Alzheimer's disease detection using a Convolutional Neural Network

July 2021

Fernando Herrán Albelda

KSchool

Contents

List of Figures

List of Tables

Abstract

The creation of a Convolutional Neural Network (CNN) and its performance is analysed in this thesis. The main objective of this project is to build a CNN to predict if a patient has signs of Alzheimer' disease (AD) using a Magnetic Resonance Image (MRI) of his head.

The main reason for analysing the use of this type of neural network is the growing interest in recent years in the use of Artificial Intelligence (AI) in medical applications, and especially in the diagnosis of diseases.

All data used to train and evaluate the CNN has been obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in the analysis or writing of this report. A complete listing of ADNI investigators can be found in this link.

The data used in the thesis consists of MRIs from patients who are cognitively normal (CN) or who have signs of Alzheimer's disease. In total, 1146 MRIs have been used in the study, which have been splitted as follows: 70% to train the CNN (802 samples), 15% to validate it (172 samples) and 15% to test it (172 samples).

Data preprocessing includes techniques of skull-stripping, MRI resize and pixel intensity normalization. Furthermore, the 3D images have been converted to TFRecords to facilitate the implementation of CNN with Tensorflow. The final CNN version presents a recall of 92% and an accuracy of 84%.

Nomenclature

AD	Alzheimer's disease
ADNI	Alzheimer's Disease Neuroimaging Initiative
AI	Artificial Intelligence
CN	Cognitively normal
CNN	Convolutional Neural Network
CT	Computed tomography
EL	Ensemble learning
MRI	Magnetic Resonance Image
NIfTI	Neuroimaging Informatics Technology Initiative
RNN	Recurrent Neural Network

1 Introduction

This study aims to implement a Convolutional Neural Network (CNN) to predict if a patient has Alzheimer's disease using MRIs of his brain. This CNN consists of a binary classification, where there are two possible categories: patient is cognitively normal or has signs of Alzheimer's disease.

This study is justified by the need to learn about the use of Deep Learning to solve problems where images are related, and to understand the impact of the different layers in the performance of the neural network. The importance of the use of Artificial Intelligence in the industry in general, and in medical applications in particular, is another reason to carry out this study. The great variety of neural networks architectures that can be applied to solve Deep Learning problems needs special attention, and knowing how to choose the correct one is an important factor to take into account.

Nowadays, Artificial Intelligence is growing in many industries and has the power to transform business and give solutions to complex problems. The use of Artificial Intelligence applications or algorithms can provide better performance than humans, and so building this algorithms is an important task to be carried out. Other advantages of Artificial Intelligence are the reduction of costs and the increase of efficiency.

According to NIH National Institute on Aging (NIA) (2017), Alzheimer's disease is an irreversible brain disorder where neurons stop functioning, lose connection with other neurons and die. This disease slowly destroys memory and thinking skills, and is the most common cause of dementia among older adults.

Multiple techniques are used to diagnose Alzheimer, which include medical evaluation, mental status testing, a physical and neurological exam, blood tests and brain imaging exams. In this thesis, has been decided to work with MRIs of the head to detect brain anomalies and the presence of Alzheimer symptoms.

1.1 Code repository

All the codes used to carry out this project are stored in a Github repository and can be accessed via this **link**. In the README.md file it is explained how to set-up a virtual python environment with all the needed packages to run the codes without dependency issues.

This project contains five main jupyter Notebooks, which are used to carry out all the process (from capturing the raw data, preprocess it, convert them to TFRecords and create the CNN). Inside the folder Notebooks, there is a folder called aux_functions, which contains some python files with functions used in the Notebooks. The Notebooks are shown below:

- **1_Capture_data**: it is used to reorganize all the raw data downloaded from ADNI database.
- **2_MRI_preprocessing**: it is used to preprocess and clean the MRI files.
- **3_TFR_creation**: it is used to convert the preprocessed data into TFRecords.
- **4_CNN_creation**: it is used to create and evaluate a CNN.
- **5_Alzheimer_prediction**: user can run this notebook to apply the CNN to a MRI file.

1.2 State of the Art

In recent years, multiple studies have been carried out to apply AI in the field of medicine. AI can be applied in multiple areas inside medicine, and one of them is the analysis of medical images. Skin cancer detection using images of the patient's skin, aortic dissection detection using abdominal CT scans, detection of fibroadenoma of the breast using a mammogram, lung cancer detection using thoracic images or Alzheimer's disease detection using MRI are only some of the cases where AI can be applied using medical images.

Although the number of studies where the detection of Alzheimer's disease is analyzed is not very big, most of them have been done in the last years. One of the reasons for this fact is the lack of data in the past, which there is currently no.

In the last years, CNNs has been imposed as the neuronal network used to solve this type of problem. Two of the most important and more modern architectures in the industry are ResNet and InceptionV3. ResNet introduced a residual learning framework that prevents the saturation of the accuracy when the depth of the network increases considerably. On the other hand, InceptionV3 stands out for being more computationally efficient.

Pan et al. (2020) proposed a classifier combining CNN and ensemble learning (EL). In this study, a set of 2D slices (sagittal, coronal and transverse) were used to train the model. to analyze three binary classification tasks. The result of the model revealed an accuracy of 84% for the binary classification task CN vs AD.

