UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Extracción de cráneo en imágenes de resonancia magnética del cerebro utilizando una red neuronal convolucional 3D

Trabajo Final de Grado Realizado en la Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

por

David Rodríguez Castelló

En cumplimiento parcial

de los requerimientos para el grado en

Ciencias y Tecnologías de la Telecomunicación

Tutores: Verónica Vilaplana Besler y Adrià Casamitjana Díaz

Barcelona, Enero 2017

Resumen

El objetivo principal de la extracción de cráneo es separar totalmente el cráneo del cerebro en imágenes de resonancia magnética, consiguiendo así todas las regiones ocupadas por tejido cerebral. Este problema ha sido ampliamente estudiado dado a su gran aplicación en diferentes técnicas en el campo de la Neuroimagen.

Los métodos usados actualmente obtienen buenos resultados pero no perfectos. El objetivo de este proyecto es estudiar soluciones basadas en redes convolucionales 3D sobre imágenes de resonancia magnética para conseguir un método robusto que pueda llevar a cabo esta extracción con los mejores resultados posibles.

Además, este proyecto utilizará una base de datos pública usada típicamente en la literatura de la extracción de cráneo. El usar estos datos facilitará la comparación de los métodos implementados con los propuestos en los diferentes papers.

Resum

L'objectiu principal de l'Skull Stripping és separar totalment el crani del cervell en imatge de ressonància magnètica, aconseguint així totes les regions ocupades per teixit cerebral. Aquest problema ha sigut àmpliament estudiat donat a la seva gran aplicació en diferents tècniques en el camp de la Neuroimatge.

Els mètodes utilitzats actualment presenten bons resultats però no perfectes. L'objectiu d'aquest projecte és estudiar solucions basades en xarxes convolucionals 3D sobre imatges de ressonància magnètica per aconseguir un mètode robust que pugui portar a terme l'extracció del cervell amb els millors resultats possibles.

A més a més, aquest projecte utilitzarà bases de dades públiques típicament utilitzades en la literatura de Skull Stripping. Utilizar aquestes dades ens ajudarà a comparar els mètodes implementats sobre els proposts als diferents papers.

Abstract

The skull-stripping problem focuses on removing the skull from a magnetic resonance image in order to obtain the regions occupied by brain tissue. The problem has been extensively studied since brain tissue extraction from MRI is crucial for many neuroimaging workflows.

Current methods provide good results but not perfect. The goal of this project will be to study solutions based on 3D-CNNs that are as robust as possible to make this brain extraction focusing in obtaining good results.

Moreover, in our work we will use some of the public databases used in the skull-tripping literature to facilitate the comparison between our final implemented method and the different methods proposed in those papers.

Agradecimientos

Me gustaría agradecer en primera instancia toda la dedicación y ayuda proporcionada por Verónica y Adrià tanto en temas teóricos como de implementación. He aprendido muchísimo en el desarrollo de este proyecto gracias a ellos.

También me gustaría dar las gracias a todos los compañeros y amigos del grado, puesto que sin ellos todo habría sido mucho más difícil el llegar a la meta final del mismo.

Por último me gustaría agradecer todo el apoyo y paciencia que ha tenido conmigo mi familia y, sobre todo, mi pareja, siempre estando para mí en los momentos más arduos y difíciles.

Historial de revisiones y registro de aprobación

Revisión	Fecha	Objetivo
0	27/11/2016	Creación del documento
1	02/01/2017	Revisión del documento
2	07/01/2017	Revisión del documento
3	09/01/2017	Revisión del documento
4	11/01/2017	Revisión del documento
5	13/01/2017	Aprobación del documento

Lista de distribución del proyecto:

Nombre	e-mail
David Rodríguez Castelló	davidrodri.c@gmail.com
Verónica Vilaplana Besler	veronica.vilaplana@upc.edu
Adrià Casamitjana Díaz	adria.casamitjana@upc.edu

Escrito por:		Revisado y aprobado por:	
Fecha	09/01/2017	Fecha	09/01/2017
Nombre	David Rodríguez Castelló	Nombre	Verónica Vilaplana Besler Adrià Castamitjana Díaz
Puesto	Autor del proyecto	Puesto	Tutores del proyecto

