Aplicación de deep-learning en neuroimágenes para el diagnóstico de enfermedades neurodegenerativas

Por

Álvaro Corrochano López, Pablo Álvarez García y Antonio Fernández Martín

UNIVERSIDAD COMPLUTENSE
MADRID

Grado en Ingeniería Informática

Facultad de informática

Dirigido por

José Luis Ayala Rodrigo y Fernando García Gutiérrez

Application of deep-learning techniques for diagnosis of neurodegenerative diseases

Madrid, 2021–2022

Deep-learning in neuroimaging

Aplicación de deep-learning en neuroimágenes para el diagnóstico de enfermedades neurodegenerativas

Memoria que se presenta para el Trabajo de Fin de Grado

Álvaro Corrochano López, Pablo Álvarez García y Antonio Fernández Martín

Dirigido por

José Luis Ayala Rodrigo y Fernando García Gutiérrez

Facultad de informática

Universidad Complutense de Madrid

Madrid, 2022

Parte A

Intro

Resumen

El Alzhéimer es la principal causa de demencia en el mundo [1], típicamente caracterizado por la pérdida de memoria seguido por una decline cognitivo. Gracias a avances recientes en el campo de la neuroimagen, como las Resonancias magnéticas (MRI) o la tomografía de emisión de positrones (PET) para el diagnostico de esta enfermedad, se han propuesto diferentes técnicas dentro del campo del *machine learning* para dar lugar a soluciones y diagnósticos. Aquí hemos desarrollado una aplicación basada en aprendizaje profundo para identificar patrones de características comunes entre las neuroimágenes de pacientes diagnosticados con Alzhéimer, este tipo de herramientas tienen una naturaleza de caja negra al tratarse de tecnologías muy complejas. Esto abre la puerta al segundo objetivo del proyecto, usar varios algoritmos y técnicas de interpretación para explicar el diagnostico obtenido de nuestros modelos mediante mapas de calor.

Palabras clave: Neuroimagen, PET, MRI, Machine Learning (ML), Aprendizaje profundo (DL), diagnóstico, caja negra, interpretación, modelo, Alzhéimer (EA).

Abstract

Alzheimer's is the leading cause of dementia in the world [1], typically characterized by memory loss followed by cognitive decline. Thanks to recent advances in the field of neuroimaging, such as Magnetic Resonance Imaging (MRI) or positron emission tomography (PET) for the diagnosis of this disease, different techniques have been proposed within the field of machine learning to give rise to solutions and diagnoses. Here we have developed an application based on deep learning to identify patterns of common characteristics in the neuroimaging of patients diagnosed with Alzheimer's. These types of tools have a black box nature as they are very complex technologies. This opens the door to the second objective of the project, the use of various algorithms and interpretation techniques to explain the diagnosis obtained from our models through heat maps.

Key words: Neuroimaging, PET, MRI, Machine Learning (ML), Deep Learning (DL), diagnosis, black-box, interpretation, model, Alzheimer (AD).

Cita

"Las notas musicales son sólo cinco, pero sus melodías son tan numerosas que no podemos oírlas todas. Los colores primarios son sólo cinco, pero sus combinaciones son tan infinitas que no podemos verlas todas. Los gustos son sólo cinco, pero sus mezclas son tan variadas que no podemos saborearlas todas."

Sun Tzu

Agradecimientos

- **Álvaro**: Agradezco a nuestros tutores, José Luis y Fernando, por toda la ayuda que nos han brindado durante todo el proyecto, a mis dos compañeros, Pablo y Toni, por el buen trabajo que hemos realizado juntos y agradezco especialmente a mi madre, que me ha apoyado siempre en todo y sin ella no podría haberlo logrado.
- **Pablo**: Quiero agradecer en especial a mi familia y amigos por apoyarme en este año tan duro de carrera, a nuestro tutor y director del proyecto por ayudarnos semana tras semana. En especial dedico este agradecimiento a mis dos compañeros del proyecto, que han sabido superar problemas y trabajar en equipo ante todo tipo de situaciones, les deseo un gran futuro profesional y que sigan con la motivación que demostraron en este trabajo de fin de grado.
- **Antonio**: Quiero agradecer a nuestros tutores José Luis y Fernando por la ayuda que nos han brindado semanalmente durante todo un curso, el objetivo de que el proyecto diese los frutos que ha dado era en todo momento común y no simplemente un trabajo más para finalizar la carrera. También quiero agradecerle a mis familiares y amigos por el apoyo que me han dado. Por último y más importante, quiero agradecerle a mis dos compañeros Álvaro y Pablo su implicación y el buen trabajo en grupo que hemos desarrollado durante todo el año y a lo largo de toda la carrera.

Índice general

Parte B

Capítulos

Capítulo 1

Introducción

1.1. La demencia y el Alzhéimer

La demencia es actualmente la séptima causa principal de muerte entre todas las enfermedades y una de las principales causas de discapacidad y dependencia entre las personas mayores a nivel mundial. Además, entre las enfermedades neurodegenerativas, el Alzhéimer (EA) cubre entre el 60 y el 70 % de los casos según datos de la Organización mundial de la salud (OMS) [2], lo que la convierte en la principal causa de demencia en todo el mundo [1].

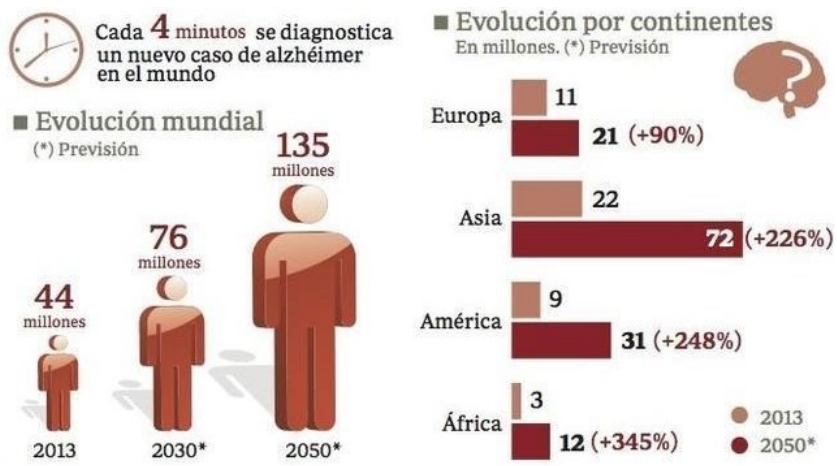


Figura 1.1: Estadística del crecimiento del Alzhéimer [3]

La demencia es un síndrome crónico o progresivo caracterizado por un deterioro cognitivo más rápido al esperado por el envejecimiento normal. La demencia afecta a múltiples funciones cognitivas como por ejemplo la memoria, el pensamiento, la orientación, la comprensión, el cálculo, la capacidad de aprendizaje, el lenguaje o el juicio.
Este deterioro suele ir acompañado o precedido por la alteración al control emocional, el comportamiento social o la motivación. [2]

El desarrollo más común del Alzhéimer se caracteriza por una pérdida de memoria inmediata y de otras capacidades mentales a medida que las neuronas van muriendo y se atrofian diferentes regiones cerebrales. A nivel neuropatológico, la enfermedad se define por la acumulación extracelular de placas de ß-amiloide y del depósito intracelular de proteína tau hiperfosforilada. La acumulación de estos dos sustratos desencadena una serie de eventos que finalmente conducen a la muerte neuronal y la consiguiente atrofia (disminución del volumen cerebral) [4].

1.2. Diagnóstico del Alzhéimer

Convencionalmente, en la práctica clínica, el diagnóstico del Alzhéimer se ha basado exclusivamente en pruebas cognitivas e historial del paciente. Sin embargo, en los últimos años se han comenzado a incorporar otro tipo de biomarcadores como las pruebas de neuroimagen [5]. Entre las técnicas de neuroimagen actualmente establecidas se encuentra la resonancia magnética nuclear (RMN), que permite estudiar la atrofia cerebral [6]; o la Tomografía por Emisión de Positrones (PET), que en combinación con el radiotrazador 2-[^{18}F] fluoro-2-desoxi-D-glucosa (FDG) que permite cuantificar la acumulación de ß-amiloides y el metabolismo cerebral respectivamente [7]. Adicionalmente, en 2020 la FDA aprobó el uso de Tauvid^{TM} para la práctica clínica, un radiotrazador que mide la acumulación de proteína tau en el cerebro [8]. Recientemente, el interés en este biomarcador como herramienta de diagnóstico ha ido en aumento, ya que la evidencia sugiere que los patrones de depósito de tau pueden ser críticos en la detección y el estudio de esta enfermedad [9].

Aunque cada vez se puede extraer más información que permita realizar un diagnóstico más preciso del EA, la capacidad de integrar, analizar y tomar decisiones en base a todos estos datos es todavía limitada. La interpretación correcta requiere un experto capacitado con varios años de experiencia en medicina nuclear. Este hecho se hace especialmente evidente con los datos de neuroimagen, dada su complejidad y el cada vez más creciente número de técnicas disponibles.

1.3. Machine-learning

El *Machine-learning* (ML) es una rama dentro de la inteligencia artificial cuyo objetivo es el desarrollo del aprendizaje en los computadores sin ser programadas directamente [10]. Utilizamos este termino ya que el propio algoritmo de entrenamiento del computador y la parametrización si se programan, no obstante, una vez diseñada su arquitectura, es el propio computador el que resuelve gran parte de la toma de decisiones basándose en su aprendizaje y en los datos que recibe. En el ML un computador observa datos, construye un modelo basado en esos datos y utiliza ese modelo a la vez como una hipótesis acerca del mundo y una pieza de software que puede resolver problemas [11]. Mediante

la observación directa de los datos, los algoritmos de ML son capaces de construir una hipótesis acerca del funcionamiento de un determinado sistema, extrayendo patrones que permitan realizar predicciones sobre su dinámica [12][13].

1.4. Deep-learning

En los últimos años, el campo del ML ha generado una gran variedad de avances notables en algoritmos de aprendizaje y técnicas de procesamiento con gran cantidad de datos.

Su funcionamiento parte de una unidad lógica que emulan el comportamiento de una neurona, un conjunto de estas unidades conectadas dan lugar a una red neuronal. Cada neurona procesa la información y la transmite a otras neuronas. Cada salida cuenta con una función de activación que es la que determina el peso del resultado que pasa a la siguiente neurona, esta jerarquiza la importancia de cada neurona para que se tenga en cuenta en la siguiente entrada[14].

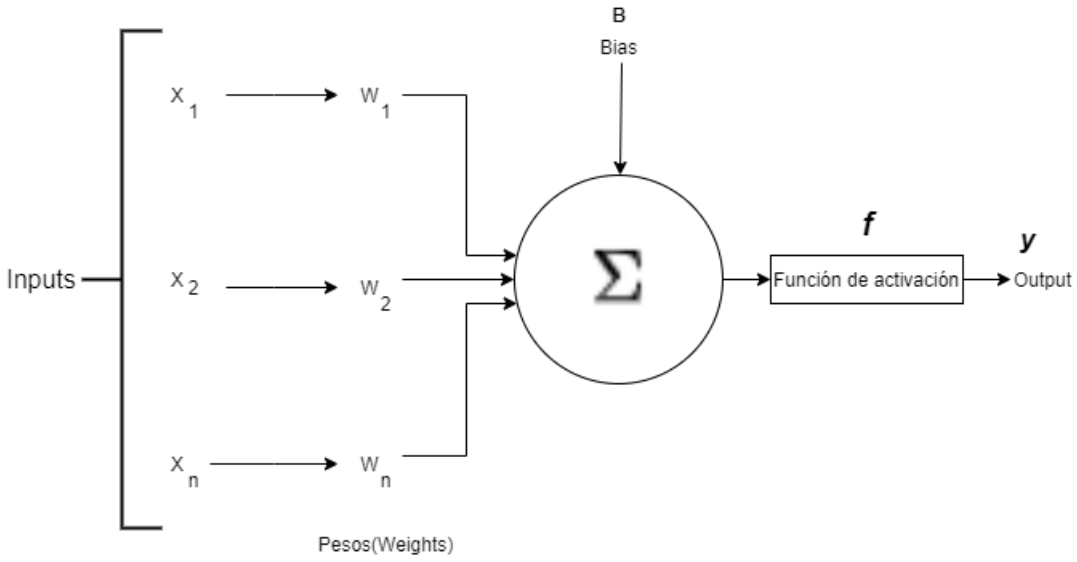


Figura 1.2: Neurona artificial

Donde la salida sería:

$$y = f(\sum_{i=1}^{n} w_n x_n + b)$$

El perceptrón multicapa agrupa sus neuronas en capas, tiene una de entrada, entre ellas tienen sus capas ocultas y por último tiene la capa de salida. La capa de entrada simplemente pasa la información a red sin necesidad de ningún procesamiento de sus neuronas, Las capas ocultas tienen tanto entrada como salida y se conectan entre si. Por último la capa de salida presenta los resultados.
La principal diferencia de la red neuronal convolucional con el perceptrón multicapa viene en que cada neurona no se une con todas y cada una de las capas siguientes, sino que se especializa con un subgrupo de ellas, con esto se consigue reducir el número de neuronas necesarias y la complejidad computacional necesaria para su ejecución.

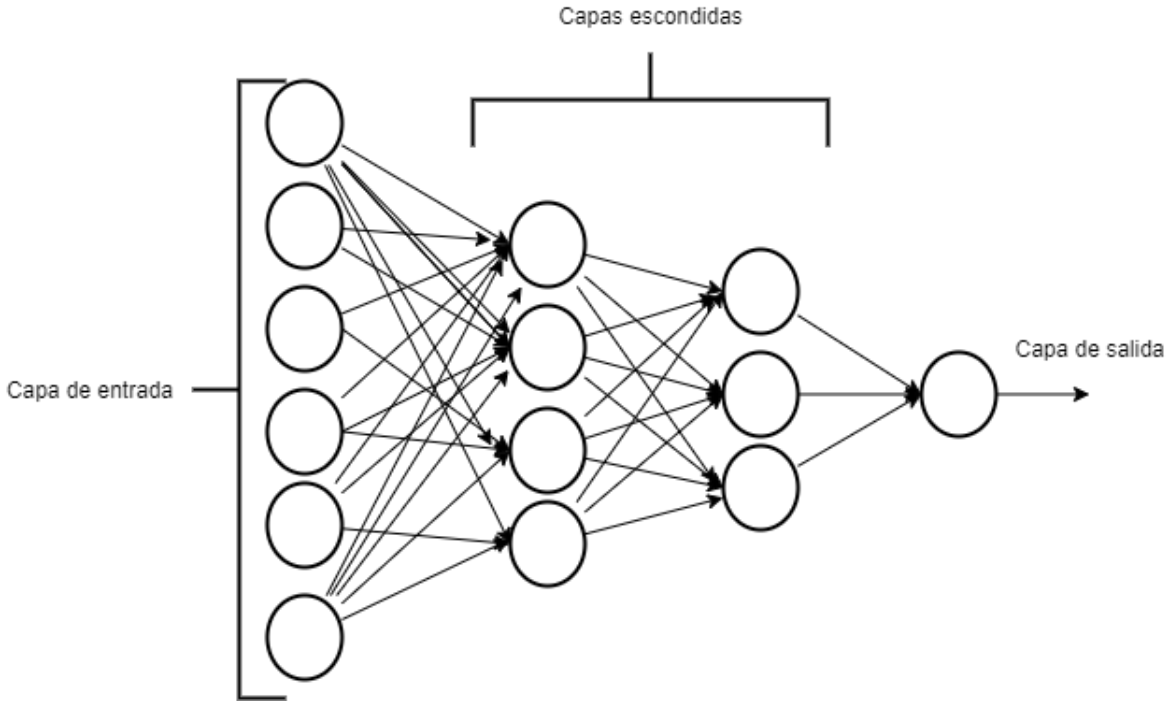


Figura 1.3: Perceptrón multicapa (cada círculo representa una neurona)

Un avance, fue la evolución de las redes neuronales artificiales hacia arquitecturas de redes neuronales más profundas y complejas, que le agregan capacidades de aprendizaje mejoradas a las que había existentes en ML.[15] A esta tecnología se le llama *Deep-Learning* (DL) o aprendizaje profundo, se trata de un algoritmo que imita la percepción humana inspirándose en nuestro cerebro y su sinapsis neuronal[16], forma redes neuronales capaces de identificar clasificar y emular la asociación de los patrones cognitivos humanos. En ciertas aplicaciones específicas superan las capacidades de percepción humana. A diferencia del ML el DL clasifica sus percepciones en jerarquías, filtra su información

mediante capas dentro de sus módulos con lo que su nivel de representación se eleva a la vez que su nivel de abstracción. Esto elimina más información no importante y filtra la más relevante[17].

Los modelos de DL han supuesto un gran avance en la visión artificial [18], extendiéndose al ámbito de la neuroimagen donde se han desarrollado numerosos modelos de predicción basados en MRI [19, 20, 21], PET- amiloide [22, 23] y FDG-PET [24, 25, 26, 27]. Además, algunos autores han explorado la aplicación de técnicas de DL para la extracción de patrones de las de imágenes PET-tau evidenciando su potencial como biomarcador de la enfermedad [28, 29, 30]. No obstante, el número de estudios basados en PET-tau es escaso.

A pesar del éxito de las técnicas de DL en una multitud de aplicaciones [18], su incorporación en el entorno clínico sigue siendo limitada debido a su naturaleza de caja negra. En el contexto de la neuroimagen, la incertidumbre sobre las ubicaciones específicas del cerebro en las que un modelo basa su decisión de diagnóstico limita un uso más práctico [31]. En consecuencia, si el modelo además de diagnosticar al paciente justifica su decisión, la información proporcionada resultaría de interés para el médico al indicarle a qué partes de la imagen el modelo ha dado más importancia. Esto tiene como consecuencia que el uso de modelos DL para respaldar la decisión médica sea valioso y, además, pueda proporcionar información sobre la neuropatología de la enfermedad.

Respecto a las preocupaciones sobre la interpretabilidad de algunos modelos de ML, en los últimos años este campo ha progresado sustancialmente [32], La gran cantidad de métodos propuestos ha llevado al uso generalizado de este tipo de técnicas en el desarrollo actual de modelos predictivos para enfermedades neurodegenerativas [**REF METHS**]. [33, 34, 35, 29, 36, 37, 28, 30]. Sin embargo, se han presentado pocas comparaciones sobre los diferentes métodos actualmente disponibles [38, 39, 40].

Por tanto, ya que el papel de la proteína tau en la neuropatología del EA sugiere que sus patrones de acumulación podrían ser relevantes para su diagnóstico incluso en etapas prodrómicas [9] y que existen pocos estudios que apliquen DL a PET-tau, el primer objetivo de este estudio fue desarrollar y validar un modelo basado en técnicas de DL para la detección de EA en base a este biomarcador. Por otro lado, la limitada interpretabilidad de los modelos de DL representa una restricción sustancial en su incorporación al

entorno clínico. Además, la aplicación de técnicas de interpretabilidad a modelos de DL en imágenes PET-tau es bastante limitada [28, 29, 30], y no conocemos ningún estudio que compare los resultados obtenidos por diferentes métodos en esta modalidad de neuroimagen. Un ejemplo de como sería una neuroimagen sería la figura 4.2, este proyecto se basa en el análisis de una base de datos de imágenes de este tipo.

1.5. Objetivos

Este trabajo tiene como objetivo abordar la brecha existente comparando la efectividad de varias técnicas para interpretar las predicciones de los modelos DL. Por lo tanto, buscamos establecer un punto de partida hacia el desarrollo de modelos basados en PET-tau más apropiados para la práctica clínica.

Para ello definimos dos fases,la fase de modelado y la fase de interpretación.

Comenzaremos por la fase del modelo:

- Crearemos un entorno *framework* en *Python*.
- A partir de la base de datos de neuroimágenes PET-TAU, incorporaremos la información de las imágenes al *framework* creado.
- Crearemos la red neuronal convolutiva, en la cual se ajustarán la arquitectura de la red, los parámetros de entrenamiento, la distribución de datos, la acotación de tiempos y la normalización de parámetros.
- Se analizarán los resultados en varios modelos y se escogerá uno.

Fase de interpretación:

- Con ese modelo escogido se procederá a la interpretación de los datos en la que exploraremos algoritmos de interpretación.
- Seleccionaremos los que resulten ser más interesantes y útiles.
- Exportamos los resultados en imágenes del cerebro en forma de mapa de calor 3D, donde un experto pueda tener acceso a la distribución de la proteína TAU en el cerebro con la enfermedad. Un ejemplo de este tipo de imagen sería la figura 1.4.

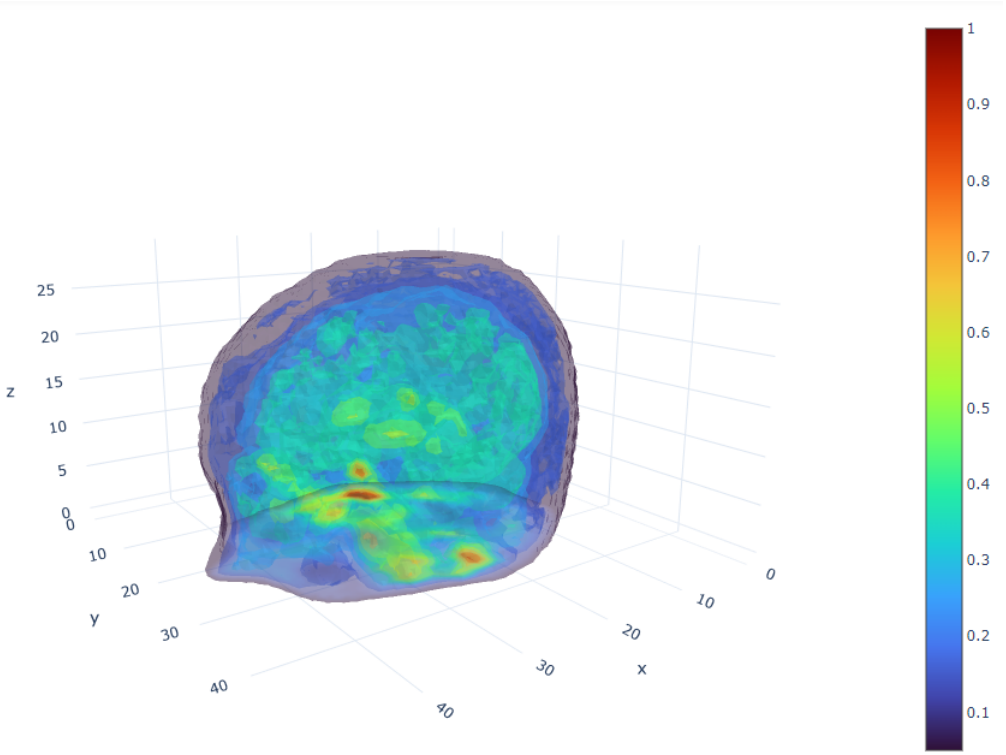


Figura 1.4: Una de las neuroimágenes utilizadas visualizada en 3D

Capítulo 2

Introduction

2.1. Dementia and Alzheimer

Dementia is currently the seventh leading cause of death among all diseases and one of the major causes of disability and dependency among older people globally. Also, Among neurodegenerative diseases, Alzheimer's disease (AD) covers between 60 and 70 % of cases according to WHO data [2], what makes it the major cause of dementia worldwide [1].

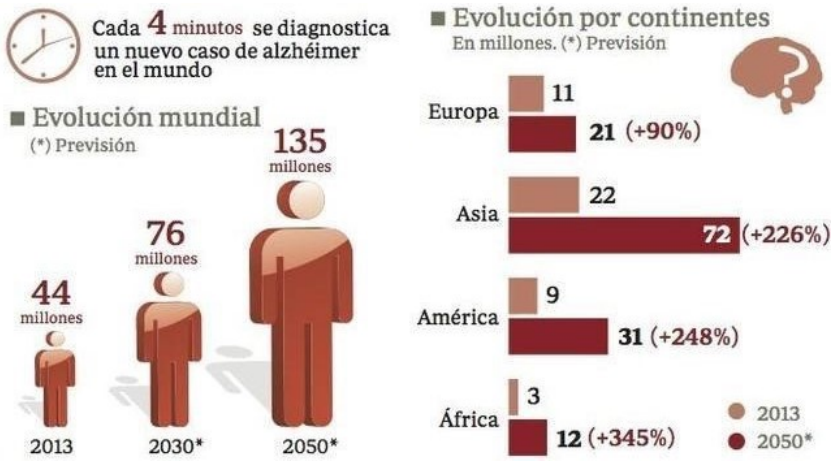


Figura 2.1: Alzheimer's Growth Statistics [3]

Dementia is a chronic syndrome characterized by a faster cognitive decline than the normal expected. Dementia affects a lot of cognitive functions such as memory, orientation, comprehension, learning or language abilities...[41]

This impairment is often preceded by impaired emotional control, social behavior, or motivation.[2]

The most common development of AD is characterized by a loss of short-term memory and other mental abilities as neurons die and different brain regions atrophy. If we talk about neuropathology, the disease is defined by the extracellular accumulation of ß-amyloid plaques and the intracellular deposition of hyperphosphorylated tau protein. The accumulation of these two substrates triggers a series of events that ultimately lead to neuronal death and consequent atrophy (decreased brain volume) [4].

