UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Clasificación de imágenes de resonancia magnética cerebral mediante redes neuronales para el diagnóstico médico

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Álvaro López Chilet

Tutor: Jon Ander Gómez Adrián
María de la Iglesia Vayá

Curso 2018-2019

Agradecimientos

En primer lugar me gustaría agradecer la implicación en el proyecto y el apoyo de mis dos tutores Jon Ander Gómez Adrián y María de la Iglesia Vayá. Ya que han sido unos excelentes mentores que me han aportado mucho durante el desarrollo de este trabajo.

También agradecer a mis compañeros de la carrera Rafael Vicente Sánchez Romero y Silvia Nadal Almela; a las personas del Centro de Investigación Príncipe Felipe (CIPF), con especial mención a José Manuel Saborit Torres, por su aportación al trabajo y gran compañía que han hecho más agradable el día a día en el centro.

Finalmente dar las gracias a NVIDIA por la tarjeta gráfica QUADRO P5000 donada al CIPF. Y a la conselleria por los fondos FEDER con los que se ha podido adquirir el cluster del CIPF con el que también hemos podido trabajar.

Resum

El problema que s'aborda en este treball és el d'usar tècniques de aprenentatge profund per a etiquetar imatges de ressonància magnètica cerebral en distints graus de la malaltia d'Alzheimer: deteriorament cognitiu lleu, deteriorament cognitiu mig, deteriorament cognitiu greu i finalment Alzheimer; a més de detectar el cas de no malaltia. És important detectar les primeres fases del deteriorament cognitiu ja que l'Alzheimer és una malaltia sense cura de moment i l'única cosa que es pot fer és diagnosticar-la com més prompte millor per a poder disminuir al màxim el seu impacte a llarg termini per mitjà d'alguns tractaments.

Per a això, primer s'estudiaran les ferramentes i tècniques de preprocessat utilitzades actualment per a traure el major partit a les imatges. A continuació es revisaran els models i estratègies seguits per l'estat de l'art. Per a finalment proposar algunes topologies de xarxes neuronals noves que aborden el problema des d'un nou punt de vista.

Paraules clau: Alzheimer, aprenentatge profund, xarxes neuronals, ressonància magnètica cerebral

Resumen

El problema que se aborda en este trabajo es el de usar técnicas de aprendizaje profundo para etiquetar imágenes de resonancia magnética cerebral en distintos grados de la enfermedad de Alzheimer: deterioro cognitivo leve, deterioro cognitivo medio, deterioro cognitivo grave y finalmente Alzheimer; además de detectar el caso de no enfermedad. Es importante detectar las primeras fases del deterioro cognitivo ya que el Alzheimer es una enfermedad sin cura por el momento y lo único que se puede hacer es diagnosticarla lo antes posible para poder disminuir al máximo su impacto a largo plazo mediante algunos tratamientos.

Para ello, primero se estudiarán las herramientas y técnicas de preprocesado utilizadas actualmente para sacar el mayor partido a las imágenes. Seguidamente se revisarán los modelos y estrategias seguidos por el estado del arte. Para finalmente proponer algunas topologías de redes neuronales nuevas que aborden el problema desde un nuevo punto de vista.

Palabras clave: Alzheimer, aprendizaje profundo, redes neuronales, resonancia magnética cerebral

Abstract

The problem addressed in this work is to use deep learning techniques to label brain magnetic resonance images in different degrees of Alzheimer's disease: mild cognitive impairment, medium cognitive impairment, severe cognitive impairment and finally Alzheimer; besides detecting the case of no disease. It is important to detect the first phases of cognitive deterioration since Alzheimer is a disease without cure at the moment and the only thing that can be done is to diagnose it as soon as possible in order to minimize its long-term impact through some treatments.

For this purpose, first, the preprocessing tools and techniques currently used will be studied to get the most out of the images. Then we will review the models and strategies followed by the state of the art. To finally propose some new neural network topologies that approach the problem from a new point of view.

Key words: Alzheimer, deep learning, neural network, brain magnetic resonance

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

El Alzheimer es una enfermedad actualmente sin cura y con un gran impacto en la sociedad. Algunas cifras que arroja son: 46 millones de personas afectadas en todo el mundo; aproximadamente 1 de cada 10 personas mayores de 65 la sufre y el pronostico es que para el año 2050 estos números se hayan triplicado si no se encuentra una cura[2].

Actualmente lo único que se puede hacer es diagnosticar la enfermedad lo antes posible para poder aplicar algunos tratamientos que disminuyan el impacto de la enfermedad y se haga más llevadera a largo plazo.

Nuestro trabajo propone utilizar técnicas *deep learning* y preprocesado de imagen con las que a partir de imágenes de resonancia magnética cerebral se consiga diagnosticar el nivel de la enfermedad que tiene el paciente. Para poder agilizar el tiempo que tarda en diagnosticarse y poder empezar cuanto antes los tratamientos.

1.1 Motivación

La motivación de este trabajo es ayudar a los médicos en la toma de decisiones al realizar un diagnóstico de la enfermedad del Alzheimer para que se realice lo antes posible. Ya que por el número de pruebas que se pueden llegar a hacer y la complejidad de algunas de ellas[1], el proceso de diagnóstico puede complicarse y durar mucho tiempo. Afectando negativamente a los pacientes que sufren la enfermedad ya que debe ser diagnosticada lo antes posible para poder empezar con los tratamientos antes de que empeoren los síntomas.

1.2 Objetivos

Los objetivos de este trabajo son:

- Estudiar las técnicas de *deep learning* y procesamiento de imagen utilizadas en imágenes de resonancia magnética.
- Comprobar la viabilidad de diagnosticar la enfermedad del Alzheimer a partir de IRM.[2]
- Replicar las técnicas de preprocesado y modelos de redes neuronales utilizados en el estado del arte para posteriormente intentar mejorar sus resultados.

[1] El número de pruebas es variable según el estado del paciente. Se pueden consultar en [1].
[2] Imágenes de resonancia magnética.

1.3 Estructura de la memoria

Primero, se realizará un estudio del estado del arte. Se verán algunas de las propuestas con mayor porcentaje de acierto en el diagnóstico y se comentaran sus resultados. Seguidamente, se hará una descripción detallada del problema junto a nuestra solución propuesta. Y finalmente se mostrarán los resultados de la experimentación y las conclusiones sacadas al respecto.

Al final del documento se puede encontrar un apéndice con la explicación de algunos términos y conceptos utilizados.

1.4 Colaboraciones

Este proyecto ha sido desarrollado en un entorno de prácticas de 4 meses en el Centro de Investigación Príncipe Felipe junto a dos compañeros de la carrera: Silvia Nadal Almela y Rafael Vicente Sánchez Romero.

La primera parte del proyecto ha sido realizada en conjunto; donde se ha analizado el problema y los datos que teníamos para abordarlo, investigado el estado del arte, visto las posibles formas de preprocesar los datos y algunos experimentos iniciales con redes neuronales. A partir de aquí cada uno se ha centrado en un tipo de experimentación según el tipo de procesado de los datos utilizado: Silvia ha estudiado el problema desde el uso de datos de volumetría extraídos con FreeSurfer de las imágenes, Rafael Vicente ha experimentado con la clasificación mediante cortes 2D extraídos de las imágenes tridimensionales preprocesadas con herramientas de extracción del cerebro como BET y yo me he centrado en la clasificación mediante las imágenes 3D aplicando las herramientas BET y FLIRT para el preprocesado.

CAPÍTULO 2
Estado del arte

En la literatura se han visto diversas estrategias para abordar este problema de clasificación mediante el uso de redes neuronales. Variando sobretodo en la manera de procesar y seleccionar los datos. Por ello, para facilitar la explicación de estas técnicas se van a separar en dos apartados. El primero de datos y preprocesado donde se hablará de las herramientas y planteamientos que se usan para extraer los datos de mayor interés de las imágenes y como se tratan para mejorar el entrenamiento. Y segundo, la parte de los modelos donde se verán las técnicas y topologías de redes neuronales convolucionales (CNN)[1] usadas para la extracción de características de las imágenes y su clasificación.

2.1 Datos y preprocesado

Los estudios abordan este problema principalmente con IRM de tipo T1. Este tipo de imagen está caracterizado por mostrar más brillantes los tejidos sólidos y más oscuras las zonas que contengan líquidos. Por lo que es muy útil para poder apreciar atrofias a nivel estructural del cerebro. Pero además, algunas publicaciones utilizan las imágenes de tipo PET[2] [18, 9]. Que a diferencia de las imágenes T1 que son de tipo estructural, las PET son de carácter funcional, ya que muestran el funcionamiento del cerebro captando la actividad metabólica celular con la que se ha visto que los pacientes con Alzheimer presentan un patrón característico [14].

A partir de estas imágenes se suele intentar extraer las zonas de interés para reducir la cantidad de datos que introducen ruido a la red y para disminuir las dimensiones de las imágenes, cosa que permite acelerar la rapidez del entrenamiento de las redes o hacer modelos más grandes; ya que al trabajar con imágenes en 3D se necesita de una gran capacidad de memoria en las GPUs para poder entrenar modelos con suficiente capacidad.

Esta extracción de las zonas de interés se suele hacer con las siguientes herramientas[3]:

- BET: *Brain Extraction Tool* es utilizada en [11, 16] para eliminar de la imagen los tejidos que no son parénquima cerebral. Como el cráneo y el cuello que no aportan información relevante para el problema.
- FLIRT: *FMRIB's Linear Image Registration Tool* utilizada en [11, 9] para transformar las imágenes a un mismo espacio de representación y que todas estén en la misma

[1] *Convolutional neural network*. La explicación de este concepto está en el apéndice A

[2] *Positron Emission Tomography.*

[3] Se encuentran explicadas en más detalle en el apartado "Descripción de las herramientas software y hardware utilizadas"

posición y tamaño. Así se pueden obtener las imágenes con las mismas dimensiones, ya que originalmente el tamaño y proporción del conjunto de datos original son variantes y para introducirlas en la red han de ser iguales en todas las dimensiones de la imagen.

- FreeSurfer: Con esta herramienta se pueden extraer segmentaciones por parcelas siguiendo diferentes modelos del cerebro con distintas parcelaciones. Además saca los datos de volumetría de las parcelas, como el grosor y la superficie. Se han visto trabajos como [9] que usan las segmentaciones para extraer múltiples zonas de interés e introducirlas en la red de forma separada. Y otros como [8] que utilizan los datos numéricos de volumetría para entrenar *random forests*.

Otra técnica utilizada en algunos trabajos como [18, 6] es la de extracción del hipocampo que se ha visto que es una de las primeras zonas en verse afectadas por el atrofiamiento. Y se realiza mediante la búsqueda del centro de la cabeza y realizando a partir de ahí un recorte da la zona de interés.

En general, todas estas técnicas nombradas buscan limpiar los datos para reducir la información ruidosa que se introduce a las redes y que pueden alterar el aprendizaje de estas. Además de suponer una mejora en la velocidad y eficiencia de los entrenamientos ya que se reduce la dimensionalidad y en consecuencia la cantidad de operaciones a realizar.

Una estrategia poco utilizada en la literatura consultada es el *data augmentation*. Que consiste en realizar transformaciones a las imágenes para generar otras nuevas, aumentando así artificialmente el tamaño del conjunto de datos. Normalmente estas transformaciones son: rotaciones, traslaciones, zooms, cambios en el brillo, deformaciones, etc. El único trabajo que hemos encontrado que propone el uso de *data augmentation* es el [6]. Donde utilizan las siguientes estrategias:

- *Blurring*: Aplican un suavizado intentando imitar posibles variaciones de contraste. Y es realizado mediante la aplicación de filtros de ruido gaussiano de dimensiones 3x3, 5x5 y 7x7; y unos valores de desviación estándar σ = 0.7, 0.7 y 0.6 respectivamente.
- Traslación: Mueven la imagen de su región de interés extraída, que es el hipocampo, 1 píxel en cada dimensión. Obteniendo así 6 imágenes adicionales.
- *Flipping*: Ya que el hipocampo es una estructura simétrica del cerebro, se le aplica un volteo respecto a su eje de simetría. En caso de utilizar todo el cerebro esta técnica no seria válida ya que los hemisferios izquierdo y derecho del cerebro no son iguales.

2.2 Modelos

Para la clasificación de estas imágenes los trabajos revisados usan principalmente redes neuronales convolucionales debido a su gran potencial en tareas de clasificación y reconocimiento de imagen. A continuación vamos a comentar algunas de las técnicas y variantes de CNNs vistas en la literatura.

En la mayoría de estudios se utilizan redes convolucionales convencionales, con bloques de convolución más pooling y finalmente una red *fully connected*[4]. Algunas variantes más complejas que se han visto son las de [9, 11].

[4]Red neuronal convencional con capas de neuronas densamente conectadas entre ellas.

En [9] se utiliza una red multimodal que consiste en tener varias entradas a la red donde cada una pasará por sus propias capas convolucionales para finalmente concatenar las matrices resultantes antes de entrar en la *fully connected* que realiza la clasificación a partir de las características extraídas. Esto hace que la red pueda dedicar toda una rama de convoluciones a cada una de las diferentes entradas para así poder extraer características más concretas de cada una de ellas. En este estudio disponían de diferentes regiones de interés que habían extraído mediante la segmentación con FreeSurfer, de esta forma dedicaban cada una de las entradas para una zona de interés. Así podían hacer un entrenamiento de grano más fino respecto a cada una de las entradas pudiendo extraer mejor las características de cada una de ellas.

En [11] utilizan bloques convolucionales densamente conectados. Estos están formados por capas convolucionales cuyas salidas están conectadas a cada una de las capas siguientes [10], como se puede ver en la figura 2.1.