Sarraf et al. (2016) used 270,900 AD and 522,900 CN 2D images, where 75% were used to train the model. In this study, two different architectures were adjusted and tested: LeNet and GoogleNet. An Amazon AWS Linux G2.8xlarge, including four high-performance NVIDIA GPUs, each with 1,536 CUDA cores and with 60 GB memory overall was used to train the models. An accuracy of 99.99% and 98.84% was obtained for each model respectively.

Basaia et al. (2019) also used transfer learning to build a CNN model using cross-sectional brain slices of the MRI scans. An accuracy of 98% in the binary classification problem AD vs CN was obtained.

Looking into other studies, the application of transfer learning to use pre-trained and depth neural networks has been extended considerably. The reason for this fact is that, with enough data, deeper neural networks provide better accuracy than simple ones. In this project, transfer learning has not been used to create the neural network.

1.3 Outline

This report is organised as follows. In section one, an introduction of the project explaining the main objectives, its justification, the state of the art and the study outline is described. In section two, all data preprocessing steps are explained, going from data acquisition to data cleaning and data preparation in TFRecords format. Section three comprises the methodology applied in this thesis to solve the original problem, together with the explanation of the CNN applied. The results obtained with the CNN are shown in section four, whilst the conclusions of this thesis are explained in section five. Finally, the interactive frontend created to be used by any user, expert or not, is shown in section six.

2 Dataset

2.1 Data acquisition

The dataset needed to work with, train and test the CNN has been obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI). This organization unites researchers which objective is to study and define the progression of Alzheimer's disease (AD). ADNI organization collects different type s of data, including MRI and PET images, genetics, cognitive tests, blood biomarkers, etc. Access to this data is not opened to the public in general, and an application must be sent in order to get access to the data (here).

ADNI includes data from 800 subjects, where 200 are normal controls, 400 are individuals with MCI and 200 present signs of Alzheimer. From all the data accessible in ADNI, only MRI has been used in this thesis. A MRI is a type of scan that uses strong magnetic fields and radio waves to produce detailed images of different parts of the body. In this case, it produces 3D images of the brain.

Regarding the magnet strengths used by the MRI machines, both 1.5T and 3.0T are presented in the datasets. In this thesis, no distinction has been made between the two magnet strengths. According to General Electric (2018), both 1.5T and 3.0T have pros and cons. The main advantage of 3.0T over 1.5T is that it provides higher clarity and better detail. However, 3.0T can also present artificial features in the image that were not presented in the original image.

2.2 Data preprocessing

As detailed in notebook **1_Data_Acquisition**, MRIs are downloaded from ADNI in NiFTI format (.nii). NIfTI (Neuroimaging Informatics Technology Initiative) files include the 3D image data of the brain and information about the coordinate system and other metadata.

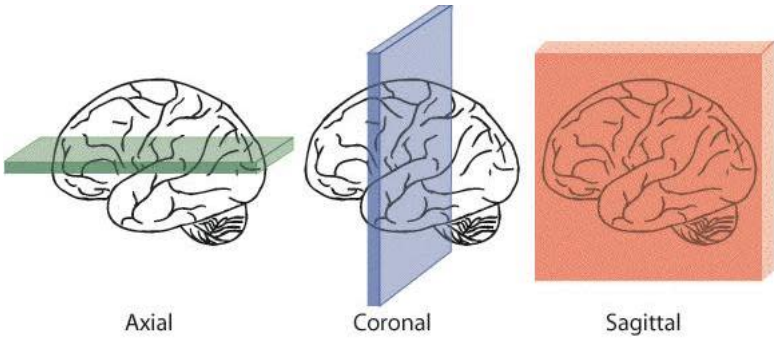

Figure 2.1. Views from a 3D brain.

2.2.1 Skull-stripping

As shown in figure 2.2, the 3D image of the brain presents not only the brain itself but also the skull and other muscles of the patients that should not be considered by the CNN in order to analyze Alzheimer's disease. For this reason, a skull stripping technique must be carried out during the data preprocessing stage, in order to remove these useless parts of the head.

To do that, a Python library called *deepbrain* has been implemented in the code (more information about this package can be found here). This package consists of a CNN, which was trained with different datasets and under a data augmentation process that involved all kinds of rotations and orientations of the brain, and which gives you the probability for each pixel of being brain tissue.

In figure 2.2 it can be seen how the brain mass has been captured for different slices of the brain.

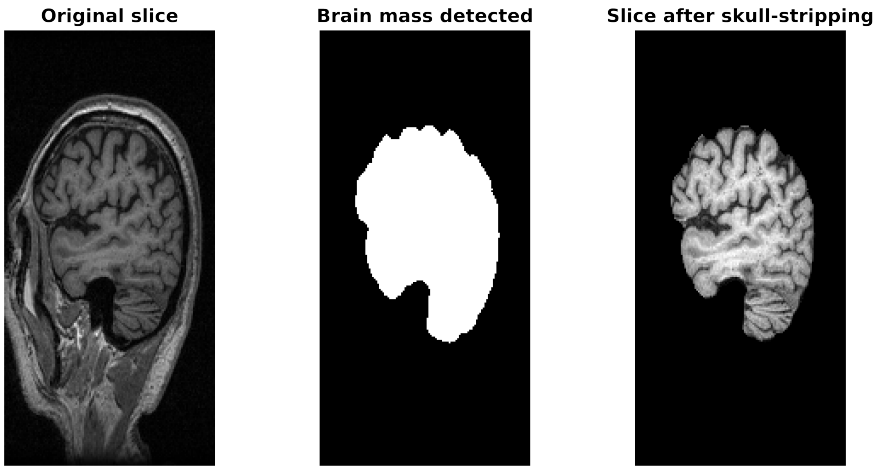

Figure 2.2. Skull-stripping process

2.2.2 Pixel normalization

The next task to carry out during the data preprocessing is the normalization of the pixel values for each sample. To do that, it has been used the equation 2.1.