2.2. Alzheimer's Diagnosis

Usually, in clinical practice, the diagnosis of AD has been based on cognitive tests and the medical record of the patient. However, A few years ago, other types of biomarkers have begun to be included, such as neuroimaging tests [5]. Among the currently established neuroimaging techniques is nuclear magnetic resonance (NMR), which allows for studying the brain atrophy [6]; or Positron Emission Tomography (PET), which in combination with the radiotracer 2-[^{18}F] fluoro-2-deoxy-D-glucose (FDG), allows to quantify the accumulation of ß-amyloid and brain metabolism [7]. In addition, in 2020 the FDA approved the use of Tauvid^{TM} for clinical practice, a radiotracer that measures the accumulation of tau protein in the brain [8]. Recently, this biomarker as a diagnostic tool has been growing, as a result of tau deposition patterns may be critical in detecting and studying this disease [9].

There is more and more information that can be extracted to allow a more precise diagnosis of AD, but the ability to integrate, analyze and make decisions based on all this data is still limited. A Correct interpretation requires an expert with several years of experience in nuclear medicine. This fact is especially obvious with neuroimaging data, because of its complexity and the increasing number of available techniques.

2.3. Machine-learning

Machine-learning (ML) is a branch of artificial intelligence whose objective is the development of learning in computers without being directly programmed [10]. We use this term

since the computer's training algorithm and parameterization are programmed, however, their architecture has been designed, it's the computer that resolves a large part of the decision based on its learning and the data that receives. In ML a computer looks at data, builds a model based on that data, and it uses that model as a hypothesis about the world and a piece of software that can resolve [11] problems. Through direct observation of the data, ML algorithms are able of building a hypothesis about the logic of a certain system, extracting patterns that allow predictions about its performances [12][13].

2.4. Deep-learning

In recent years, the field of machine learning has generated a variety of advances in learning algorithms and data processing techniques.

Its operation starts from a logic unit that imitates the behavior of a neuron, a set of these connected units shape a neural network. Each neuron processes the information and carries it to other neurons. Each output has an activation function that defined the weight of the result that carries to the next neuron, this prioritizes the importance of each neuron.[14].

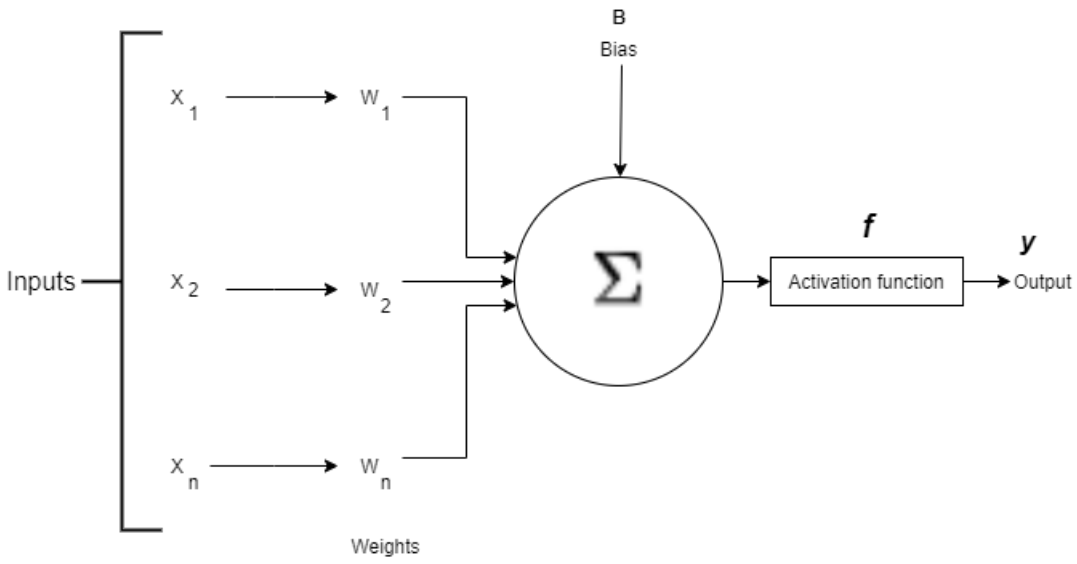


Figura 2.2: Artificial neuron

Output:

$$y = f(\sum_{i=1}^{n} w_n x_n + b)$$

The multilayer perceptron groups its neurons by layers, it has an input layer, between them they have hidden layers, and finally, it has the output layer. The input layer carries the information to the network without processing. The hidden layers have both, input and output, they are connected. Finally, the output layer presents the results.

The main difference between the convolutional neural network (CNN) and the multilayer perceptron is that each neuron doesn't connect with all of the following layers, but specializes with a subgroup of them, reducing the number of neurons needed and the computational complexity necessary for its execution.

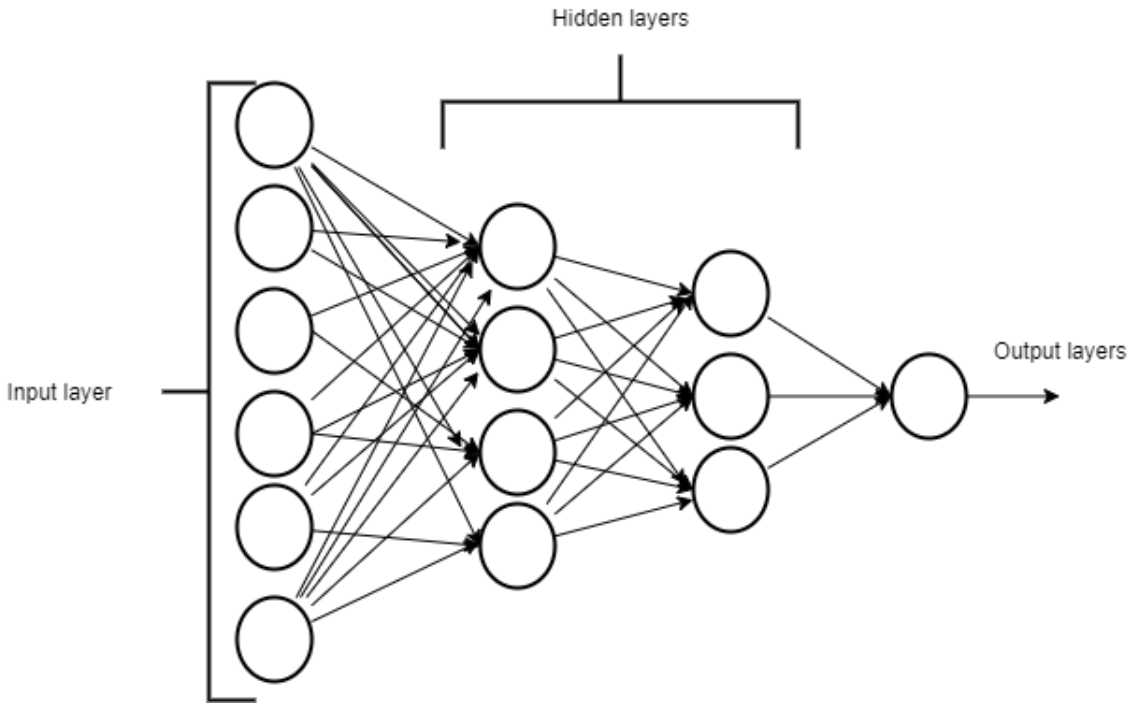


Figura 2.3: Multilayer perceptron (Each circle represents a neuron)

An important advance in this field was the evolution of artificial neural networks to deeper and more complex neural network architectures, which add improved learning habilities to ML.[15] This technology is called Deep-Learning (DL), is an algorithm that imitates human perception inspired by our brain and its neuronal synapse[16], it forms

neural networks able of identifying, classifying, and emulating the association of human cognitive patterns. In some specific applications, they get better results than human perception. Unlike Machine learning, DL classifies its perceptions in hierarchies and filters its information through layers with its modules, which raise the level of representation and at the same time, its level of abstraction. This removes more unimportant information and filters the most important information[17].

Deep-learning (DL) models have been a great advance in artificial vision [18], Also in the field of neuroimaging where numerous MRI-based prediction models have been developed [19, 20, 21], amyloid-PET [22, 23] and FDG-PET [24, 25, 26, 27]. In addition, some authors have explored the application of DL techniques to extract patterns from PET-tau images, showing its potential habitability as a biomarker of disease [28, 29, 30]. However, Nowadays there are few studies based on PET-tau.

Despite the success of DL techniques in a lot of applications [18], their incorporation into the clinical setting remains limited because of their black box nature. In the context of neuroimaging, uncertainty about the specific brain locations where a model bases the diagnostic decision limits the use [31]. Also, if the model is able to diagnosing the patient and justifies its decision, the information provided would be of useful to the doctor by showing which parts of the image are more importance to. Consequently the use of DL models supports medical decisions and can provide information of the neuropathology in the disease.

Regard to concerns about the interpretability of some ML models, In recent years the field of interpretability has a greatly improbed [32], Many of the proposed methods has led to the use of this type of techniques in the current development of predictive models for neurodegenerative diseases [**REF METHS**]. [33, 34, 35, 29, 36, 37, 28, 30]. However, there are few comparisons of the different methods that nowadays are available [38, 39, 40].

Therefore, since the role of tau protein in the neuropathology of AD suggests that its accumulation patterns could be useful for the diagnosis even in prodromal stages [9] and that there are few studies that apply DL to PET-tau , the first objective of this study was to develop and validate a model based on DL techniques for the detection of AD based on this biomarker. On the other hand, the limited interpretability of DL models 13 Deep-

learning in neuroimaging UCM represents a substantial restriction in their incorporation into the clinical settings. In addition, the application of interpretability techniques to DL models in PET-tau images is quite limited [28, 29, 30], and we're not aware of any study that compares the results obtained by different methods in this neuroimaging modality. An example of what a neuroimaging look like is the figure 4.2, this project is based on the analysis of a database of this type of images.

2.5. Goals

This work aims to present the existing scarcity by comparing the effectiveness of some techniques to interpret the predictions of DL models. Consequently, we seek to establish a starting point to the development of PET-tau based on models for clinical practice.
For this work, we define two phases, the modeling phase and the interpretation phase.
We will start with the model phase:
- We will create a framework environment in Python.
- From the PET-TAU neuroimaging database, we will incorporate the information from the images into the created framework.
- We create the convolutional neural network(CNN),then the network architecture, training parameters, data distribution, time bounding and parameter normalization will be adjusted.
- The results will be analyzed in several models and one will be chosen. Interpretation phase:
- With this chosen model, we will proceed to the interpretation of the data in which we will explore interpretation algorithms.
- We will select the ones that turn out to be the most interesting and useful.
- We export the results in images of the brain in the form of a 3D heat map, where an expert can have access to the distribution of the TAU protein in the brain with the disease. An example of this type of image is the figure 2.4.

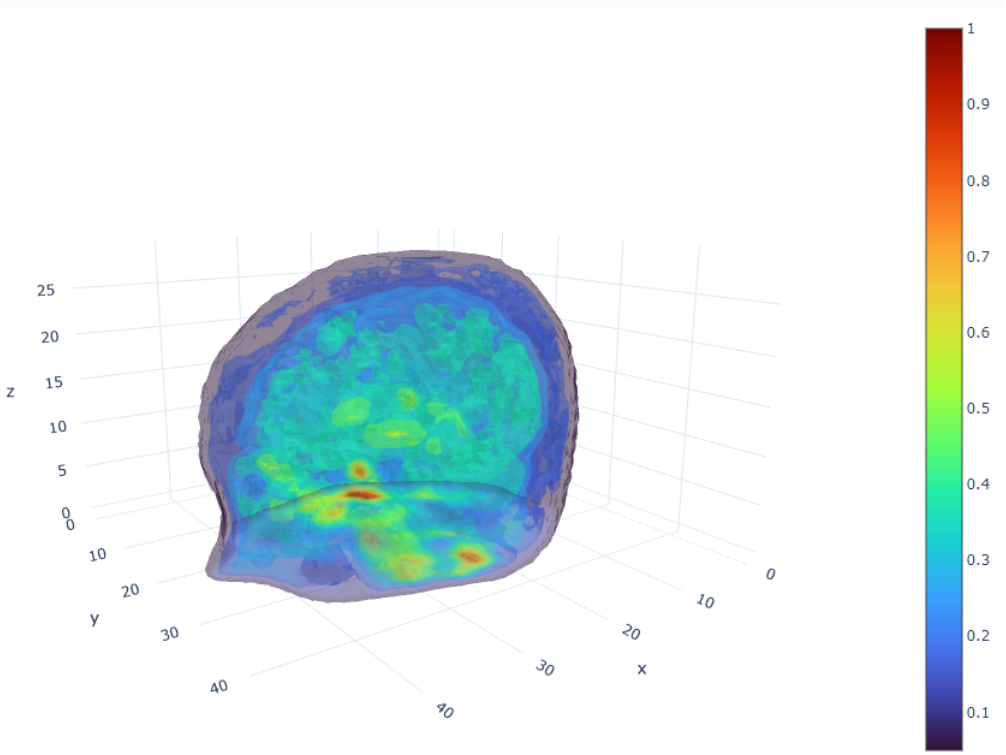


Figura 2.4: Neuroimaging in 3D

Capítulo 3

Estado del arte

3.1. Trabajos relacionados

En este capítulo vamos a hablar de los trabajos relacionados que hay publicados actualmente.

El enfoque más común de estos trabajos, se basa en redes neuronales convolutivas (CNN) aplicado a datos *MRI* (imágenes de resonancias magnéticas).[29, 42, 43, 33, 34, 35, 44] No obstante, en los últimos años están surgiendo multitud de estudios que han demostrado resultados prometedores empleando otro tipo de técnicas de neuroimagen [25, 45, 30, 44].

Por otra parte, en el contexto de los métodos de interpretabilidad encontramos que los más utilizados son los métodos basados en el análisis del flujo del gradiente en modelos como *class saliency* (CS)[46] o *guided backpropagation*[47] son algunos de los más utilizados. Por ejemplo, al investigar los mapas de activación de estos métodos, los autores de [29, 43] identificaron una reducción del volumen del hipocampo y una alteración en la región temporal y la parietal como algunas de las características más discriminatorias de los pacientes con EA.

En contraposición, los autores de[25], basándose en datos PET-FDG, encontraron que no siempre era posible identificar claramente regiones usando CS a pesar de que sí obtuvieron buenos resultados de clasificación. Sin embargo, en [44], usando *guided backpropagation*, los autores detectaron áreas como el giro cingulado y el hipocampo como las más decisivas para el diagnóstico de pacientes EA. Otros estudios aplicaron el método *layer-wise relevance propagation* (LRP) para explicar las decisiones diagnósticas del modelo [33, 34]. En [33] los autores aplicaron LRP a una CNN entrenada para discernir pacientes EA, concluyendo que las áreas más relevantes fueron consistentes con la neuropatología de la enfermedad.

De forma similar, en [34] se propusieron dos nuevas variantes de LRP, las cuales identificaron regiones típicamente relacionadas con pacientes con EA como el giro temporal medio, el giro temporal inferior o el giro hipocampal.

Asimismo, mediante el uso de técnicas de oclusión, los autores de [35] desarrollaron un modelo de diagnóstico y concluyeron que el hipocampo resulta ser una región crítica en el modelo para identificar EA.

Por último, otras investigaciones han explorado estrategias alternativas a través de métodos de interpretabilidad temporales. Por ejemplo, [37] presentó un enfoque basado en

aprendizaje por refuerzo (*Reinforcement Learning*) que analizaba la trayectoria de un agente en imágenes MRI para interpretar las predicciones del modelo.

El uso del DL en técnicas de interpretación en imágenes PET-tau es más limitado. Hasta la fecha y conocimiento de los autores, sólo existen dos artículos que combinen DL con técnicas de interpretabilidad en esta modalidad de imágenes.
Por un lado, los autores de [45] desarrollaron una CNN capaz de discriminar pacientes con EA de pacientes de control (sin Alzhéimer) con una exactitud del 90'8 %. Posteriormente, emplearon LRP para explorar las regiones de la imagen más informativas sobre las etapas prodómicas de la EA. Fueron capaces de identificar regiones directamente relacionadas con la memoria.
En segundo lugar, en [30], los autores fueron capaces de alcanzar una exactitud del 80 % en detección de daño cognitivo medio (DMC, un estado previo al desarrollo de EA) y EA. Posteriormente, aplicando la técnica de oclusión, pudieron determinar que las regiones temporales estaban más estrechamente relacionadas con el diagnóstico de DMC/EA.

En todos los trabajos mencionados, los investigadores presentaron modelos con una buena capacidad discriminatoria. Además, la evidencia sugiere que sus predicciones están basadas en regiones típicamente afectadas por el EA. Sin embargo, los autores suelen basar sus conclusiones exclusivamente de los resultados dados por uno o dos métodos sin explorar la consistencia con otras alternativas. Esto es un problema, ya que cada método tiene sus propios sesgos y conviene comparar la consistencia de varios para poder concluir (en nuestro contexto) a que partes de la imagen está prestando atención un modelo. Por este motivo, varios autores han comparado los diferentes métodos de interpretabilidad actualmente disponibles usando MRI, proporcionando información sobre las propiedades de cada técnica cuando se utiliza con este tipo de datos [38, 39, 40]. Sin embargo, en la literatura actual no existe una comparación formal aplicada a los datos PET-tau.

3.1.1. Principal referencia

Como se ha comentado en el anterior punto, Este trabajo se ha basado en el estudio publicado por Jo et al.[45] para comenzar a desarrollar nuestro modelo.

En este trabajo los autores realizaron una red neuronal convolutiva y aplicaron el algoritmo de interpretación *layer-wise relevance propagation* (LRP) para poder observar las regiones del cerebro que más relevancia tenían en el diagnóstico.

Para aplicar todo esto, utilizaron los datos de trescientos pacientes a través de ADNI, los cuales se habían realizado una PET-Tau, de donde se obtuvieron todos los datos necesarios.

La arquitectura de su CNN 3D fue la que se puede observar en la figura 3.1.

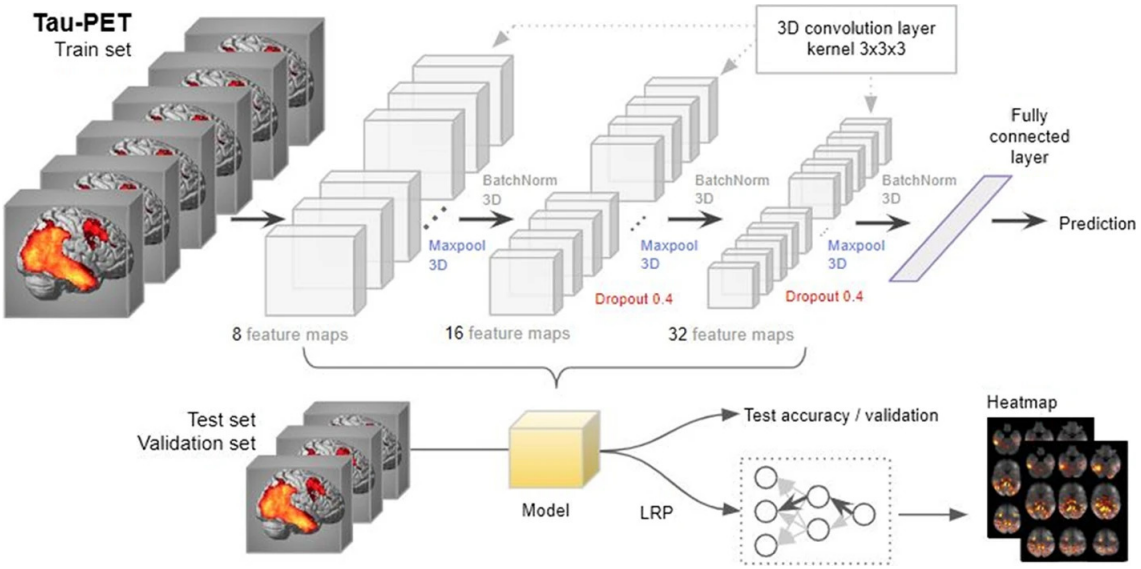


Figura 3.1: Arquitectura de la red usada en el trabajo de Jo et al. [45]

Para medir el rendimiento del clasificador, los autores utilizaron validación cruzada con cinco *folds* estratificados (es decir, con la misma proporción en cada tipo de paciente a clasificar) sobre su modelo, viendo así cómo de bien distinguía entre controles (CN) y EA.

La validación cruzada es una técnica utilizada para poder evaluar los resultados garantizando que estos son independientes de la partición de datos de entrenamiento y prueba. Esto se logra partiendo los datos en k particiones diferentes (a las que llamaremos *folds*), aplicando como conjunto de entrenamiento y validación todos los datos menos los del fold correspondiente y usando para las pruebas este. Se tratará más a fondo en qué consiste la validación cruzada en el capítulo 4.11.

En este trabajo, de entre los cuatro *folds* de entrenamiento/validación, uno fue usado para validación y los otros tres para entrenamiento. En los tres *folds* de entrenamiento,

los autores usaron la técnica *image augmentation* para poder poder entrenar con más datos. Esto fue aplicado girando los datos de las imágenes, cambiando la posición de dos vóxeles de la misma y cambiando la posición simultáneamente con el giro.
Este proceso de validación cruzada se repitió cuatro veces para que los resultados fuesen correctamente verificados y la exactitud media fue usada como exactitud final. La red fue diseñada usando Pytorch 1.0.1 y los programas fueron ejecutados con Python 3.5.

En el trabajo los autores obtuvieron los resultados mostrados en el cuadro 3.1 después de aplicar el proceso de validación cruzada mencionado anteriormente.

Como se ha mencionado, los autores del artículo aplicaron el algoritmo LRP para identificar las características informativas y visualizar los resultados de clasificación. El algoritmo LRP, tal y como ellos explican en su trabajo, determina la contribución de cada píxel de la imagen de entrada en la predicción dada por el modelo. Para más detalles sobre el LRP, los autores recomiendan leer [48], aunque nosotros también recomendamos [49], artículo que consultamos para informarnos sobre este y más algoritmos de interpretación.

Aplicando el algoritmo LRP se obtienen mapas de calor donde se indican que regiones del cerebro han tenido mayor relevancia de cara a clasificar las imágenes. En estos mapas de calor se puede observar la relevancia de cada zona marcada con valores crecientes de rojo a amarillo.
Los autores aplicaron LRP sobre los modelos entrenados con mayor precisión de cada fold, generando así cinco mapas de calor y generando un top 10 de las regiones con mayor contribución. Los autores destacan el hipocampo, el giro parahipocampal, el tálamo, el giro fusiforme y el diencéfalo de entre las regiones coloreadas.
Para poder comparar los resultados que obtuvieron con LRP, los autores analizaron los vóxeles de todo el cerebro con SPM12[50] para identificar las regiones del cerebro con las diferencias más significativas entre EA y CN en la disposición tau cerebral.
Los autores también calcularon las puntuaciones de probabilidad de EA en participantes DMC usando el modelo de clasificación con el que han trabajado todo lo anterior, consiguiendo buenos resultados.