Contenido

Resumen 1
Resum 2
Abstract 3
Agradecimientos 4
Historial de revisiones y registro de aprobación 5
Lista de Figuras 8
Lista de Tablas 10
1. Introducción 11
1.1. Motivación 11
1.2. Objetivos 11
1.3. Organización del proyecto 12
1.3.1. Estructura 12
1.3.2. Diagrama de Gantt 13
1.4. Incidencias 13
2. Conceptos básicos de Redes Neuronales Convolucionales 14
2.1. Definición 14
2.2. Estructura 15
2.2.1. Tipos de Capas 15
2.2.2. Función de activación 16
2.2.3. Función Softmax 17
2.3. Entrenamiento 17
2.3.1. Partición de la base de datos 18
2.3.2. Backpropagation 18
2.4. Métricas utilizadas 18
2.4.1. Accuracy 19
2.4.2. Dice 19
2.4.3. Sensitivity 19
2.4.4. Specificity 20
2.4.5. Distancia de Hausdorff 20
3. Análisis del estado del arte 20
3.1. Técnicas más usadas 20

3.1.1. BET (Brain Extraction Tool) 20
3.1.2. BEaST (Brain Extraction based on nonlocal Segmentation Technique) 21
3.1.3. BSE (Brain Surface Extraction) 21
3.1.4. HWA (Hybrid Watershed Algorithm) 21
3.1.5. 3dSkullStrip 21
3.1.6. RoBEx (Robust Brain Extraction) 21
3.1.7. CNN 22
4. Metodología y desarrollo del proyecto 22
4.1. Descripción del proyecto 22
4.2. Entorno de programación 22
4.3. Arquitectura u_net 23
4.4. Base de datos LPBA40 24
4.5. Muestreo 24
4.6. Aplicación 25
4.6.1. Dimensión de entrada variante 25
4.6.2. K-Fold 27
4.6.3. Data augmentation 30
4.6.4. Resumen de resultados en test 31
5. Resultados 32
5.1. Comparativa con el estado del arte 32
6. Presupuesto 35
7. Conclusiones y futuros desarrollos 36
Bibliografía 37

Lista de Figuras

Figura 1 – Diagrama de Gantt 13
Figura 2 – Neurona 14
Figura 3 – Estructura CNN 15
Figura 4 – Fórmula ReLU 16
Figura 5 – Función ReLU 16
Figura 6 – Fórmula Softmax 17
Figura 7 – Fórmula Accuracy 19
Figura 8 – Fórmula Dice 19
Figura 9 – Fórmula Sensitivity 19
Figura 10 – Fórmula Specificity 20
Figura 11 – Fórmula Distancia Hausdorff 20
Figura 12 – U_net 23
Figura 13 – Distribución real 25
Figura 14 – Distribución equilibrada 25
Figura 15 – Sujeto 3 (32^3) 25
Figura 16 – Sujeto 19 (32^3) 25
Figura 17 – Sujeto 3 (Simple) 26
Figura 18 – Sujeto 19 (Simple) 26
Figura 19 – Gráfica Loss (Simple) 26
Figura 20 – Gráfica Dice (Simple) 26
Figura 21 – Gráfica Accuracy (Simple) 26
Figura 22 – Sujeto 3 (Fold 1) 27
Figura 23 – Sujeto 19 (Fold 2) 27
Figura 24 – Sujeto 26 (Fold 3) 27
Figura 25 – Sujeto 34 (Fold 4) 27
Figura 26 – Accuracy (Fold 1) 28
Figura 27 – Dice (Fold 1) 28
Figura 28 – Loss (Fold 1) 28
Figura 29 – Accuracy (Fold 2) 28
Figura 30 – Dice (Fold 2) 28
Figura 31 – Loss (Fold 2) 28
Figura 32 – Accuracy (Fold 3) 29

Figura 33 – Dice (Fold 3) 29
Figura 34 – Loss (Fold 3) 29
Figura 35 – Accuracy (Fold 4) 29
Figura 36 – Dice (Fold 4) 29
Figura 37 – Loss (Fold 4) 29
Figura 38 – MRI S01 30
Figura 39 – MRI S01 invertido 30
Figura 40 – S03 (DA) 30
Figura 41 – S19 (DA) 30
Figura 42 – Accuracy (DA) 31
Figura 43 – Dice (DA) 31
Figura 44 – Loss (DA) 31
Figura 45 – Dice estado del arte 34

Lista de Tablas

Tabla 1 – Arquitectura CNN 22

Tabla 2 – Estructura de capas u_net 23

Tabla 3 – Métricas Train (64^3) 26

Tabla 4 – Particiones K-fold 28

Tabla 5 – Métricas Validación (K-Fold) 29

Tabla 6 – Métricas en validación (DA) 31

Tabla 7 - Métricas Simple vs 4-fold vs Data Augmentation 32

Tabla 8 – Resultados implementación 33

Tabla 9 – Resultados estado del arte 33

1. Introducción

1.1. Motivación

En la actualidad un importante problema médico que se le plantea a la sociedad es el incremento de aparición de tumores cerebrales y patologías que afectan directamente al tejido cerebral. Una de las complicaciones que traen estas enfermedades es el desconocimiento de ellas y, por tanto, la dificultad de poderlas detectar de forma temprana para poder aplicar al paciente un tratamiento precoz y más efectivo.

