical CNNs. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[25] T. S. Cohen and M. Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pages 2990–2999, 2016.

[26] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[27] J. Cooley, P. Lewis, and P. Welch. Application of the fast Fourier transform to computation of Fourier integrals, Fourier series, and convolution integrals. *IEEE Transactions on Audio and Electroacoustics*, 15(2):79–84, 1967.

[28] M. Sugiura. *Unitary Representations and Harmonic Analysis*. North-Holland Mathematical Library, 1990.

[29] G.B. Folland. *A Course in Abstract Harmonic Analysis*. CRC Press, 1995.

[30] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[31] J. D. Mcauliffe and D. M. Blei. Supervised topic models. In *Advances in Neural Information Processing Systems 20*, pages 121–128. Curran Associates, Inc., 2008.

[32] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[33] D. Zhang and D. Shen. Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in Alzheimer's disease. *NeuroImage*, 59(2):895–907, 2012.

[34] B. Cheng, D. Zhang, D. Shen, B. C. Munsell, and D. Shen. Domain transfer learning for MCI conversion prediction. In *Medical Image Computing and Computer-Assisted Intervention*, pages 82–90, 2012.

[35] K. Hu, Y. Wang, K. Chen, L. Hou, and X. Zhang. Multi-scale features extraction from baseline structure MRI for MCI patient classification and AD early diagnosis. *Neurocomputing*, 175(PA):132–145, 2016.

[36] M. R. Sabuncu and E. Konukoglu. Clinical prediction from structural brain MRI scans: A large-scale empirical study. *Neuroinformatics*, 13(1):31–46, 2015.

[37] I. Beheshti, H. Demirel, and H. Matsuda. Classification of Alzheimer's disease and prediction of mild cognitive impairment-to-Alzheimer's conversion from structural magnetic resource imaging using feature ranking and a genetic algorithm. *Computers in Biology and Medicine*, 83:109–119, 2017.

[38] S. Ramos Bernardes da Silva Filho, J. H. Oliveira Barbosa, C. Rondinoni, A. C. Dos Santos, C. E. Garrido Salmon, N. K. da Costa Lima, E. Ferriolli, and J. C. Moriguti. Neuro-degeneration profile of Alzheimer's patients: A brain morphometry study. *NeuroImage: Clinical*, 15:15–24, 2017.

[39] E. Westman, S. Muehlboeck, and A. Simmons. Combining MRI and CSF measures for classification of Alzheimer's disease and prediction of mild cognitive impairment conversion. *NeuroImage*, 62(1):229–238, 2012.

[40] S. F. Eskildsen, P. Coupe, D. Garcia-Lorenzo, V. Fonov, J. C. Pruessner, and D. L. Collins. Prediction of Alzheimers disease in subjects with mild cognitive impairment from the ADNI cohort using patterns of cortical thinning. *NeuroImage*, 65:511–521, 2013.

[41] E. Moradi, A. Pepe, C. Gaser, H. Huttunen, and J. Tohka. Machine learning framework for early MRI-based Alzheimer's conversion prediction in MCI subjects. *NeuroImage*, 104:398–412, 2015.

[42] Alzheimer's Disease Neuroimaging Initiative. `http://adni.loni.usc.edu`. Accessed: 2019-09-06.

[43] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, volume 37, pages 448–456, 2015.

[44] P. J. Kostelec and D. N. Rockmore. FFTs on the rotation group. *Journal of Fourier Analysis and Applications*, 14(2):145–179, 2008.