$$I_N = (I - I_{MIN}) \frac{I_{newmax} - I_{newmin}}{I_{MAX} - I_{MIN}} + I_{newmin} \tag{2.1}$$

where I_{MAX} and I_{MAX} represent the current maximum and minimum pixel intensity present in the 3D image, I_{newmax} and I_{newmin} the desired maximum and

minimum pixel intensity (in this case 1 and 0), I the current pixel value and I_N the new pixel value. More information about image normalization can be found **here**.

2.2.3 Resize 3D images

Each sample of the data can have a different image shape. However, the input shape of the CNN must be always the same. To achieve that, it has to be done the last step of data preprocessing: resize each 3D image to have the same shape for all the samples used to train and test the neural network. In table 2.1 it is shown the different shapes found in the ADNI dataset, together with the number of samples associated with each shape.

Table 2.1. Shapes of the images in the ADNI dataset.

Shape	Nº samples
(256, 256, 166)	587
(192, 192, 160)	558
(256, 256, 180)	157
(256, 256, 170)	218
(256, 256, 161)	2
(170, 256, 256)	4
(256, 256, 160)	4
(256, 256, 146)	1
(256, 256, 162)	1
(256, 256, 184)	41
(256, 256, 150)	2
(160, 192, 192)	4
(192, 192, 176)	1
(166, 256, 256)	1

It can be seen how (256, 256, X) or (192, 192, X) are the predominant shapes among all the samples. Furthermore, looking into more detailed in figure 2.2, it can be seen how after carrying out the skull stripping, there is a huge amount of pixels without brain mass. This phenomenon happens in all the samples of the dataset, and so, the image can be cut in order to reduce the image size and the number of parameters of the neural network. After carrying out the resize of the image, the final shape for all the samples used to train and test the CNN is (110, 130, 80).

2.2.4 TFRecords

Once the MRIs have been preprocessed, it has been decided to convert them into TFRecords in the Notebook **3_TFR_Creation**. According to Tensorflow, TFRecords are used to store the data as a sequence of binary strings. The main advantage of using TFRecords is that it speeds up data reading. More information about TFRecords can be found here.

3 Methodology

As explained previously, the objective of this thesis is to build a classification model to predict if a patient has signs of Alzheimer's disease looking at his MRI. To achieve that, the type of neural network needed to analyse images is the Convolutional Neural Network. In the same way that Recurrent Neural Networks (RNNs), the CNN also learns weights and biases, but each one is associated with a different type of problem. CNNs are used to solve spatial problems, such as an image, whilst RNNs are used to solve temporal or sequential problems, such a text. Another interesting aspect to consider is that CNNs work with fixed inputs size, whilst RNN's inputs sizes can vary.

Below is explained the different layers used in the CNN:

- **Convolutional layer**: it applies filters to all the pixels in the input image.
- **Pooling layer**: it is used to reduce the dimension of an image. This is useful to decrease the computational cost of the CNN and to keep only the important information of the image. The image is splitted into portions according to the kernel size, and depending on if it is used max pooling or average pooling, the layer will return the maximum value of the portion or an average.
- **Batch normalization**: it is a regularization technique that normalizes the inputs to a layer for each mini-batch. The main advantage of using this type of layer is that stabilizes the learning process and reduce the number of epochs required to train the CNN.
- **GlobalAveragePooling3D**: it is a flatten layer used to replace the connected layers from the previous convolutional block and averages out all features within a feature map.
- **Dense**: it is the default connected NN layer.
- **Dropout**: it randomly sets input units to 0 during training time. It is useful to prevent overfitting.

Figure 3.1 shows the structure of the CNN used in this thesis. How this structure was selected is explained in Section 4 Results. As it can be seen, the CNN starts with four convolutional blocks, each one containing a convolutional layer, a pooling layer and a batch normalization. After applying the four convolutional blocks, a flattening is done using a global average pooling layer. Finally, two blocks of dense plus dropout layers are applied before the output layer, which contains one neuron

and uses a sigmoid activation. Table 3.1 shows the number of parameters in the CNN.

Table 3.1. CNN structure

Total parameters	5,045,633
Trainable parameters	5,043,713
Non-trainable parameters	1,920