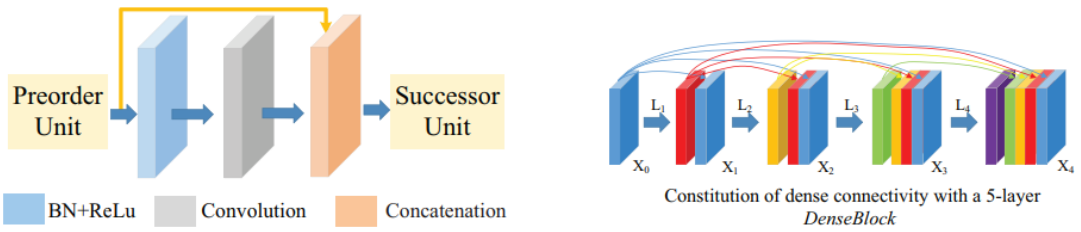

Figura 2.1: En la figura de la izquierda se pueden ver las conexiones de un bloque convolucional densamente conectado. Y a la derecha se muestra el flujo de los tensores para la combinación de 5 de estos bloques. Estas imágenes han sido extraídas de la publicación [11].

Con esto se consigue mejorar la conectividad de la red y por lo tanto una mejor retroprogación del error con la que se reduce el problema de *vanishing-gradient*.[5] Pero estos bloques aumentan mucho las dimensiones de los tensores debido a las múltiples concatenaciones que se realizan, por lo que para reducir la dimensionad, la imagen de entrada pasa por una capa convolucional con el *stride* de 2x2x2 para reducir el tamaño a un cuarto. Además, tras pasar por cada bloque densamente conectado se aplica una capa convolucional con kernels de 1x1x1 y otra capa de *pooling* para reducir todas las dimensiones del tensor resultante de cada bloque. Se puede ver el esquema de esta red en la figura 2.2.

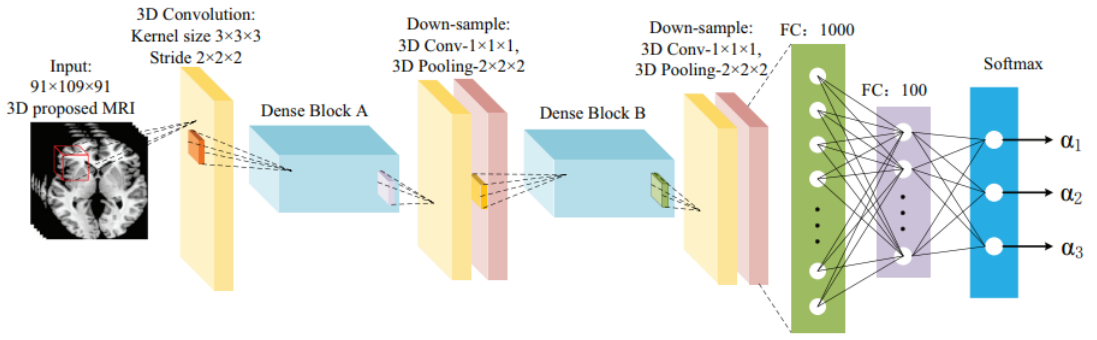

Figura 2.2: Esta imagen ha sido extraída de la publicación [11].

Además, en este estudio se entrenan 10 modelos con la misma topología mediante validación cruzada para finalmente hacer una clasificación por votación entre todos ellos para el conjunto de test. Obteniendo así los mejores resultados vistos y que se muestran en la tabla 2.1.

[5] Descripción de este concepto en el apéndice D

Tabla 2.1: Porcentajes de acierto de la propuesta de [11] para diferentes combinaciones de etiquetas.

Etiquetas	Precisión (%)
MCI/AD	93.61
AD/CN	98.83
MCI/CN	98.42
AD/MCI/CN	97.52

Finalmente, una técnica muy utilizada[6] es el pre entrenamiento de los bloques convolucionales mediante autoencoders[7]. Para ello, primero se entrena un autoencoder que aprenda a reconstruir las imágenes, esto hace que las capas convolucionales aprendan los *kernels* para reconocer características de ese tipo de imágenes ya que la parte del encoder ha de crear una representación compacta de la imagen en el cuello de botella de la red para que luego el decoder pueda reconstruir la imagen lo mejor posible. Una vez acabado el entrenamiento se extraen las capas del encoder para utilizarlas como bloque convolucional de la red que vamos a utilizar para la clasificación. El entrenamiento de esta red con pesos pre entrenados se hace en dos pasos. Primero se entrena con los pesos del encoder pre entrenado congelados, para que sólo aprenda primero los pesos restantes y tras este paso se descongelan los pesos del autoencoder para realizar un entrenamiento de grano fino que acabe de ajustar los pesos de toda la red en conjunto.

2.3 Crítica al estado del arte

Una parte muy importante a la hora de abordar estos problemas es la forma de hacer el particionado de los datos en los conjuntos de entrenamiento, validación y test. Ya que no siempre es válida una partición aleatoria de las muestras para separar los datos en estos tres conjuntos. En el caso del problema que se está abordando en este trabajo, este tipo de partición aleatoria no es válida ya que hay muestras que comparten características que pueden ser aprovechadas por la red para hacer "trampas"durante el entrenamiento y sacar resultados que aparentan ser correctos pero en realidad no está generalizando para la tarea que se les está queriendo entrenar.

En este caso se trata de que hay pacientes con varias imágenes de resonancia magnética y una partición aleatoria podría hacer que sus imágenes queden separadas en estas tres particiones. Por lo que la red puede aprender características particulares de ese paciente en lugar de patrones del deterioro cognitivo con los que poder generalizar. Y aun así sacaría buenos resultados en las particiones de validación y test ya que al volver a ver el cerebro del paciente que ha memorizado lo clasificará correctamente.

Para verificar esto hemos entrenado un mismo modelo con dos particiones distintas, una aleatoria y otra a nivel de sujeto para que todas las imágenes de un mismo sujeto sólo puedan aparecer en una partición. Los resultados obtenidos son totalmente diferentes, para el modelo con la partición aleatoria se ha obtenido una precisión entorno al 99 % para todas las particiones como se puede ver en la figura 2.3, mientras que el modelo correctamente particionado ha acabado haciendo un sobreentrenamiento en la partición de *training* (entrenamiento) mientras que en *development* (validación) y test se ha quedado estancado cerca del 70 %, figura 2.4.

[6] Usada en [7, 9]
[7] La explicación de este concepto se encuentra en B.

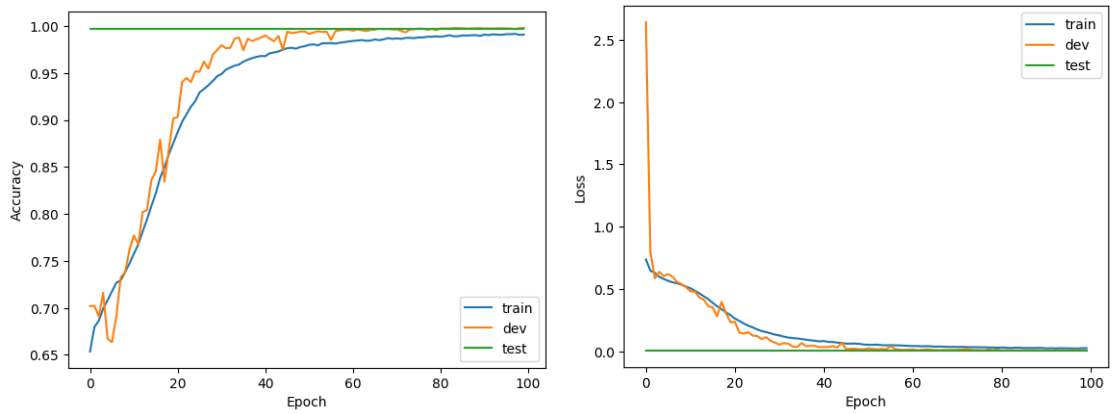

Figura 2.3: Resultados de clasificación para una red entrenada con un particionado incorrecto separando las imágenes de forma aleatoria.

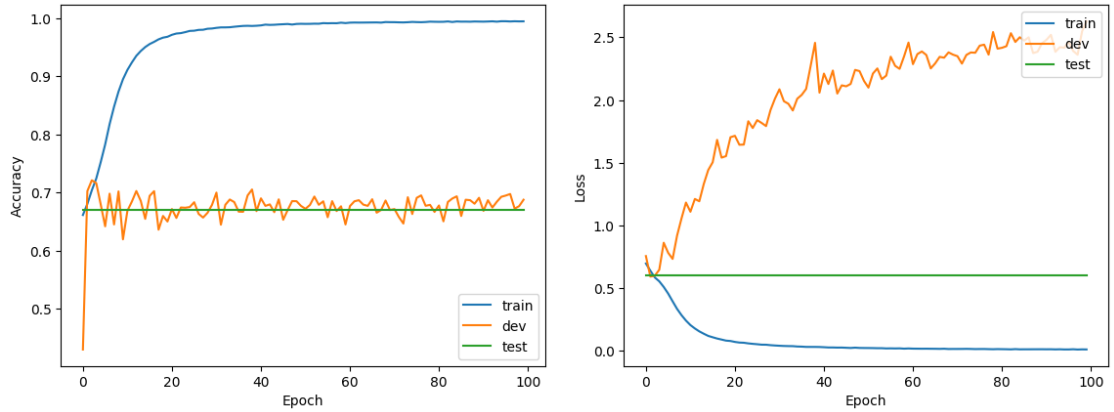

Figura 2.4: Resultados de clasificación para una red entrenada con un particionado del dataset a nivel de sujeto.

Esto es debido a que con la partición correctamente realizada la red no puede generalizar bien aprendiéndose solamente los sujetos de memoria, por lo que al ver nuevos de estos en *development* y test los resultados no son tan buenos. Por lo que si se llegara a implementar y poner en producción el modelo con una partición aleatoria, los resultados de clasificación con casos nuevos no serian buenos.

En las publicaciones hemos visto pocos casos que especifiquen con claridad como han hecho la partición y que esta sea correcta, estos han sido [11, 18]. Otros casos en los que no se han especificado como se ha realizado la partición son [7, 9, 6], por lo que sus resultados podrían ser erróneos en caso de una mala partición. Y finalmente se encontró una publicación [16] donde se especificaba que la partición era aleatoria, por lo tanto no se está haciendo uso de buenas prácticas pudiendo llevar a resultados no válidos.

El objetivo de esta crítica es destacar esta mala práctica con el particionado y la falta de detalle en la explicación de algunas publicaciones que puede llevar a resultados aparentemente correctos y que pueden pasar desapercibidos.

CAPÍTULO 3
Descripción del problema y solución propuesta

El problema que nos encontramos al abordar este trabajo es la clasificación de imágenes de resonancia magnética cerebral para realizar un diagnóstico de la enfermedad del Alzheimer. Esta clasificación puede ser en 5 clases que pertenecen a 4 niveles del desarrollo de la enfermedad de más grave a menos y el caso de control: AD (*Alzheimer's Disease*), LMCI (*Late Mild Cognitive Impairment*), MCI (*Mild Cognitive Impairment*), EMCI (*Earlier Mild Cognitive Impairment*) y CN (*Cognitive Normal*).[1]

Gracias a esto se podría ayudar a los médicos a realizar diagnósticos más rápido, y con mayor precisión. Cosa que es crucial de cara a preparar los tratamientos ya que el Alzheimer no tiene cura sino que existen formas de disminuir su impacto, pero para ello hay que detectar la enfermedad lo antes posible para poder empezar los tratamientos en las fases iniciales y obtener así mejores resultados a largo plazo.

Para resolver esta tarea se propone utilizar reconocimiento de imagen mediante redes neuronales para detectar biomarcadores en las imágenes que permitan clasificar a los sujetos. Uno de los biomarcadores conocidos del Alzheimer es la detección de atrofias en las estructuras cerebrales. Algunas de ellas ya empiezan a verse afectadas en fases tempranas del desarrollo de la enfermedad. Por ello, gracias a las imágenes de resonancia magnética podemos obtener información estructural útil para abordar este problema de diagnóstico incluso en el principio del desarrollo de la enfermedad.

3.1 Datos y preprocesado

Trabajar con imágenes de resonancia magnética presenta ciertas dificultades para preparar bien los datos antes de utilizarlos. Debido a que pueden contener información que no es de interés como el cuello en el caso de las imágenes de ADNI. Por lo que hay que desarrollar alguna técnica para extraer las zonas de interés o utilizar una herramienta de segmentación de IRM como FreeSurfer o BET con la que poder hacer una extracción más precisa.

Nosotros hemos optado por utilizar BET para extraer el parénquima cerebral y así poder eliminar todo el tejido que no es de interés. FreeSurfer ha sido utilizado en el trabajo de la compañera Silvia Nadal Almela para extraer datos de volumetría de las imágenes y realizar la clasificación a partir de datos numéricos en lugar de imágenes.

[1] Estas categorías vienen dadas por el dataset utilizado que es ADNI.

Además del BET, también se ha utilizado otra herramienta llamada FLIRT con la que se han transformado todas las imágenes para llevarlas al mismo espacio de representación. Con esto se consigue tener todas las imágenes en la misma posición y forma, perdiendo así la variabilidad respecto a características personales de cada sujeto como la forma de la cabeza o la posición de esta durante la adquisición de la resonancia magnética.

Con la combinación de estas herramientas hemos obtenido un dataset con mucha menos información ruidosa para que las redes puedan captar más fácilmente los patrones de las imágenes que generalizan mejor para esta tarea.

La descripción más detallada de las herramientas comentadas anteriormente se encuentra en el apartado de descripción de las herramientas software y hardware utilizadas.

Debido al desbalanceo de las etiquetas en las 5 clases[2] y a la poca diferencia que hay en cuestión de atrofiamiento para las clases con un deterioro intermedio: MCI, EMCI y LMCI. El problema se ha reducido a una clasificación en dos etiquetas: AD y CN. Ya que con un mayor número de etiquetas aumenta mucho la dificultad del problema y las redes no consiguen aprender a clasificar correctamente.

3.2 Modelos propuestos

De todos los modelos con los que se ha experimentado se proponen a continuación los dos que mejores resultados han obtenido. Estos se han utilizado con diferentes tipos de preprocesado: sólo con BET y con BET más FLIRT. Y los resultados de estos se encuentran descritos en el apartado de experimentación y resultados.