	Conjunto de entrenamiento (Entrenamiento)	Conjunto de entrenamiento (Validación)	Prueba	Exactitud r1	Época	Exactitud r2	Época	Exactitud r3	Época	Exactitud r4	Época	Exactitud media	Desviación Típica
fold1	78 (312)	28	26	85.7	23	92.9	24	89.3	21	85.7	27	88.4	3.0
fold2	78 (312)	28	26	96.2	20	96.2	21	92.3	21	88.5	22	93.3	3.2
fold3	80 (320)	26	26	100	35	92.3	28	88.5	27	88.5	30	92.3	4.7
fold4	80 (320)	26	26	92.3	24	88.5	31	88.5	38	88.5	29	89.4	1.7
fold5	80 (320)	26	26	92.3	50	80.8	36	96.2	35	92.3	34	90.4	5.8

Cuadro 3.1: Resultados obtenidos tras la validación cruzada en el artículo de Jo et al. [45]

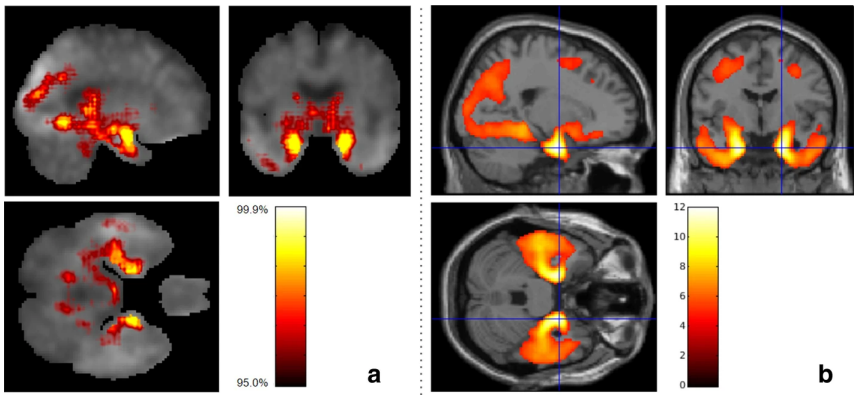


Figura 3.2: Mapas de calor generados con LRP (a) comparados con mapas de diferencia de grupo en vóxeles entre EA y CN (b) en el artículo de Jo et al. [45].

Finalmente, los autores concluyeron que el DL se puede utilizar imágenes PET-tau de pacientes EA con pacientes de control. Nosotros nos vamos a centrar en comprar formalmente distintos algoritmos de interpretación en este tipo de imágenes para poder observar cuál de ellos es mejor.

Capítulo 4

Metodología

4.1. Plan de trabajo

Para poder llevar a cabo el proyecto y conseguir los objetivos propuestos, el trabajo se estructuró como se puede observar en la figura 4.1

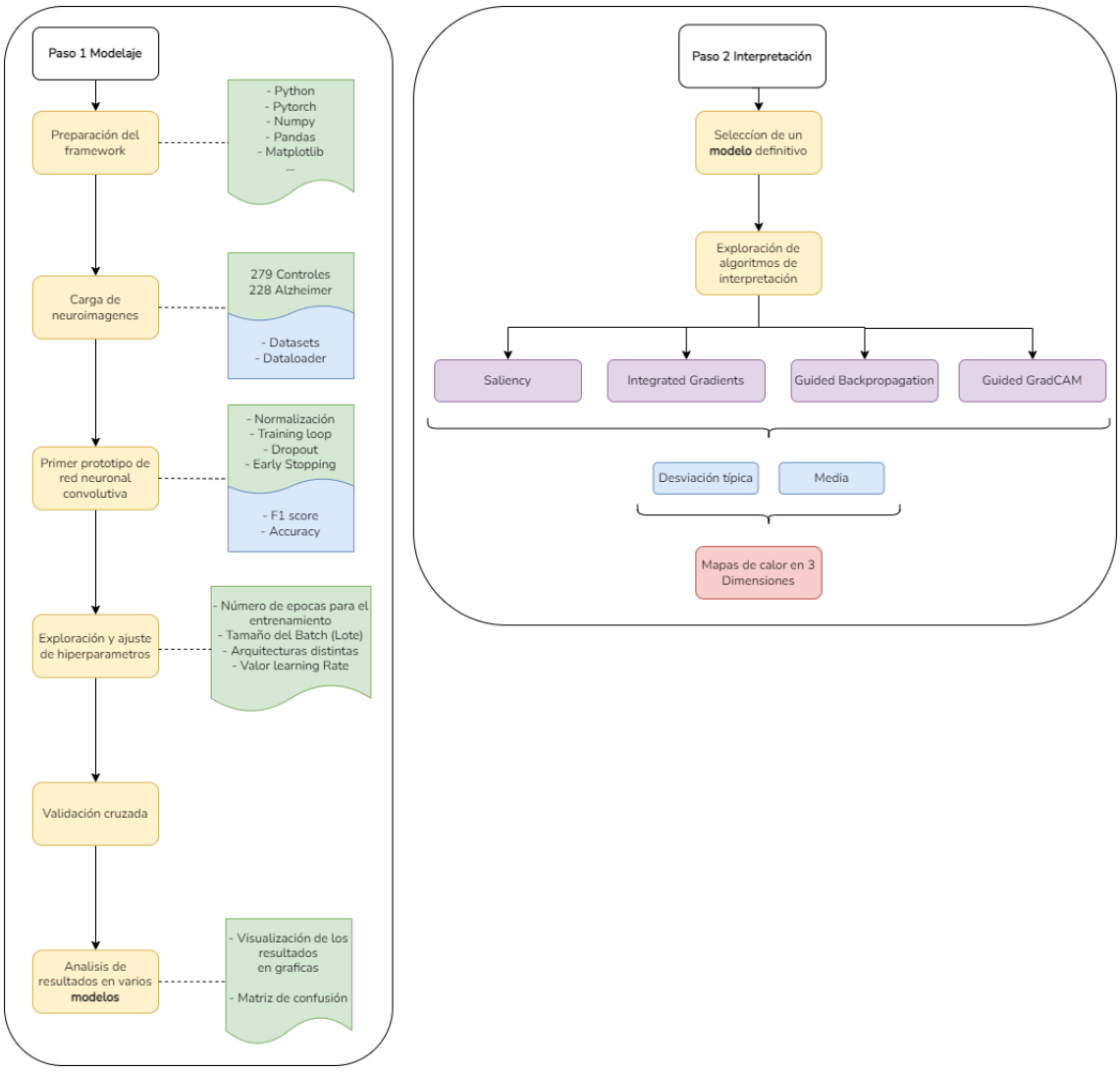


Figura 4.1: Plan de trabajo

En este capítulo entraremos a analizar cada una de las fases estructuradas en dicha figura.

4.2. Frameworks y herramientas

En esta sección vamos a hablar sobre los distintos *frameworks* y herramientas que hemos utilizado a lo largo del proyecto.

4.2.1. Anaconda Navigator y Jupyter Notebook

Anaconda Navigator es una interfaz gráfica de usuario (GUI) para escritorio que permite gestionar paquetes de *Conda*.
De todos estos paquetes, hemos utilizado *Jupyter Notebook* (que viene instalado por defecto en *Anaconda Navigator*).

Jupyter Notebook es un entorno informático interactivo basado en la web cuya funcionalidad es la de crear y modificar documentos de *Jupyter notebook* (.ipynb). Este tipo de archivos nos ha permitido programar en distintas celdas las cuales se pueden ejecutar independientemente unas de otras pero nos permite mantener datos como las variables o las definiciones entre ellas. En las celdas hemos escrito *Python* (lenguaje del que se hablará en el capítulo 4.2.2) y estos archivos han sido con los que hemos desarrollado toda la programación del proyecto.

4.2.2. Python

Python es el lenguaje de programación con el que se ha desarrollado este proyecto. Es un lenguaje de alto nivel, de código abierto, interpretado, multiparadigma y multiplataforma.
Es un lenguaje de código abierto y multiplataforma que destaca sobre todo por la gran cantidad de bibliotecas que se han desarrollado en este lenguaje para facilitar el trabajo en inteligencia artificial, *big data*, ML y DL entre otros...
En este proyecto hemos utilizado este lenguaje enfocado al campo del DL[17]. Estos métodos enfocados a la clasificación de imágenes a partir de el diseño de una red convolucional se apoyan en el uso de las bibliotecas *Pytorch* y *Tensorflow*, Existe un gran debate en la dicotomía de cual elegir de estas dos para desarrollar tu diseño[51]. En nuestro caso hemos utilizado *Pytorch* a lo largo de todo el proyecto.
Profundizaremos sobre cada una de las librerías de *Python* que hemos utilizado en su

correspondiente sección.

Pytorch

Pytorch[52] es una biblioteca de *Python* de aprendizaje automático basado en *Torch* que permite la creación de redes neuronales de DL.
Con el paquete ***Autograd*** implementa la diferenciación automática. Su funcionamiento está basado en grafos computacionales dónde cada operación es representada como un nodo y cada nodo almacena información sobre el gradiente de la función.
Para computar los gradientes:
- Se construye el grafo computacional que sigue las operaciones del flujo hacia delante de la red neuronal.
- Las operaciones se transforman en sumas y multiplicaciones para calcular sus derivadas.
- Por ultimo se utiliza el algoritmo de ***backpropagation*** calculando los gradientes respecto a los parámetros.
Este método es muy interesante, ya que ahorra tiempo en una época en la que el cálculo de la diferenciación de parámetros es mediante un paso adelante.
Autograd permite diferenciar automáticamente y calcular gradientes fácilmente.

Pytorch tiene la clase ***tensor*** (*torch.Tensor*), la cual funciona de forma parecida a un vector pero también permite operarse mediante GPUs compatibles con *CUDA* (Nvidia).

El paquete ***optim*** (*torch.optim*) implementa varios algoritmos de optimización utilizados en la creación de redes neuronales, primero se construye un objeto optimizador que se va actualizando con los parámetros en función de los gradientes operados.

Numpy

Numpy[53] es una biblioteca de *Python* con la cual se permite la creación de vectores y matrices multidimensionales de gran tamaño. También aporta funciones matemáticas de alto nivel y facilita el manejo de y la modificación de vectores multidimensionales.

Pandas

Pandas[54] es un paquete de *Python* que proporciona estructuras de datos diseñados de forma relacional, A su vez incluye funciones para la manipulación de esas estructuras. Ofrece herramientas de reordenación y permite la lectura y escritura de los archivos csv y de bases de datos: de una dimensión (Series), de dos dimensiones (*Dataframe*) y de tres (*WidePanel*, funciona con 3 índices: *items*, *major axis* y *minor axis*).

Matplotlib

Matplotlib[55] es una biblioteca de *Python* enfocada a la visualización gráfica de datos en 2D, computa listas, *arrays* apoyado por la librería *numpy* se completa su funcionalidad para sacar representaciones de funciones matemáticas relacionadas con los datos computados.

Scikit-learn

Sklearn[56] es una biblioteca de *Python* de aprendizaje automático para el uso de algoritmos de clasificación regresión y análisis de grupos, Dentro de esta biblioteca hemos utilizado el paquete *Sklearn.metrics* para conseguir los valores de las métricas *accuracy* y valor F1.

Captum

Captum es una biblioteca de *Python* enfocada a la interpretación de modelos de *Pytorch*, proporcionando diversos algoritmos para ayudar a entender la salida de un modelo de inteligencia artificial. En nuestro caso hemos utilizado algoritmos de interpretación para probar su capacidad de análisis con imágenes 3D. En concreto usaremos los algoritmos *Integrated Gradients*, *Saliency Maps*, *Guided Backpropagation* y *Guided GradCAM*.

4.2.3. LaTeX y Overleaf

LaTeX es un sistema de composición de textos orientado a la creación de documentos de alta calidad tipográfica y que está orientado a la creación de artículos, libros y documentos científicos. Su principal ventaja con respecto a otros editores es que nos permite

separar el formato y el contenido del documento de forma clara, además de proporcionar una gran ayuda con el índice y la bibliografía.
LaTeX ha sido utilizado para poder escribir la memoria del trabajo.

Para poder realizar esta tarea de escritura de memoria se usó Overleaf, un editor colaborativo basado en la nube de documentos en LaTeX que permite la compilación automática del código para observar los resultados de manera simultánea.

4.2.4. Github

Github es una plataforma de desarrollo colaborativo para alojar proyectos y controlar las versiones de los mismos usando el sistema de control de versiones *Git*.
Github ha sido utilizado a lo largo de todo el proyecto para poder trabajar de forma paralela y llevar un control de versiones adecuado.

4.3. Procesamiento de imágenes

Los datos fueron descargados del repositorio *Alzheimer's Disease Neuroimaging Initiative* (ADNI) [57]. ADNI es una base de datos pública lanzada en el año 2003 con el objetivo de investigar cómo diferentes biomarcadores pueden ser combinados para la medición de la progresión del deterioro cognitivo leve (DCL) y el Alzheimer (EA). De entre los diferentes biomarcadores recogidos en ADNI, hemos utilizado imágenes cerebrales de Tomografía por Emisión de Positrones (PET), en combinación con el radiotrazador [^{18}F] flortaucipir.

Se descargaron un total de 1321 imágenes pertenecientes a 1321 pacientes. Las imágenes fueron preprocesadas siguiendo el protocolo número 3 descrito en ADNI. Adicionalmente se aplicó un suavizado con un *full width at half máximum* (FWHM) de tamaño [8, 8, 8]. Por otro lado, de cara a disminuir el coste computacional en el desarrollo de los modelos, el tamaño de los vóxeles fue reescalado a 2.5mm^3.

En nuestro caso las imágenes a estudiar serían tomografías de cerebros en **tres dimensiones**, concretamente **[97, 97, 59]**, estos índices son el ancho, fondo y alto respectivamente. Este objeto tridimensional se compone de múltiples **vóxeles** (píxeles volumétricos) y es la unidad mínima procesable de una matriz tridimensional y es, por tanto el equivalente del píxel en una imagen de dos dimensiones. Cada vóxel contiene un valor diferente haciendo referencia a la importancia de cada uno, los vóxeles mas relevantes tendrían valores mas altos y los que no tengan ninguna importancia contendrían un cero.

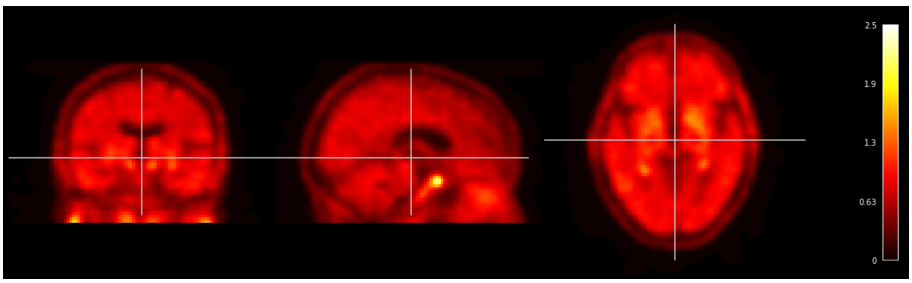


Figura 4.2: Una de las neuroimágenes utilizadas.

Del total de imágenes descargadas se seleccionaron 507, dónde el 44.97 % correspondían a pacientes con EA, y el 55.03 % restante a pacientes de control. Los pacientes de control se definieron a partir de los casos diagnosticados como sanos con unos valores de *standardized uptake ratio* (SUVR) para el biomarcador 18F-AV45 (SUVR-AV45) inferiores a 1.11 [58].

El grupo de pacientes con EA se compuso a partir de las imágenes con un diagnóstico de DMC o EA, y unos valores de SUVR-AV45 superiores o iguales a 1.11. El resto de los pacientes que no entraban dentro de los grupos CN, DMC o EA para los puntos de corte definidos, fueron descartados bajo la asunción de posibles dudas en su diagnóstico.
Estas imágenes están en formato **NIFTI (.nii)**, un formato muy usado para datos de neuroimagen multidimensional.

Con el uso de este formato de imágenes y a través de un *script* recogemos la información de cada paciente iterando a través de un *dataframe* de la librería Pandas, junto al UID (identificador) de cada paciente, para poder luego emparejarlo con su imagen correctamente. Obtuvimos así los UIDs de los pacientes y su diagnóstico, pero nos quedaba asociarlos a sus respectivas imágenes.

Para lograr esto, creamos otro *script* que iteraba entre los distintos UIDs obtenidos para cargar sus respectivas imágenes. Las imágenes están todas en un mismo directorio y se llaman "UID.nii". Para cada imagen, se creó una estructura llamada *nifticontainer* (perteneciente a la librería *Miitools* [59]), que guarda sus respectivos datos junto a un parámetro que contiene un diccionario con el UID y el diagnóstico de la imagen.

4.4. Convoluciones, Redes neuronales convolutivas

Una de las grandes funcionalidades de las redes neuronales son las **Redes neuronales convolutivas** (*Convolutional neural networks*, CNN). Las redes neuronales convolucionales son una clase de redes multicapa *feedforward* diseñadas para el reconocimiento y clasificación de imágenes. Las CNNs usan estas dos herramientas para detectar aquello que se quiere estudiar: caras, objetos o, en nuestro caso, enfermedades neurodegenerativas. Las redes están parametrizadas mediante dos vectores, un vector de pesos y un vector de sesgos, que son ajustados durante el aprendizaje. Cada neurona esta conectada a diversas entradas y después toma una suma ponderada sobre ellas, después la transmitirá a través de una **función de activación** y responde con una salida.

El esquema de la figura 4.3 muestra la arquitectura matemática para representar una red neuronal convolutiva de tres dimensiones **[7, 7, 3]**. En este caso observaríamos la estructura de una imagen de dos dimensiones con tres capas de colores (RGB) que en las CNNs son interpretadas como canales.

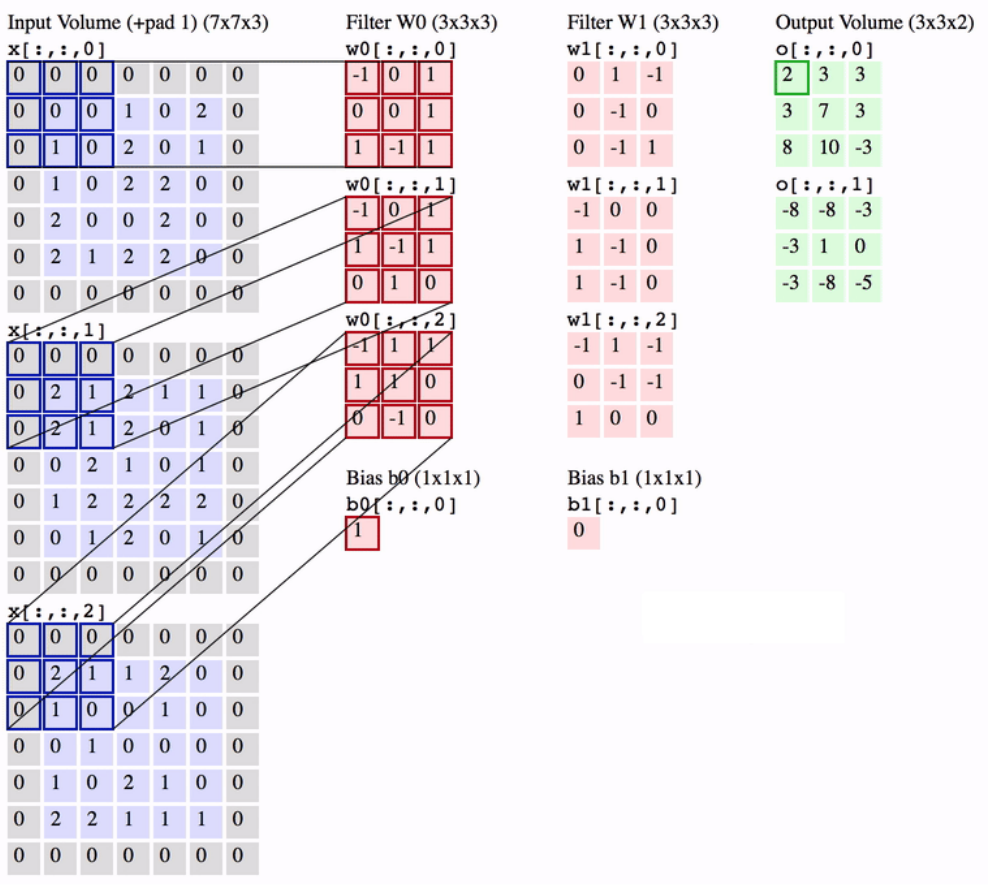


Figura 4.3: Arquitectura matemática de una operación de convolución 7x7x3 [60]

La **primera capa** en una CNN es la **capa convolutiva** 4.4, que es la base de este método y realiza la mayoría del trabajo computacional. Los datos son convolucionados usando *filtros o núcleos*, los filtros son pequeñas unidades que aplicamos a lo largo de los datos mediante una ventana deslizante. El submuestreo reduce la dimensión de la matriz de entrada dividiéndola en subregiones y permitiendo generalizar las características.

En las redes neuronales, la función de activación es responsable de transformar la entrada ponderada sumada del nodo, a la activación del nodo o salida para esa entrada. La **función de activación lineal rectificada**, **ReLu** para abreviar, es una función lineal por partes que generará la entrada directamente si tiene un valor positivo, de lo contrario generará un cero. Se ha convertido en la función de activación predeterminada para muchos tipos de redes neuronales porque un modelo que la usa es más fácil de entrenar y, a menudo, logra un mejor rendimiento [61].

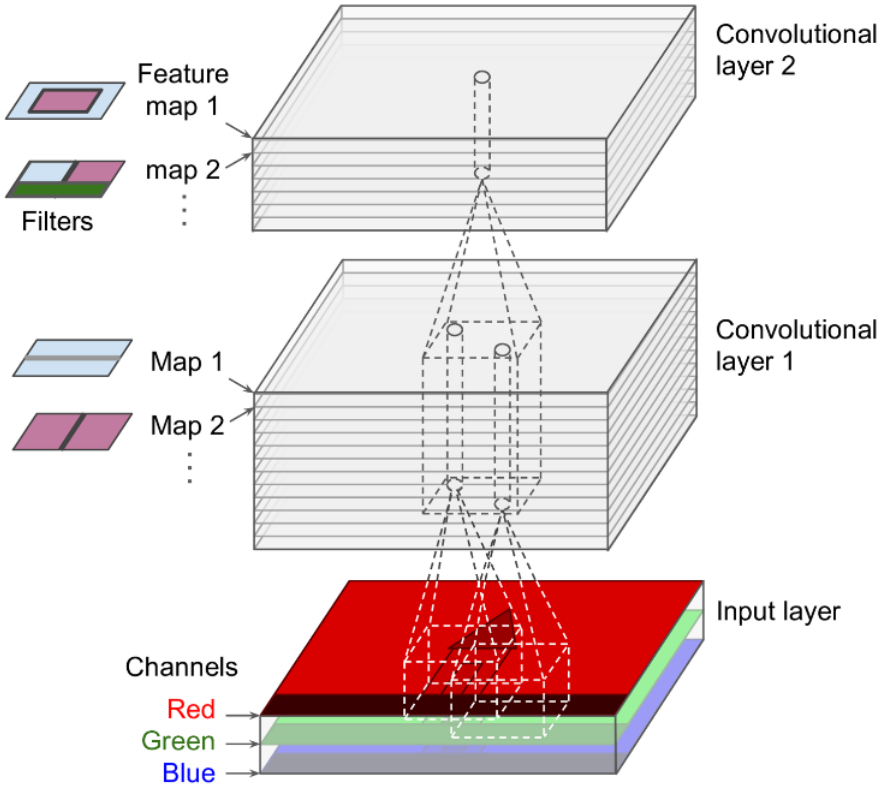


Figura 4.4: Arquitectura gráfica de capas convolutivas [62]

Otro elemento importante dentro del funcionamiento de las CNNs son las **capas de agrupación** (*pooling layers*), su objetivo es agrupar o encoger la entrada con el propósito de reducir la carga computacional, el uso de memoria y el número de parámetros. La capa

de agrupación mas usada es la denominada *max pooling layer*, su funcionamiento es el siguiente: solo el valor de entrada máximo en cada campo receptivo llega a la siguiente capa, mientras que las otras entradas se eliminan: 4.5.

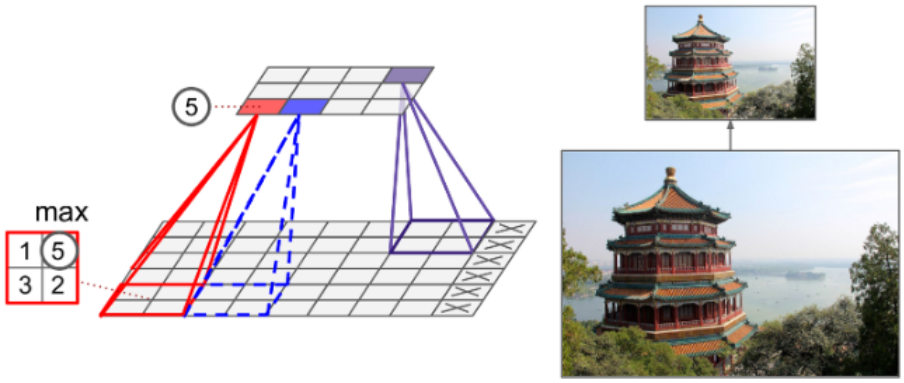


Figura 4.5: Capa de agrupación tipo *max pooling layer* [62]

Por último encontramos la capa totalmente conectada (*fully connected layer*), esta capa implica transformar toda la matriz del mapa de características (*feature map*) agrupada en una sola columna que luego se inyectará a la red neuronal para su procesamiento. En la figura 4.6 podemos ver estas últimas capas en la parte derecha de la imagen.