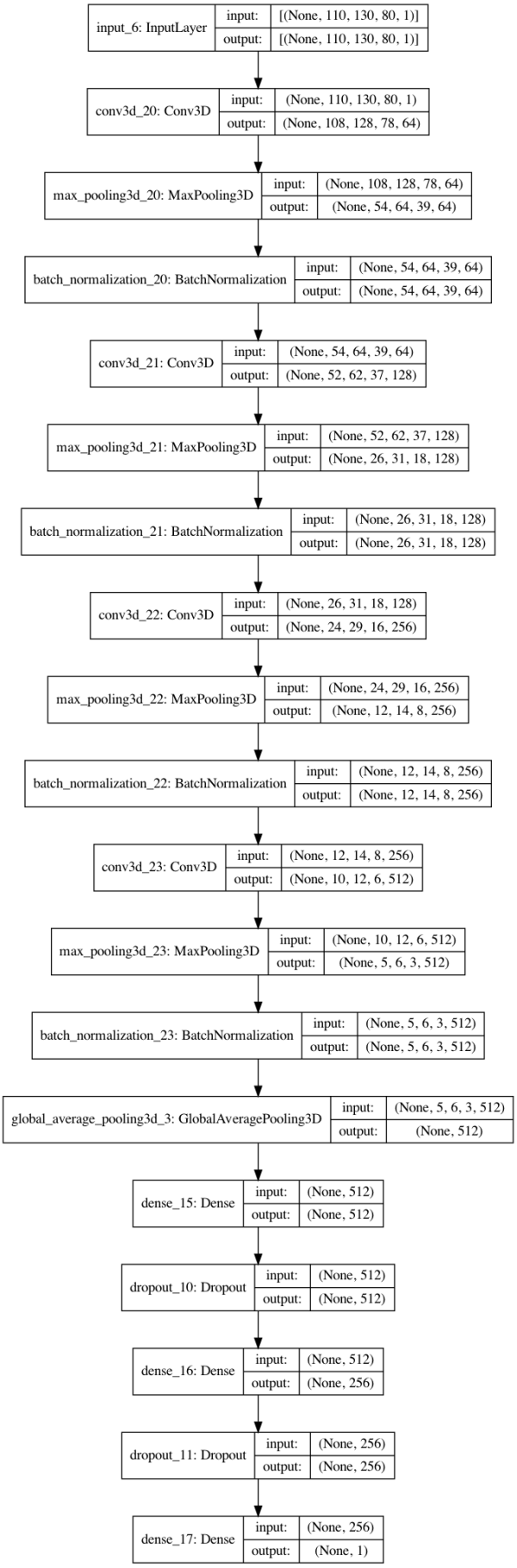

Figure 3.1. CNN structure.

4 Results

4.1 Hyperparameters optimization

One of the most important tasks to create a CNN is to know which set-up (neural network structure, type of optimizer, number of neurons, etc.) is the best one to achieve the best performance possible. In this case, it has been decided to use the metric Recall as the one to optimize and maximize.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4.1}$$

As it can be seen in equation 4.1, the Recall metric takes into account the number of false negatives. When the confusion matrix is calculated, the false negatives are the samples who have been misclassified as false when they are true. In the case of this thesis, a false negative means that the patient has been classified as cognitively normal when really has Alzheimer. Obviously, this type of scenario is the worst possible and should be avoided as much as possible. Therefore the main objective is to get a Recall as high a recall as possible, even if that means increasing the number of false positives.

When defining and compiling the CNN, before train it, multiple variables must be specified by the user: network structure, number of neurons, type of layer activation, type of optimizer, learning rate, batch size, number of epochs, etc.). However, by default, there is not a correct value to be applied to all these variables, and an iterative process must be carried out to find which set-up gives the best performance of the CNN.

To do this iterative process, it has been decided to use the library **Optuna**. This library is an automatic hyperparameter optimization framework, particularly designed for machine learning or deep learning algorithms. The main advantage of using this library is the simplicity with which it is implemented.

The user must define an objective function, where the model is defined according to the hyperparameters suggested and trained. This objective function returns a metric, in this case, the Recall, where the "objective" will be to maximize it. The hyperparameters must be defined by the user and give them a range of possible values (for example, the number of epochs can go from 20 to 150). More information about the implementation of Optuna is detailed in the Notebook **4_CNN_creation**.

From figures 4.1 to 4.4 it is shown an example of the output provided by the Optuna study. In this particular study, 3 different parameters were analysed: the type of optimizer, with four possible values (Adam, Adagrad, Adadelta and RMSprop); the learning rate, with values from 0.00001 to 0.01; and the flatten layer, with two possible values (global average pooling or global max pooling).

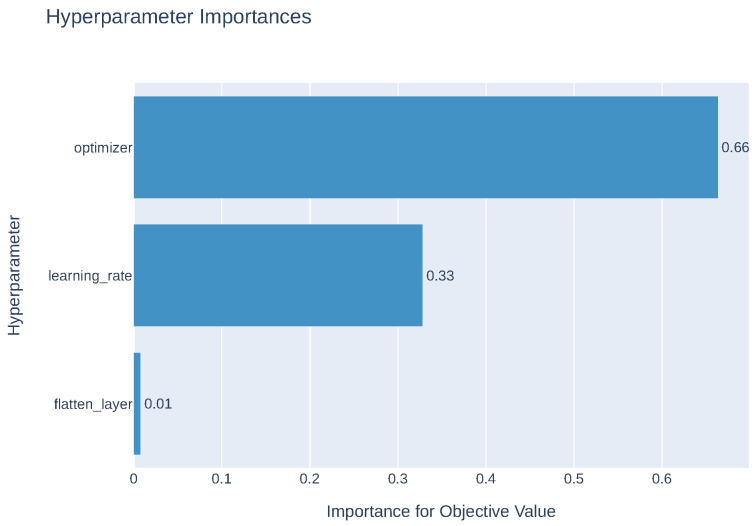

Figure 4.1. Optuna study: importance of the hyperparameters to the metric Recall.