3.2.1. Modelo 1

Este modelo consta de una primera capa de convolución que utiliza *kernels* de 5x5x5 para extraer características de mayor tamaño de las imágenes de entrada. A partir de esta se utilizan 3 bloques de 2 capas convolucionales con *kernels* de 3x3x3 más una capa de *average pooling* con los que se consigue reducir el tamaño de los tensores extrayendo las características de mayor interés. La salida de estos bloques pasa por otras 3 capas convolucionales con *kernels* de 3x3x3 cuya salida se le aplica una operación de *flatten* para convertir la matriz de salida en un vector unidimensional para la entrada a la red *fully connected* que realiza la clasificación que está formada por dos capas: 1 de 100 neuronas y la de salida con 2.[3] La red completa se puede ver en la figura 3.1

Todas las capas de convolución de la red y la primera *fully connected* tienen *batch normalization* para evitar el *overfitting*. La función de activación de todas las capas es lineal mientras que la última utiliza una *softmax* para que la predicción de la red sea una distribución de probabilidad entre las dos etiquetas.[4]

3.2.2. Modelo 2

El segundo modelo también utilizan *kernels* de 5x5x5 pero en lugar de sólo en la primera capa los aplica en las 4 primeras. Esto es debido a que este modelo no utiliza capas

[2] Estos valores se pueden consultar en la tabla 5.1.
[3] Una para cada etiqueta: AD Y CN
[4] La suma de los dos valores resultantes es 1.

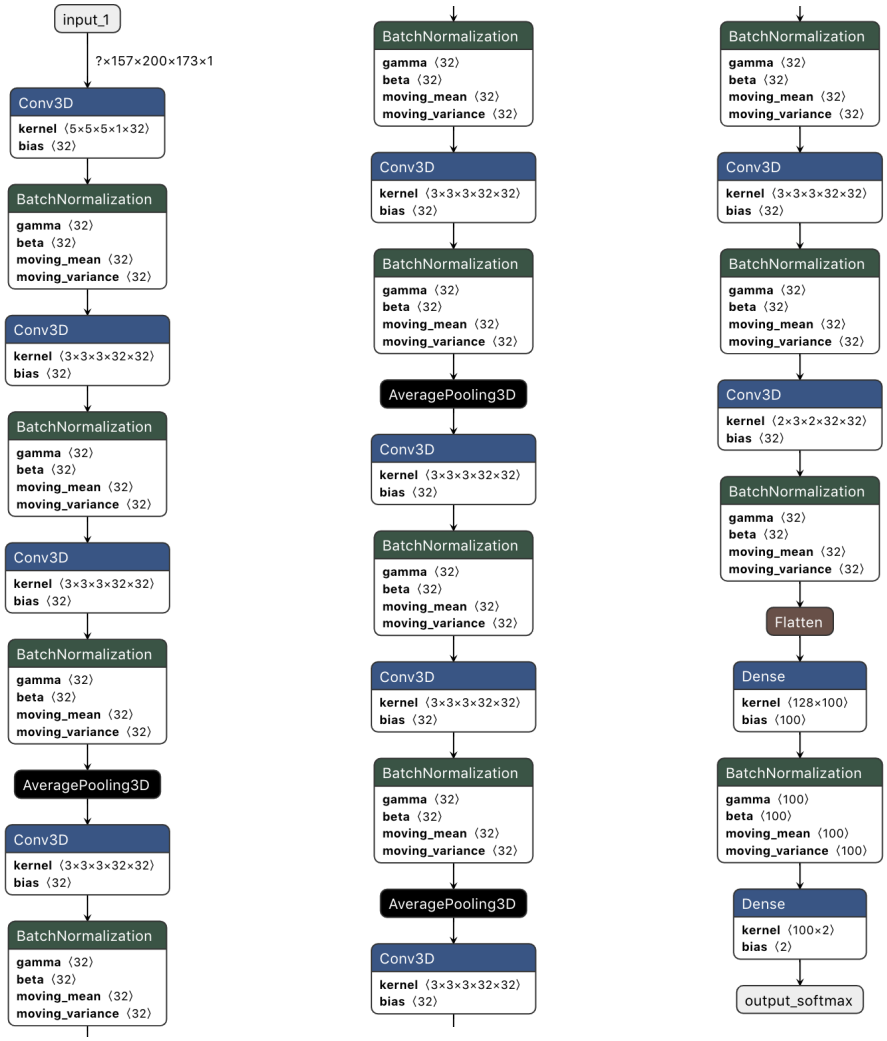

Figura 3.1: Topología del modelo 1 propuesto.

de *pooling* para reducir las dimensiones, sino que se utiliza un *stride* de 2x2x2 en las convoluciones que reduce el tamaño a un cuarto[5]. Debido a esto se utilizan *kernels* más grandes para intentar perder la mínima información espacial posible. Después de estas capas se pasa por 3 capas convolucionales más pero esta vez con *kernels* de 3x3x3 debido a que las dimensiones ya están muy reducidas. Y a la matriz resultante se le aplica la operación de *flatten* y pasa a la red *fully connected* formada por 4 capas: las 3 primeras con 100 neuronas y la de salida con 2. La red completa se puede ver en la figura 3.2

Al igual que la red anterior se utiliza *batch normalization* como método de regularización y las activaciones de todas las capas son lineales excepto la última que es *softmax*.

La característica principal de este modelo respecto a una red convolucional más tradicional es que no utiliza la operación *pooling* para reducir las dimensiones de los tensores. Sino que se utiliza convoluciones con *strides* de 2x2x2 en su lugar.

[5]Esta es la misma reducción que aplica una capa de *pooling* con una ventana de 2x2x2.

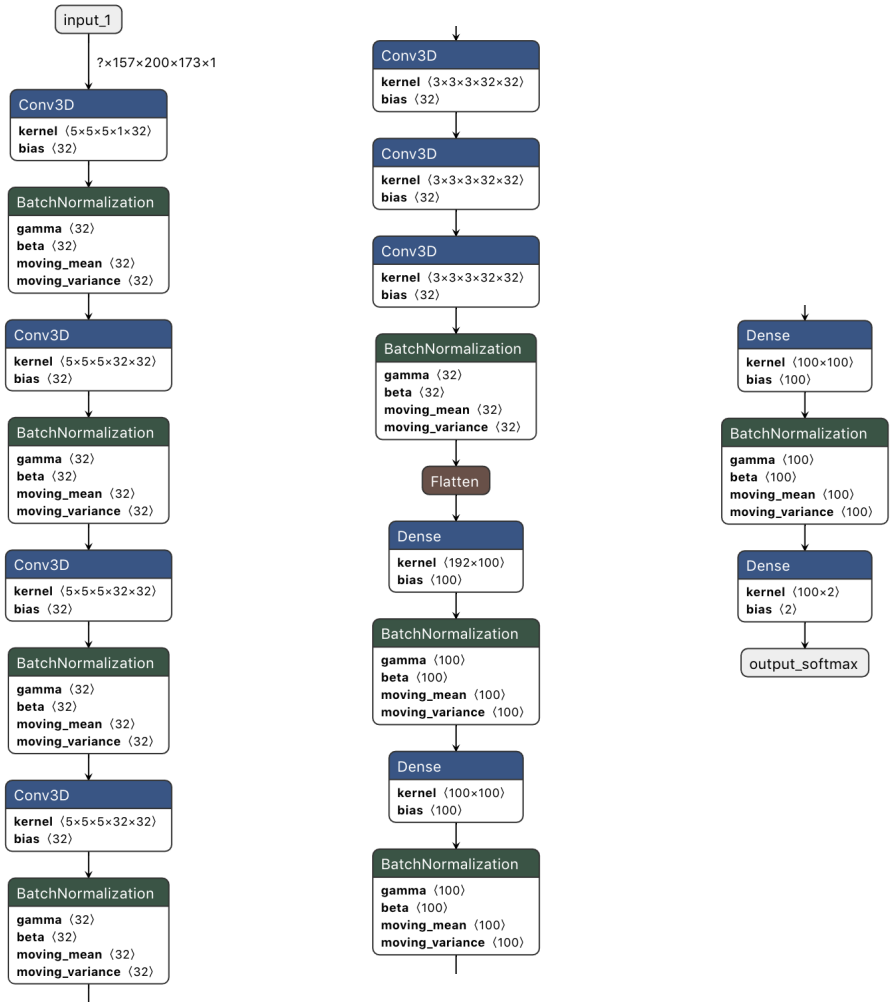

Figura 3.2: Topología del modelo 2 propuesto.

3.3 Validación cruzada

Al igual que en [11] nosotros hemos realizado la evaluación del test mediante validación cruzada. Esta se ha realizado a partir del modelo 2 ya que es el que mejores resultados nos ha proporcionado.

Para una mayor precisión en la evaluación de los resultados se ha realizado la validación cruzada para 5 particiones distintas del dataset. Y en cada una de ellas se ha entrenado 10 modelos del tipo 2 propuesto para realizar la clasificación de la partición de test por votación.

El sistema de votación aplicado es el utilizado en [11]. Y consiste en realizar la votación dándole más peso a las probabilidades más altas que aparecen en el vector de salida de la *softmax*. La fórmula que representa este sistema de votación se puede ver en la ecuación 3.1, donde y es la clase resultante de la votación; m el número de modelos entrenados; α_0 y α_1 los componentes del vector de salida de la red.

$$y = argmax(\prod_{i=0}^{m-1} \alpha_0^i, \prod_{i=0}^{m-1} \alpha_1^i) \tag{3.1}$$

CAPÍTULO 4
Descripción de las herramientas software y hardware utilizadas

4.1 Software

El lenguaje de programación utilizado para el desarrollo de todos los experimentos ha sido python, debido a múltiples razones:

- Es un lenguaje de alto nivel que nos permite realizar los experimentos centrándonos más en el problema que en detalles de implementación.
- Gran popularidad entre la comunidad y sobretodo en el campo del *deep learning*. Que pone a nuestro alcance una gran cantidad de tutoriales y resoluciones a problemas que podamos encontrar.
- Dispone de Jupyter Notebook, una herramienta que saca partido a la característica de python de ser un lenguaje interpretado y no compilado para crear archivos donde puedes ir ejecutando lineas de código por bloques (a modo de interprete de python) e ir guardando las salidas en el mismo archivo. Esto es de gran utilidad a la hora de analizar los datos ya que podemos ir sacando gráficas y estadísticas poco a poco de una manera más dinámica.

A continuación se explicarán las herramientas software utilizadas junto a python.

4.1.1. Keras y TensorFlow

Para el desarrollo de las redes neuronales hemos utilizado Keras, una API que puede funcionar por encima de varios *frameworks*[1] de *deep learning* añadiendo una capa de abstracción por encima de un nivel más alto. Lo que simplifica mucho el código y convierte a Keras en una gran herramienta para introducirse en este campo además de dar una gran rapidez al hacer prototipado y experimentos sin casi perder funcionalidades respecto a usar solamente el *framework* que usa por debajo.

Nosotros hemos elegido usar Tensorflow con Keras. Al ser este el más popular entre el resto y tener una comunidad y una API más madura que pone a nuestro alcance una gran cantidad de contenido en Internet para aprender y resolver dudas.

[1] Estos son: TensorFlow, CNTK y Theano.

4.1.2. Nibabel

Las imágenes de nuestro conjunto de datos vienen en un formato llamado NIfTI[2]. Este tipo de imagen nace a partir de una iniciativa para mejorar las herramientas de tratamiento de IRM y establecer un punto de convergencia entre estas [3]. En nuestro caso, este ha sido el formato aceptado por las dos herramientas de preprocesado que hemos utilizado y que describiremos en los siguientes apartados.

Los NIfTI contienen un vector tridimensional con la representación de la imagen 3D de la cabeza del paciente y una cabecera adicional con metadatos de la imagen. Entre ellos se encuentra una matriz afín con la que se puede realizar una transformación a la imagen para llevarla a un espacio de representación dado. Esto es debido a que a veces las cabezas no están centradas o alineadas con los ejes del imán que produce la IRM por lo que la imagen resultante no sale centrada. Así pues, se puede corregir usando esta matriz.

Para tratar con este tipo de imágenes hemos utilizado Nibabel, una librería que nos permite leer y escribir imágenes con formato NIfTI entre otros. Con esta librería se puede además acceder a la cabecera de las imágenes y realizar operaciones sobre estas como la transformación con la matriz afín previamente comentada.

4.1.3. FSL

FMRIB Software Library (FSL) [4] es una librería creada por el grupo de análisis de la Universidad de Oxford con multitud de herramientas de análisis de IRM. De entre todas ellas nosotros hemos utilizado *Brain Extraction Tool* (BET) [17] y *FMRIB's Linear Image Registration Tool* (FLIRT) [13, 12], las cuales vamos a describir a continuación:

BET

BET es un software que se encarga de eliminar el tejido de la imagen que no sea cerebro. Además puede segmentar la superficie interna y externa del cráneo.

Esta herramienta es utilizada por muchos de los estudios revisados. Ya que como se puede ver en la figura 4.1, hace un buen trabajo al quitar el tejido no deseado de la imagen dejando sólo el parénquima y reduciendo los datos ruidosos que introducimos en las redes neuronales.

Para poder realizar una extracción de parénquima ajustada y sin perder tejido de interés hay que modificar los parámetros de entrada del BET según lo estricto que se quiera realizar el recorte. Unos valores demasiado altos pueden resultar en dejarse demasiados trozos de tejidos que no son de interés por recortar, mientras que valores demasiado bajos podrían quitar trozos del parénquima. Tras hacer pruebas e investigar[3] encontramos que los valores que mejor resultado nos daban eran: un umbral de 0.1[4] y activar la opción de quitar el cuello.[5]

[2] *Neuroimaging Informatics Technology Initiative.*

[3] Encontramos este articulo [15] donde compara los resultados para distintas combinaciones de parámetros.