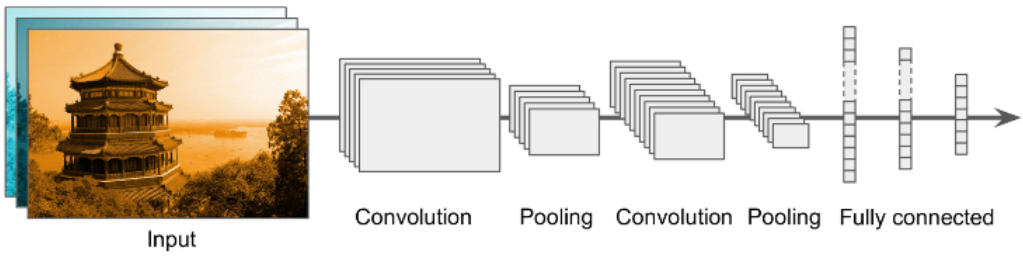


Figura 4.6: Arquitectura típica de una red neuronal convolutiva [62]

4.5. Desarrollo de un modelo

En esta sección se detallarán las diferentes técnicas y características que hemos escogido implementar.

Para optimizar los resultados de una red convolutiva, se tienen en cuenta diferentes hiperparámetros de entrenamiento como podrían ser el tamaño de *batch*, el optimizador o la función de perdida, los cuales deben ser los que garanticen los mejores resultados. A su vez, la red está compuesta de distintas capas (como hemos podido ver en el capítulo 4.4) y a parte necesitará tener los datos en un formato específico, en concreto habrá que crear una estructura de *Pytorch* llamada *Dataset*, la cual detallaremos más a fondo en su correspondiente sección.

4.5.1. Implementación de un Dataset

El *dataset* es una parte indispensable del aprendizaje automático, la información contenida en él (imágenes, datos, etc) será el sujeto de estudio para nuestro modelo, en nuestro caso el repositorio Alzheimer's Disease Neuroimaging Initiative (ADNI) [57] nos facilita la base para completar este componente. Este será usado para entrenar a nuestro *modelo* con el objetivo de encontrar patrones predecibles dentro de todo el *dataset*.

Nuestro *dataset* fue creado pasando como parámetro una lista de los *nifticontainers* a utilizar, que conseguimos coger tal y como comentamos en el capítulo 4.3.

Para poder utilizar el *dataset* con esta estructura, lo redefinimos para poder adaptarlo, quedando de la siguiente forma:

```
class Dataset(Dataset):
    def __init__(self, patients: list, label_mapping: dict = {'AD': 1, 'CN': 0}, transform=None):
        self._patients = patients
        self._label_mapping = label_mapping
        self.transform = transform

    def __len__(self):
        return len(self._patients)
```

```
    def __getitem__(self, idx: int):
        patient = self._patients[idx]
        image = torch.Tensor(patient.data)

        image = image[None, :]

        label = torch.tensor(self._label_mapping[patient.params['diagnosis']])

        if self.transform:
            image = self.transform(image)

        return image, label
```

Dataloaders, Conjuntos de entrenamiento, validación y testing

Un detalle importante es el uso que se le da a la información contenida en el *dataset* completo. Esta no será usada únicamente para el entrenamiento del algoritmo, en cambio sera dividida en tres subconjuntos que serán necesarios para comprobar como de preciso es nuestro modelo.

- El **conjunto de entrenamiento** es literalmente la **información** que usaremos **para entrenar** al modelo. Este observa y aprende de estos datos.
- El **conjunto de validación** es una muestra de datos directamente separado del conjunto de entrenamiento del modelo, que se utiliza para dar una estimación de la habilidad, precisión o experiencia del algoritmo **mientras se ajustan sus hiperparámetros**. La evaluación se vuelve mas sesgada a medida que la habilidad en el conjunto de datos de velicación se incorpora a la configuración del modelo.
- El **conjunto de prueba o test** se usa para aportar una estimación imparcial de la habilidad del **modelo ajustado final** al comparar o seleccionar entre modelos finales. Este conjunto debería tratar de contener datos que cubran todos los posibles casos que el modelo vaya a afrontar, como si fuera a ser usado en el mundo real.

En nuestro proyecto usamos una división de los datos en los que **el 70 % son de entrenamiento, el 10 % son de validación y el 20 % de prueba**.

Ahora bien, cuando estamos entrenando nuestro modelo, normalmente querremos usar distintos lotes (cada una de las divisiones que se forman con el *batch-size*, esto se explicará en mayor profundidad en el capítulo 5), barajar los datos cada época para reducir el entrenamiento y usar multiproceso para hacer más rápido el entrenamiento. Todo esto se logra usando una estructura de *Python* llamada *Dataloader*. [63]

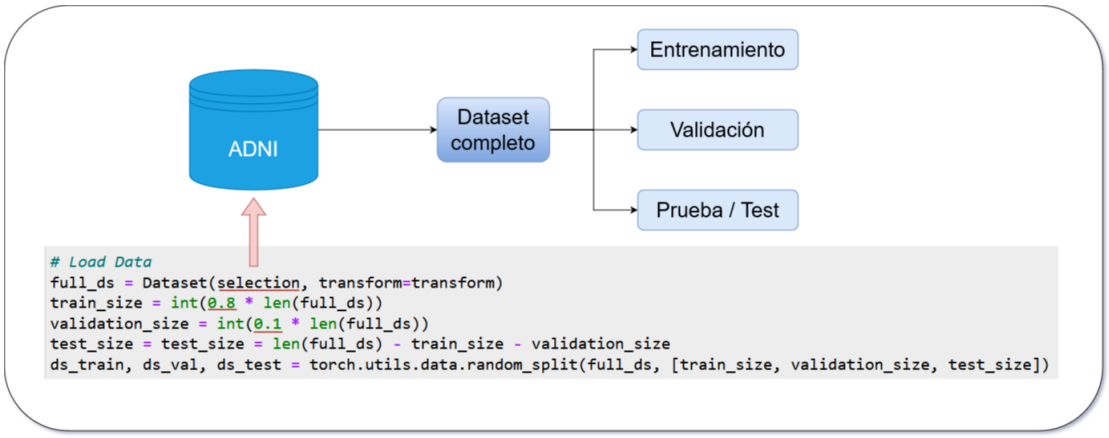


Figura 4.7: División del *dataset*

4.5.2. Normalización

Para hacer que todos los atributos tengan la misma escala se realiza mediante normalización. La normalización de imágenes es una buena práctica cuando trabajamos con redes neuronales de DL. Normalizar las imágenes significa transformar las imágenes en valores tales que la media y la desviación estándar de la imagen se conviertan en 0 y 1 respectivamente. Para hacer esto, primero se resta la media de cada entrada y luego el resultado se divide por la desviación estándar. Las estadísticas de esta transformación son computadas sobre los datos de entrenamiento y posteriormente se utilizan para normalizar los datos de validación y test.[62].

La normalización reduce la asimetría, por tanto favorece la rapidez del aprendizaje, mejora la resolución de problemas sobre los gradientes decrecientes y explosivos. Para normalizar la imagen del tensor aplicamos la transformación *torchvision.transforms.Normalize()*.

---

torch_dataset = Dataset(selection)

imgs = torch.stack([img_t for img_t, _ in ds_train], dim=1)

```
imgs.view(1, -1).mean(dim=1) # media
0.3136

imgs.view(1, -1).std(dim=1) #desviación estandar
0.4671

transforms.Normalize((0.3136), (0.4671))
```

4.5.3. Entrenamiento, el descenso por gradiente

El descenso por gradiente (*Gradient descent*) es un algoritmo genérico capaz de encontrar soluciones óptimas para un amplio rango de problemas. La mecánica que reproduce es **minimizar** la función de pérdida:

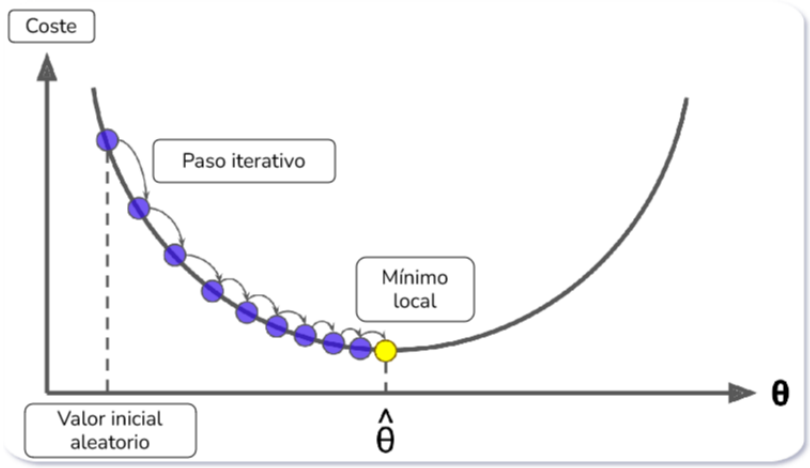


Figura 4.8: Ilustración sobre la mecánica del descenso por gradiente. [62]

El algoritmo se implementa siguiendo un bucle, el *training loop* (donde siempre que se usa una semilla de aleatoriedad, se usa una fijada para que los resultados puedan ser replicables), que se ejecutará n veces, siendo n el número de épocas 4.7, siempre y cuando el *early stopping* (método de regularización para prevenir el sobreajuste del modelo. Para más información sobre este método ver el capítulo 4.5.6.) no interrumpa la ejecución de este. Para una mejora en el tiempo de ejecución siempre que *CUDA* este disponible haremos uso de esta para usar computación por **GPU**:

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

CUDA está disponible para las tarjetas gráficas desarrolladas por Nvidia. Esto puede dar problemas si dispones de tarjetas fabricadas por AMD u otras marcas.

El último paso para perfeccionar el entrenamiento sería escoger validación cruzada de *k-folds*, detallada en la sección 4.6.

4.5.4. Dropout

Este método es una de las técnicas de regularización mas populares en redes neuronales. Fue propuesto en un articulo [64] por Geoffrey Hinton en 2012 aportando a las mejores arquitecturas de la época una mejora en la exactitud 1 % a 2 %. Esta cifra puede parecer poca cosa, sin embargo un modelo que obtiene una mejora del 2 % con una exactitud del 95 % reducirá su tasa de error en un 40 %.

El algoritmo que implementa esta funcionalidad es sencillo, en cada paso del entrenamiento cada neurona *(Incluyendo las neuronas de entrada, pero siempre excluyendo las de salida)* tiene una probabilidad p de ***quedar desactivada*** temporalmente, de tal forma que no contribuye a la salida final de la red, en nuestro proyecto queda representado como:

```
out = F.dropout(out, p=0.45)
```

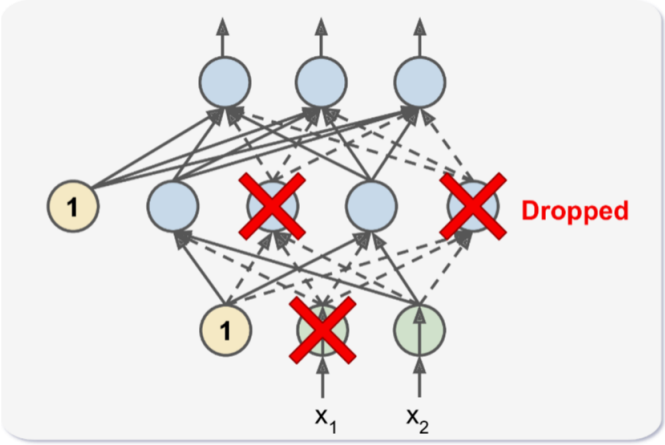


Figura 4.9: Funcionalidad del método *dropout* [62]

4.5.5. Función de pérdida

La función de pérdida (*loss function*) se utiliza para comparar los resultados obtenidos por el modelo con los resultados deseados (es decir, lo que realmente son las imágenes). El trabajo que realiza concretamente es el de computar un solo valor numérico que el proceso de aprendizaje debe tratar de minimizar. Dicho valor es más grande en función de como de grande sea el error y más pequeño cuanto menor sea este, por lo que lo ideal será que este valor sea lo menor posible (y por lo tanto, más cercano a un resultado perfecto.[65]

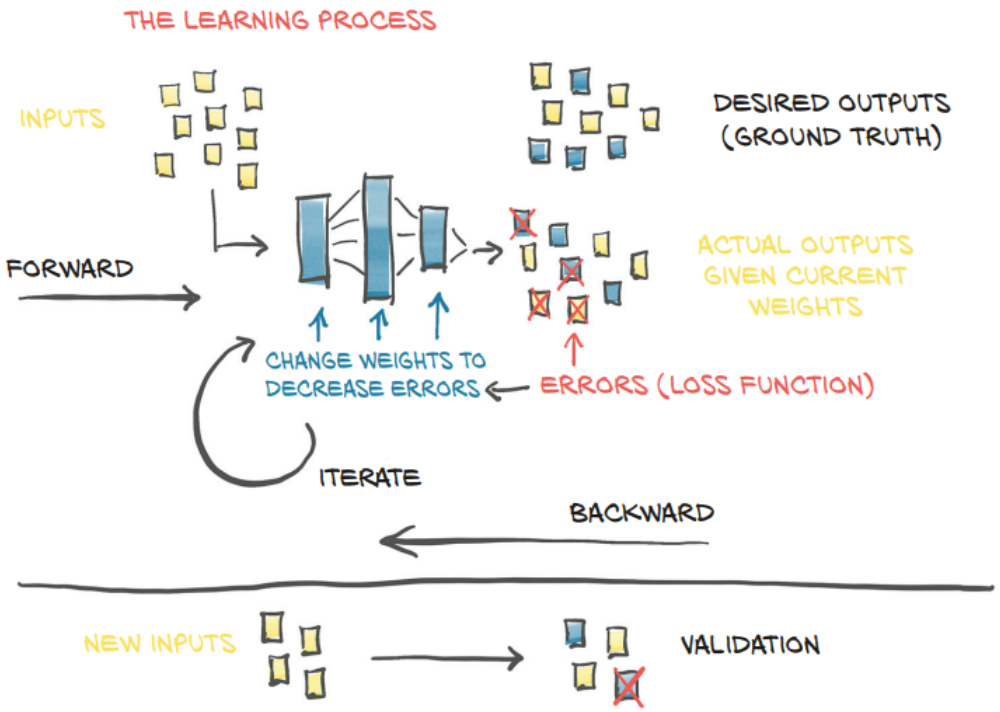


Figura 4.10: Modelo de un proceso de aprendizaje. En él podemos apreciar como actúa la función de pérdida. [65]

Existen muchas funciones de pérdida diferentes (BCE, MSE, MAE, Cross Entropy...), cada una mejor para un tipo de problema concreto.

En nuestro caso, se ha usado BCE (*Binary Cross Entropy*) al tratarse de un problema de clasificación binario (ya que esta función está diseñada para este tipo de problemas). Esta función crea un criterio para calcular la entropía cruzada binaria entre los objetivo y las probabilidades de entrada.

El valor de pérdida (siempre y cuando no se modifiquen los parámetros utilizados al invocar a la función de *Pytorch*) es calculado de la siguiente manera:

$l(x, y) = mean(L) = (l_1, ..., l_N)^T, l_n = -w_n[y_n \cdot logx_n + (1 - y_n) \cdot log(1 - x_n)]$

donde N es el batch size.

Para utilizar esta función se debe tener en cuenta que los objetivos (los diagnósticos de las imágenes) deben ser 0 o 1.[66]

4.5.6. Early Stopping

Una vía diferente de controlar el proceso de aprendizaje iterativo de un algoritmo como el descenso por gradientes es parar de entrenar cuando el error de validación (valor devuelto por la función de pérdida sobre los datos de validación) alcanza un cierto valor. Nosotros optamos por evaluarlo siguiendo la media de las últimas 10 iteraciones, es decir, si el valor de la media de las últimas 10 iteraciones era mejor que el obtenido en la iteración actual considerábamos que el algoritmo estaba sobreajustándose a los datos, por tanto parábamos la ejecución del entrenamiento en ese punto. En la gráfica 4.12 se aprecia cuándo uno de los *folds* de la ejecución deja de mejorar según la media de las últimas 10 ejecuciones y por tanto es interrumpida.

4.6. Validación cruzada

El proceso de validación cruzada consiste en dividir los datos en n (en nuestro caso cinco) subconjuntos llamados *folds*, entrenando así n redes distintas, donde en cada iteración uno de los *folds* se aparta para usar como datos de test, mientras que el resto se dividen para ser usados como entrenamiento y validación (nosotros usamos una proporción 70 % entrenamiento, 10 % validación y 20 % prueba). El conjunto de validación se utilizará para evitar que el modelo se sobre-entrene, aplicando la técnica *early stopping* (Para más detalles sobre esta técnica, consultar el capítulo 4.5.6), estando el modelo en modo evaluación (es decir, sin 'aprender' al clasificar estos datos). Con esos datos de entrenamiento, se calcula la media y la desviación estándar que se usará como normalización en dicho *fold*. Para cada una de las iteraciones, entrenamos un modelo con sus correspondientes datos de entrenamiento y usamos para validar sus datos de validación. Cuando la red acaba de entrenar, se testea usando los datos de test (los asignados al *fold* que toque, tal y como se comentaba anteriormente) y guardamos las estadísticas de cada uno de estos modelos (exactitud, valor-F1, matriz de confusión...), además de guardar el modelo entrenado. Cabe destacar que los datos que usamos están siempre estratificados, lo que implica que los datos de cada *fold* tienen la misma proporción de pacientes enfermos y sanos, para que no haya más datos EA que CN en ninguno de los casos y la semilla de aleatoriedad (al igual que en el *training loop*) queda siempre fijada para que los resultados puedan ser replicables.

En la validación cruzada lo ideal sería también hacer un ajuste de hiper-parámetros para el modelo, en lugar de utilizar siempre los mismos, pero por simplicidad se adaptó de esta manera.
La figura 4.11 define todo este procedimiento de forma visual.

La validación cruzada sirve para evaluar los resultados obtenidos y poder garantizar que estos son independientes de la partición que se realice en los datos, siendo usada en nuestro caso en los modelos que decidimos como mejores de los probados anteriormente, tal y como se explica en el capítulo 4.7.

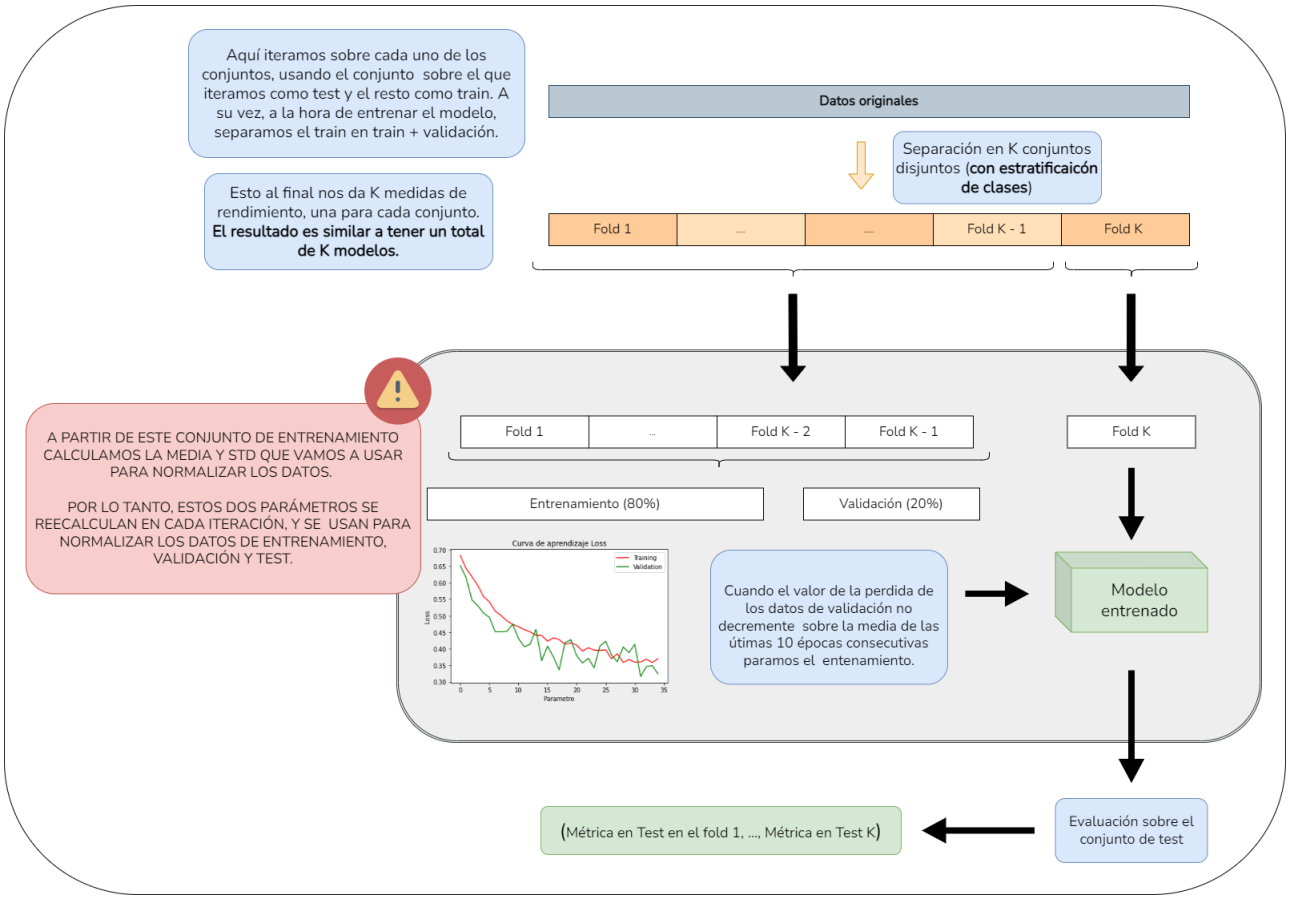


Figura 4.11: Esquema de la validación cruzada implementada en nuestro proyecto.

4.7. Evaluación de los modelos

En esta sección presentaremos los resultados obtenidos con los diferentes modelos, cada uno con cambios parciales en sus parámetros para averiguar que diferencias hacen a una arquitectura mas precisa que otra en nuestro proyecto.

Las métricas usadas para evaluar las diferencias de cada modelo son las siguientes:

- **Gráficas** representando la **curva de aprendizaje** tanto de la exactitud como de la pérdida. Las curvas de aprendizaje reflejan el rendimiento del modelo sobre la experiencia o el tiempo. El modelo se evalúa tanto en conjunto de datos de entrenamiento como en el conjunto de validación. Estas gráficas son especialmente útiles para diagnosticar problemas con el aprendizaje como por ejemplo el sobreajuste.