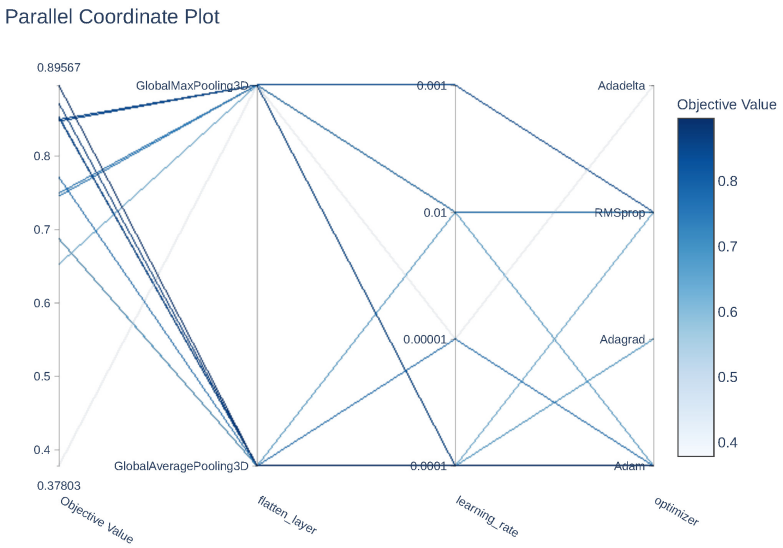

Figure 4.2. Optuna study: coordinate plot with all the results for each iteration.

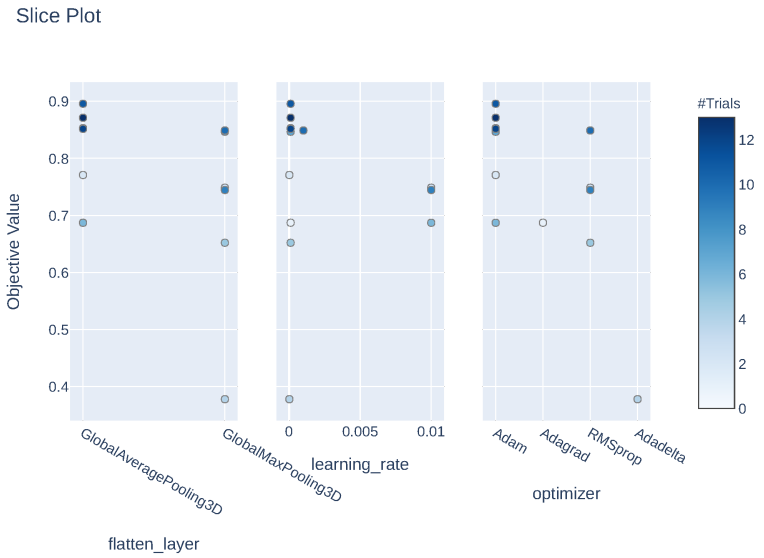

Figure 4.3. Optuna study: slice plot with all the results for each iteration.

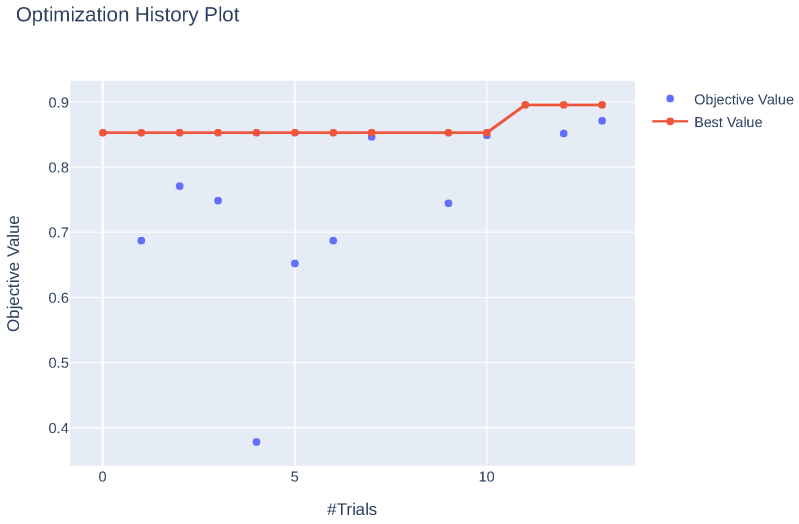

Figure 4.4. Optuna study: Recall value over the different iterations.

From this Optuna study, it was concluded that a global average pooling after the blocks of convolutional layers, together with the optimizer Adam and a low learning rate (0.00001) was the set-up to achieve a higher Recall.

To avoid adding multiple and similar figures with all the Optuna studies carried out during the project, below is presented table 4.1 with the values extracted from the hyperparameter optimization analysis.

Table 4.1. Hyperparameters analysis

Hyperparameter	Value
Number of convolutional blocks	4
Flatten layer	Global Average Pooling
Optimizer	Adam
Learning rate	0.00001
Number of filters first convolutional block	64
Batch size	32
Number of epochs	60
Dropout	0.3

4.2 Model results

In this section, it is presented the results obtained by the CNN created using the hyperparameters extracted from the Optuna studies and shown in the section above.

Figure 4.5 shows the values of the metrics loss, accuracy and F1 score during the training. It can be seen how in the initial epochs, the CNN was not learning and the validation metrics were oscillating, whilst the training metrics were increasing. However, from epochs 50-55, CNN starts to stabilize and stops the oscillation.