Hay muchos campos que se dedican a estudiar diferentes patologías cerebrales (como tumores, Alzheimer, esclerosis múltiple, etc.) basándose en los vóxeles del tejido cerebral de imágenes de resonancia magnética. Sobre todo en sujetos con patologías esta extracción no es para nada trivial y los métodos usados para los sujetos sanos introducen errores que afectan directamente sobre la información del tejido cerebral extraído.

Es por ello que esta extracción ha de hacerse minuciosamente, evitando errores para así obtener la información real del tejido cerebral y poder mejorar tanto los estudios de estas patologías como ayudar en el diagnóstico de las mismas.

1.2. Objetivos

El principal objetivo de este proyecto es desarrollar un sistema, haciendo uso de las redes convolucionales 3D, que pueda llevar a cabo esta extracción de cráneo de imágenes de resonancia magnética con las mejores prestaciones posibles.

Para ello será importante el hacer un estudio sobre los métodos más utilizados recientemente y, sobre todo, de la aplicación de las redes neuronales en estos campos (dado que es una tecnología en descubrimiento y con un gran crecimiento en los últimos años en muchas aplicaciones de clasificación y detección).

Un importante objetivo, como se ha comentado anteriormente, sería el investigar y conseguir que el sistema sea robusto al tipo de sujeto que haya en la entrada, idealmente consiguiendo una perfecta separación del cerebro.

1.3. Organización del proyecto

1.3.1. Estructura

Este documento consta de diferentes partes:

2 - Conceptos básicos de Redes Neuronales: En este apartado se hará un breve resumen de redes neuronales y, en concreto, redes neuronales convolucionales puesto que han sido las estudiadas y utilizadas durante el desarrollo del proyecto.

3 - Análisis del estado del arte: En este apartado se hará un pequeño recorrido histórico de las herramientas y metodologías que han existido para resolver el problema de la extracción del cráneo y, sobre todo, enfatizar en las últimas y mejores técnicas del estado del arte. Es importante hacer este análisis dado a la posterior necesidad de tener alguna tabla comparativa para poder evaluar la eficacia final de nuestro método respecto los creados por la comunidad científica.

4 - Metodología y desarrollo del proyecto: En este apartado se resumirá cómo ha sido el enfoque llevado en el proyecto y qué decisiones, testeos y mejoras se han ido aplicando al sistema hasta llegar al prototipo final.

5 - Resultados: En este apartado se estará centrado en los resultados del sistema final, comparándolos con los resultados de otros sistemas anteriormente mencionados. Con esta comparativa se pretende evaluar finalmente cómo funciona el sistema final.

6 - Presupuesto: Este breve apartado consta de un pequeño estudio teórico de los costes que representaría a cualquier empresa o entidad el llevar a cabo el sistema final. Teniendo en cuenta las horas dedicadas y recursos utilizados para la realización del proyecto.

7 - Conclusiones y vías de desarrollo: Por último se acabará con las conclusiones finales del proyecto y, por tanto, del sistema que se ha acabado implementando. También se comentarán las posibles mejoras y pruebas que se podrían aplicar al sistema para poder mejorar los resultados. Además también se dejarán abiertos otros caminos de interés por los cuales este proyecto podría extenderse.

1.3.2. Diagrama de Gantt

A continuación se añade el diagrama de Gantt sobre cómo ha sido la planificación y organización del proyecto:

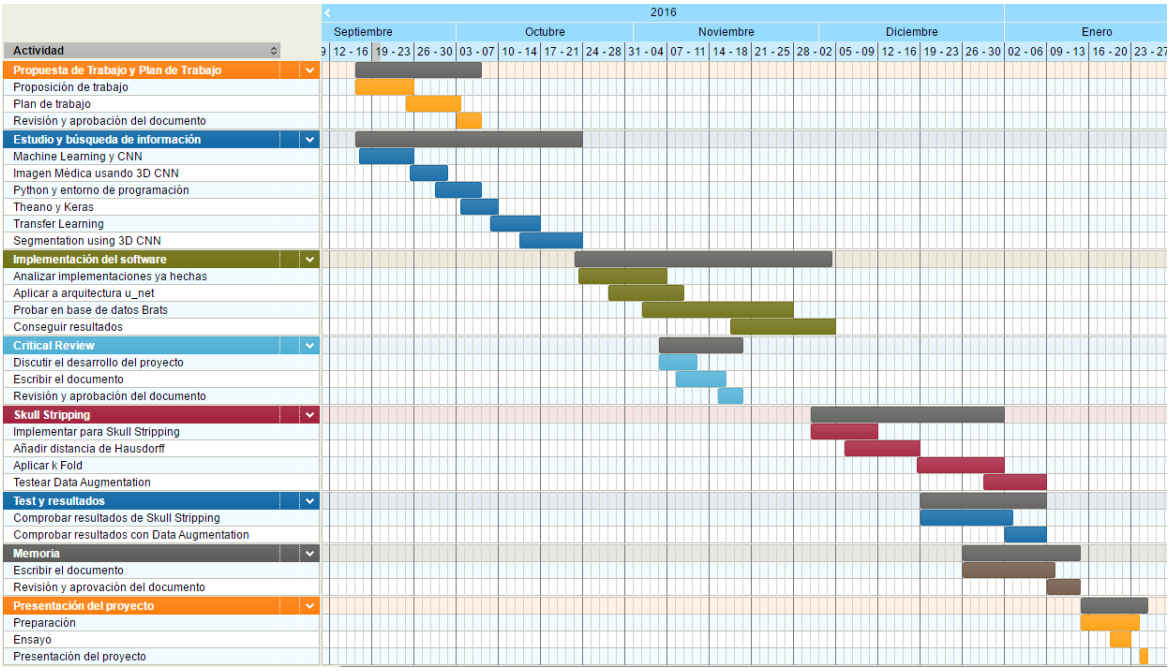


Figura 1 - Diagrama de Gantt

1.4. Incidencias

Después de hacer un estudio intensivo de las diferentes aplicaciones y enfoques de las redes convolucionales en la imagen médica se decidió el continuar el proyecto hacia la rama de la extracción del cráneo en imágenes de resonancia magnética.

Esta decisión fue afectada por la necesidad de tener un método preciso y robusto de segmentación de cráneo dado que los que están ya implementados dan ciertos problemas con sujetos con patologías.

Esta elección dejó de lado otras aplicaciones de las redes convolucionales que en un primer momento se estudiaron como podrían ser la segmentación de tumores o el trasladar estos mismos sistemas en la segmentación de tumores en otros órganos.

2. Conceptos básicos de Redes Neuronales Convolucionales

2.1. Definición

Antes de entrar en los conceptos de las redes neuronales convolucionales se ha de definir lo que es una red neuronal. Una red neuronal es un sistema de aprendizaje automático inspirado en el sistema nervioso biológico. Este sistema se crea a partir de la interconexión de diferentes elementos inteligentes e independientes llamados neuronas. Estos sistemas siguen una estructura por capas, con diferente número de neuronas en cada capa. Estas interconexiones son ponderadas por ciertos pesos (que se irán automáticamente aprendiendo y mejorando), los cuales multiplican la salida de la neurona anterior. Seguidamente, se aplica un sumador a todas las entradas de cada neurona de la siguiente capa para acabar calculando una función no lineal (llamada función de activación) a cada una de estas sumas. Este comportamiento se representa en la siguiente figura:

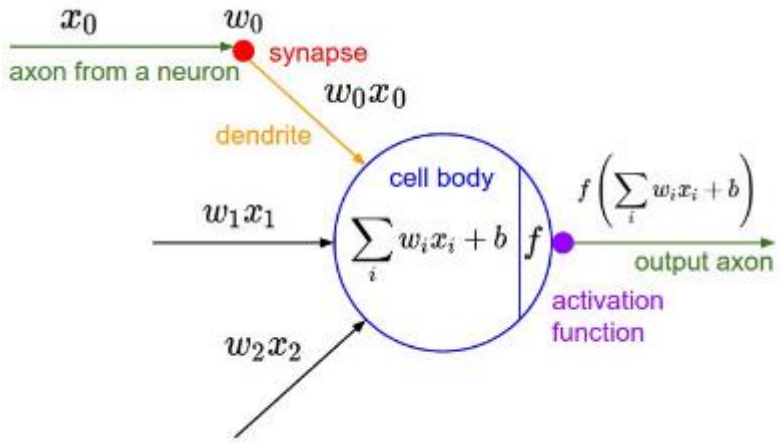


Figura 2 - Neurona

Una red convolucional, es una generalización de la red neuronal clásica. Su nombre viene dado a que al hacer este incremento de dimensión, nuestros pesos ya no serán números si no que se convertirán en matrices n-dimensionales, conocidas como filtros, que aplican una convolución a la entrada para obtener su salida.