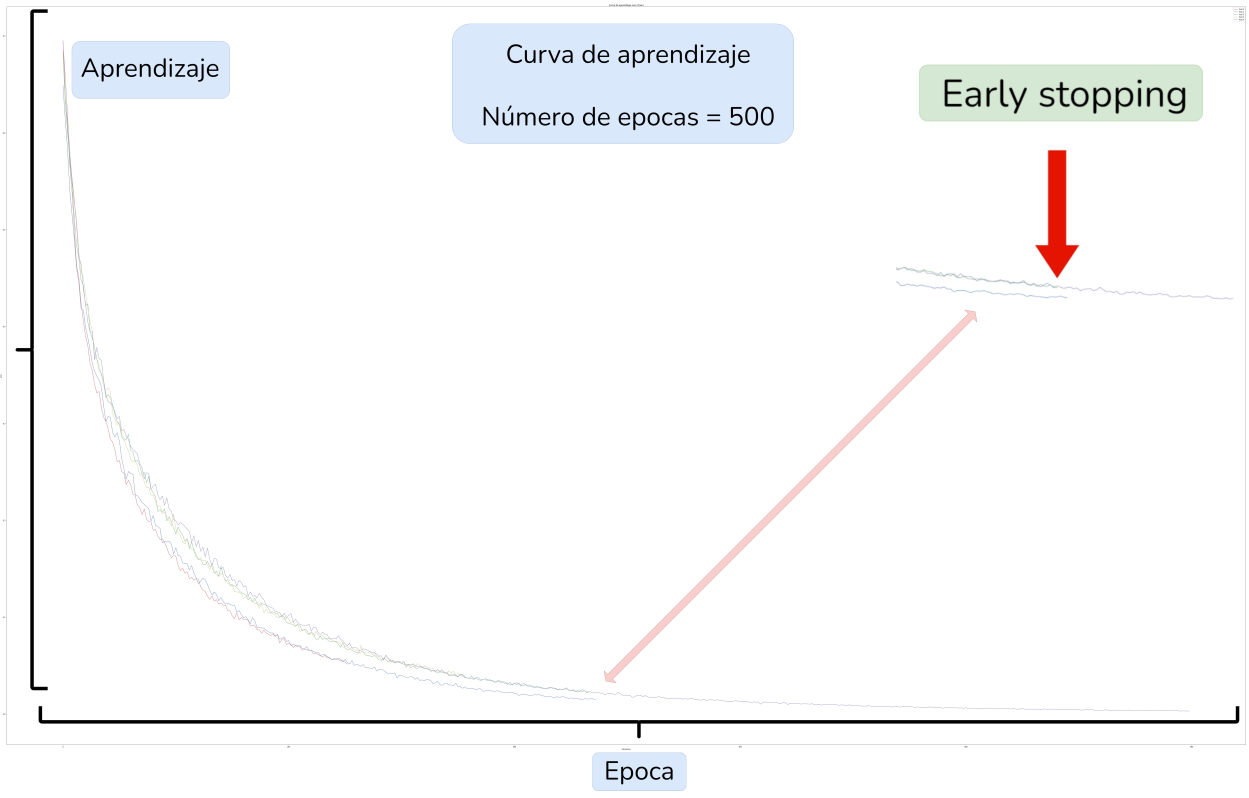


Figura 4.12: Esquema de una curva de aprendizaje

- **Matriz de confusión** reflejan el porcentaje de diagnósticos acertados para los pacientes con EA y los de control, también muestran el porcentaje de diagnósticos mal pronosticados, es decir, representan todas las posibles salidas del modelo:

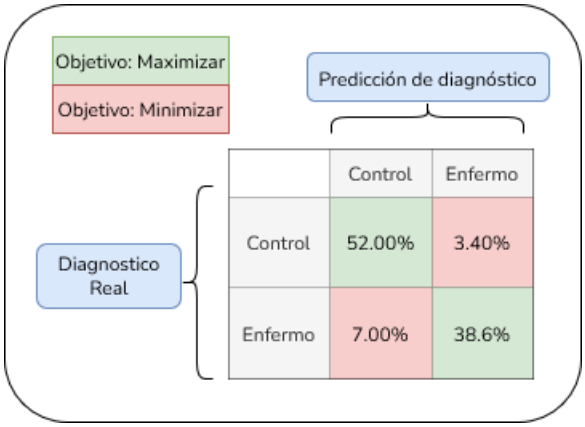


Figura 4.13: Esquema de una matriz de confusión

- **Valores de exactitud y Valor-F** para cada *fold* y la cantidad general del modelo.
- **Arquitectura y parámetros** particulares de cada uno de los modelos, en ellos variaran:
  - **Número de épocas** (*epochs*) con las que se entrenará el modelo.
  - **Algoritmo de Optimización** El proceso de aprendizaje se resume en una idea: la minimización de la función de pérdida mediante la actualización de los parámetros, w, de la red. Convirtiéndolo en un problema de optimización.

    Adam o *Adaptative Moment Estimation* [67] es un algoritmo de optimización que permite modificar el ritmo de aprendizaje o *learning rate* a medida que se desarrolla el entrenamiento. En concreto se trata de una variante del descenso de gradiente estocástico o *stochastic gradient descent* 4.5.3 [68].

$$w_a = w_j - \alpha \frac{\partial f(w_j)}{\partial w_j} \tag{4.1}$$

    El descenso por gradiente es un proceso iterativo donde los pesos se actualizan en cada iteración, es por ello que para obtener el mejor resultado es necesario procesar el *dataset* más de una vez. Debido a que procesar un *dataset* completo

en una única iteración es complicado por restricciones de memoria, este es dividido en partes llamadas *batch* o *lotes*.

- **Tamaño del lote** *(Batch Size)*, para ponerlo en perspectiva, pongamos que tenemos 1000 imágenes y se establece un tamaño de *batch* de 100 unidades, en total se tendrán 10 divisiones (10 lotes).
- **Valor de la tasa de aprendizaje** *(Learning Rate)*. Este hiperparametro indica que tan largo será el camino que tome el algoritmo de optimización. Si el valor es muy pequeño la actualización se puede quedar atrapada en un mínimo local y los valores de W y b no cambiarían correctamente, igualmente la red neuronal tardara mucho más tiempo en optimizar. Por otro lado si los valores son muy altos la actualización puede pasarse del punto perfecto que sería el mínimo global y nunca encontrarlo y aunque el aprendizaje sea más rápido nunca llegara al mínimo global:

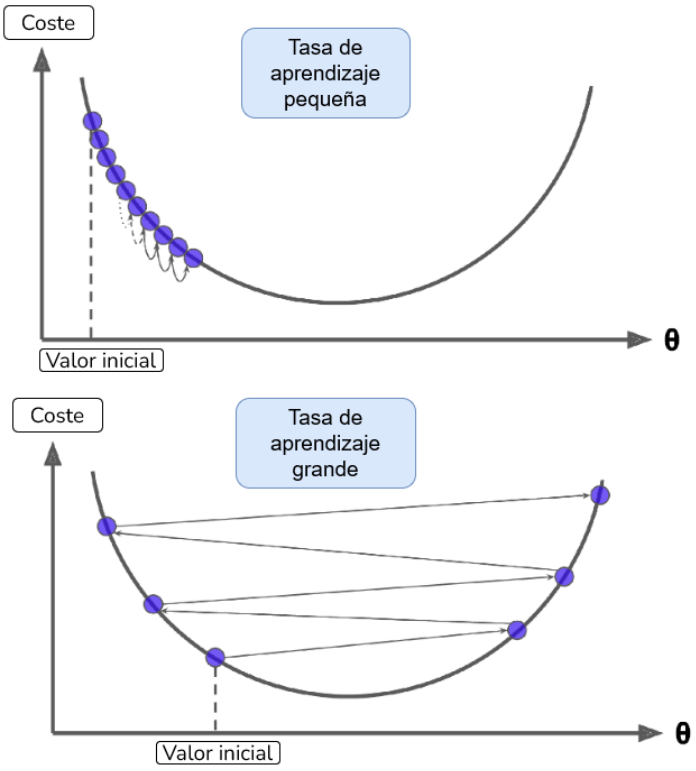


Figura 4.14: Comparación entre las tasas de aprendizaje bajas y altas. [62]

- **Tipo de arquitectura**, las opción que implementamos fue **Net12()**, una versión modificada de la estructura usada en la figura 3.1 y la propia arquitectura de 3.1 **Pnet()**.

```
class PNet(nn.Module):
    def __init__(self):
        super().__init__()
        device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        self.conv1 = nn.Conv3d(1, 8, kernel_size=3).to(device)
        self.conv2 = nn.Conv3d(8, 16, kernel_size=3).to(device)
        self.conv3 = nn.Conv3d(16, 32, kernel_size=3).to(device)
        self.fc1 = nn.Linear(16000, 32).to(device)
        self.fc2 = nn.Linear(32, 1).to(device)
        self.bn1 = nn.BatchNorm3d(8).to(device)
        self.bn2 = nn.BatchNorm3d(16).to(device)
        self.bn3 = nn.BatchNorm3d(32).to(device)

    def forward(self, x):
        out = F.max_pool3d(torch.relu(self.conv1(x)), 2)
        out = self.bn1(out)
        out = F.max_pool3d(torch.relu(self.conv2(out)), 2)
        out = self.bn2(out)
        out = F.dropout(out, p=0.45)
        out = F.max_pool3d(torch.relu(self.conv3(out)), 2)
        out = self.bn3(out)
        out = F.dropout(out, p=0.45)
        out = out.view(-1, 16000)
        out = torch.relu(self.fc1(out))
        out = torch.sigmoid(self.fc2(out))
        return out
```

```
class Net12(nn.Module):
    def __init__(self):
        super().__init__()
        device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        self.conv1 = nn.Conv3d(1, 8, kernel_size=3).to(device)
        self.conv2 = nn.Conv3d(8, 16, kernel_size=3).to(device)
        self.fc1 = nn.Linear(100672, 32).to(device)
        self.fc2 = nn.Linear(32, 1).to(device)
        self.bn1 = nn.BatchNorm3d(8).to(device)
        self.bn2 = nn.BatchNorm3d(16).to(device)

    def forward(self, x):
        out = F.max_pool3d(torch.relu(self.conv1(x)), 2)
        out = self.bn1(out)
        out = F.max_pool3d(torch.relu(self.conv2(out)), 2)
        out = self.bn2(out)
        out = F.dropout(out, p=0.45)
        out = out.view(-1, 100672)
        out = torch.relu(self.fc1(out))
        out = torch.sigmoid(self.fc2(out))
        return out
```

4.8. Métodos de interpretabilidad

Nuestra principal aportación con este trabajo es interpretar la red pero utilizando distintos algoritmos de interpretabilidad y visualización sobre datos PET-tau para poder observar cuál es mejor y porqué (es decir, compararlos formalmente).

Para un primer vistazo sobre los modelos de interpretabilidad con los que podríamos trabajar, consultamos T. Huff et al. (2021)[49] donde estudiamos en que consistían varios modelos, como *LRP*, *Occlusion* o *Saliency Maps*.

Tras esto, pasamos a ver la web de la librería de Python que hemos utilizado para aplicar estos algoritmos: Captum[69]. En la documentación de Captum[70] pudimos encontrar más información sobre los algoritmos que hemos utilizado, además de muchos otros.

De todos estos métodos se han seleccionado cuatro para interpretar las predicciones de nuestro modelo: *Saliency maps*, *Guided Backpropagation*, *Integrated gradients* y *Guided Grad-CAM*.

4.8.1. Saliency Maps

Saliency maps utiliza gradientes para visualizar la clasificación de una imagen evaluada por una red neuronal convolutiva. En el artículo [71], los autores ofrecen dos usos para este método: visualización de la maximizaron de clases y *saliency maps* imágenes-específicas.

La maximización de clases usa gradiente ascendente para producir una imagen que maximice la activación de esa clase, y después pueda ser interpretada como la más representativa de ese conjunto. Formalmente la *maximización de clases* encuentra una imagen para la cual la puntuación de clase es máxima, y por tanto puede ser interpretada como la mas representativa de dicha clase:

$$argmax S_c(I) - \lambda \|I\|_2^2 \tag{4.2}$$

Donde es un parámetro de regularización.

Los *Saliency Maps* de clases imagen-específicas son mapas de calor específicos que representan la importancia de cada píxel individual a la asignación de una imagen a una

clase, dando una evaluación de cuáles son las partes de la imagen más importantes para el modelo. *Saliency mapping* a veces es referido como análisis sensitivo *(sensitivity analysis)*.

Dada su simplicidad los *Saliency Maps* son unos de los métodos de interpretación mas usados en el ámbito de la imagen médica hasta la fecha.

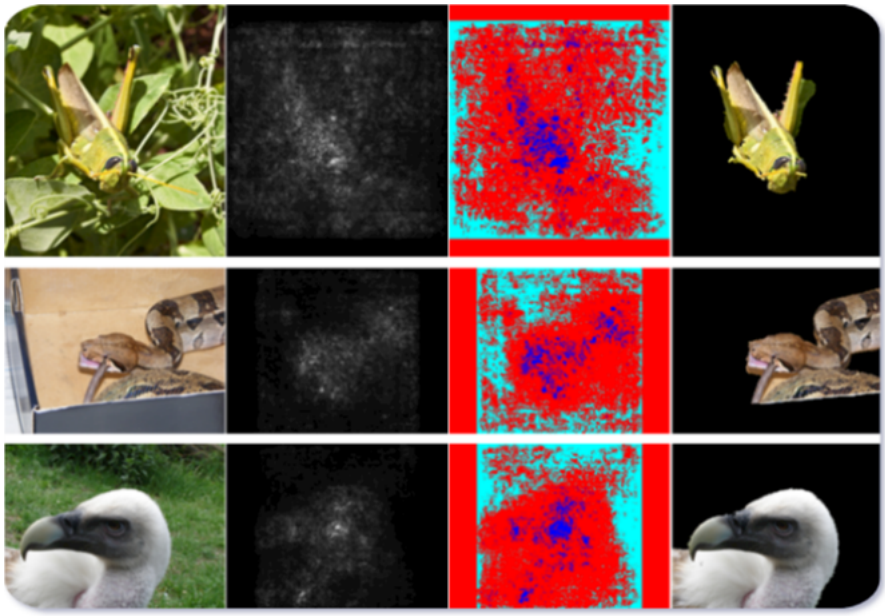


Figura 4.15: Segmentación de objetos débilmente supervisada usando Redes convolutivas. [71]

4.8.2. Guided Backpropagation

Se trata de una extensión de los *saliency maps* (para más información sobre ellos, consultar el capítulo 5.2.1), cuya diferencia se centra en cómo la realimentación (*backpropagation*) a través de las capas de activación tipo ReLU (*Rectified Linear Unit*) son manejadas. ReLU es una capa de activación comúnmente usada en las redes neuronales convolutivas. Durante el paso hacia adelante (*forward pass*), las neuronas con salidas de valor negativo son truncadas a cero por ReLU por definición $\rightarrow [ReLU(x) = max(0, x)]$. *Guided Backpropagation* combina esta primera idea y añade que los gradientes con valores negativos también sean truncados a 0, eliminando la señal a través de las neuronas que tengan una salida negativa en el paso hacia adelante o un gradiente negativo durante el paso hacia atrás (*backward pass*).

El resultado es un mapa de calor que señala únicamente píxeles que proporcionan evi-

dencia positiva para la clasificación.[49]

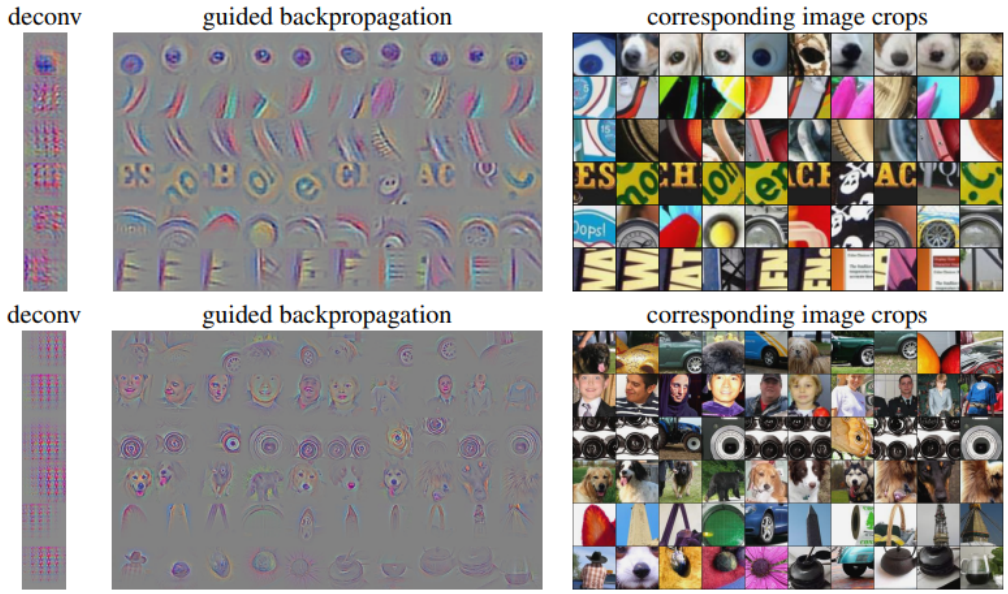


Figura 4.16: Visualización utilizando *Guided Backpropagation* de patrones aprendidos por dos capas convolutivas de una CNN. [72]

4.8.3. Integrated Gradients

El método *Integrated Gradients*[49] considera la imagen de entrada y una imagen de referencia con todo ceros en los gradientes, también llamada *baseline*. A partir de la imagen *baseline*, se produce un conjunto de imágenes intermedias entre la *baseline* y la imagen de entrada. En cada tramo intermedio entre los dos extremos, el gradiente de la salida del modelo con el de cada píxel de la imagen intermedia es calculado. Posteriormente estos gradientes son añadidos sobre las imágenes intermedias desde la imagen *baseline* hasta la imagen de entrada. Esto produce el mapa de calor con la importancia de cada píxel.[73]el mapa de calor de *Integrated Gradients* IG(x) es producido por:

$$IG(x) = (x - x') \int_0^1 \frac{\partial F(x' + \alpha(x - x'))}{\partial x} d\alpha$$

Dónde:

x es la imagen de entrada.

x' es la imagen de *baseline*.

F: Rn [0,1] representa el modelo CNN.

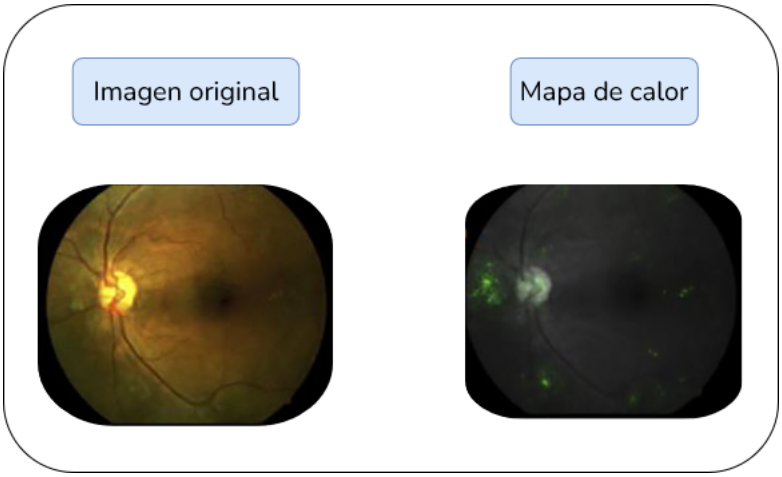


Figura 4.17: *Integrated Gradients* para visualizar evidencia de retinopatía diabética en imágenes del fondo retinal. [74]

4.8.4. Guided GradCAM

Es una asignación de activación de clase ponderada por gradiente guiada, fusiona *Guided Backpropagation* y *GradCAM*, calcula el producto por elementos de las atribuciones de *backpropagation guided* con atribuciones de *GradCAM* sobremuestreadas no negativas. utiliza la información de gradiente que fluye hacia el última capa convolucional de la CNN para asignar importancia a cada píxel[69].

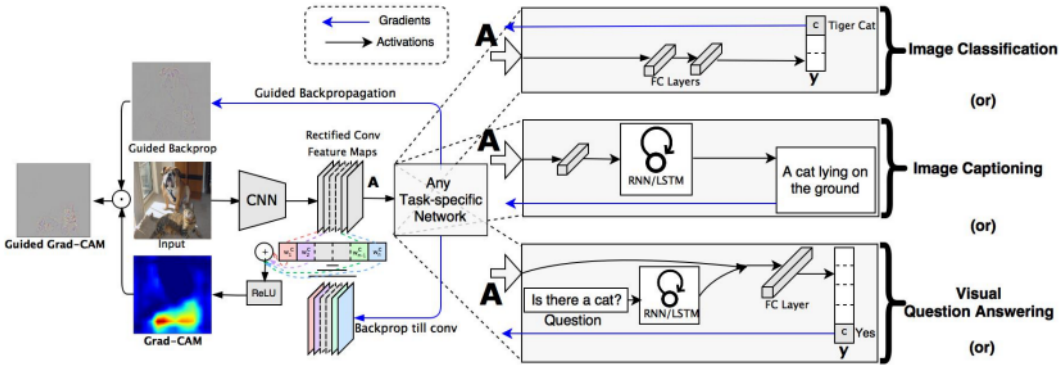


Figura 4.18: Esquema de *Guided GradCAM*. [75]

Dada una imagen y una clase como entrada, la imagen se propaga a través de la CNN del modelo, luego tras los cálculos pertinentes de la tarea se obtiene una puntuación bruta para cada categoría. Los gradientes se establecen en cero para todas las clases excepto la clase deseada, que se establece en 1. Esta señal luego se propaga de vuelta

a los mapas de características convolucionales de interés, los cuales se combinan para calcular la localización aproximada de *GradCAM*, que representa hacia dónde debe mirar el modelo para tomar una decisión en particular. Finalmente, multiplicamos puntualmente el mapa de calor con *Guided Backpropagation* pada para obtener visualizaciones guiadas de *GradCAM* que son tanto de alta resolución como específicas del concepto[75].

Capítulo 5

Resultados

5.1. Resultados obtenidos de los modelos

A continuación compararemos los resultados obtenidos en los 3 modelos mas relevantes. En las figuras podemos encontrar 6 gráficas en la parte superior, reflejando los datos de la curva de aprendizaje, la exactitud *(accuracy)* y el valor-F1 *(F1 score)* correspondientes al conjunto de entrenamiento o al de validación. Cada *fold* de la validación cruzada 4.11 está representado con un color diferente. En la zona inferior se recoge la matriz de confusión 4.7 obtenida del promedio de todos los *folds*, los parámetros con los que se configuró cada uno de los modelos, y los valores obtenidos de precisión y exactitud medios.

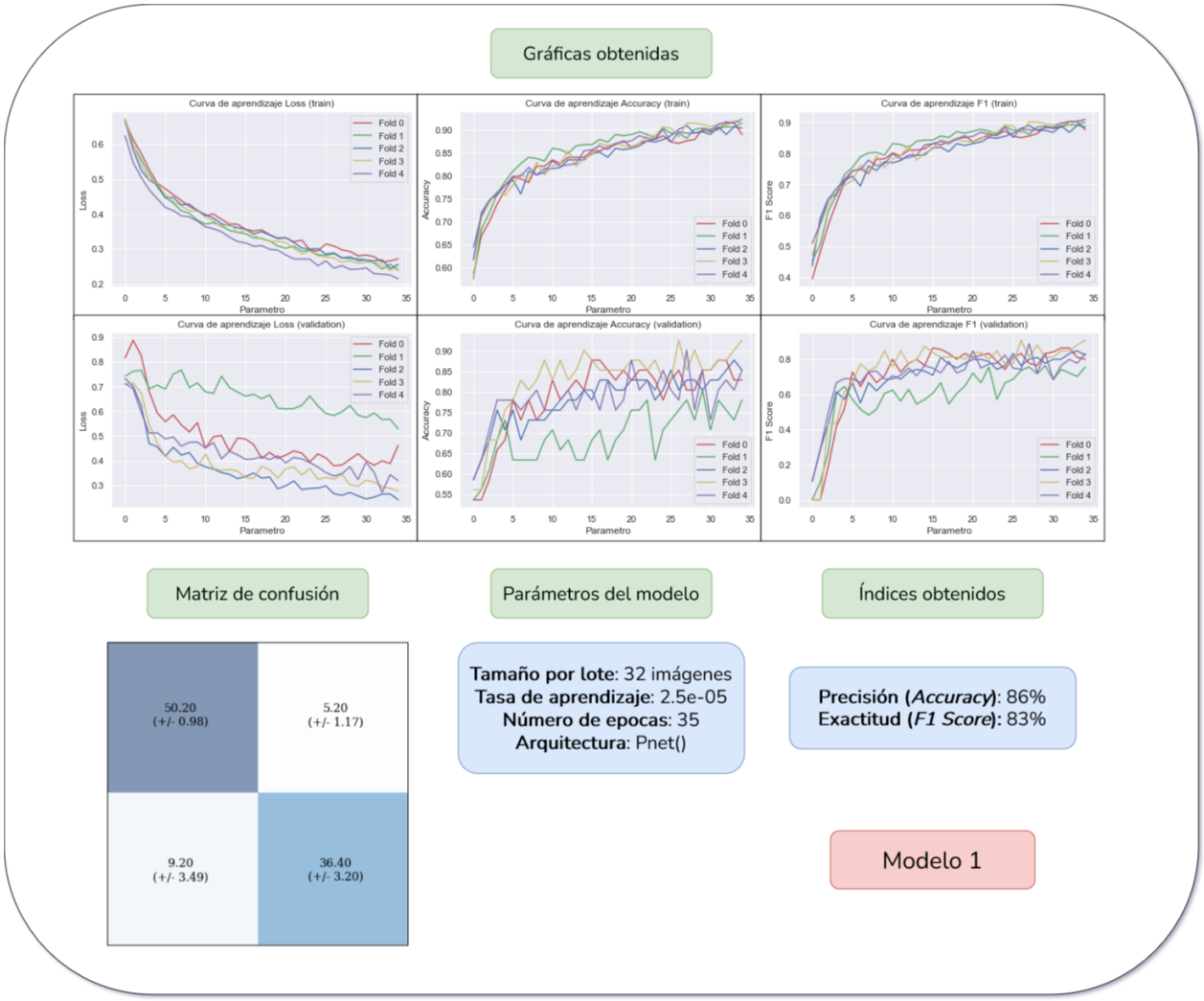


Figura 5.1: Resultados del primer modelo.