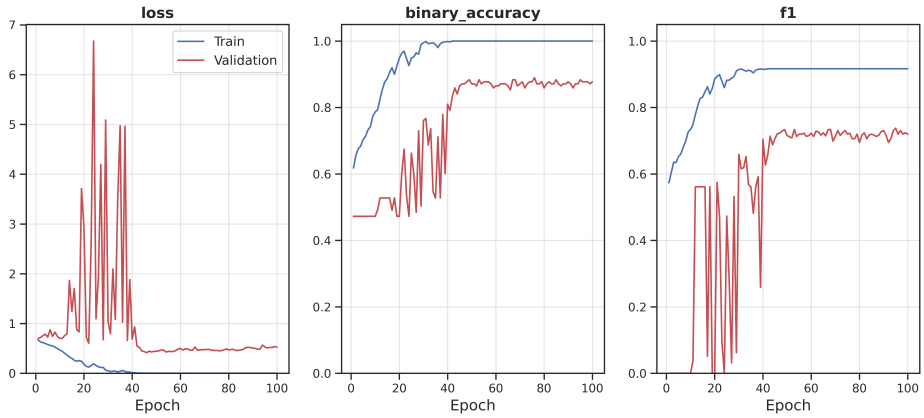

Figure 4.5. Train and validation metrics over training history.

Looking into more detail into the training metrics, it can be seen how the CNN has some overfitting, as loss is close to 0 and accuracy is close to 1. This overfitting can be also observed in figure 4.6, where it is shown the confusion matrix with the training data.

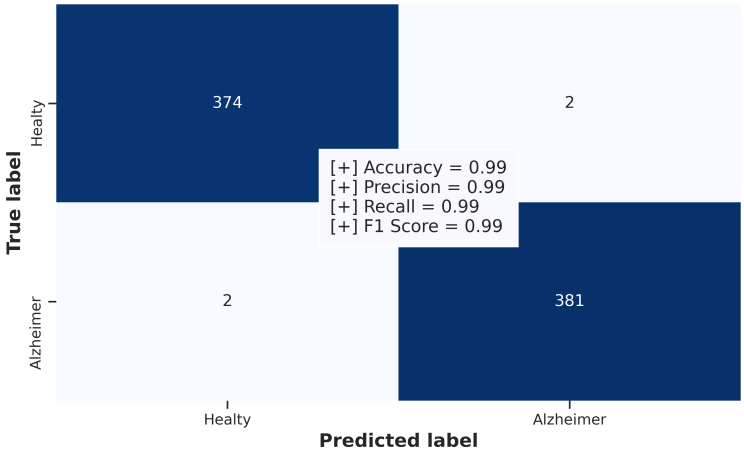

Figure 4.6. Confusion matrix with training dataset.

However, when the CNN is applied to the testing dataset, it can be seen how the performance is also good, achieving a Recall of 92% and an accuracy of 84% (figure 4.6). As expected, the results with the training dataset are better than with the testing data, also taking into account that the model has some overfitting. However, the good metrics obtained with the testing dataset shows that the CNN is able to generalise. Figure 4.8 shows the roc curve, where it can be seen how both training and testing curves are close.

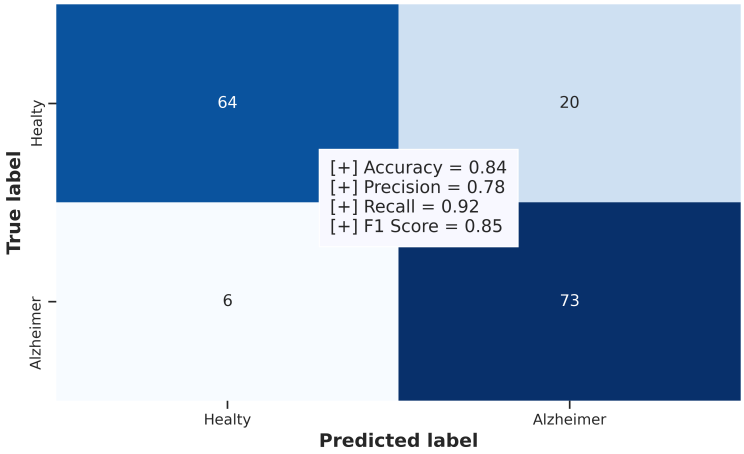

Figure 4.7. Confusion matrix with testing dataset.

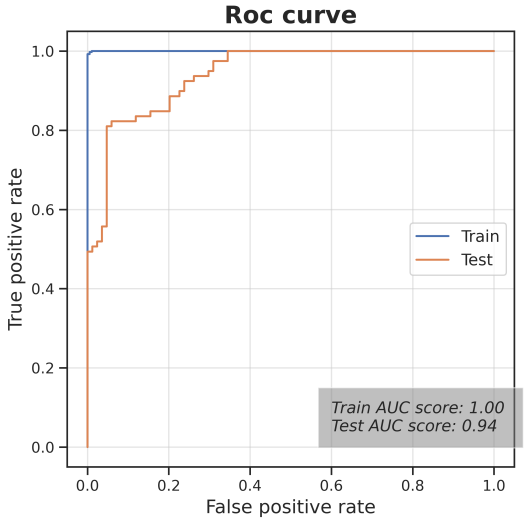

Figure 4.8. Roc curve.

A k-fold cross validation analysis has been also carried out to analyze the performance of the CNN with different blocks of training and testing data, and see if the model can generalize well or if it has been biased. In this case, it has been decided to use 5 folds.