Este incremento de pesos que calcular y entrenar nos obligará a tener una potencia computacional mucho mayor, siendo estos sistemas bastante más costosos (en recursos y tiempo) de entrenar y utilizar que los clásicos. A partir de este momento vamos a especializarnos en hablar sobre las redes neuronales convolucionales puesto que son las utilizadas en el proyecto.

2.2. Estructura

Como se ha comentado anteriormente, una red tiene una estructura por capas. Cada capa puede tener una función diferente respecto los datos de entrada puesto que no todas las capas son utilizadas para clasificar o sacar las dependencias, como explicaremos en el siguiente subapartado.

Además de la división de capas por funcionalidad también se hace otra definición por posición según su lugar físico en la arquitectura. Hay tres tipos de capa diferentes: capa de entrada, capa oculta y capa de salida. La capa de entrada y salida son la primera y última capa de la red respectivamente. Todas las capas que quedan entre estas dos son las llamadas ocultas. Esta estructura se representa de forma esquemática en la siguiente figura:

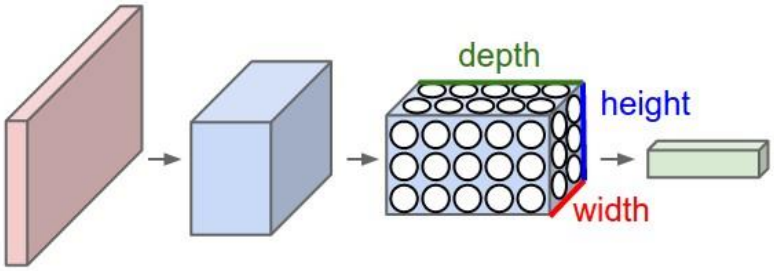


Figura 3 - Estructura CNN

2.2.1. Tipos de Capas

Como se ha comentado anteriormente hay diferentes tipos de capas importantes para usar en las redes convolucionales, cada una con diferente uso.

Las más habituales y usadas son:

- **Convolucional:** Esta capa aplica la matriz de pesos como un filtro, calculando sus salidas mediante una convolución dónde, seguidamente, se le aplicará una función de activación a la salida. Esta capa es la capa más usada en cualquier arquitectura puesto que es el núcleo de la red. Es importante tener claro que cada capa convolucional puede aplicar n filtros diferentes a la información de la entrada, creando así n diferentes resultados que se irán concatenando en profundidad.
- **Pooling:** Esta capa fue creada por las dificultades de cálculo cuando la entrada de la red tiene demasiada información. Se encarga de reducir en parámetros la información de su entrada aplicando una operación llamada pooling. Esta capa se encarga de definir un filtro con unas dimensiones WxHxD que recorrerá el volumen reduciendo bloques de WxHxD vóxeles en un vóxel de salida. El tipo de pooling más utilizado es el llamado Max-Pooling, el cual hace este proceso escogiendo el valor más grande de entre todos los vóxeles de dentro del volumen del filtro.
- **Upsampling:** Esta capa es la contraria al Pooling, transforma la matriz de información en una más grande transformando 1 voxel en un volumen de WxHxD vóxeles.
- **Fully-Connected:** Esta capa suele estar situada justo antes de la función Softmax y es la que ayuda a la red a hacer una clasificación. Esta capa se podría entender como una capa convolucional pero con filtro de tamaño unitario (transformando nuestro sistema en una red de neuronas unidimensionales) y con cada neurona totalmente conectada a cada vóxel de la capa anterior.

- **Capa Softmax:** Esta capa suele ser la última de la red convolucional. Su funcionalidad es la de transformar los resultados de la capa fully-connected en resultados probabilísticos para poder llevar a cabo una clasificación basándose en las probabilidades de cada clase.

2.2.2. Función de activación

La función de activación es aquella aplicada en cada neurona después del proceso convolutivo. Es una de las partes más importantes de la red. Esta función se caracteriza por ser no lineal y añadir el componente no lineal a la red, ya que sin ella la red sería una concatenación de operaciones lineales que no conseguirían una buena clasificación.