[4] Este valor puede ir desde 0 hasta 1.

[5] Este parámetro elimina una gran cantidad de datos ruidosos ya que en las imágenes aparece todo el cuello y este no aporta información relevante para el problema.

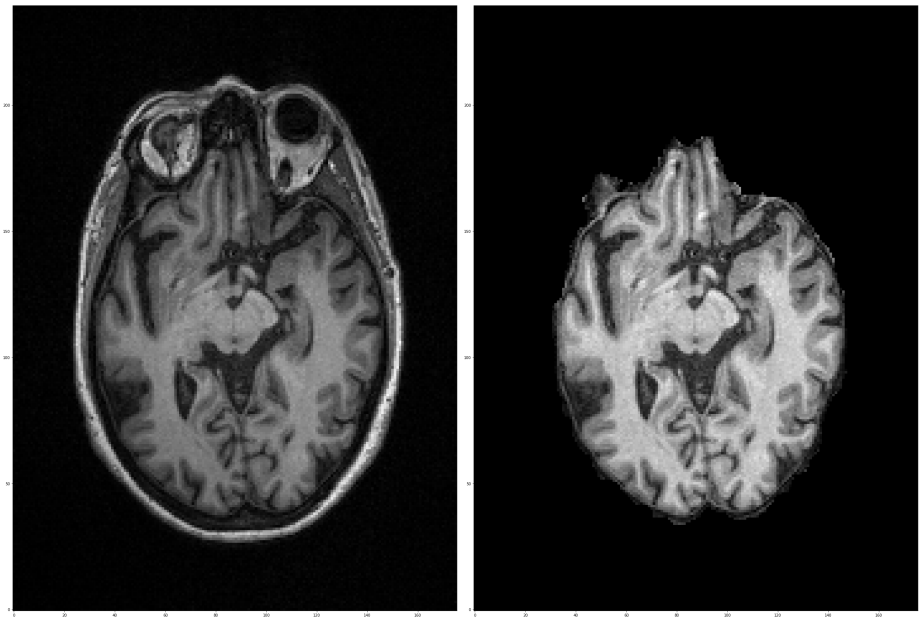

Figura 4.1: En la figura de la izquierda tenemos un corte 2D de la imagen original y en la de la derecha el mismo corte después utilizar BET.

FLIRT

FLIRT es una herramienta que realiza una transformación lineal a la imagen para llevarla a un espacio de representación de referencia tomado a partir de otra imagen de IRM. En nuestro caso hemos utilizado como referencia el MNI152, que es una imagen con un cerebro lo más representativo posible de la población obtenido a partir de realizar la media entre 152 muestras de IRM de sujetos normales.

Así pues, dada una imagen de nuestro conjunto de datos, FLIRT realiza operaciones de traslación, rotación y deformación para que la imagen dada se ajuste a la posición y forma de la cabeza de referencia. Como se puede apreciar en la figura 4.2.

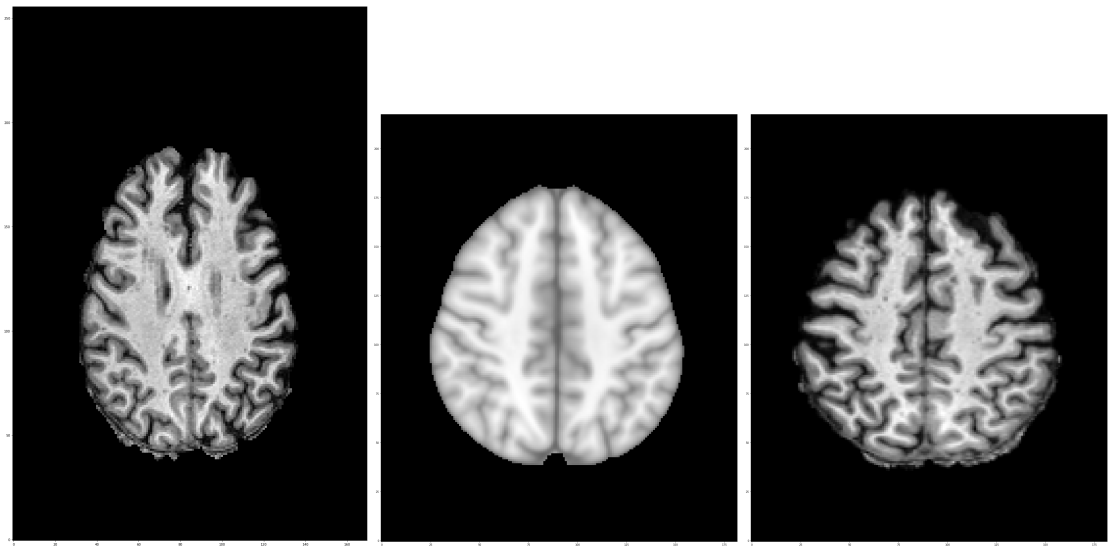

Figura 4.2: A la izquierda se puede ver un corte 2D de la imagen original, la figura central es un corte del MNI152 usado como referencia y en la derecha se observa el resultado del FLIRT.

4.1.4. FreeSurfer

FreeSurfer es un paquete software de código abierto para el análisis y la visualización de IRM. Permite extraer la segmentación del cerebro por parcelas siguiendo un atlas[6] como modelo. También extrae los datos de volumetría del cerebro para estos atlas como el grosor, volumen o superficie de las diferentes áreas. Lo cual son datos de interés ya que el deterioro cognitivo por atrofia cerebral se podría apreciar a partir de estos datos.

Aunque un punto en contra de esta herramienta es su gran demora para sacar estos datos de una imagen que puede tardar horas. En nuestro caso y para el tamaño de nuestras imágenes, se tardaba de 6 a 8 horas por cada imagen 3D de una sesión. Por lo que para procesar todo el conjunto de datos se paralelizó la tarea 100 nodos de computo donde en cada nodo se procesaba una imagen a la vez.

4.2 Hardware

Las tarjetas gráficas aportan una gran aceleración en los procesos de entrenamiento de las redes neuronales. Esto es debido a su gran potencial para realizar cálculos matriciales. Y al ser estos las operaciones más costosas llevadas a cabo durante la fase de entrenamiento, es muy útil disponer GPUs potentes para poder realizar pruebas sin tener que esperar mucho tiempo. Ya que un entrenamiento que en CPU puede durar días, en una GPU pasa a ser cuestión de horas.

En nuestro caso, al estar trabajando con imágenes 2D y 3D que requieren de mucha potencia de cálculo hemos utilizado varios ordenadores que tenían GPUs instaladas para poder trabajar con tiempos de entrenamiento más factibles.

A continuación se muestran el total de tarjetas gráficas que hemos utilizado:

- 2 NVIDIA GeForce RTX 2080 de 8GB. Estas son las más potentes en velocidad de computo que hemos utilizado gracias a su nueva arquitectura Turing [5] que se especializa más en *deep learning*. Ya que cuentan con una nueva unidad de computo llamada "*Tensor Cores*"que aceleran aun más el cálculo matriz-matriz.[7]
- 3 NVIDIA GeForce GTX 1080 de 8GB.
- 1 NVIDIA QUADRO P5000 de 16 GB. Los 16 GB de VRAM han permitido entrenar modelos de bastante más capacidad cuando entrenábamos con imágenes 3D debido a su gran dimensionalidad.

[6] La explicación de este concepto está en el apéndice C.

[7] Este cálculo es muy utilizado en las redes neuronales. Por ejemplo, al multiplicar la matriz de activaciones que sale de una capa con los pesos de la siguiente.

CAPÍTULO 5
Experimentación y resultados

En este apartado se van a describir los pasos y procesos realizados para llevar a cabo los experimentos. Primero se hablará del conjunto de datos utilizado y como se ha preparado para realizar la experimentación. Y seguidamente comentaremos las pruebas realizadas mediante redes neuronales y sus resultados.

5.1 Descripción de los datasets utilizados para la experimentación

Las imágenes han sido obtenidas del conjunto de datos *Alzheimer's Disease Neuroimaging Initiative* (ADNI). Este dataset nace a partir de una iniciativa para encontrar biomarcadores que puedan detectar la enfermedad del Alzheimer en fases tempranas de su desarrollo, ya que es el momento donde más efectivos van a ser los tratamientos. ADNI comenzó en 2004 y actualmente tiene 4 versiones del dataset. Nosotros hemos utilizado la última versión ADNI-3 de 2016, ya que es la que más muestras contiene.

Las imágenes vienen clasificadas en las siguientes categorías según el nivel de desarrollo de la enfermedad incluyendo el caso de control: AD (*Alzheimer's Disease*), LMCI (*Late Mild Cognitive Impairment*), MCI (*Mild Cognitive Impairment*), EMCI (*Earlier Mild Cognitive Impairment*) y CN (*Cognitive Normal*).

El dataset contiene dos tipos de imagen T1 y BET. Las T1 son de tipo estructural ya que gracias a su forma de adquisición muestran un buen contraste entre las zonas de diferentes tejidos e incluso oscureciendo más aquellas regiones que contengan liquido. Esto nos permite apreciar bien posibles atrofias en el cerebro ya que se habrá perdido tejido cerebral y en su lugar habrá líquido fácilmente distinguible. Por otro lado, las imágenes BET son funcionales, esto quiere decir que muestran la actividad metabólica celular, por lo que según la concentración de estas células y su actividad se verán distintas regiones contrastadas en la imagen. Esta información se ha demostrado que es de utilidad ya que los pacientes afectados por el Alzheimer muestran ciertos patrones en el metabolismo de las células cerebrales, como se demuestra en [14].

De todas estas imágenes nosotros hemos adquirido las que fueran 3D[1] y de tipo estructural T1, sumando un total de 2576 imágenes. Elegimos este subconjunto ya que el problema va a ser abordado solamente con información estructural del cerebro y para ello necesitábamos imágenes que fuesen de tipo T1 y con una buena calidad, por lo que

[1] Las imágenes están formadas por cortes 2D con vista axial. Y las imágenes 3D están formadas por un número de cortes suficientes de forma que cada uno represente un corte real de aproximadamente 1mm de grosor.

han de ser tridimensionales para aportar la mayor cantidad de información volumétrica posible. El recuento de imágenes por etiqueta, edad y sexo se puede ver en la tabla 5.1.

Tabla 5.1: En esta tabla se muestra el recuento de imágenes del dataset utilizado por edad, etiqueta y sexo.

	LMCI		AD		CN		EMCI		MCI		
Edad	M	H	M	H	M	H	M	H	M	H	Suma por edad
(50, 55]	1	0	0	0	0	0	0	0	0	0	1
(55, 60]	16	3	9	4	8	2	11	7	11	4	75
(60, 65]	19	16	10	12	18	13	51	47	18	26	230
(65, 70]	27	12	13	17	48	45	70	58	31	42	363
(70, 75]	37	34	49	32	121	97	47	79	56	81	633
(75, 80]	20	28	22	52	121	122	33	63	65	112	638
(80, 85]	15	19	24	38	53	77	29	21	56	91	423
(85, 90]	6	4	23	24	18	41	2	6	15	56	195
(90, 95]	1	0	3	0	4	3	1	0	0	5	17
(95, 100]	0	0	0	0	1	0	0	0	0	0	1
Suma por genero	142	116	153	179	392	400	244	281	252	417	2576

Cada imagen del dataset no tiene su sujeto propio sino que hay sujetos con varias imágenes que pueden haber sido tomadas en distintos años. Esto es importante saberlo para realizar un correcto particionado del dataset como se demostrará en el siguiente apartado. En la tabla 5.2 se muestra un recuento por edad, etiqueta y sexo de todos los sujetos de dataset.

Debido al tamaño del dataset no se ha podido cargar todas las imágenes en memoria para poder ejecutar el entrenamiento de las redes. Ya que gracias a Keras, si el conjunto de datos cabe en memoria se puede pasar fácilmente a la función que realiza el entrenamiento y la librería se encarga de sacar los *batches* de muestras automáticamente.

Para solucionarlo Keras tiene una clase de python llamada *DataGenerator* que nos permite extenderla para implementar un generador de los *batches* de entrenamiento personalizado que se puede pasar a la función de entrenamiento de la red en lugar del conjunto de datos. Para ello hay que implementar obligatoriamente 3 funciones dentro de la clase que necesita la función de entrenamiento: el constructor, una función que devuelve la cantidad de *batches* por *epoch* y una función que dado un índice como parámetro devuelva el *batch* correspondiente de esa *epoch*. La función que carga los *batches* va cargando las muestras en memoria a medida que se le piden, por lo que no hay que tener en la RAM todo el dataset, sino que se va cargando y eliminando de la memoria poco a poco. Además, para disminuir el posible cuello de botella, Keras automaticamente se encarga de crear un cola[2] donde ir metiendo *batches* sin necesidad de empezar a cargarlos justo en el momento que se necesitan.

Dentro del *DataGenerator* también se ha implementado el sistema de balanceo de las muestras por etiqueta. Ya que como se puede ver en la tabla 5.1 hay mucha diferencia en el número de muestras de las etiquetas. La estrategia seguida ha sido el balanceo a nivel de *batch*, donde para cada uno que se pedía se ha cogido a partes iguales muestras de cada conjunto de etiquetas. Y cuando se alcanza la última muestra de alguna etiqueta se empiezan a repetir las muestras de esa clase otra vez.

[2]El tamaño de esta cola se le puede pasar como parámetro.

Tabla 5.2: En esta tabla se muestra el recuento de sujetos del dataset utilizado por edad, etiqueta y sexo.