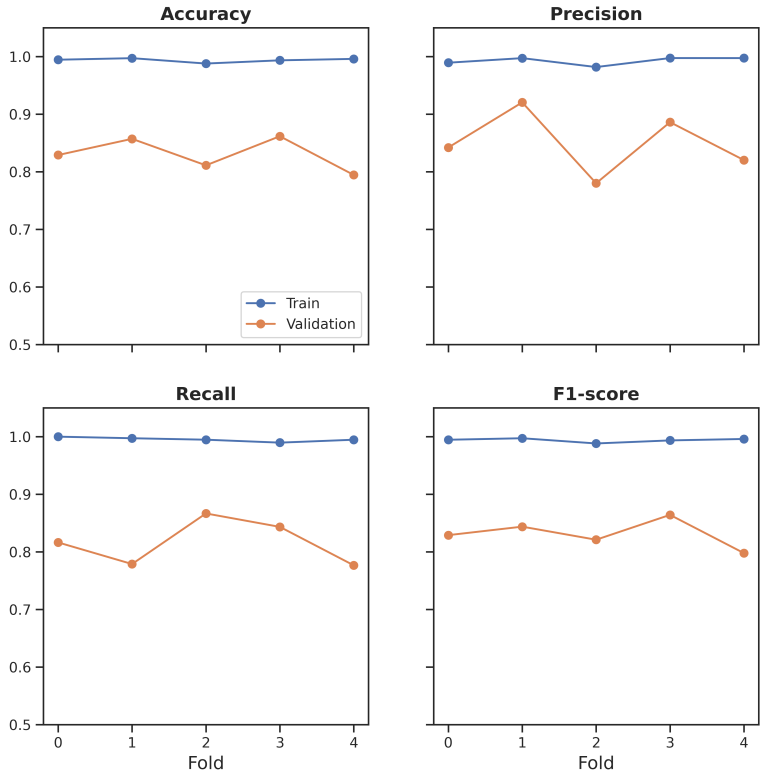

Figure 4.9. Cross-validation: train and validation metrics over folds.

Figure 4.8 shows the results of the k-fold cross validation. It can be seen how the performance of the CNN does not vary too much among the different folds, so it can be concluded that the model has not been biased.

5 Conclusions

As explained during this report, the objective of this thesis has been to apply Machine Learning or Deep Learning models to work with medical images in order to predict if a patient has signs of Alzheimer's disease. To do that, it has been decided to implement a CNN, as this type of neural network is the one used to work with images and solve spatial problems.

The first approach to solve this problem was to use 2D slices of the brain images, which were extracted from the 3D images of the MRIs. However, there was not a scientific method to choose which 2D slices to use, and during the training phase, the model had some limitations when it comes to learning. For these reasons, it was decided to move from a 2D - CNN to a 3D - CNN.

The first challenge to solve when implementing the 3D - CNN was to reduce the size of the images, as due to computational limitations was not possible to train the model with the original sizes. After carrying out the skull-stripping, which is one of the steps of the data preprocessing stage, it was observed how the 3D image could be cut "manually", as all the pixels that were showing the skull and other muscles of the brain were now empty. Furthermore, after cutting the image, a resize was also done to reduce more the size of the images.

Once the problem with the size of the images was solved, and after carrying out the preprocessing stage, the images were saved as a numpy file. However, it was observed that when loading these numpy files to train the neural network, it was taking a huge amount of memory and time. It was a big inconvenience, an after some research it was decided to convert all the images that were in numpy file into TFRecords. These TFRecords were occupying more space than the numpy files, however, when loading them into the jupyter notebook to train the model. it was taking only some seconds, instead of minutes as before.

Once all the data prepared needed to train the model was prepared, the CNN was set up after carrying out a hyperparameter analysis with the library Optuna. To verify that the model was not biased, a fold cross validation was also done.

Comparing the results obtained with previous works, a Recall of 92% seems to be in the range of results of other studies. However, some improvements could be done in future work. In this thesis, 1146 samples have been used to train and test the model. This number of samples could be increased using data augmentation techniques. However, it should also come with some computational improvements.

In order to learn how to set up a CNN and analyse the performance of different layers, it was decided to create a CNN from zero. However, as it has been also done in other studies explained in the State of the Art section, transfer learning could be applied in future works to huge CNN architectures already pre-trained.

6 Frontend

An interactive fronted has been built using Streamlit and can be accessed via this **link**. The goal of this frontend is to create a simple and user-friendly application simple, which could be used by expert people on the subject but also by non-expert people. This application has been deployed to production using Azure (more information about this tool can be found **here**).

As shown in figure 6.1, the frontend is splitted into three different pages that can be accessed via a navigation panel: Home, About and Application. The page Home shows the objective of the website and for what it is used. The page About shows a brief description of the reason to build the website, together with some useful links to the. Finally, the page Application is the one needed to carry out the process to determine if a patient has signs of Alzheimer or not uploading an MRI of his brain.

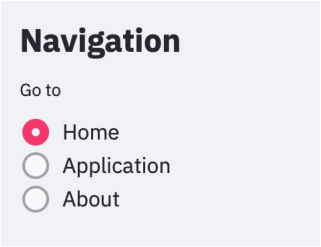

Figure 6.1. Frontend: navigation panel.

Figure 6.2 shows the initial view of Application page that the user will see when access to it. As it can be seen, there is a tool to upload an MRI file. This tool only accepts NiFTI files (.nii) or zipped NiFTI files (.nii.gz). Once the user uploads an MRI, a button to start the process will appear (figure 6.3).