Hay muchos tipos de función de activación que se han ido utilizando y probando como las funciones sigmoide o tanh. Aun así la función ReLU (Rectified Linear Unit) es la más utilizada a día de hoy puesto que es una de las más sencillas y de las más efectivas en términos de la clasificación y computación. Uno de los problemas más importantes que soluciona es el del desvanecimiento de gradientes. Este problema ocurre en el entrenamiento de una red utilizando algoritmos basados en gradiente y backpropagation. En estos métodos, cada peso recibe una actualización proporcional al gradiente de la función de error respecto el valor del peso en ese instante. El algoritmo de backpropagation calcula estos gradientes utilizando la regla de la cadena. Esto conlleva a multiplicar n valores pequeños para calcular los gradientes de los pesos de las primeras capas en una red de n-capas, implicando un entreno muy lento en las primeras capas de la red.

2.2.2.1. ReLU

Se basa en hacer un máximo entre 0 y el valor de entrada, eliminando así cualquier valor negativo que la red haya podido crear. Su ecuación sigue esta forma:

$$\boldsymbol{f(x) = \max(0, x)}$$

Figura 4 – Fórmula ReLU

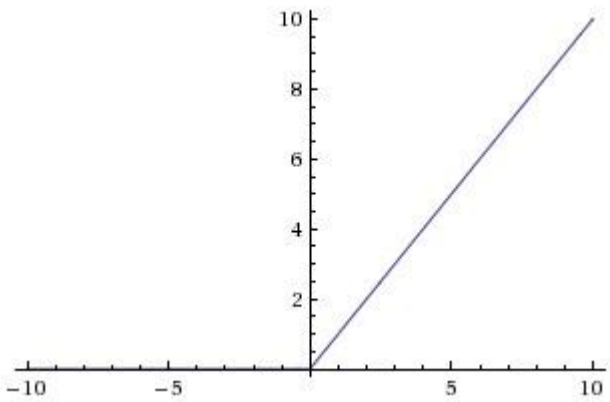


Figura 5 – Función ReLU

En la gráfica anterior se puede observar función ReLU.

Su simpleza trae a la red muchos beneficios computacionales sin perder capacidad de clasificación, siendo por eso la más utilizada en el momento.

El uso de esta función tiene estas ventajas y desventajas:

- (+) Acelera importantemente la convergencia del algoritmo de Gradiente Descendente comparando con otras funciones como tanh o sigmoide.
- (+) Puede ser implementada con operaciones simples. Simplemente ha de aplicar un umbral en cero.
- (-) Son bastante frágiles en el entrenamiento y pueden llegar a "morir". Puede ser que de la forma que se actualicen los pesos lleguen a un punto dónde estas unidades ReLU nunca más volverán a ser activadas, siendo cero para siempre.

2.2.3. Función Softmax

Esta función, como ha sido anteriormente comentado, transforma todos los valores de la capa Fully-connected a valores probabilísticos (limitando el rango de valores entre 0 y 1), ayudando así a la clasificación. La función se puede expresar como:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Figura 6 – Fórmula Softmax

Dónde z_j es el valor de una neurona de la capa Fully-connected.

Esta función transforma los valores de la capa fully connected a una estimación sobre la probabilidad de que cada vóxel forme parte de cada una de las clases definidas. De esta forma, se clasificará ese vóxel por el valor de la probabilidad más grande de entre todas las probabilidades de clase.

2.3. Entrenamiento

El entrenamiento es la etapa por la que pasa cualquier red para aprender todos sus pesos hasta clasificar (dado un problema o función de coste) los datos de entrada. Este proceso es uno de los más complejos y costosos en la creación de una red puesto que es dónde se computarán iterativamente todos los valores de los pesos de los filtros. El objetivo que sigue este proceso es el llegar a obtener la mejor clasificación posible de los datos de entrada.

Antes de empezar con el entrenamiento se han de definir ciertas particiones en la base de datos. Se definen la partición de entrenamiento y de test. La partición de entrenamiento será aquella que se use para entrenar la red, por tanto tendrá que estar etiquetada (deberemos saber los valores reales de cada vóxel de información). La partición de test será utilizada para evaluar el funcionamiento de la red cuando ésta esté entrenada.

Cuando se haya definido esta partición, se puede empezar con el entrenamiento. Para empezar con el proceso, una vez creada la arquitectura de la red, hemos de inicializar los pesos en algún valor para poder empezar a iterar el algoritmo y empezar a encontrar los pesos idóneos para nuestra entrada. Es muy importante entender que los pesos ideales de la red dependen totalmente de cómo son los datos de entrada y con los que entrenamos, o sea la red es muy dependiente de

la aplicación para la cual haya sido entrenada. Esta inicialización de los datos puede hacerse de muchas formas diferentes pero en este proyecto se utiliza una inicialización aleatoria de los pesos entre unos valores acotados.