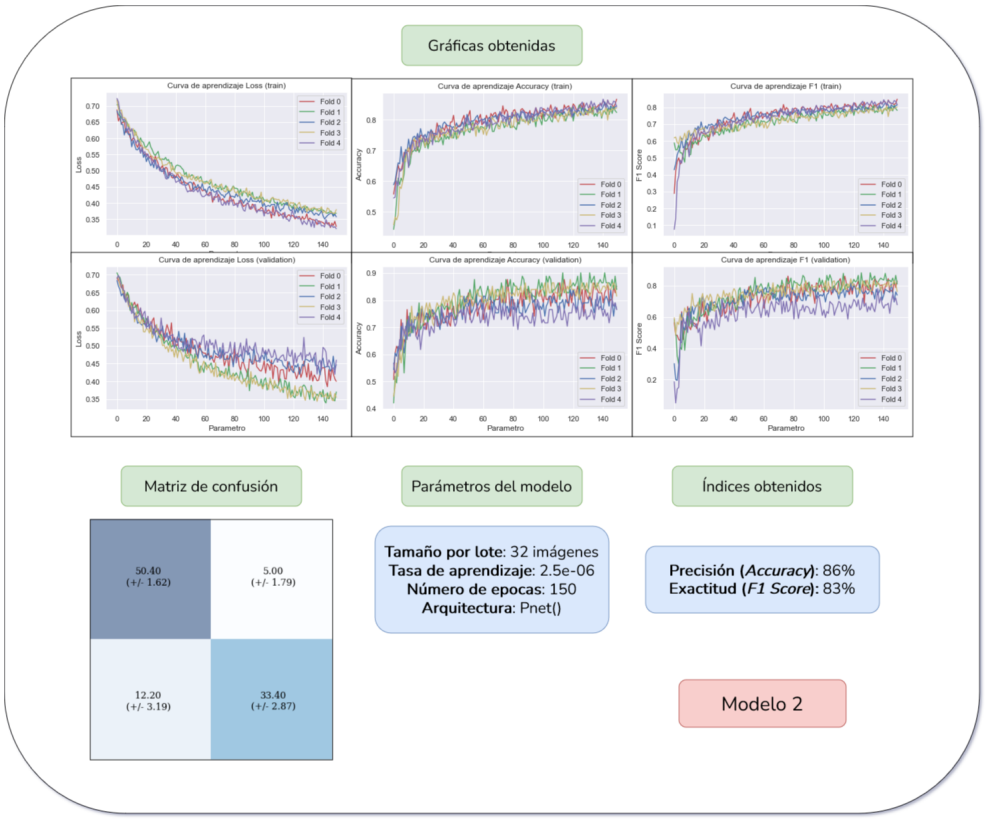


Figura 5.2: Resultados del segundo modelo.

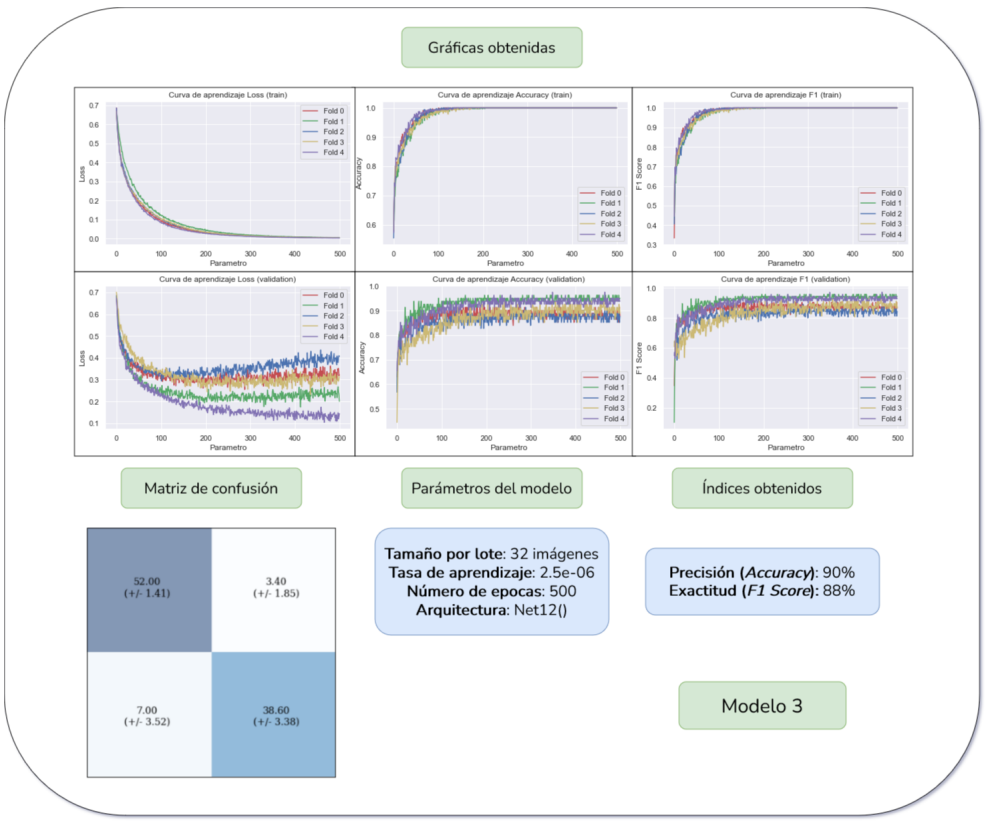


Figura 5.3: Resultados del tercer modelo.

5.1.1. Modelo 1

El primer prototipo que desarrollamos obtuvo muy buenos resultados en relación a la poca experiencia previa que teníamos al implementarlo. El **tamaño del lote** se estableció en 32, una cantidad de imágenes que no requerían demasiados recursos, en concreto este número se fue ajustando con modelos previos que no funcionaron debido a requerir mas **memoria RAM** de la disponible con el computador que entrenaba al modelo. Ajustamos la **tasa de aprendizaje** *(learning rate)* manualmente tras probar una serie de valores que teóricamente encajarían con nuestro problema. Para poder entrenar varios modelos y obtener diferentes respuestas en un tiempo no demasiado excesivo, el **número de épocas** se fijó en 32, permitiendo como decíamos repetir la ejecución y evaluar los parámetros de modelos diferentes. Para comenzar con una arquitectura probada y funcional, nos basamos en el artículo [45] mencionado en el capitulo 3.1.1. La precisión y exactitud obtenidas de este modelo como decíamos fueron las mejores de la primera generación de modelos y se siguió desarrollando este prototipo.

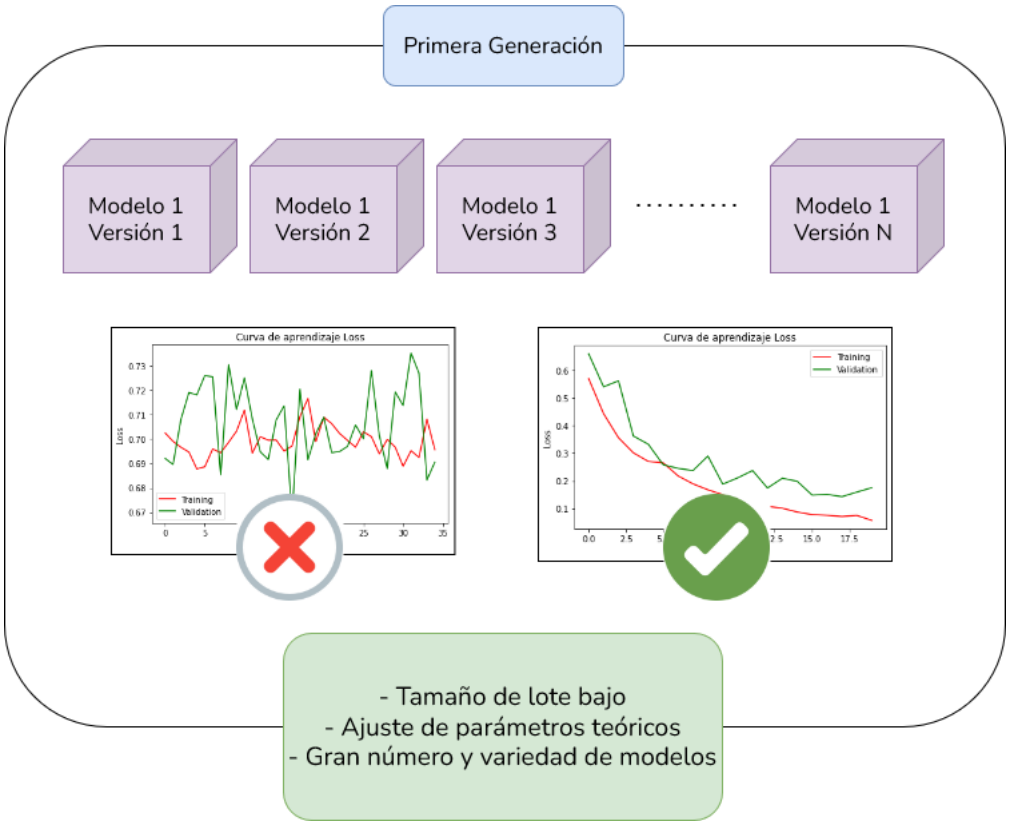


Figura 5.4: Enfoque para evaluar los resultados de la primera generación.

5.1.2. Modelo 2

El punto de partida para este segundo modelo era mejorar su rendimiento al elevar el número de épocas. Para ello, el tamaño del lote y la arquitectura del modelo se mantuvieron sin cambios. Reducimos el valor de la tasa de aprendizaje de **2.5e-5** a **2.5e-6** con la intención de que los modelos no se sobre entrenaran tan rápido al pasar cierto número de iteraciones, mantuvimos los resultados obtenidos del primer prototipo con 4 veces mas épocas de entrenamiento. Una característica que mejoramos dentro del bucle de entrenamiento fue el funcionamiento del *early stopping*, ya que hasta ahora su mecánica consistía en evaluar durante cuántas épocas seguidas no disminuía el error de validación. El problema consistía en que las salidas de los modelos no respondían con una salida estable, tenían ciertos picos que a la larga hacían que esta variante de *early stopping* no funcionase. La solución, como se menciona en la sección 4.5.6, fue usar la media de las últimas 10 iteraciones para evitar que la cuenta del *early stopping* se reiniciara.

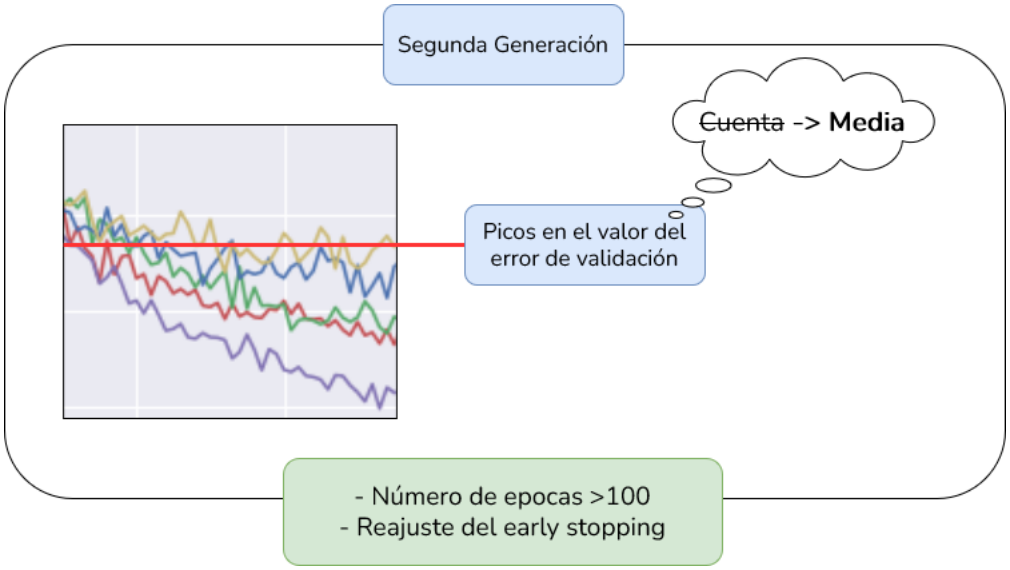


Figura 5.5: Enfoque para mejorar los resultados de la segunda generación.

5.1.3. Modelo 3

Esta última generación fue la mas concreta y precisa de todas. Habíamos obtenido experiencia tras entrenar a los modelos de las generaciones previas y decidimos hacer un último cambio en la configuración del modelo. En un estudio [76] realizado en 2020 se analiza la proporción de capas convolutivas que los modelos son capaces de manejar dependiendo del tamaño y volumen de datos disponibles. Las arquitecturas profundas con mas capas convolutivas de menor tamaño, pueden conseguir una mejora marginal a costa de hacer la red mas compleja y, por tanto, más difícil de ajustar. Además, el uso de arquitecturas complejas conlleva una mayor tendencia al sobreajuste y un coste computacional más elevado. Las redes mas anchas y superficiales, como las utilizadas en este trabajo, están más proporcionadas en relación con el volumen de datos disponibles. Como pudimos ver, el uso de este tipo de arquitecturas resultó ser la mejora de esta última generación. La configuración final se puede ver en **Net12()** 4.7. Eliminamos la última capa convolutiva con el objetivo de ajustar el tamaño de la red a la cantidad de neuroimágenes. Otro cambio fue establecer el número de épocas en un valor muy grande, la mejora en el *early stopping* nos permitía establecer 500 iteraciones porque cada *fold* pararía su ejecución al no obtener un error de validación menor en la media de las últimas 10 iteraciones.

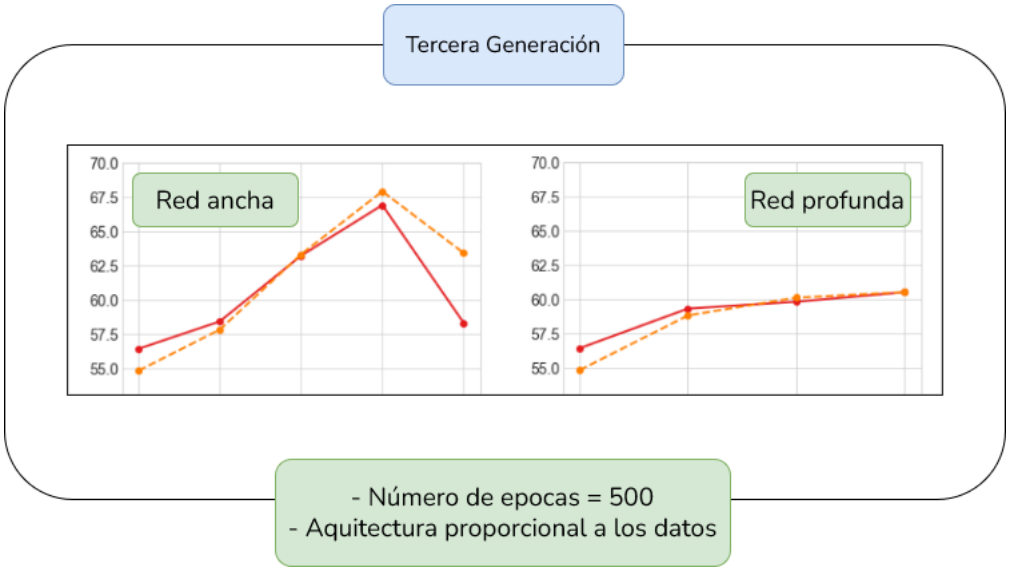
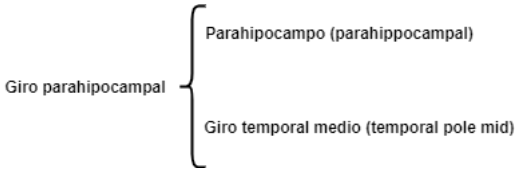


Figura 5.6: Metodología para optimizar el modelo final.

5.2. Análisis de interpretabilidad

En este capítulo se explica la visualización de las salidas de cada uno de los algoritmos de interpretación utilizados y la metodología de evaluación para concretar si los resultados concuerdan con estudios similares sobre el reconocimiento de la EA en neuroimágenes.

El lóbulo temporal medial humano *(human medial temporal lobe (MTL))* está presente en la memoria episódica, es decir, sucesos temporales pasados [77]. Al mismo tiempo el *MTL* es vulnerable a la EA [78]. Sin embargo el *MTL* no es homogéneo, por tanto, sus propiedades pueden variar en ciertas ubicaciones del espacio, estas ubicaciones serán denominadas subregiones, entre ellas destacamos para este estudio el **hipocampo** *(hippocampus)*, giro temporal medio y las subestructuras a lo largo del **giro parahipocampal** *(parahippocampal gyrus)* [79].



Los resultados en [79] se ajustan a los estudios neuropatológicos que indican cómo se involucran las cortezas perirrinal / entorrinal *(perirhinal/entorhinal cortices)* con el hipocampo anterior.

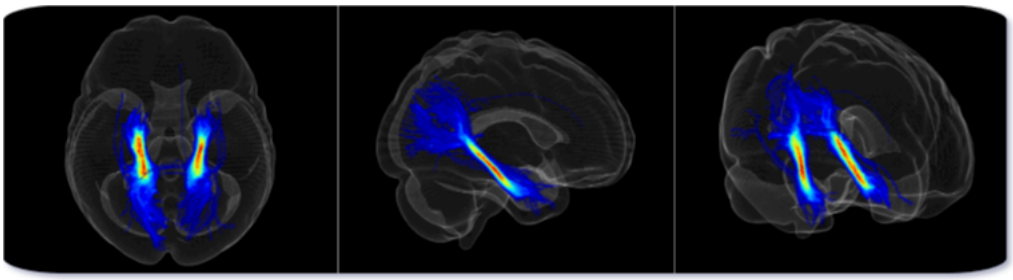


Figura 5.7: Anatomía del giro parahipocampal. [79]

De cada algoritmo de interpretabilidad se han obtenido cuatro mapas de calor *(en formato Nifti)*; imágenes de la **media** y la **desviación típica** obtenidas con el método, para los pacientes que presentan Alzhéimer; los pacientes sanos; y pacientes sanos y EA. Para conseguir estas imágenes, se aplicó el algoritmo de interpretabilidad sobre las imágenes

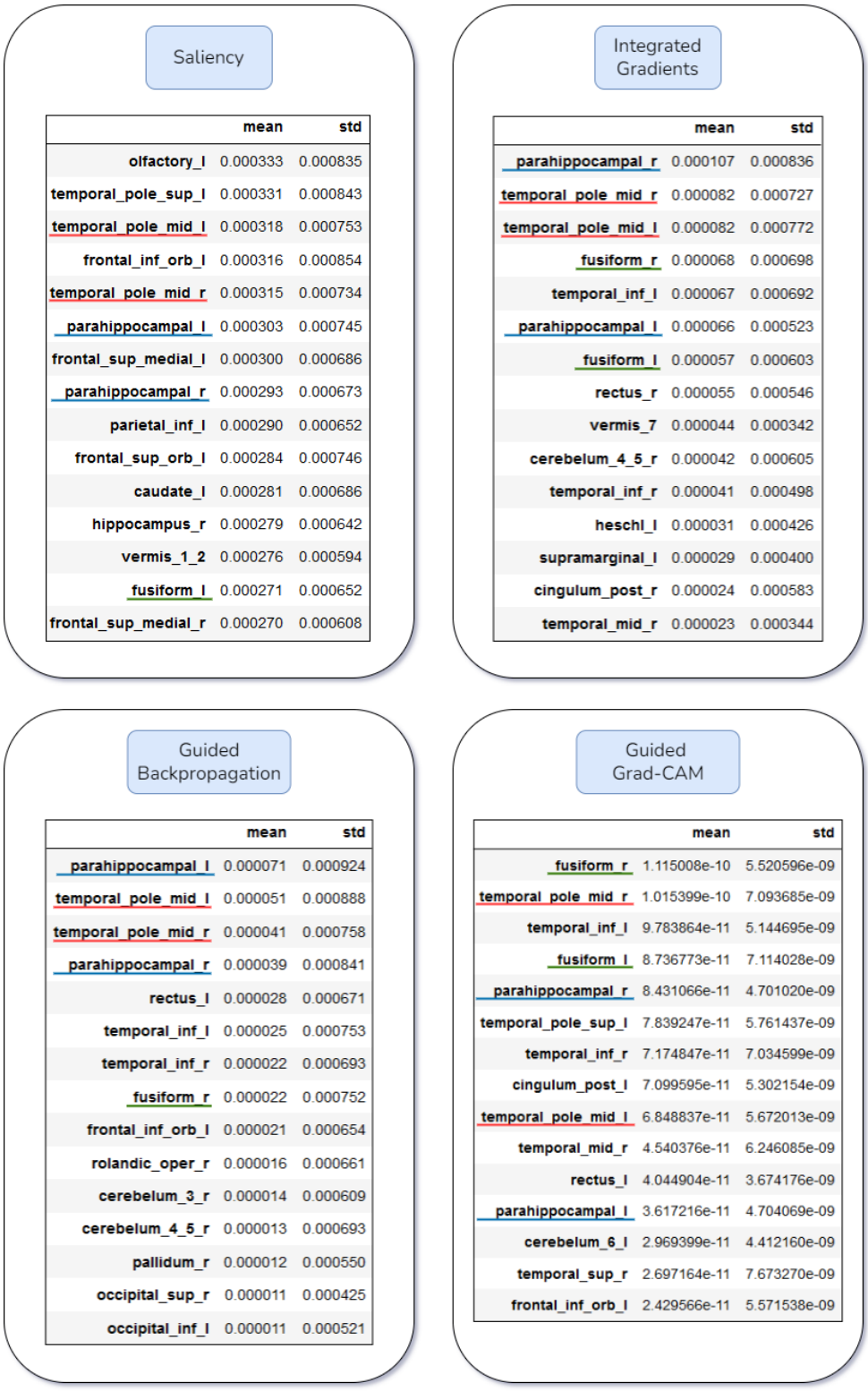
de testeo de cada uno de los modelos entrenados con los cinco *folds* (es decir, para cada modelo entrenado en cada *fold* con sus datos, se aplica el algoritmo sobre este modelo cargado interpretando sus imágenes de testeo).
Para poder ver exactamente qué zonas del cerebro son más relevantes, no sólo se visualizaron los mapas de calor tanto en 2D como en 3D, sino que también se utilizó el atlas AAL [80] para distinguir las distintas regiones de interés en el cerebro. Para cada método visualizamos la media y la desviación típica de las 15 regiones más representativas ordenadas por mayor media.

Realizaremos un análisis formal de todos los algoritmos de interpretabilidad y compararemos sus características, como por ejemplo las regiones en común con mas importancia asignada por cada método o sus valores. Esto será especialmente relevante para la identificación de patrones en la clasificación y diagnostico de pacientes con EA. Concluiremos con algunas ventajas y desventajas entre los distintos métodos que hemos usado en este proyecto.

En la gráfica 5.8 se han resaltado con un mismo color las distintas regiones en común entre los diferentes métodos de interpretación, las tablas están **representadas en orden descendente** de importancia por tanto las regiones en la parte superior son las mas relevantes para cada algoritmo.

En relación al estudio mencionado en 5.2 y a [81], las regiones en común mas relevantes son el **giro temporal medio** *(Temporal pole middle gyrus)* resaltado en color rojo y el parahipocampo resaltado en color azul, ambas regiones forman parte del **giro parahipocampal** tal como se ve en la figura 5.7 que concuerdan con los resultados obtenidos de los tres métodos de interpretación que usamos para nuestro estudio. Esto demuestra que los mapas de calor sí se relacionan y representan elementos y regiones significativas para el diagnostico de EA. Las regiones cerebrales fusiformes se han marcado en verde dada su relevancia según [81].