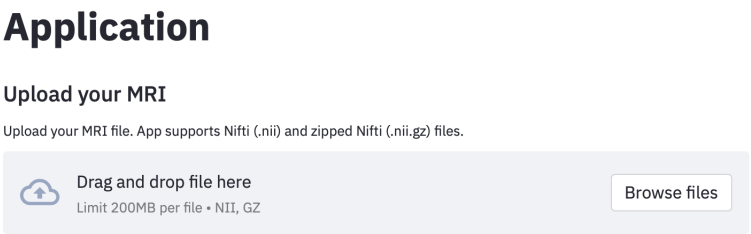

Figure 6.2. Frontend: application page.

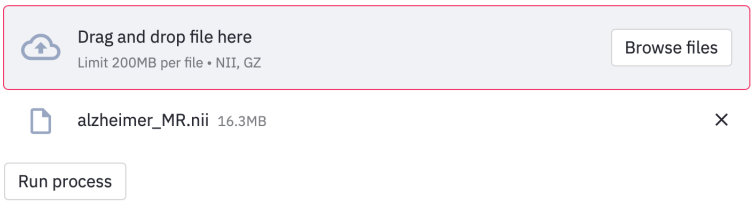

Figure 6.3. Frontend: application page after uploading a MRI file.

Once the user clicks the **"Run process"** button, the algorithm will be activated behind the website and the MRI will start the process: first, it will preprocess the file, carrying out the steps of skull-stripping, image resize and pixel intensity normalization, and finally, it will apply the CNN to the preprocessed image in order to predict the class. Once the process ends, the class predicted and the probability will be displayed on the screen (figure 6.4).

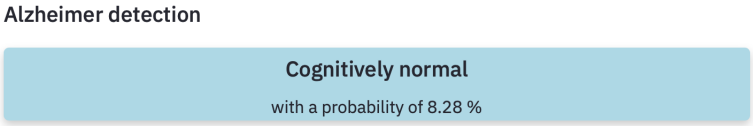

Figure 6.4. Frontend: application page showing class prediction.

Finally, the user will have the option to show the activation maps of the CNN. The user can visualize the activation map from three different views (sagittal, coronal and axial), which must be selected from a dropdown list. Figure 6.5 shows how the activation map is displayed on the screen.

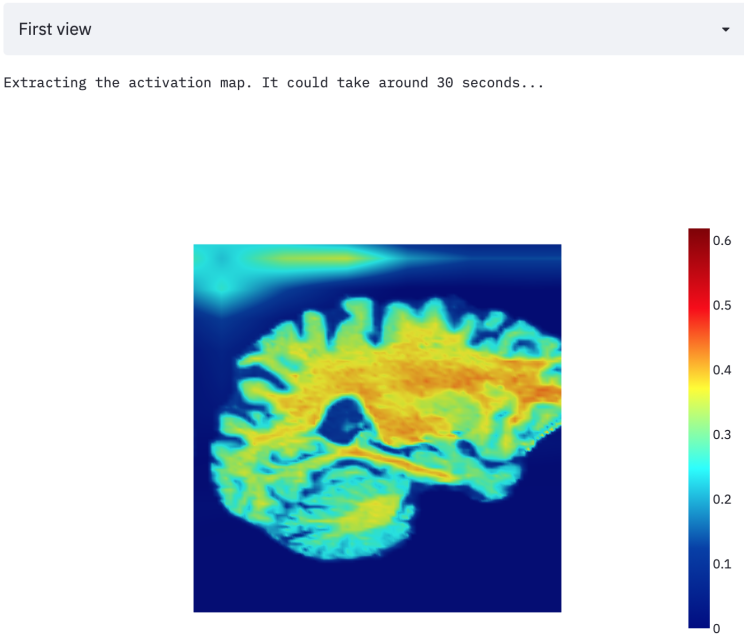

Figure 6.5. Frontend: application page showing activation maps.

The Github repository with all the codes needed to create the application can be accessed via this **link**.

Bibliography

Basaia, S., Agosta, F., Wagner, L., Canu, E., Magnani, G., Santangelo, R. & Filippi, M. (2019), 'Automated classification of alzheimer's disease and mild cognitive impairment using a single mri and deep neural networks', *NeuroImage: Clinical* **21**, 101645.
URL: *https://www.sciencedirect.com/science/article/pii/S2213158218303930*

General Electric (2018), '1.5T Compared to 3.0T MRI Scanners'.
URL: *https://www.gehealthcare.com/feature-article/15t-compared-to-30t-mri-scanners*

NIH National Institute on Aging (NIA) (2017), 'What Is Alzheimer's Disease?'.
URL: *https://www.nia.nih.gov/health/what-alzheimers-disease*

Pan, D., Zeng, A., Jia, L., Huang, Y., Frizzell, T. & Song, X. (2020), 'Early detection of alzheimer's disease using magnetic resonance imaging: A novel approach combining convolutional neural networks and ensemble learning', *Frontiers in Neuroscience* **14**, 259.
URL: *https://www.frontiersin.org/article/10.3389/fnins.2020.00259*

Sarraf, S., Tofighi, G. & for the Alzheimer's Disease Neuroimaging Initiative (2016), 'Deepad: Alzheimer's disease classification via deep convolutional neural networks using mri and fmri', *bioRxiv* .
URL: *https://www.biorxiv.org/content/early/2016/08/21/070441*