Seguidamente, todos los sujetos de la partición de entrenamiento serán utilizados como entrada para clasificarlos uno a uno. Puesto que tenemos el valor real de cada vóxel de información en las etiquetas, podemos comprobar mediante una comparativa de las etiquetas con la clasificación de la red cómo está clasificando. En este momento se aplica un algoritmo elegido (normalmente el algoritmo de backpropagation) para optimizar los valores de los pesos de los filtros de la red para conseguir una mejor clasificación. Este proceso se itera para cada sujeto de la base de datos de entrenamiento y durante unas cuantas épocas. Se le denomina Época al instante en que todos los sujetos de la base de datos de entrenamiento han sido clasificados por el sistema y los pesos han sido actualizados.

2.3.1. Partición de la base de datos

Como ya se ha comentado, se definen diferentes particiones en la base de datos para conseguir entrenar de una forma adecuada la red. Una mala elección de estas particiones nos podría llevar a una mala generalización o a un mal entrenamiento de la red.

Toda base de datos para entrenar debe tener al menos dos particiones: entrenamiento y test. Aproximadamente la partición de entrenamiento debe tener un 70%/80% de los casos y el resto la de test. En algunos casos se añade una partición extra que divide la partición de entrenamiento, llamada partición de validación. Esta partición suele ser de un 25/30% de la partición de entrenamiento. La partición de validación es utilizada para evaluar (normalmente después de cada época) el entrenamiento que se ha llevado a cabo con los sujetos de entrenamiento y utilizar esta información para optimizar los valores de los filtros de la red.

2.3.2. Backpropagation

Uno de los pasos más importantes del entrenamiento, como ya ha sido comentado, es la optimización de los pesos de los filtros de la red. Esta optimización se hace mediante el proceso de backpropagation, el cual para cada volumen clasificado calcula el error que se ha producido en la clasificación y actualiza los valores de estos filtros para minimizarlo mediante un algoritmo de optimización (en nuestro caso el de Gradiente Descendente).

Como anteriormente se ha comentado, este algoritmo se basa en aplicar la regla de la cadena para calcular de una forma más eficiente los gradientes del algoritmo de optimización. Estos cálculos de gradientes se efectúan de capas finales hacia el principio de la red, de aquí viene su nombre.

La aplicación de esta técnica mejora mucho la eficiencia de cálculo de los gradientes y reduce el tiempo de entrenamiento de la red.

2.4. Métricas utilizadas

Para poder evaluar el trabajo que desempeñan los sistemas de clasificación se han definido diferentes métricas. Normalmente se calculan en la fase de entrenamiento y validación para saber cómo está evolucionando el entrenamiento del sistema (dado que es aquí dónde disponemos de las etiquetas de cada dato de entrada de la red) y en test para poder evaluar su trabajo. En nuestro caso vamos a considerar que tenemos dos clases binarias, por tanto nuestras dos posibles clases serán 1 y 0.

Estas métricas se calculan mediante la comparación de la clasificación obtenida por la red y los valores reales que muestran las etiquetas. La mayoría de ellas se calculan mediante los conceptos de true positive, true negative, false positive y false negative.

- **True positive:** Son aquellos resultados clasificados como 1 cuando la etiqueta es 1.
- **True negative:** Son aquellos resultados clasificados como 0 cuando la etiqueta es 0.
- **False positive:** Son aquellos resultados clasificados como 1 cuando la etiqueta es 0.
- **False negative:** Son aquellos resultados clasificados como 0 cuando la etiqueta es 1.

Unos valores de false negative y false positive altos nos estarían denotando un mal funcionamiento de nuestra red para el tipo de entrada que le estamos dando.

2.4.1. Accuracy

La accuracy es la métrica (acotada entre 0 y 1) que calcula cuantos resultados hemos clasificado bien respecto todos los valores clasificados. Su fórmula:

$$A = \frac{TP + FP}{TP + FP + TN + FN}$$

Figura 7 – Fórmula Accuracy

Esta medida puede darnos una idea global de cómo funciona de bien la red pero no es una buena métrica en el caso de tener clases no balanceadas (puesto que sólo evalúa la clase positiva).

2.4.2. Dice

Esta métrica, al contrario que el accuracy, tiene en cuenta los errores y los aciertos de la red neuronal, y no solamente los aciertos. Su fórmula:

$$D = \frac{2 * TP}{2 * TP + FN + FP}$$

Figura 8 – Fórmula Dice

Como podemos observar sólo conseguirá su valor ideal (D = 1) cuando los false negative y los false positive sean igual a 0, o sea, la clasificación haya sido perfecta, sin ningún error.