	LMCI		AD		CN		EMCI		MCI		
Edad	M	H	M	H	M	H	M	H	M	H	Suma por edad
(50, 55]	1	0	0	0	0	0	0	0	0	0	1
(55, 60]	6	2	5	2	5	2	8	4	6	3	43
(60, 65]	9	9	6	7	14	7	27	22	12	15	128
(65, 70]	14	9	6	11	33	29	40	31	18	20	211
(70, 75]	20	20	29	24	69	53	26	40	31	46	358
(75, 80]	15	16	14	30	65	67	17	37	36	59	356
(80, 85]	12	12	15	25	34	43	17	16	30	44	248
(85, 90]	3	3	13	15	13	24	2	5	10	29	117
(90, 95]	1	0	1	0	2	3	1	0	0	5	13
(95, 100]	0	0	0	0	1	0	0	0	0	0	1
Suma por genero	81	71	89	114	236	228	138	155	143	221	1476

5.2 Descripción de los experimentos realizados

5.2.1. Primeros experimentos

Inicialmente se ha abordado el problema en forma de una clasificación de imágenes 2D en lugar de intentar clasificar directamente la imagen 3D de la sesión[3] del sujeto. Por lo que para clasificar una sesión se procedería a etiquetar cada uno de los cortes 2D para luego realizar una clasificación por votación de la imagen 3D.

Para ello, cada imagen 3D se ha particionado en cada uno de los cortes axiales que posee dándonos como resultado 481952 imágenes 2D. Estas se particionaron de forma aleatoria en los subconjuntos de entrenamiento, validación y test en un 60 %, 20 % y 20 % respectivamente. Este particionado aleatorio fue incorrecto debido a que no se tuvo en cuenta la importancia de no separar imágenes de un mismo sujeto en diferentes particiones. Ya que la red puede aprender las características personales de cada sujeto y con ello obtener un buen resultado en la clasificación de todas las particiones de entrenamiento a pesar de no estar generalizando correctamente. Además, este error no se puede apreciar en los resultados de clasificación de ninguna de las tres particiones ya que las imágenes de los sujetos se han repartido por todas ellas.

A continuación se muestra el experimento realizado con el que hemos visto que esta forma de particionar los datos hace que la red se aprenda las características específicas de cada sujeto en lugar de generalizar correctamente para la tarea.

Autoencoder con imágenes 2D

Con este conjunto de datos se ha entrenado primero un autoencoder que aprenda a reconstruir las imágenes 2D, para así preentrenar los pesos de las capas convolucionales que luego han sido utilizadas en la red de clasificación. Este está formado por bloques de capa de convolución más *batch normalization* para evitar *overfitting*; capas *max pooling* en la

[3]Una sesión equivale a una imagen, de forma que un sujeto puede tener varias sesiones.

parte del encoder para reducir la dimensionalidad y capas *upsampling* en el decoder que realizan la inversa del *pooling* para volver al tamaño original de la imagen. La topología completa se puede ver en la figura 5.1.

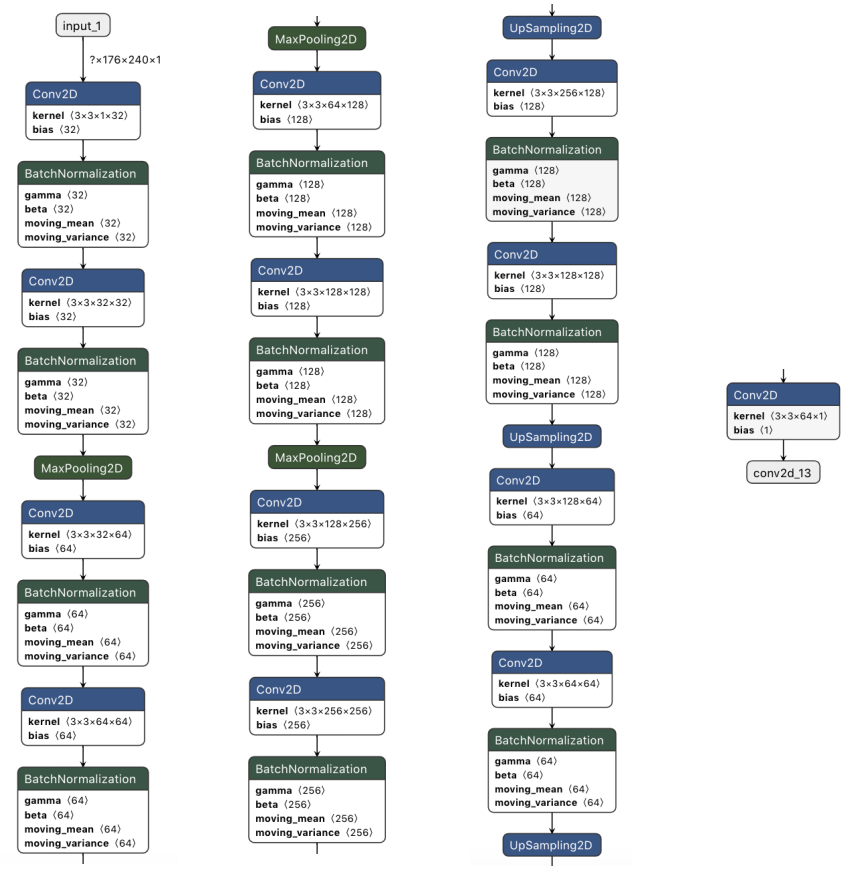

Figura 5.1: Topología del autoencoder utilizado en el primer experimento.

Una vez entrenado el autoencoder se han extraído las capas preentrenadas del encoder y se han conectado a una red *fully connected* con una capa de salida de 5 neuronas con una función de activación *softmax* para obtener una distribución de probabilidad entre las 5 etiquetas posibles.

Esta red se ha entrenado en dos pasos. Primero se han congelado los pesos obtenidos del autoencoder para realizar primero un entrenamiento de los pesos de las capas *fully connected*. Y una vez entrenada se han descongelado los pesos para hacer un entrenamiento de grano fino para que se ajusten todos los pesos de todas las capas en conjunto.

El resultado de la partición de validación nos ha dado un valor de *accuracy* del 98 % en la clasifiación en 5 etiquetas. Este valor no puede ser cierto para la aproximación tomada ya que ese 98 % es obtenido al clasificar todas las imágenes 2D de los cortes, pero cada imagen posee muchos cortes pertenecientes a las partes inferiores y superiores que son cortes negros, con ruido o que pertenecen al cuello del sujetos. Estas imágenes forman mucho más del 2 % de los cortes y el hecho de que los esté clasificando correctamente para la enfermedad del Alzheimer no puede ser posible ya que no tienen ninguna información a partir de la cual poder clasificarlas para esta tarea.

Nuestra deducción al respecto ha sido que se trataba del particionado, por lo que probamos a volver a entrenar otra red con la misma técnica del preentrenamiento mediante un autoencoder pero con la partición hecha a nivel de sujeto, de forma que todas las imágenes de un sujeto estén en sólo una partición. Obteniendo unos resultados mucho

peores del 41 % de *accuracy* en validación que demuestran que la partición no se estaba realizando correctamente.

Una vez obtenidos los primeros resultados con la partición a nivel de sujeto se ha visto que no se puede alcanzar una buena clasificación sin limpiar los datos y extraer las zonas de interés como se ha visto en el estado del arte, donde se utilizan múltiples técnicas para resolver este problema.

Para los primeros experimentos como el del autoencoder comentado anteriormente sólo se ajustaron los tamaños de las imágenes 2D para que fueran el mismo añadiendo un *padding* de ceros a las imágenes haciéndolas de 256x256 píxeles. Se escogió este tamaño por ser 256 la dimensión más grande quitando unos pocos casos que fueron descartados porque aumentaban el tamaño de las imágenes para incluir unas pocas más. Este incremento en el tamaño se evitó para mayor eficiencia del entrenamiento.

El siguiente paso en el tratamiento de las imágenes fue elegir el tamaño más repetido en el dataset para luego transformar las imágenes a este tamaño. Se ha realizado un estudio de las dimensiones de los cortes 2D y el tamaño elegido ha sido 176x240 por ser el más abundante como se puede ver en la tabla 5.3.

Para llevar a este tamaño al resto de imágenes se ha desarrollado un script que primero añade un *padding* de ceros en aquellas dimensiones en las que la imagen fuera más pequeña. Seguidamente se hace un *crop* de las imágenes que fueran más grandes cogiendo un recorte centrado y dos adicionales por cada dimensión que fuera más grande que la del tamaño objetivo,[4] pudiendo sacar hasta cinco recortes de una imagen. De esta manera también conseguimos algo de *data augmentation* a modo de desplazamiento horizontal y vertical.

Tabla 5.3: Recuento de sesiones por tamaño de imagen 2D

Tamaño	Sesiones
176x240	1078
160x192	682
170x256	590
150x256	1
160x240	190
208x240	46
176x256	8
211x256	24
184x256	19
170x288	3
230x230	6
176x248	3
230x240	5
140x256	2
512x174	1
160x256	1

Con este segundo procesado de las imágenes los resultados han sido los mismos ya que realmente el problema de los datos ruidosos como el cuello y el cráneo siguen estando. Así que hemos estudiado herramientas con las que poder eliminarlos y poder limpiar y preparar mejor los datos. Estas herramientas han sido: FreeSurfer, utilizado en el trabajo de Silvia Nadal Almela para entrenar utilizando datos de volumetría; BET y FLIRT

[4] Si la imagen es más grande en el eje horizontal se coge un recorte desde el lado izquierdo y otro desde el derecho, y en eje vertical de la misma manera pero cogiéndolo desde arriba y abajo.

utilizado por Rafael Vicente Sánchez Romero y por mí para la eliminación de los datos ruidosos. En el trabajo de Rafael Vicente se aborda la clasificación con imágenes 2D en profundidad mientras que yo lo haré mediante las imágenes en 3D.

5.2.2. Clasificación con imágenes 3D

Después de obtener los primeros resultados con la clasificación en 5 etiquetas se decidió reducir el problema a una clasificación en 2: *Alzheimer's Disease* (AD) y *Cognitive Normal* (CN). Ya que debido al desbalanceo[5] en la cantidad de muestras de cada etiqueta y a las pocas diferencias estructurales que pueden haber entre las etiquetas LMCI, MCI y EMCI la tarea se complica mucho, así que optamos por este planteamiento al igual que en la literatura. Donde suelen hacer las clasificaciones con 2 etiquetas y en algunos casos con 3 como se puede ver en la tabla 2.1.

A continuación se van a describir los experimentos realizados con las imágenes 3D en dos apartados; el primero utilizando BET como herramienta de preprocesado y el segundo donde además se utiliza el FLIRT. En todos ellos se realiza una clasificación en las dos etiquetas: AD y CN.

Preprocesado con BET

Mediante BET hemos extraído los parénquimas cerebrales de las imágenes para así eliminar todos los tejidos que no pertenecen al cerebro. Además también se consigue eliminar el ruido que hay en el fondo oscuro de la imagen, ya que no es completamente negro sino que presenta variaciones de intensidad producidas por el escáner de resonancia magnética.

Para conseguir una buena extracción se experimentó con los parámetros de entrada del BET para poder quitar la mayor cantidad de cráneo sin perder tejido cerebral. Tras hacer pruebas e investigar encontramos que los parámetros presentados en [15] eran los que mejor funcionaban; que son f=0.1[6] y activar la opción de eliminar el cuello.[7]

El BET ha sido ejecutado sobre las imágenes originales las cuales no poseen todas el mismo tamaño, por lo que las imágenes producidas tampoco son iguales. Para solucionar esto se aplicó primero un *padding* de ceros en 3 dimensiones para llevar a todas las imágenes al tamaño 256x256x256 pero eso impedía poder entrenar modelos con suficiente capacidad ya que el tamaño de los tensores era muy grande utilizando estas dimensiones de entrada. Así que se realizo un script para encontrar el cubo más pequeño posible en el que todos los cerebros cupiesen y posteriormente llevar a todas las imágenes a ese tamaño recortando o añadiendo *padding* de ceros. Obteniendo unas dimensiones de 157x200x173. Se decidió eliminar una muestra la cual era más grande para no tener que aumentar mucho el tamaño por sólo una imagen.

La normalización utilizada ha sido dividir el valor de los píxeles entre 2000. Este valor ha sido elegido tras sacar estadísticas sobre los valores de los píxeles de todas las imágenes, donde se vio que 2000 es el valor máximo aproximado para las imágenes como se puede ver en la figura 5.2.

Existen unos pocos casos que exceden el valor 2000 por mucha diferencia, llegando a valores de 60000 que pueden ser debidos a imprecisiones del escáner de resonancia magnética y aparecen en el recuento de la figura 5.3, los cuales fueron tomados como el valor 2000 en el momento de la normalización.

[5] Estos valores se pueden consultar en la tabla 5.1.
[6] Este parámetro represanta el umbral de recorte.
[7] Se activa con el *flag* -B.

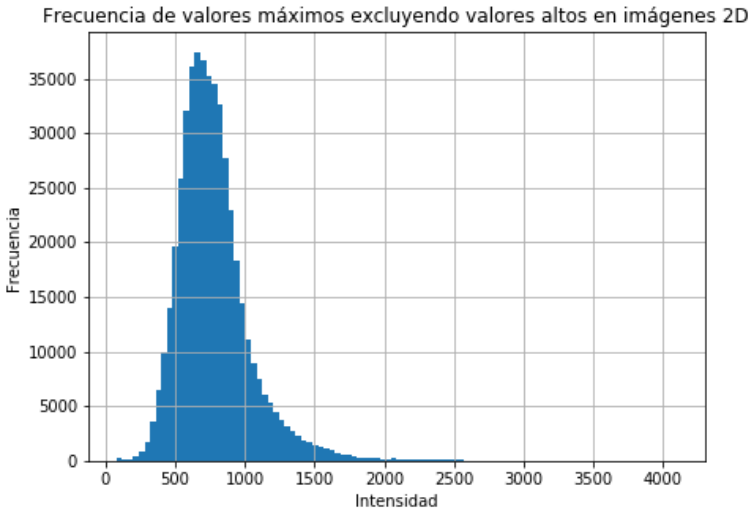

Figura 5.2: Recuento de los valores máximos de los píxeles para cada corte 2D de todas las imágenes 3D excluyendo algunos valores muy altos.