En esta gráfica reflejamos las regiones en común a las que los distintos métodos de interpretabilidad asignan mas importancia:



	mean	std
olfactory_l	0.000333	0.000835
temporal_pole_sup_l	0.000331	0.000843
temporal_pole_mid_l	0.000318	0.000753
frontal_inf_orb_l	0.000316	0.000854
temporal_pole_mid_r	0.000315	0.000734
parahippocampal_l	0.000303	0.000745
frontal_sup_medial_l	0.000300	0.000686
parahippocampal_r	0.000293	0.000673
parietal_inf_l	0.000290	0.000652
frontal_sup_orb_l	0.000284	0.000746
caudate_l	0.000281	0.000686
hippocampus_r	0.000279	0.000642
vermis_1_2	0.000276	0.000594
fusiform_l	0.000271	0.000652
frontal_sup_medial_r	0.000270	0.000608

	mean	std
parahippocampal_r	0.000107	0.000836
temporal_pole_mid_r	0.000082	0.000727
temporal_pole_mid_l	0.000082	0.000772
fusiform_r	0.000068	0.000698
temporal_inf_l	0.000067	0.000692
parahippocampal_l	0.000066	0.000523
fusiform_l	0.000057	0.000603
rectus_r	0.000055	0.000546
vermis_7	0.000044	0.000342
cerebelum_4_5_r	0.000042	0.000605
temporal_inf_r	0.000041	0.000498
heschl_l	0.000031	0.000426
supramarginal_l	0.000029	0.000400
cingulum_post_r	0.000024	0.000583
temporal_mid_r	0.000023	0.000344

	mean	std
parahippocampal_l	0.000071	0.000924
temporal_pole_mid_l	0.000051	0.000888
temporal_pole_mid_r	0.000041	0.000758
parahippocampal_r	0.000039	0.000841
rectus_l	0.000028	0.000671
temporal_inf_l	0.000025	0.000753
temporal_inf_r	0.000022	0.000693
fusiform_r	0.000022	0.000752
frontal_inf_orb_l	0.000021	0.000654
rolandic_oper_r	0.000016	0.000661
cerebelum_3_r	0.000014	0.000609
cerebelum_4_5_r	0.000013	0.000693
pallidum_r	0.000012	0.000550
occipital_sup_r	0.000011	0.000425
occipital_inf_l	0.000011	0.000521

	mean	std
fusiform_r	1.115008e-10	5.520596e-09
temporal_pole_mid_r	1.015399e-10	7.093685e-09
temporal_inf_l	9.783864e-11	5.144695e-09
fusiform_l	8.736773e-11	7.114028e-09
parahippocampal_r	8.431066e-11	4.701020e-09
temporal_pole_sup_l	7.839247e-11	5.761437e-09
temporal_inf_r	7.174847e-11	7.034599e-09
cingulum_post_l	7.099595e-11	5.302154e-09
temporal_pole_mid_l	6.848837e-11	5.672013e-09
temporal_mid_r	4.540376e-11	6.246085e-09
rectus_l	4.044904e-11	3.674176e-09
parahippocampal_l	3.617216e-11	4.704069e-09
cerebelum_6_l	2.969399e-11	4.412160e-09
temporal_sup_r	2.697164e-11	7.673270e-09
frontal_inf_orb_l	2.429566e-11	5.571538e-09

Figura 5.8: Regiones con mas importancia.

5.2.1. Saliency Maps

El primer método utilizado fue *Saliency Maps* (Para más información sobre este método consultar el capítulo 4.8.1).
Las 15 regiones con mayor media de este método en pacientes con EA (ver figura 5.8) incluyen varias de las más relevantes para diagnosticar Alzhéimer. Estas son el parahipocampo, el giro temporal medio y el fusiforme izquierdo.

Se pueden observar los mapas de calor de este método para pacientes EA y control en la figura 5.9. En esta figura podemos ver como, según *Saliency Maps*, la media en pacientes con EA en la red neuronal se centra en una zona mucho más concreta que en los pacientes de control, donde se observan muchas más regiones marcadas.

La mayor desviación típica se observa localizada en la zonas de mayor relevancia para pacientes con EA, algo esperable al ser la zona más sensible en este tipo de pacientes, mientras que en pacientes de control esta no está especialmente marcada en ninguna zona concreta.

La conclusión final es que este método proporciona buenos resultados al dar especial relevancia a las regiones pertenecientes al giro parahipocampal sobre otras.

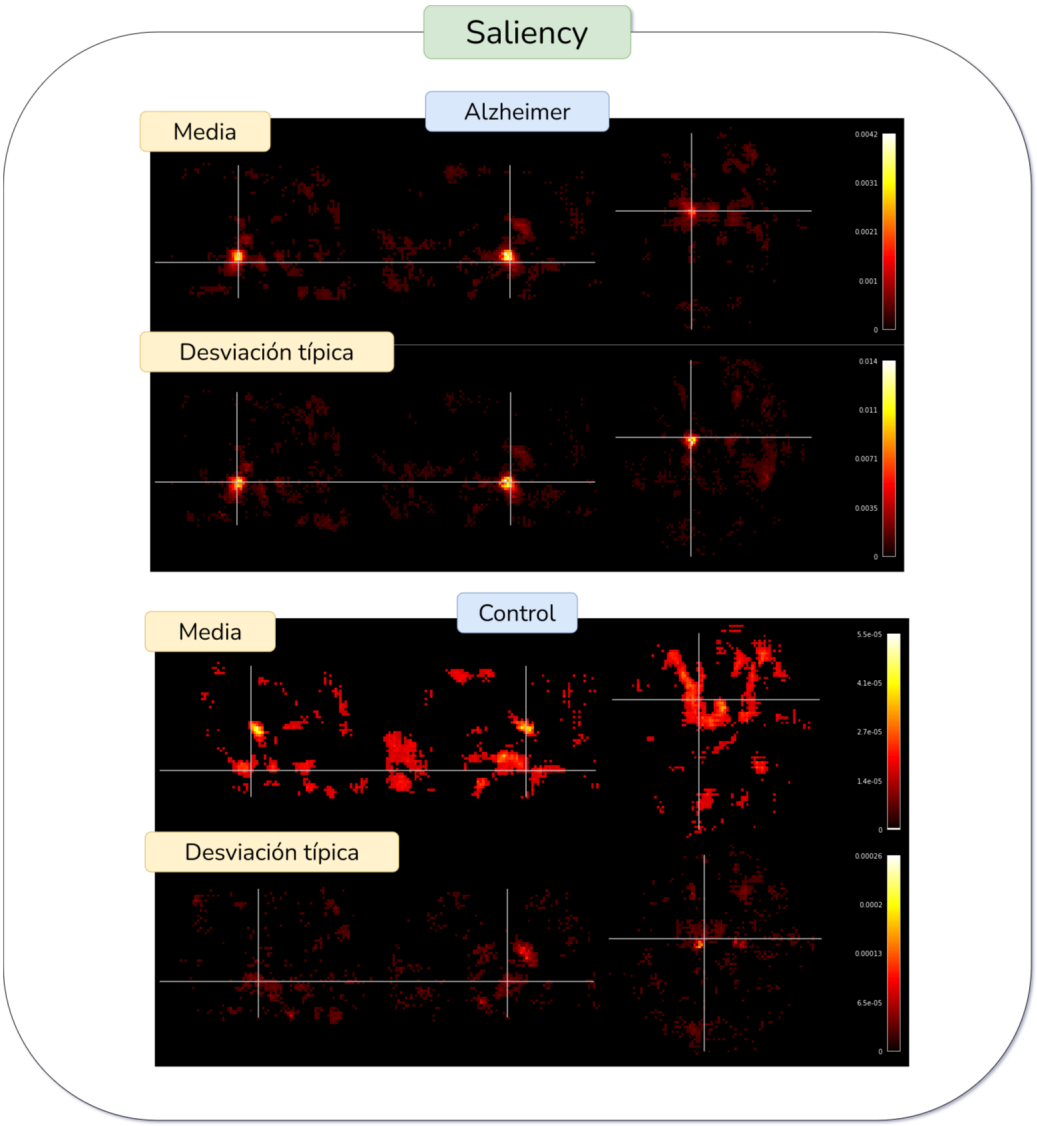


Figura 5.9: Resultados de Saliency Maps.

5.2.2. Guided Backpropagation

A continuación vamos a analizar los resultados obtenidos con *guided backpropagation*. Las cuatro regiones a las que mayor importancia da este método en pacientes EA (ver figura 5.8) son regiones relevantes en el diagnóstico del Alzhéimer. En concreto, estas son (por orden de importancia) el parahipocampo izquierdo, el giro temporal medio izquierdo, el giro temporal medio derecho y el parahipocampo derecho. Además de estas, el fusiforme derecho es la octava región más relevante. Esto demuestra que este método es capaz de encontrar las zonas relevantes de forma efectiva, aunque con algunas pegas respecto al método *Saliency Maps*, y es que los valores de la media son mucho más pequeños que en este método, lo que significa que le da relevancia a más zonas del cerebro.

Esto se puede observar visualmente en la figura 5.10, donde encontramos los mapas de calor para pacientes EA como control. En esta figura vemos como ninguna zona está especialmente marcada como sí ocurría en *Saliency Maps* (ver figura 5.9). Esto es algo relevante dado que las desviaciones típicas tienen valores parecidos a los presentados en *Saliency Maps*, resultando especialmente negativo al dar en algunos casos valores relevantes a regiones cerebrales que no lo son para diagnosticar Alzhéimer. Aun así, podemos observar como la desviación típica es especialmente relevante en zonas importantes para el diagnóstico de EA.

Además, podemos observar como los mapas de calor para ambos tipos de pacientes son muy parecidos, algo que choca de nuevo con *Saliency Maps* y que indica más homogeneidad a la hora de mirar zonas independientemente de si los pacientes presentan o no EA.

Concluimos que este método tiene un buen resultado, aunque funciona mejor *Saliency Maps* al no dar relevancia a tantas regiones (y por tanto centrarse más en unas pocas) y ser el que más importancia asocia a las regiones pertenecientes al giro parahipocampal.

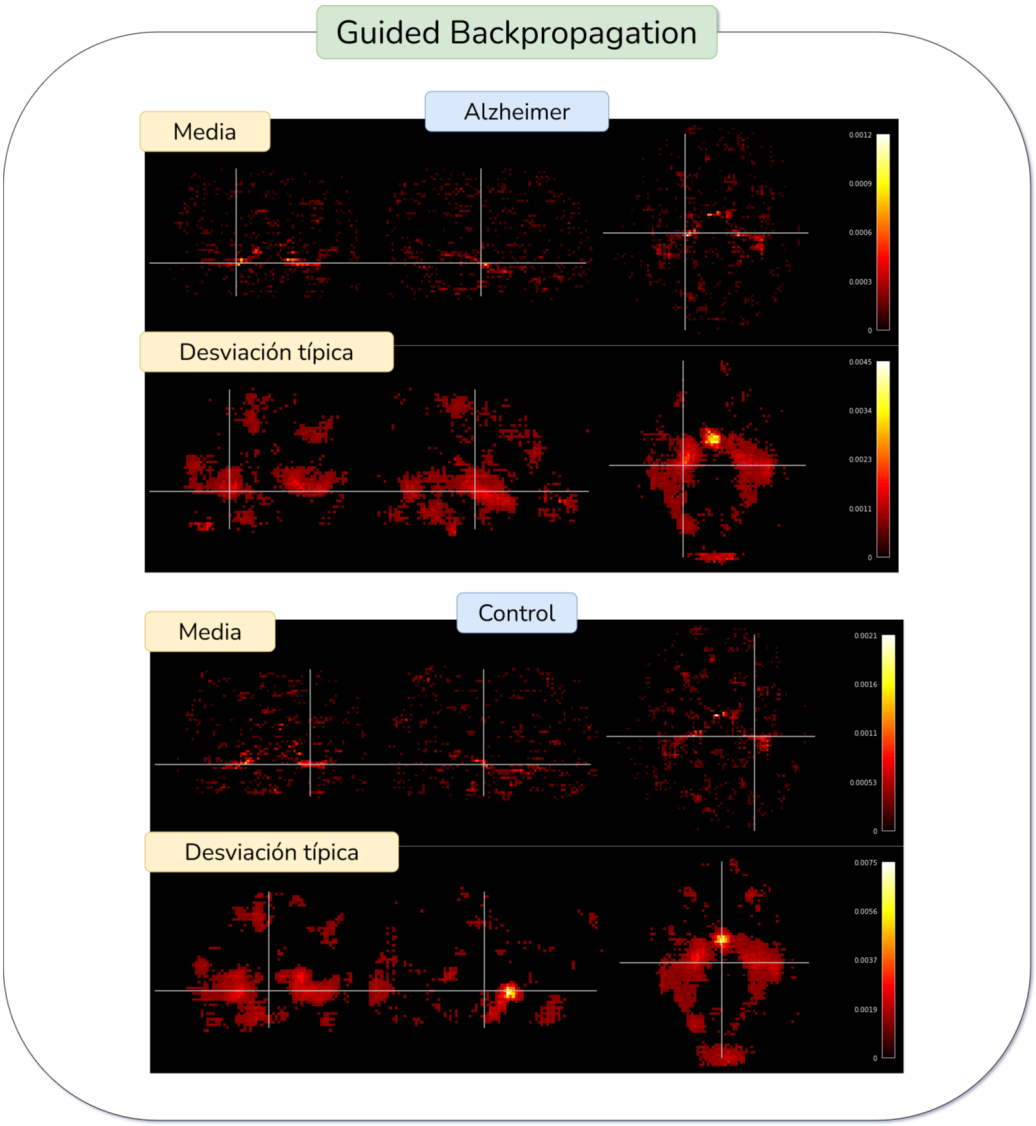


Figura 5.10: Resultados de Guided Backpropagation

5.2.3. Integrated Gradients

En esta sección vamos a analizar los resultados obtenidos al aplicar el método *Integrated Gradients*. Entre las quince regiones con más importancia para *Integrated Gradients* en pacientes EA (ver figura 5.8) encontramos de nuevo varias de las más relevantes para el diagnóstico de Alzhéimer. En concreto, estas son (por orden de relevancia) el parahipocampo derecho, el giro temporal medio derecho, el giro temporal medio izquierdo, el fusiforme derecho, el parahipocampo izquierdo y el fusiforme derecho. Este método tiene de nuevo consistencia con lo observado en los dos anteriores respecto a que zonas son más relevantes, aunque con más semejanzas en los resultados a *Guided Backpropagation* (ver capítulo 5.2.2) que a *Saliency Maps* (ver capítulo 5.2.1).

En concreto, al igual que ocurre en *Guided Backpropagation*, encontramos que los valores de la media son más pequeños que los obtenidos en *Saliency Maps*, por lo que *Integrated Gradients* daría relevancia a más zonas, no concretando en áreas importantes. Las medias aún así son algo mayores que las vistas en *Guided Backpropagation*. En cuanto a la desviación típica, encontramos de nuevo un acercamiento a los resultados vistos en *Guided Backpropagation*, con unas desviaciones típicas similares a la encontrada en tanto *Saliency Maps* como en *Guided Backpropagation*, pero con la misma sutileza que en el segundo, al ser la media menor que el primero pero la desviación típica similar, podría darse el caso de asignar importancia a regiones cerebrales no relevantes para el diagnóstico de EA.

Esto es algo que podemos observar visualmente en la figura 5.11, donde encontramos los mapas de calor tanto en pacientes EA como control.

En esta imagen podemos observar cómo la relevancia está más extendida por distintas regiones que concretada en una sola, como sí ocurre en *Saliency Maps* (ver figura 5.9), respaldando lo comentado anteriormente. En este método volvemos a observar con mayor claridad como en pacientes de control se da importancia en la media a un rango de regiones más amplio que en pacientes EA, pero con la desviación típica siendo especialmente importante en ambos tipos de pacientes dentro de áreas relevantes para el diagnóstico del Alzhéimer.

Finalmente, podemos concluir que este método nos reporta un buen resultado, aunque

seguimos observando que *Saliency Maps* realiza un mejor trabajo al no dar relevancia a tantas regiones, además de ser la que más importancia asocia a las regiones pertenecientes al giro parahipocampal.

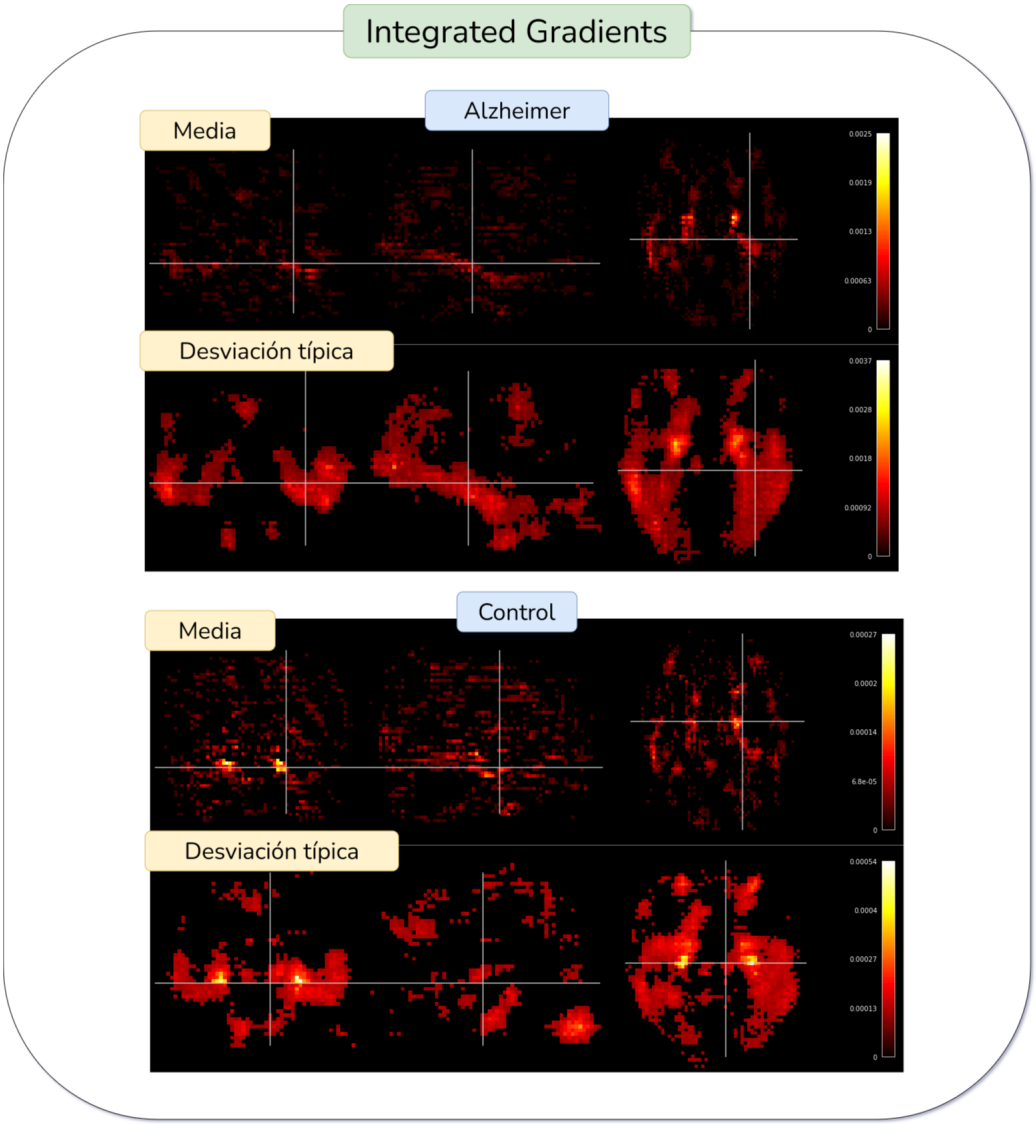


Figura 5.11: Resultados de integrated gradients

5.2.4. Guided GradCAM

El método *Guided GradCAM* ha aportado resultados con valores especialmente bajos y poco fiables, los cuales no son aptos para poder asegurar nada sobre ellos. Este problema puede estar relacionado con lo mencionado en [40]. Ocurre que la imagen de salida tiene una menor resolución, ya que procede de la última capa convolutiva en lugar de venir de la capa linear del final, como ocurre con los otros tres métodos utilizados.

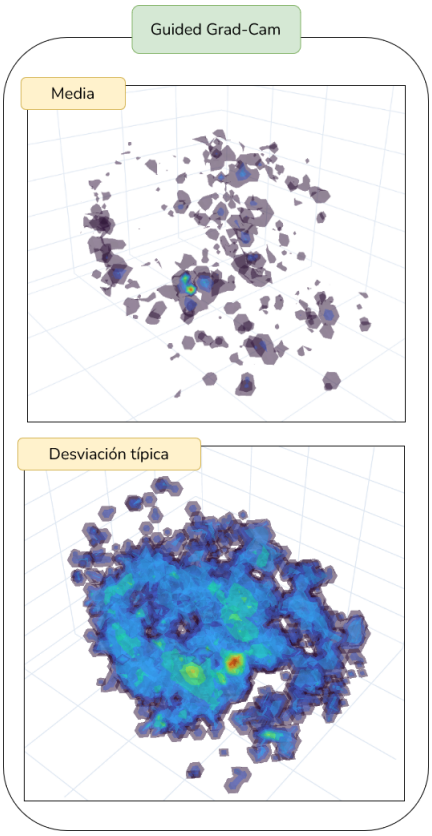


Figura 5.12: Resultados de guided Grad Cam

5.3. Comparación de los métodos

En base a los resultados obtenidos, concluimos que de todos los métodos de interpretación usados, el algoritmo de *Saliency* tiene una mejor generalización, asignando una importancia mayor a toda la región del para-hipocampo *(temporal pole mid r + temporal pole mid i + parahippocampal r + parahippocampal i)* con un valor total del **0.001229**, El método de *Integrated gradients* otorga a esta misma región un total de **0.000337** y el algoritmo de *Guided Backpropagation* **0.000202**. Por consiguiente la suma de estas 4 áreas pasa a ser la más representativa de cada uno de los métodos, ya que no hay ninguna otra región con valores mayores de media. A las regiones cerebrales fusiformes *Saliency* le asigna un valor de **0.000271**, *integrated gradients* un total de **0.000125** sumando el área izquierda y derecha y el método de *Guided Backpropagation* **0.000022**.

Basándonos en que estas dos regiones deben ser las más representativas respecto al resto, podemos concluir que *Integrated gradients* es el algoritmo que más las prioriza, quedando en su tabla como los dos primeros valores con mas media.

Capítulo 6

Trabajo individual

Este proyecto ha seguido un total de cuatro fases, en primera instancia el trabajo consistió en investigar y estudiar todo el conocimiento necesario, todos los participantes comenzamos con *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlowConcepts* [62], un volumen con una base suficiente para comenzar a desarrollar un prototipo de prueba. Fueron necesarias lecturas adicionales en relación a cada una de las tecnologías descritas en el capitulo 4.2. Esta fase nos supuso dos meses (agosto y septiembre).

La segunda fase consistió en implementar el modelo basado en *deep-learning*. En esta parte del trabajo fue donde tuvimos que comenzar a poner a prueba lo estudiado en los meses previos. Hemos utilizado una metodología ágil (*Scrum*) en la que avanzábamos con el proyecto simultáneamente dividiéndolo en tareas cortas con múltiples iteraciones realizadas por los tres miembros del grupo. Los integrantes del grupo nos reuníamos tres días a la semana mediante videoconferencias y herramientas de trabajo simultaneo como discord para desarrollar el proyecto a la vez, alternando los roles de programador, analista y recopilador de información. Este funcionamiento de grupo lo mantuvimos durante el resto de fases, a excepción de las lecturas individuales. Semanalmente nos reuníamos con el tutor y director del TFG para mantener un correcto flujo de trabajo y resolver las dudas que nos iban surgiendo. Para la resolución de algunas dudas también nos comunicábamos por medio de correos electrónicos. Esta segunda fase duró tres meses y medio (noviembre, diciembre, enero y parte de febrero).

La tercera fase concluyó con el desarrollo de los métodos de interpretación, comenzando

por varías lecturas teóricas sobre los distintos algoritmos y pasando a implementarlos en nuestro proyecto. A esta tercera fase le dedicamos tres meses y medio ya que fue la más larga (enero, febrero, marzo y abril).

Por último redactamos la memoria. Gran parte del contenido ya lo habíamos redactado de forma previa, de manera que tuvimos que maquetarlo, introducir gráficas e imágenes y terminar de darle el formato adecuado para su presentación. Esta fase final fue redactada durante el mes de mayo.

6.1. Diagrama de Gantt

Figura 6.1: Diagrama de Gantt simplificado del proyecto.