Por lo que la función de normalización queda como se muestra en la fórmula 5.1.

$$norm(x) = min(1, \frac{x}{2000}) \tag{5.1}$$

A continuación se muestran los resultados obtenidos con este preprocesado con BET para los modelos 1 y 2 propuestos en el apartado de descripción del problema y solución propuesta.

Modelo 1

Con este modelo se ha obtenido un *accuracy* en el conjunto de test del 78,4 %. Por lo que podemos decir que la red está aprendiendo a detectar algunas características del atrofiamiento cerebral que están siendo útiles en la clasificación. Los resultados del entrenamiento completos se pueden ver en la figura 5.4. En esta se aprecia como la red empieza a hacer *overfitting* a parir de la *epoch* 10 aproximadamente ya que el valor de *accuracy* empieza a ser muy diferente entre las particiones de entrenamiento y validación. Y en el caso de la gráfica del *loss*, a partir de ese mismo punto el valor empieza a incrementarse indefinidamente para la partición de validación.

Modelo 2

En este caso los resultados del test han sido de un 76,5 % de *accuracy*, un poco más bajos que con el modelo anterior. Los resultados del entrenamiento completos se pueden ver en la figura 5.5. Al igual que con el modelo anterior se ha empezado a hacer *overfitting* sobre la *epoch* 5 aunque en este caso la evolución del *accuracy* y del *loss* ha sido menos irregular llegando a un punto donde se ha estabilizado.

En la tabla 5.4 se muestran las estadísticas completas de la clasificación del conjunto de test para este preprocesado de BET.

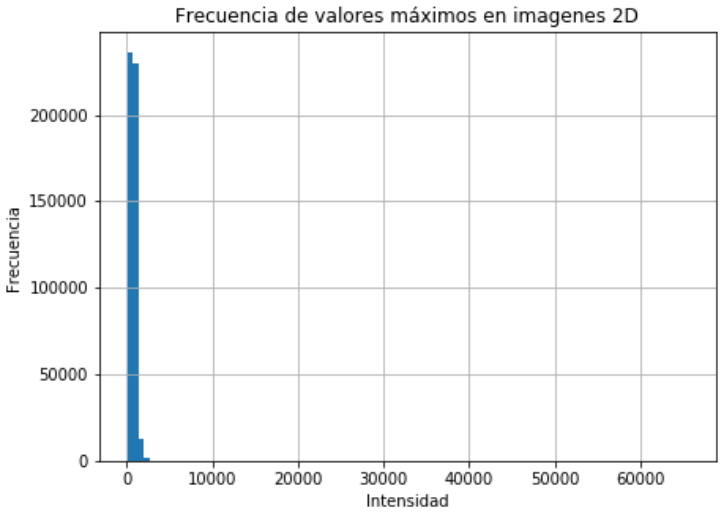

Figura 5.3: Recuento de los valores máximos de los píxeles para cada corte 2D de todas las imágenes 3D.

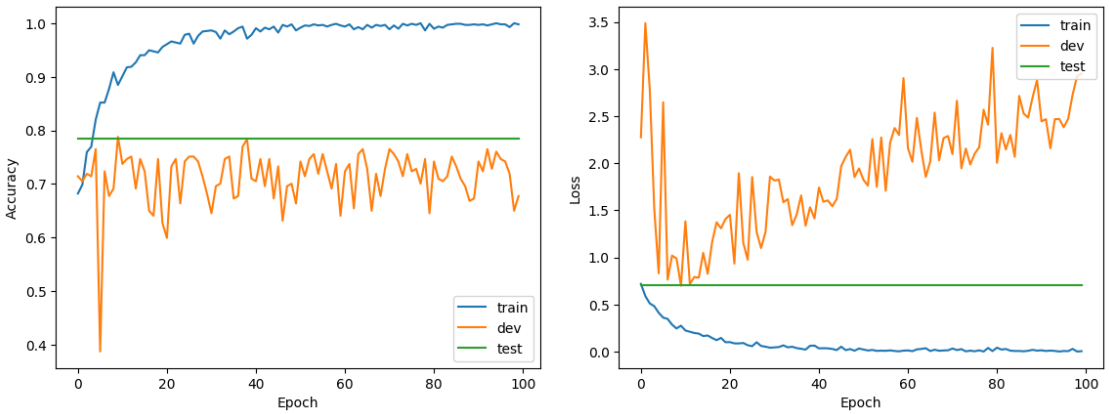

Figura 5.4: Resultados de clasificación en las clases AD y CN con el modelo 1 entrenado con el preprocesado de BET.

Preprocesado con BET más FLIRT

Con solamente el BET como herramienta de preprocesado se ha obtenido un 78,4 % de *accuracy*. Esto queda lejos de los resultados del estado del arte con un 98,83 %, como se puede ver en la tabla 2.1. Para acercarnos más a esos resultados hemos intentado replicar el preprocesado del estado del arte[8] utilizando FLIRT además del BET.

FLIRT es una herramienta que nos permite realizar transformaciones lineales a las imágenes para llevarlas a un mismo espacio de representación dado por otra imagen de referencia. Con esto podemos hacer que todas las sesiones 3D estén es la misma posición, orientación e incluso forma del volumen cerebral. De forma que si un cerebro tiene un atrofiamiento en una zona en particular, en esa misma zona de un cerebro sano se verá diferente. Debido a que ambos han sido transformados para estar en la misma posición y forma. Por lo que además se pierden características propias de los pacientes como la forma de la cabeza y que no aportan información relevante y puede ser usado por la red para hacer *overfitting*.

[8] La publicación [11].

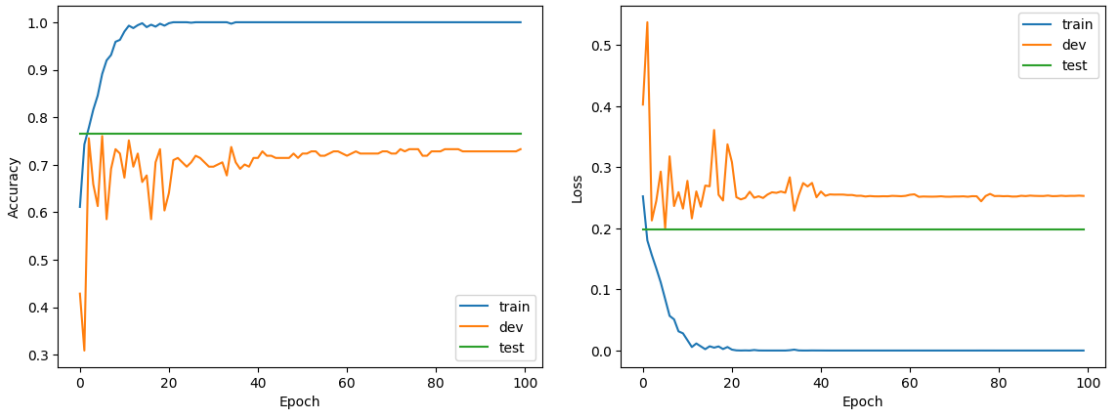

Figura 5.5: Resultados de clasificación en las clases AD y CN con el modelo 2 entrenado con el preprocesado de BET.

Tabla 5.4: Resultados de los modelos entrenados con el preprocesado de BET.

	Accuracy	*Precision*	*Recall*	*F1 score*
Modelo 1	78,46 %	80,95 %	91,27 %	85,80 %
Modelo 2	76,55 %	77,47 %	94,63 %	85,19 %

La imagen tomada como referencia para ejecutar el FLIRT es un estándar creado por *Montreal Neurological Institute* llamado MNI152. Este cerebro esta creado a partir de la media de 152 imágenes de sujetos sanos que previamente han sido estandarizadas usando su anterior modelo MNI305.

El MNI152 utilizado contiene sólo el parénquima cerebral, por lo que el FLIRT lo hemos ejecutado sobre las imágenes resultantes del BET.

La ejecución del BET y del FLIRT tarda unos 5 minutos por imagen. Esto aplicado a todo el dataset pasaría a tardar unos 9 días. Para reducir este tiempo se ha realizado un script de preprocesado que hace la ejecución de cada imagen en paralelo en cada núcleo del procesador. Este fue lanzado en un ordenador con 12 núcleos reduciendo el tiempo de preprocesado de todo el dataset a menos de un día.

FLIRT devuelve las imágenes con las mismas dimensiones de la imagen que se ha utilizado como referencia dándonos unas imágenes de 182x218x182. Para optimizar el entrenamiento de las redes y permitirnos construir redes con más capacidad, reducimos el tamaño de estas a imágenes a 164x198x172.[9]

Con estos nuevos datos se han vuelto a probar los dos modelos del apartado anterior donde sólo se utilizaba BET. Para el primer modelo[10] se ha obtenido un *accuracy* de 79,52 % y con el segundo modelo[11] un 84,76 %. Los resultados completos de los entrenamientos se pueden ver en la figura 5.6 para el primer modelo y la 5.7 para el segundo. Al igual que con el preprocesado de sólo BET el modelo 2 tiene una evolución más estable del *accuracy* y del *loss* durante el entrenamiento.

En la tabla 5.5 se muestran las estadísticas completas de la clasificación del conjunto de test para este preprocesado de BET más FLIRT.

Se aprecia una clara mejoría sobretodo con el segundo modelo, por lo que podemos concluir que el BET más el FLIRT aporta un mejor preprocesado de los datos que sólo

[9] Se ha utilizado el mismo script que elimina los bordes negros desarrollado para las imágenes con BET.
[10] Figura 3.1.
[11] Figura 3.2.

utilizando el BET. Esto puede deberse a la perdida de la variabilidad de características no relevantes como la forma de la cabeza o la posición de esta que podrían estar introduciendo ruido a la red y que son eliminadas con FLIRT.

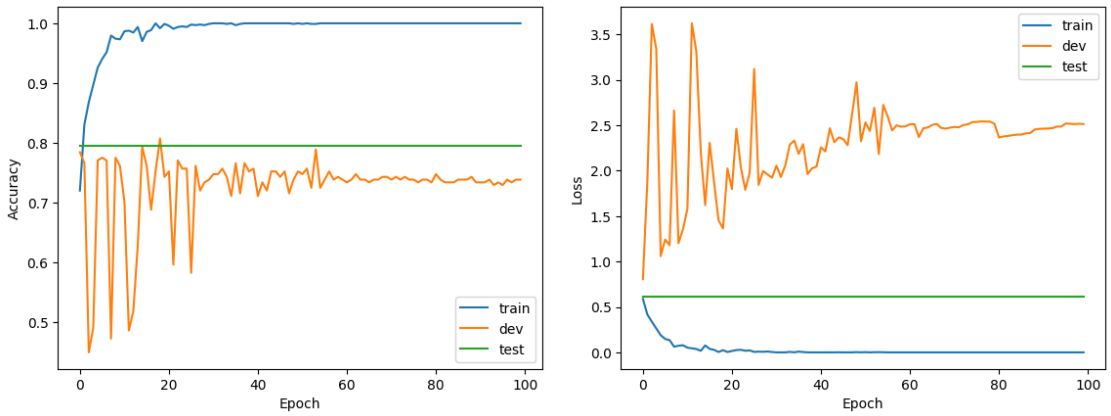

Figura 5.6: Resultados de clasificación en las clases AD y CN con el primer modelo entrenado con el preprocesado de BET más FLIRT.

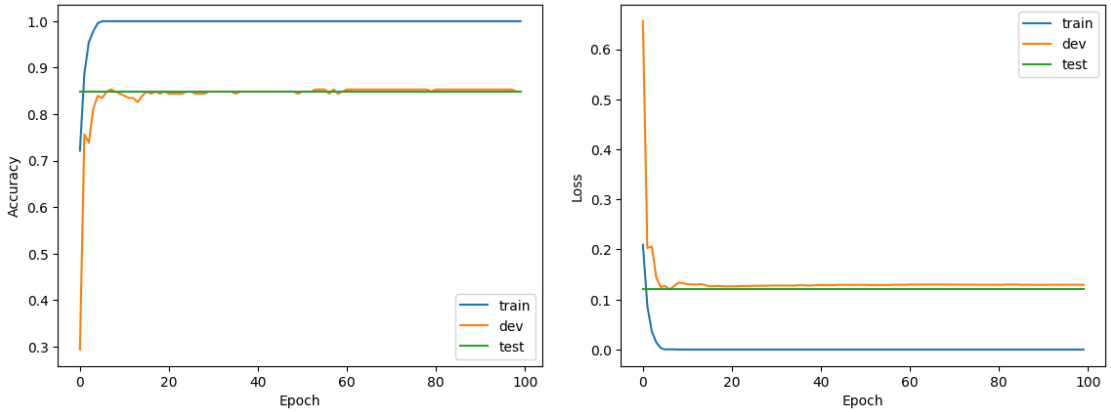

Figura 5.7: Resultados de clasificación en las clases AD y CN con el segundo modelo entrenado con el preprocesado de BET más FLIRT.

Tabla 5.5: Resultados de los modelos entrenados con el preprocesado de BET más FLIRT.

	Accuracy	*Precision*	*Recall*	*F1 score*
Modelo 1	79,52 %	79,23 %	96,66 %	87,08 %
Modelo 2	84,76 %	85,97 %	94,00 %	89,80 %

5.2.3. Reproducción del estado del arte

Nuestro mejor resultado aplicando nuestros modelos ha sido un 84,76 %, por lo que hay un gran margen de mejora hasta el 98,83 % del estado del arte [11]. Así que hemos reproducido su modelo para intentar acercarnos a esos resultados.

Preparación de los datos

En [11] utilizan el FLIRT al igual que nosotros con nuestro último preprocesado. Pero obtienen un tamaño de las imágenes de 91x109x91 el cual es mucho más pequeño que el

que nosotros tenemos de 164x198x172[12]. Esto es debido a que su imagen de referencia es una versión del MNI152 con menor calidad y como consecuencia con unas dimensiones de 91x109x91.[13] Por lo que hemos tenido que volver a ejecutar el preprocesado con la misma versión del MNI152 que ellos utilizan. Ya que la topología de la red que proponen tiene un gran coste en memoria que no podría ser soportado con nuestras dimensiones previas.

Reproducción del modelo y entrenamiento

El modelo reproducido se puede ver en la figura 2.2. La información que no se aporta en la publicación es la cantidad de filtros en las capas convolucionales y donde aplican las técnicas de regularización. Ya que explican que utilizan *dropout* pero no en que capas ni en que ratio se apagan las activaciones. Así que menos estos parámetros se ha reproducido la misma estructura de la red implementado los bloques densamente conectados[14] como ellos explican y con el mismo número y tipo de capas.

El optimizador del entrenamiento también es el mismo que ellos utilizan y es el *stochastic gradient descent* (SGD) con *momentum* de 0,9. El entrenamiento ha sido de 200 *epochs* a diferencia de los anteriores experimentos debido a que en [11] dicen que entrenan el modelo durante 1000 *epochs*, de forma que hemos dejado el entrenamiento más tiempo por ver si podía mejorar. Tras las 200 iteraciones, el resultado obtenido en la partición de test para el modelo de la *epoch* con menor *loss* en la partición de validación ha sido de un 76,19 %, más bajo que los previamente obtenidos con nuestra propuesta del modelo 2 con un 84,76 % y también peores que los obtenidos en el estado del arte como se puede ver en la tabla 5.6. El resultado completo del entrenamiento se puede ver en la figura 5.8.

Tabla 5.6: Comparación de los resultados obtenidos por el estado del arte con los de nuestra implementación de su propuesta.

	Accuracy	*Precision*	*Recall*	*F1 score*
Estado del arte	98,83 %	98,70 %	98,70 %	98,70 %
Nuestra implementación	76,19 %	84,72 %	81,33 %	82,99 %

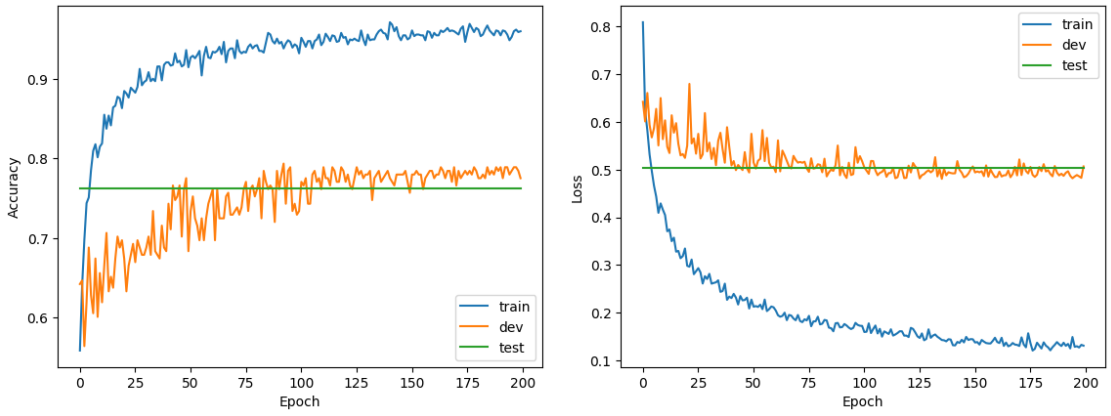

Figura 5.8: Resultados de clasificación en las clases AD y CN con el modelo del estado del arte [11] entrenado con el preprocesado de BET más FLIRT.

[12] A pesar de haber reducido el tamaño de las imágenes tras pasar el FLIRT.
[13] Nuestra versión tiene una precisión de 1 mm mientras que la ellos utilizan una de 2 mm.
[14] Se puede ver un esquema de la estructura de este bloque en la figura 2.1.

5.2.4. Validación cruzada

Después de probar todos los modelos por separado hemos decidido coger el que mejores resultados nos ha dado para realizar la clasificación por votación. Para obtener unos resultados más certeros se ha hecho el entrenamiento y la votación para 5 particiones distintas del dataset con preprocesado de BET más FLIRT. En cada una de ellas se han entrenado 10 modelos del tipo 2 propuesto ya que ha obtenido los mejores resultados.

El sistema de votación empleado utiliza directamente los vectores resultantes de las capas *softmax* de salida. Debido a que se tiene en cuenta la distribución de probabilidad completa a la hora de realizar la votación, y no sólo si la red ha clasificado a una clase o a otra. En la fórmula 3.1 se muestra el cálculo realizado para la clasificación mediante este sistema.

La configuración de la red completa para cada partición quedaría como se muestra en la figura 5.9.

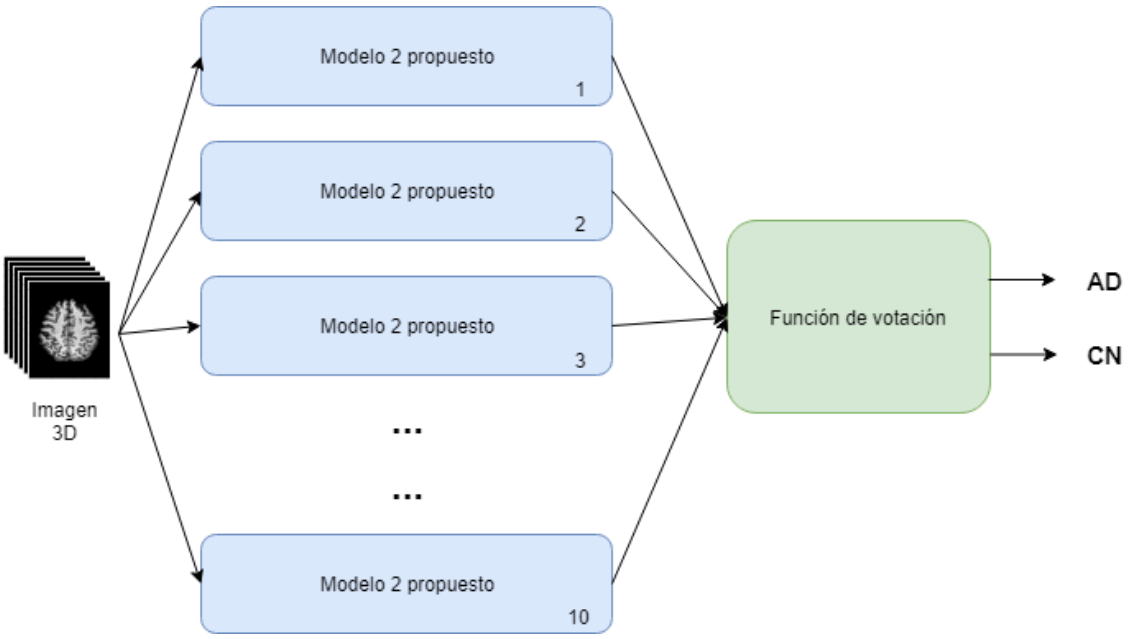

Figura 5.9: Estructura del sistema para la validación cruzada. Formado por 10 modelos del tipo 2 propuesto; entrenados sobre la misma partición y con sus salidas conectadas a la función de votación que realiza la clasificación final para el conjunto de test.

En la tabla 5.7 se muestran los resultados para cada partición. Como se puede apreciar, el resultado es muy dependiente de la partición realizada, esto es debido a que el número de muestras no es muy elevado[15] haciendo que los cambios en la distribución de las particiones tengan mucho impacto en el aprendizaje de la red. Por lo que se han obtenido desde valores más bajos que los previamente obtenidos sin votación, como en las particiones 1, 2 y 4; hasta valores más altos como las particiones 3 y 5, alcanzando en la 3 un 87,94 % que es nuestro máximo *accuracy* alcanzado.

5.2.5. Resultados y valoración del experimento

En este apartado se van a comentar y valorar los resultados obtenidos en los experimentos descritos anteriormente. Se puede consultar la recopilación de todos ellos en la tabla 5.8.

[15] Para la clasificación en AD y CN se disponen de 1124 imágenes.

Tabla 5.7: Resultados de la validación cruzada para 5 particiones distintas del dataset y utilizando 10 modelos en cada una de ellas para realizar la votación en el conjunto de test.

	Accuracy	*Precision*	*Recall*	*F1 score*
Partición 1	83,41 %	88,37 %	90,47 %	89,41 %
Partición 2	77,61 %	81,48 %	88,59 %	84,88 %
Partición 3	87,94 %	88,69 %	94,90 %	91,69 %
Partición 4	75,12 %	82,31 %	82,87 %	82,59 %
Partición 5	85,59 %	87,01 %	94,47 %	90,58 %

Gracias a las pruebas realizadas con diferentes preprocesados se ha visto como la utilización del BET con el FLIRT aporta mejores resultados que sólo utilizando el BET. Por lo que podemos deducir que la eliminación de factores como la rotación y posición de la cabeza; y la forma de esta, que es particular de cada paciente y no aporta información relevante, sólo introducen ruido a la red. Por lo que unos datos más estandarizados como los que proporciona el FLIRT permiten obtener unas imágenes con una mayor proporción de características relevantes para el problema. Obteniendo de esta forma los mejores resultados de un 84,76 % de *accuracy* para la clasificación mediante el modelo 2 y de un 87,94 % para la validación cruzada con 10 modelos del tipo 2.

A pesar de obtener un 87,94 % en la mejor partición mediante validación cruzada, este valor no se puede tomar como el *accuracy* real del sistema ya que posee un sesgo debido a que este resultado se ha obtenido en sólo una de las particiones y viene condicionado por la distribución del dataset y no por su mejor capacidad de generalización para el problema, ya que para otras particiones se han obtenido valores peores a pesar de utilizarse los mismos modelos. Por lo que un valor más fiel para evaluar este sistema de validación cruzada sería hacer la media de todos los valores obtenidos de las particiones, de forma que obtenemos un *accuracy* de 81,93 %.

Por lo que podemos concluir que la cantidad de muestras empleada para estos experimentos no ha sido realmente la suficiente para poder obtener unos resultados fiables. Ya que la escasez de muestras y el desbalanceo de las etiquetas en el dataset hacen que no se pueda obtener un conjunto de datos con una representación del problema real lo suficientemente amplia y bien equilibrada. Ya que se ha visto reflejado este problema al obtener resultados muy diferentes sólo cambiando la distribución de las muestras entre las tres particiones de entrenamiento, validación y test.

Tabla 5.8: Recopilación de los resultados obtenidos para todos los sistemas de preprocesado y clasificación utilizados.

		Accuracy	*Precision*	*Recall*	*F1 score*
BET	**Modelo 1**	78,46 %	80,95 %	91,27 %	85,80 %
	Modelo 2	76,55 %	77,47 %	94,63 %	85,19 %
BET más FLIRT	**Modelo 1**	79,52 %	79,23 %	96,66 %	87,08 %
	Modelo 2	84,76 %	85,97 %	94,00 %	89,80 %
Validación cruzada (10 modelos del tipo 2) BET más FLIRT	**Partición 1**	83,41 %	88,37 %	90,47 %	89,41 %
	Partición 2	77,61 %	81,48 %	88,59 %	84,88 %
	Partición 3	87,94 %	88,69 %	94,90 %	91,69 %
	Partición 4	75,12 %	82,31 %	82,87 %	82,59 %
	Partición 5	85,59 %	87,01 %	94,47 %	90,58 %

CAPÍTULO 6
Conclusiones

En este trabajo se han podido estudiar algunas de las herramientas que principalmente se utilizan para el tratamiento de IRM. Se ha a aprendido a trabajar con el formato de imagen NIfTI que es el estándar empleado para el tratamiento de este tipo de imágenes y es el empleado como formato de entrada por las herramientas de preprocesado utilizadas.

Hemos estudiado las funcionalidades de las herramientas FSL[1] y FreeSurfer que podían proporcionarnos beneficios en nuestro preprocesado de los datos. Con FSL se ha visto como tratar las imágenes para limpiarlas y obtener unos datos lo más preparados posible de cara a mejorar el entrenamiento de las redes. En el caso de FreeSurfer, no se ha puesto en práctica en este trabajo ya que se ha profundizado en el de la compañera Silvia Nadal Almela. Pero aun así se han visto las funcionalidades que ofrece como la extracción de datos de volumetría y segmentaciones complejas del cerebro, pudiendo extraer parcelas concretas siguiendo el modelo proporcionado por un atlas cerebral dado.

Además del procesamiento de los datos también se ha aprendido a trabajar con redes neuronales en un proyecto de estas características utilizando el *framework* de Keras junto a Tensorflow para implementar las redes.

Con todos estos conocimientos aplicados se ha visto como el problema de detección del deterioro por la enfermedad del Alzheimer puede ser abordado mediante técnicas de *deeplearning*. Obteniendo resultados de clasificación entorno al 80 % de *accuracy*, que aún siguen siendo más bajos que los obtenidos en el estado del arte. Estos resultados se han intentado replicar pero no se han alcanzado cifras parecidas a pesar de utilizar el mismo preprocesado y un modelo de red neuronal parecido, ya que no ha podido ser el mismo que ellos utilizan debido a que no dan datos concretos sobre toda la topolgía de la red; dejándose datos como el número de *kernels* en las capas convolucionales y donde y en que medida aplican la regularización con *dropout*.

Gracias a los experimentos realizados también se ha podido apreciar una de las dificultades principales al abordar este tipo de problemas y es la cantidad de datos disponibles. Debido a que son imágenes con información sensible y de difícil adquisición en grandes cantidades. Y como consecuencia hemos trabajado con un conjunto de datos no lo suficientemente extenso y balanceado que nos permita obtener unos mejores resultados.

[1] Contiene las herramientas BET y FLIRT.

6.1 Trabajos futuros

Para poder disminuir el impacto de la falta de datos, un desarrollo futuro es la adquisición de otro dataset denominado OASIS[2] el cual contiene 2168 sesiones de resonancia magnética que supondrían un gran incremento de los datos.