6.2. Pablo Álvarez García

6.2.1. Estudio e investigación

- Lectura de bibliografías relacionadas con el proyecto
- Estudio de las tecnologías escogidas en cada fase
- Estudio sobre los métodos de interpretabilidad

6.2.2. Desarrollo del modelo

- Procesamiento de imágenes y carga de la base de datos
- Instalación y configuración del framework de trabajo
- Implementación de los algoritmos necesarios
- Desarrollo de los modelos
- Entrenamiento de los modelos
- Generación de gráficas para evaluar modelos

6.2.3. Algoritmos de interpretación

- Implementación de los métodos y generación de mapas de calor

6.2.4. Redacción de la memoria

- Resumen y *abstract*
- Introducción e *introduction*
- Estado del arte
- Metodología
- Resultados
- Trabajo futuro
- Conclusiones y *conclussions*

6.3. Álvaro Corrochano López

6.3.1. Estudio e investigación

- Lectura de bibliografías relacionadas con el proyecto
- Estudio de las tecnologías escogidas en cada fase
- Estudio sobre los métodos de interpretabilidad

6.3.2. Desarrollo del modelo

- Procesamiento de imágenes y carga de la base de datos
- Instalación y configuración del framework de trabajo
- Implementación de los algoritmos necesarios
- Desarrollo de los modelos
- Entrenamiento de los modelos
- Generación de gráficas para evaluar modelos

6.3.3. Algoritmos de interpretación

- Implementación de los métodos y generación de mapas de calor

6.3.4. Redacción de la memoria

- Resumen y *abstract*
- Introducción e *introduction*
- Estado del arte
- Metodología
- Resultados
- Trabajo futuro
- Conclusiones y *conclussions*

6.4. Antonio Fernández Martín

6.4.1. Estudio e investigación

- Lectura de bibliografías relacionadas con el proyecto
- Estudio de las tecnologías escogidas en cada fase
- Estudio sobre los métodos de interpretabilidad

6.4.2. Desarrollo del modelo

- Procesamiento de imágenes y carga de la base de datos
- Instalación y configuración del framework de trabajo
- Implementación de los algoritmos necesarios
- Desarrollo de los modelos
- Entrenamiento de los modelos
- Generación de gráficas para evaluar modelos

6.4.3. Algoritmos de interpretación

- Implementación de los métodos y generación de mapas de calor

6.4.4. Redacción de la memoria

- Resumen y *abstract*
- Introducción e *introduction*
- Estado del arte
- Metodología
- Resultados
- Trabajo futuro
- Conclusiones y *conclussions*

Capítulo 7

Trabajo futuro

En este capítulo vamos a comentar el trabajo futuro que pueda complementar el trabajo de este proyecto.

En primer lugar, se podría tratar de optimizar los hiperparámetros del modelo, algo que no pudimos implementar dado su elevado coste en tiempo. Esto se podría hacer por ejemplo con la librería RayTune [82].
Otra mejora sería realizar un desenfoque gaussiano (*gaussian smoothing*) de forma previa al cálculo de los mapas de calor obtenidos por los algoritmos de interpretabilidad. Este procedimiento, suaviza los mapas de bits de las imágenes, reduciendo así el ruido visual y el detalle de la imagen. Podemos observar como actúa esta técnica en la figura 7.1. Utilizar esta técnica creemos que reduciría la desviación típica en los resultados obtenidos en los algoritmos de interpretabilidad.

Figura 7.1: Ejemplo de aplicación de desenfoque gaussiano [83].

También se podrían haber implementado más métodos de interpretabilidad.
Bajo nuestro punto de vista, LRP sería el más interesante (ya que es usado en diversos artículos como por ejemplo en [45]), que no fue implementado por problemas tecnológicos y temporales al no estar adaptada para capas convolutivas tridimensionales en Captum.
Otro método de interpretabilidad que también sería de interés implementar es Oclusión, dado que su funcionamiento es simple y se ha utilizado en algunos de los trabajos relacionados que hemos comentado (ver capítulo 3.1), como por ejemplo en [30].
Sobre los métodos utilizados en nuestro trabajo, llama la atención que, sin contar el giro parahipocampal como la suma de todas sus regiones ROIs, la región con más importancia dada por *Saliency Maps* fue la región olfativa izquierda. Investigando sobre el tema, encontramos que recientes estudios han encontrado relación entre la región olfativa y los síntomas precoces de Alzhéimer [84], por lo que podría ser de interés ampliar el estudio de la relación de esta zona cerebral con la enfermedad del Alzhéimer.

Por otro lado, hubiera sido interesante contrastar los resultados obtenidos con neurólogos añadiendo credibilidad a los mismos y ayudando a poder analizarlos desde un enfoque más técnico en el ámbito médico.

Otra línea de trabajo futuro podría incluir la validación del modelo con un conjunto de datos independiente de ADNI de cara a evaluar su capacidad de generalización. Adicionalmente, el desarrollo de modelos centrados en la detección de estadios previos a la EA, como el DCM (daño cognitivo medio); o la combinación de diferentes técnicas de neuroimagen, podrían ser aspectos de gran interés.

Capítulo 8

Conclusiones

En este proyecto hemos desarrollado un modelo de **deep learning** para detectar patrones informativos en imágenes cerebrales de Tomografía por Emisión de Positrones (**PET**) para la clasificación de pacientes con Alzheimer. Obtuvimos una **red neuronal convolutiva** de 3 dimensiones capaz de clasificar pacientes con EA y pacientes de control de un total de **507 imágenes**, con una **exactitud del 90 %** y una **valor F1 del 0'88**.

En una etapa posterior al la obtención del modelo definitivo, aplicamos diferentes **métodos de interpretación** para generar mapas de calor reflejando la **importancia** de distintas **regiones** para la clasificación de los diagnósticos.

Varios estudios [85] [79] han examinado cuales son las regiones clave en las que se produce una mayor acumulación de proteína TAU en el transcurso de la EA. Estas se superponen con las principales regiones de interés del estadio III de Braak [78], correspondientes al **parahipocampo** y la **región fusiforme**. Al comparar los resultados dados por los métodos: *saliency maps*, *integrated gradients*, *Guided Backpropagation* y *Guided GradCam*, pudimos ver que los mapas de calor obtenidos coincidían con lo esperado respecto al reconocimiento de patrones en estas áreas. En particular, los mapas de calor muestran que el parahipocampo y la región fusiforme son usadas ante todo para diferenciar pacientes con EA de sujetos sanos. Comparando los resultados dados por los diferentes métodos de interpretación aplicados, pudimos ver que el ***Saliency Maps*** es el que muestra una mayor consistencia con el conocimiento actual de la enfermedad. El producto de nuestro estudio respalda la investigación existente demostrando que la acumulación de TAU en

áreas relacionadas con la memoria juega un papel importante en el desarrollo de la EA [45].

Un diagnóstico temprano y preciso de la enfermedad de Alzhéimer es importante para marcar el inicio de un tratamiento efectivo. Las técnicas de *deep learning* han demostrado tener un gran potencial al respecto. Por otro lado, la introducción de técnicas de interpretabilidad representan un paso más hacia una posible incorporación de este tipo de modelos en el ámbito clínico como herramientas de soporte diagnóstico.

Capítulo 9

Conclussions

In this project we have developed a model implemented through **deep learning** to detect patterns in brain images of Positron Emission Tomography (**PET**) for the classification of patients with AD. We obtained a 3D **convolutional neural network** which is able to classifying AD patients and control patients from **507 images**, they have a **90 % of accuracy** and **0'88 value in F1** .

In the stage after the final model, we apply different **interpretation methods** to generate heat maps that presents the **importance** of different **regions** for the classification of diagnoses.

There are many studies [85] [79] that examined which key regions contributing to a high global TAU signal and to the main Braak stage III important regions [78],they're **parahippocampus** and the **fusiform region**. When we compare the outputs of the algorithms of interpretation *saliency maps*, *integrated gradients*, *Guided Backpropagation* and *Guided GradCam*, we obtained results that coincided with those expected regarding pattern recognition in these regions, in particular, the heat maps show that the parahippocampus and the fusiform region are primarily used to classify AD patients, specifically among all the interpretation methods used, ***Saliency Maps*** is the one with the best results. The conclusion of our project supports the published studies about it, the accumulation of TAU in areas related to memory is important in the development of AD [45].

An early and accurate diagnosis of AD is important to start an effective treatment. Deep

learning techniques have shown great potential in this regard. On the other hand, the introduction of interpretability techniques represents another step towards a possible incorporation of this type of models in the clinical setting as diagnostic support tools.

Parte C

Índices

Índice de figuras

Índice de cuadros

Bibliografía

[1] P. Scheltens, B. De Strooper, M. Kivipelto, H. Holstege, G. Chételat, C. E. Teunissen, J. Cummings, and W. M. van der Flier, "Alzheimer's disease," *The Lancet*, vol. 397, no. 10284, pp. 1577–1590, 2021.

[2] "Dementia." https://www.who.int/news-room/fact-sheets/detail/dementia/, 2021.

[3] "Algunos cerebros resisten al daño inicial que produce el alzhéimer." https://www.abc.es/salud/noticias/20140915/abci-alzheimer-cerebro-resistencia-201409151415.html/, 2014.

[4] J. M. Long and D. M. Holtzman, "Alzheimer disease: an update on pathobiology and treatment strategies," *Cell*, vol. 179, no. 2, pp. 312–339, 2019.

[5] G. M. McKhann, D. S. Knopman, H. Chertkow, B. T. Hyman, C. R. Jack Jr, C. H. Kawas, W. E. Klunk, W. J. Koroshetz, J. J. Manly, R. Mayeux, *et al.*, "The diagnosis of dementia due to alzheimer's disease: Recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease," *Alzheimer's & dementia*, vol. 7, no. 3, pp. 263–269, 2011.

[6] A. Chandra, G. Dervenoulas, and M. Politis, "Magnetic resonance imaging in alzheimer's disease and mild cognitive impairment," *Journal of neurology*, vol. 266, no. 6, pp. 1293–1302, 2019.

[7] A. Nordberg, J. O. Rinne, A. Kadir, and B. Långström, "The use of pet in alzheimer disease," *Nature Reviews Neurology*, vol. 6, no. 2, pp. 78–87, 2010.

[8] C. V. Jie, V. Treyer, R. Schibli, and L. Mu, "Tauvid™: The first fda-approved pet tracer for imaging tau pathology in alzheimer's disease," *Pharmaceuticals*, vol. 14, no. 2, p. 110, 2021.

[9] H. Cho, J. Y. Choi, M. S. Hwang, Y. J. Kim, H. M. Lee, H. S. Lee, J. H. Lee, Y. H. Ryu, M. S. Lee, and C. H. Lyoo, "In vivo cortical spreading pattern of tau and amyloid in the alzheimer disease spectrum," *Annals of neurology*, vol. 80, no. 2, pp. 247–258, 2016.

[10] A. Samuel, "Eight-move opening utilizing generalization learning,(see appendix b, game g-43.1 some studies in machine learning using the game of checkers)," *IBM Journal*, pp. 210–229, 1959.

[11] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, eBook*. Pearson Higher Ed, 2021.

[12] H. Llinás Solano, *Estadística inferencial*. 2018.

[13] Q. Bi, K. E. Goodman, J. Kaminsky, and J. Lessler, "What is machine learning? a primer for the epidemiologist," *American journal of epidemiology*, vol. 188, no. 12, pp. 6–6, 2019.

[14] E. Stevens, L. Antiga, and T. Viehmann, *Deep learning with PyTorch*. Manning Publications, 2020.

[15] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021.

[16] N. J. Majaj and D. G. Pelli, "Deep learning—using machine learning to study biological vision," *Journal of vision*, vol. 18, no. 13, pp. 2–2, 2018.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[18] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[19] J. Islam and Y. Zhang, "A novel deep learning based multi-class classification method for alzheimer's disease detection using brain mri data," in *International conference on brain informatics*, pp. 213–222, Springer, 2017.

[20] M. Hon and N. M. Khan, "Towards alzheimer's disease classification through transfer learning," in *2017 IEEE International conference on bioinformatics and biomedicine (BIBM)*, pp. 1166–1169, IEEE, 2017.

[21] A. Valliani and A. Soni, "Deep residual nets for improved alzheimer's diagnosis," in *Proceedings of the 8th ACM international conference on bioinformatics, computational biology, and health informatics*, pp. 615–615, 2017.

[22] H. J. Son, J. S. Oh, M. Oh, S. J. Kim, J.-H. Lee, J. H. Roh, and J. S. Kim, "The clinical feasibility of deep learning-based classification of amyloid pet images in visually equivocal cases," *European Journal of Nuclear Medicine and Molecular Imaging*, vol. 47, no. 2, pp. 332–341, 2020.

[23] H. Choi, K. H. Jin, A. D. N. Initiative, *et al.*, "Predicting cognitive decline with deep learning of brain metabolism and amyloid imaging," *Behavioural brain research*, vol. 344, pp. 103–109, 2018.

[24] J. Díaz-Álvarez, J. A. Matias-Guiu, M. N. Cabrera-Martín, V. Pytel, I. Segovia-Ríos, F. García-Gutiérrez, L. Hernández-Lorenzo, J. Matias-Guiu, J. L. Carreras, J. L. Ayala, *et al.*, "Genetic algorithms for optimized diagnosis of alzheimer's disease and frontotemporal dementia using fluorodeoxyglucose positron emission tomography imaging," *Frontiers in aging neuroscience*, p. 983, 2022.

[25] Y. Ding, J. H. Sohn, M. G. Kawczynski, H. Trivedi, R. Harnish, N. W. Jenkins, D. Lituiev, T. P. Copeland, M. S. Aboian, C. Mari Aparici, *et al.*, "A deep learning model to predict a diagnosis of alzheimer disease by using 18f-fdg pet of the brain," *Radiology*, vol. 290, no. 2, pp. 456–464, 2019.

[26] I. Illán, J. M. Górriz, J. Ramírez, D. Salas-Gonzalez, M. López, F. Segovia, R. Chaves, M. Gómez-Rio, C. G. Puntonet, A. D. N. Initiative, *et al.*, "18f-fdg pet imaging analysis for computer aided alzheimer's diagnosis," *Information Sciences*, vol. 181, no. 4, pp. 903–916, 2011.

[27] M. Liu, D. Cheng, W. Yan, A. D. N. Initiative, *et al.*, "Classification of alzheimer's disease by combination of convolutional and recurrent neural networks using fdg-pet images," *Frontiers in neuroinformatics*, vol. 12, p. 35, 2018.

[28] T. Jo, K. Nho, S. L. Risacher, and A. J. Saykin, "Deep learning detection of informative features in tau pet for alzheimer's disease classification," *BMC bioinformatics*, vol. 21, no. 21, pp. 1–13, 2020.

[29] K. Oh, Y.-C. Chung, K. W. Kim, W.-S. Kim, and I.-S. Oh, "Classification and visualization of alzheimer's disease using volumetric convolutional neural network and transfer learning," *Scientific Reports*, vol. 9, no. 1, pp. 1–16, 2019.

[30] J. Zou, D. Park, A. Johnson, X. Feng, M. Pardo, J. France, Z. Tomljanovic, A. M. Brickman, D. P. Devanand, J. A. Luchsinger, *et al.*, "Deep learning improves utility of tau pet in the study of alzheimer's disease," *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring*, vol. 13, no. 1, p. e12264, 2021.

[31] M. Hengstler, E. Enkel, and S. Duelli, "Applied artificial intelligence and trust—the case of autonomous vehicles and medical assistance devices," *Technological Forecasting and Social Change*, vol. 105, pp. 105–120, 2016.

[32] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[33] M. Böhle, F. Eitel, M. Weygandt, and K. Ritter, "Layer-wise relevance propagation for explaining deep neural network decisions in mri-based alzheimer's disease classification," *Frontiers in aging neuroscience*, p. 194, 2019.

[34] T. Pohl, M. Jakab, and W. Benesova, "Interpretability of deep neural networks used for the diagnosis of alzheimer's disease," *International Journal of Imaging Systems and Technology*, 2021.

[35] S. Esmaeilzadeh, D. I. Belivanis, K. M. Pohl, and E. Adeli, "End-to-end alzheimer's disease diagnosis and biomarker identification," in *International Workshop on Machine Learning in Medical Imaging*, pp. 337–345, Springer, 2018.

[36] F. Eitel, E. Soehler, J. Bellmann-Strobl, A. U. Brandt, K. Ruprecht, R. M. Giess, J. Kuchling, S. Asseyer, M. Weygandt, J.-D. Haynes, *et al.*, "Uncovering convolutional neural network decisions for diagnosing multiple sclerosis on conventional mri

using layer-wise relevance propagation," *NeuroImage: Clinical*, vol. 24, p. 102003, 2019.

[37] D. Wood, J. Cole, and T. Booth, "Neuro-dram: a 3d recurrent visual attention model for interpretable neuroimaging classification," *arXiv preprint arXiv:1910.04721*, 2019.

[38] M. Dyrba, A. H. Pallath, and E. N. Marzban, "Comparison of cnn visualization methods to aid model interpretability for detecting alzheimer's disease," in *Bildverarbeitung für die Medizin 2020*, pp. 307–312, Springer, 2020.

[39] J. Rieke, F. Eitel, M. Weygandt, J.-D. Haynes, and K. Ritter, "Visualizing convolutional networks for mri-based diagnosis of alzheimer's disease," in *Understanding and Interpreting Machine Learning in Medical Image Computing Applications*, pp. 24–31, Springer, 2018.

[40] C. Yang, A. Rangarajan, and S. Ranka, "Visual explanations from deep 3d convolutional neural networks for alzheimer's disease classification," in *AMIA annual symposium proceedings*, vol. 2018, p. 1571, American Medical Informatics Association, 2018.

[41] B. N. Dugger and D. W. Dickson, "Pathology of neurodegenerative diseases," *Cold Spring Harbor perspectives in biology*, vol. 9, no. 7, p. a028035, 2017.

[42] F. Ren, C. Yang, Q. Qiu, N. Zeng, C. Cai, C. Hou, and Q. Zou, "Exploiting discriminative regions of brain slices based on 2d cnns for alzheimer's disease classification," *Ieee Access*, vol. 7, pp. 181423–181433, 2019.

[43] M. Raju, V. P. Gopi, V. Anitha, and K. A. Wahid, "Multi-class diagnosis of alzheimer's disease using cascaded three dimensional-convolutional neural network," *Physical and Engineering Sciences in Medicine*, vol. 43, no. 4, pp. 1219–1228, 2020.

[44] E. Yee, K. Popuri, M. F. Beg, and A. D. N. Initiative, "Quantifying brain metabolism from fdg-pet images into a probability of alzheimer's dementia score," *Human brain mapping*, vol. 41, no. 1, pp. 5–16, 2020.

[45] T. Jo, K. Nho, S. L. Risacher, and A. J. Saykin, "Deep learning detection of informative features in tau pet for alzheimer's disease classification," *BMC bioinformatics*,

vol. 21, no. 21, pp. 1–13, 2020.

[46] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *In Workshop at International Conference on Learning Representations*, Citeseer, 2014.

[47] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[48] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, pp. 1–46, 07 2015.

[49] D. T. Huff, A. J. Weisman, and R. Jeraj, "Interpretation and visualization techniques for deep learning models in medical imaging," *Phys Med Biol*, pp. 7–14, 2021.

[50] "Spm12 github." https://github.com/spm/spm12.

[51] M. C. Chirodea, O. C. Novac, C. M. Novac, N. Bizon, M. Oproescu, and C. E. Gordan, "Comparison of tensorflow and pytorch in convolutional neural network-based applications," in *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–6, IEEE, 2021.

[52] S. Imambi, K. B. Prakash, and G. Kanagachidambaresan, "Pytorch," in *Programming with TensorFlow*, pp. 87–104, Springer, 2021.

[53] T. E. Oliphant, *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.

[54] W. McKinney, "Pandas, python data analysis library," *URL http://pandas. pydata. org*, pp. 3–15, 2015.

[55] S. Tosi, *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[56] R. Garreta and G. Moncecchi, *Learning scikit-learn: machine learning in python*. Packt Publishing Ltd, 2013.

[57] "Alzheimer's disease neuroimaging initiative." adni.loni.usc.edu.

[58] V. Camus, P. Payoux, L. Barré, B. Desgranges, T. Voisin, C. Tauber, R. La Joie, M. Tafani, C. Hommet, G. Chételat, *et al.*, "Using pet with 18f-av-45 (florbetapir) to

quantify brain amyloid load in a clinical environment," *European journal of nuclear medicine and molecular imaging*, vol. 39, no. 4, pp. 621–631, 2012.

[59] F. G. Gutiérrez, "Medical image tools." https://github.com/FernandoGaGu/Medical-Image-Tools.

[60] S. Bansari, "Introduction to how cnns work." https://medium.datadriveninvestor.com/introduction-to-how-cnns-work-77e0e4cde99b, 2019.

[61] J. Brownlee, "Introduction to the rectified linear unit (relu)." https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/, 2019.

[62] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* .ºReilly Media, Inc.", 2019.

[63] "Datasets and dataloaders." https://pytorch.org/tutorials/beginner/basics/data_tutorial.html/.

[64] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[65] E. Stevens, L. Antiga, and T. Viehman, *Deep learning with Pytorch.* "Manning", 2020.

[66] "Bceloss." https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html/.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[68] J. Martínez Llamas, "Reconocimiento de imágenes mediante redes neuronales convolucionales," 2018.

[69] "Captum website." https://captum.ai/.

[70] "Captum api reference." https://captum.ai/api/.

[71] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, pp. 2–8, 2013.

[72] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014.

[73] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*, pp. 3319–3328, PMLR, 2017.

[74] R. Sayres, A. Taly, E. Rahimy, K. Blumer, D. Coz, N. Hammel, J. Krause, A. Narayanaswamy, Z. Rastegar, D. Wu, *et al.*, "Using a deep learning algorithm and integrated gradients explanation to assist grading for diabetic retinopathy," *Ophthalmology*, vol. 126, no. 4, pp. 552–564, 2019.

[75] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

[76] S. Liu, C. Yadav, C. Fernandez-Granda, and N. Razavian, "On the design of convolutional neural networks for automatic detection of alzheimer's disease," in *Machine Learning for Health Workshop*, pp. 184–201, PMLR, 2020.

[77] H. Eichenbaum, A. P. Yonelinas, and C. Ranganath, "The medial temporal lobe and recognition memory," *Annu. Rev. Neurosci.*, vol. 30, pp. 123–152, 2007.

[78] M. Schöll, S. N. Lockhart, D. R. Schonhaut, J. P. O'Neil, M. Janabi, R. Ossenkoppele, S. L. Baker, J. W. Vogel, J. Faria, H. D. Schwimmer, *et al.*, "Pet imaging of tau deposition in the aging human brain," *Neuron*, vol. 89, no. 5, pp. 971–982, 2016.

[79] D. Berron, D. van Westen, R. Ossenkoppele, O. Strandberg, and O. Hansson, "Medial temporal lobe connectivity and its associations with cognition in early alzheimer's disease," *Brain*, vol. 143, no. 4, pp. 1233–1248, 2020.

[80] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot, "Automated anatomical labeling of activations

in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain," *Neuroimage*, vol. 15, no. 1, pp. 273–289, 2002.

[81] Q. Zhao, M. Liu, L. Ha, Y. Zhou, M. W. Weiner, P. Aisen, M. Weiner, R. Petersen, C. R. Jack Jr, W. Jagust, *et al.*, "Quantitative 18f-av1451 brain tau pet imaging in cognitively normal older adults, mild cognitive impairment, and alzheimer's disease patients," *Frontiers in Neurology*, vol. 10, p. 486, 2019.

[82] "Raytune." https://docs.ray.io/en/latest/tune/index.html.

[83] A. Jana, "Applying gaussian smoothing to an image using python from scratch." http://www.adeveloperdiary.com/data-science/computer-vision/applying-gaussian-smoothing-to-an-image-using-python-from-scratch/.

[84] J. Klein, X. Yan, A. Johnson, Z. Tomljanovic, J. Zou, K. Polly, L. S. Honig, A. M. Brickman, Y. Stern, D. Devanand, S. Lee, and W. C. Kreisl, "Olfactory impairment is related to tau pathology and neuroinflammation in alzheimer's disease," *Journal of Alzheimer's Disease*, vol. 80, pp. 10–13, 2021.

[85] A. Maass, S. Landau, S. L. Baker, A. Horng, S. N. Lockhart, R. La Joie, G. D. Rabinovici, W. J. Jagust, A. D. N. Initiative, *et al.*, "Comparison of multiple tau-pet measures as biomarkers in aging and alzheimer's disease," *Neuroimage*, vol. 157, pp. 448–463, 2017.

[86] N. Buduma and N. Locascio, *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. .O'Reilly Media, Inc.", 2017.

Álvaro Corrochano López, Pablo Álvarez García y Antonio Fernández Martín

acorroch@ucm.es pabloa06@ucm.es antofe10@ucm.es

Junio 2022

Últ. actualización 1 de junio de 2022