Otra aproximación que se podría tomar es la utilización de las máscaras del FreeSurfer. Consistiría en estudiar que zonas del cerebro son las más afectadas por la atrofia y extraer esas regiones con la segmentación por parcelas del FreeSurfer. Posteriormente se entrenarían redes neuronales que tomen sólo los recortes de las regiones de interés para realizar la clasificación. De esta manera se podría intentar limpiar todavía más los datos eliminando aquellas regiones del cerebro que no sean de interés y como consecuencia introducen ruido en la red.

[2]Más información en https://www.oasis-brains.org/.

Bibliografía

[1] ¿Cómo se hace el diagnóstico del Alzheimer? Consultado en https://blog.fpmaragall.org/como-se-diagnostica-el-alzheimer

[2] Enfermedad del Alzheimer. Consultado en https://fpmaragall.org/alzheimer-enfermedad/enfermedad-alzheimer/

[3] Información sobre el formato NIfTI. Consultado en https://nifti.nimh.nih.gov/background

[4] Información FSL. Consultado en https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/

[5] Arquitectura Turing de las NVIDIA RTX 2080. Consultado en https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf

[6] Aderghal, Karim & Boissenin, M & Benois-Pineau, Jenny & Catheline, Gwenaëlle & Karim, Afdel. Classification of sMRI for AD Diagnosis with Convolutional Neuronal Networks: A Pilot 2-D+ϵ Study on ADNI. *Lecture Notes in Computer Science*, 10.1007/978-3-319-51811-4_56.

[7] Adrien Payan y Giovanni Montana. Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks. arXiv:1502.02506 [cs.CV], lunes 9 de febrero de 2015.

[8] S.I. Dimitriadis, Dimitris Liparas, Magda N. Tsolaki. Random forest feature selection, fusion and ensemble strategy: Combining multiple morphological MRI measures to discriminate among healhy elderly, MCI, cMCI and alzheimer's disease patients: From the alzheimer's disease neuroimaging initiative (ADNI) database. *Journal of neuroscience methods*, 15 de mayo de 2018;302:14-23, doi: 10.1016/j.jneumeth.2017.12.010.

[9] Donghuan Lu, Karteek Popuri, Weiguang Ding, Rakesh Balachandar, Mirza Faisal Beg. Multimodal and Multiscale Deep Neural Networks for the Early Diagnosis of Alzheimer's Disease Using Structural MR and FDG-PET Images. arXiv:1710.04782 [cs.CV], viernes 12 de octubre de 2017.

[10] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger. Densely Connected Convolutional Networks. arXiv:1608.06993 [cs.CV], 28 de enero de 2018.

[11] Hongfei Wang and Yanyan Shen and Shuqiang Wang and Tengfei Xiao and Liming Deng and Xiangyu Wang and Xinyan Zhao. Ensemble of 3D densely connected convolutional network for diagnosis of mild cognitive impairment and Alzheimer's disease. *Neurocomputing*, volumen 333, páginas 145-156, 14 de marzo de 2019.

[12] M. Jenkinson, P.R. Bannister, J.M. Brady, y S.M. Smith. Improved optimisation for the robust and accurate linear registration and motion correction of brain images. *NeuroImage*, 17(2):825-841, 2002.

[13] M. Jenkinson y S.M. Smith. A global optimisation method for robust affine registration of brain images. *Medical Image Analysis*, 5(2):143-156, 2001.

[14] Marcus C, Mena E, Subramaniam RM. Brain PET in the diagnosis of Alzheimer's disease. *Clin Nucl Med*, 2014;39(10):e413–e426. doi:10.1097/RLU.0000000000000547

[15] Popescu V *et al*. Optimizing parameter choice for FSL-Brain Extraction Tool (BET) on 3D T1 images in multiple sclerosis. Neuroimage. 2012 julio 16;61(4):1484-94. doi: 10.1016/j.neuroimage.2012.03.074. Epub 2012 Mar 30.

[16] Sarraf, Saman and DeSouza, Danielle D. and Anderson, John and Tofighi, Ghassem and , DeepAD: Alzheimer's Disease Classification via Deep Convolutional Neural Networks using MRI and fMRI. *bioRxiv*, 14 de enero de 2017, doi: 10.1101/070441.

[17] S.M. Smith. Fast robust automated brain extraction. Human Brain Mapping, 17(3):143-155, Noviembre 2002.

[18] Yechong Huang, Jiahang Xu, Yuncheng Zhou, Tong Tong y Xiahai Zhuang. Diagnosis of Alzheimer's Disease via Multi-modality 3D Convolutional Neural Network. arXiv:1902.09904 [cs.CV], martes 26 de febrero de 2019.

APÉNDICE A

Redes neuronales convolucionales (CNN)

Las CNN son un tipo de red neuronal diseñado para encontrar patrones en los datos a partir de los cuales extraer características útiles para la clasificación. Estos pueden ser imágenes u otro tipo de datos que se puedan representar mediante matrices. Aunque este tipo de red tiene su mayor auge en el reconocimiento de imagen.

Lo que permite a estas redes encontrar patrones son las capas convolucionales. Las cuales aplican esta operación de convolución a la matriz de entrada a partir de otra llamada *kernel* que tendrá que aprender durante el entrenamiento. La operación consiste en multiplicar el *kernel* elemento a elemento con una submatriz del mismo tamaño de la matriz de entrada y sumar los resultados de las multiplicaciones, esto nos da como resultado un número de la matriz resultante, que para obtenerla habra que repetir esta operación deslizando el *kernel* sobre la matriz a modo de ventana deslizante, como se puede ver en la figura A.1.

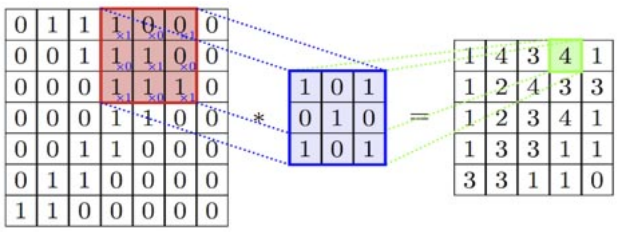

Figura A.1: La matriz roja es la submatriz que se multiplica con el *kernel* (matriz azul) para obtener el valor resultado tras la suma de las multiplicaciones (píxel verde).

Estas capas convolucionales suelen ir seguidas de otras llamadas *pooling layers* que se encargan de reducir la dimensionalidad de las matrices para así disminuir la cantidad de parámetros y operaciones de la red neuronal. Estas capas utilizan una ventana deslizante sobre la matriz de entrada y sacan un valor resultado por cada posición de la ventana, que suele ser el valor máximo o la media de los valores. Normalmente se usan ventanas de 2x2 y un deslizamiento de la ventana de 2 píxeles en horizontal y 2 en vertical, lo que resulta en una reducción a la mitad de la matriz de entrada, véase en la figura A.2. La idea de tomar el valor máximo o la media de estos es reducir las dimensiones intentando preservar donde están las activaciones más relevantes.

Estas capas de convolución y *pooling* se van apilando para ir extrayendo características cada vez más complejas de la imagen, pudiéndose reconocer elementos más básicos como líneas en las primeras capas y a medida que se aumenta el número de capas se crean conceptos más complejos como figuras u objetos. Lo que da como resultado al final de es-

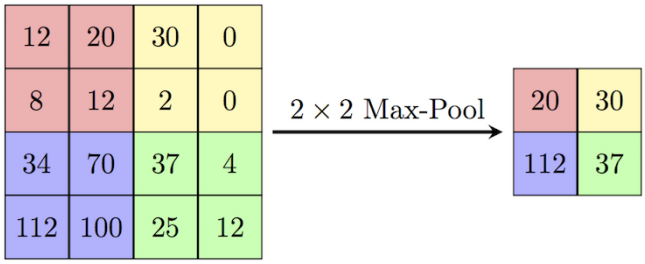

Figura A.2: Operación *max pooling* con una ventana de 2x2 y desplazamiento de 2 píxeles en horizontal y vertical.

te bloque convolucional una matriz con la representación compacta de las características de la imagen.

Finalmente, esta matriz resultante se estira[1] en forma de vector unidimensional para introducirlo a una red neuronal tradicional que sacará el resultado final de la clasificación.

[1] A esta operación se le denomina *flatten*.

APÉNDICE B
Autoencoders

Los autoencoders son redes cuyo objetivo es dada una imagen de entrada devolver la misma imagen a la salida. La dificultad de esta tarea que realizan está en el cuello de botella que estas redes poseen, como se puede ver en la figura B.1. Este estrechamiento en la dimensionalidad hace que la red tenga que aprender una representación más compacta de la imagen de entrada para poder comprimirla y posteriormente descomprimirla en una imagen lo más parecida posible a la original.

Uno de los usos de este tipo de redes es el de preentrenar los pesos de capas convolucionales que posteriormente serán utilizadas en otra red para una tarea distinta, como por ejemplo clasificación. En este trabajo se entrena un autoencoder para posteriormente utilizar sólo la parte del encoder que ya tendrá unos pesos preentrenados para extraer ciertas características de la imagen. Esto es de ayuda debido a que la nueva red partirá de unos pesos que ya serán de utilidad y facilitaran la optimización de la red durante el entrenamiento.

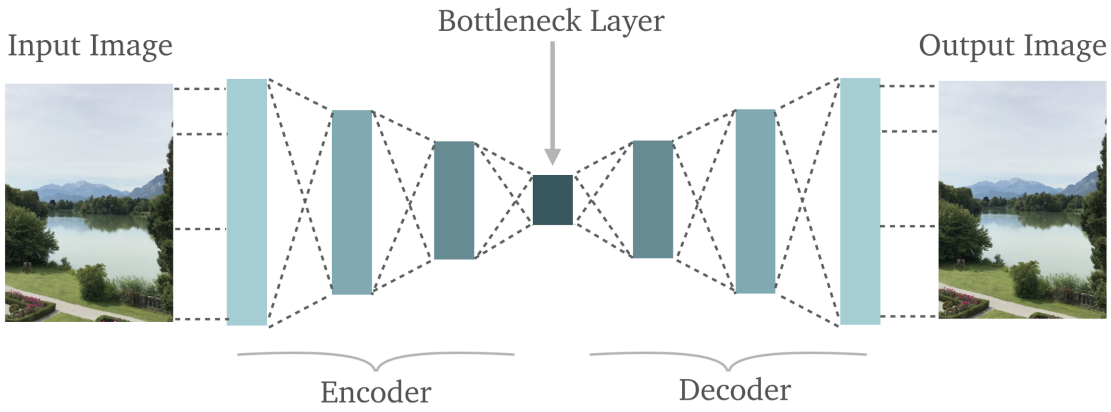

Figura B.1: Estructura de un autoencoder. La parte del encoder se encarga de extraer las características de la imagen para crear una representación compacta en el cuello de botella de la red (*bottleneck*) y el decoder reconstruye la imagen original a partir de esta representación creada por el encoder.

APÉNDICE C
Atlas cerebral

Los atlas cerebrales son modelos que determinan las zonas que tiene un cerebro. Se pueden encontrar diversos atlas según el objetivo del análisis o la procedencia. Un par de ejemplos de atlas que hemos utilizado con FreeSurfer serian los de la figura C.1.

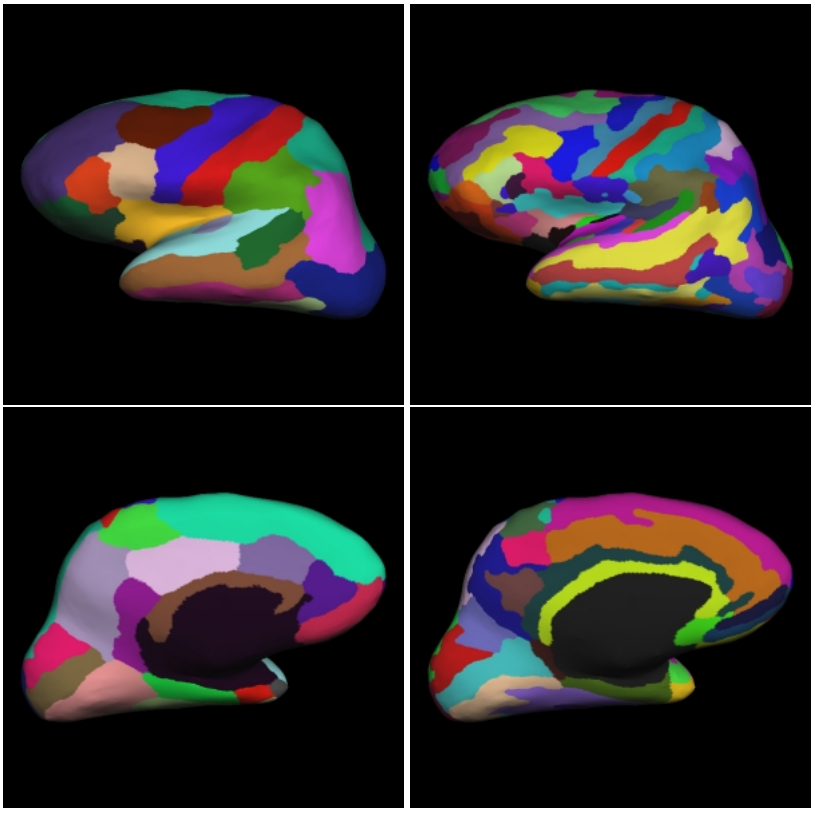

Figura C.1: En la columna de la izquierda se pueden apreciar las parcelas del atlas Desikan-Killiany y en la derecha las del Destrieux.

APÉNDICE D
Vanishing-gradient

Durante el entrenamiento de las redes, el algoritmo de *backpropagation* transmite el error desde la capa de salida hacia atrás variando los pesos de la red, donde parte de la variación depende de la cantidad de error propagado a ese punto de la red. A medida que el error va pasando por las capas va disminuyendo pudiendo llegar a ser demasiado pequeño al llegar a las primeras capas y por tanto realizando modificaciones muy pequeñas en los pesos. Esto hace que a las primeras capas les cueste mucho entrenar y por tanto que la red no aprenda bien. Ya que las siguientes capas van a depender de las activaciones que devuelvan las primeras. Y a este problema se le denomina *vanishing-gradient*.