2.4.3. Sensitivity

Esta métrica, junto con la de Specificity ha sido directamente utilizada para llevar a cabo la comparación del sistema implementado con otros sistemas dado que en los artículos científicos seleccionados se utilizan estas métricas.

La sensitivity se utiliza para comprobar el porcentaje de aciertos de la clase positiva en la red, sin tener en cuenta los false positive. Sigue la siguiente fórmula:

$$Sn = \frac{TP}{TP + FN}$$

Figura 9 – Fórmula Sensitivity

2.4.4. Specificity

La specificity se centra en calcular el porcentaje de aciertos de la clase negativa en la red, sin tener en cuenta los false negative. Su fórmula:

$$Sn = \frac{TN}{TN + FP}$$

Figura 10 – Fórmula Specificity

2.4.5. Distancia de Hausdorff

Esta distancia mide la diferencia que hay entre dos conjuntos de puntos. En el sistema implementado acaba obteniendo un volumen el cual será clasificado como cerebro, es por ello que esta métrica es interesante para la aplicación puesto que nos da una medida de distancia entre el contorno del volumen clasificado con las etiquetas del mismo. Su fórmula:

$$d_H(X,Y) = \inf\{\delta : A \subset B_\delta \wedge B \subset A_\delta\}$$

Dónde

$$A_\delta = \{x : \mathrm{dist}(x,A) < \delta\}, \quad B_\delta = \{x : \mathrm{dist}(x,B) < \delta\}$$

Figura 11 – Fórmula Distancia Hausdorff

3. Análisis del estado del arte

3.1. Técnicas más usadas

El Skull Stripping es un importante problema investigado en la neuroimagen dado a su necesidad de aplicación en cualquier estudio técnico con imágenes MRI del cerebro. Es por ello que existen una gran variedad de métodos y técnicas diferentes para intentar resolver este problema de la mejor forma posible. En este apartado se comentarán brevemente algunas técnicas para, más adelante, compararlas con los resultados que se obtendrán del sistema implementado en este proyecto.

3.1.1. BET (Brain Extraction Tool)

El método BET **[2]** es uno de los más clásicos utilizados para Skull Stripping y se basa en la diferencias de intensidad entre el background o el cráneo del cerebro y el tejido cerebral. Tiene diferentes procesos que llevar a cabo para conseguir una segmentación del cráneo.

- Primero calcula un histograma por nivel de intensidad en la imagen para así eliminar toda la zona que sea background. Este paso creará un umbral.
- Seguidamente se aplica una binarización a la imagen aprovechando el umbral calculado en el primer proceso. Además, con esta binarización, se busca el centro de gravedad de la esfera creada por la binarización y se crea una esfera con este centro y el radio del volumen.
- Por último se vuelve a los niveles normales de intensidad de la imagen y mediante una triangularización de la esfera se modela la esfera a la forma del cerebro en iteraciones de un solo vértice por iteración.

Para este método se ha de definir un momento dónde la iteración debe acabar ya que puede ser que nunca llegue a una segmentación óptima.

3.1.2. BEaST (Brain Extraction based on nonlocal Segmentation Technique)

El método BEaST **[3]** se basa en la segmentación por parches de imágenes MRI del cerebro. Cada vóxel será clasificado creando un parche junto con los vóxeles más cercanos y comparándolo con diferentes parches similares dentro de una librería con N imágenes etiquetadas. Para definir la similitud entre dos parches se utilizará la distancia euclídea, siendo 0 para dos parches totalmente iguales. Dado a la gran potencia computacional que requiere comparar cada parche creado por cada vóxel de una imagen con todos los parches posibles en las N imágenes de la librería es importante el implementar algún método para liberar de potencia computacional al método.

Como este método hace comparación directa de las intensidades de dos parches de imágenes distintas es muy importante aplicar una normalización espacial y de intensidad en todas las imágenes de la base de datos y la imagen en estudio.

3.1.3. BSE (Brain Surface Extraction)

El método BSE **[6]** se basa en la detección de diferencias en la intensidad de la imagen. Para ello difumina la imagen de tal forma que los cambios pequeños de intensidad los elimina pero los grandes cambios los mantiene. En este punto hace una detección de los cambios de intensidad que se han mantenido después del primero procesado. Después de este paso aplica una erosión al resultado, provocando que las conexiones menores de 2 vóxeles se rompan, para así eliminar los cambios bruscos de intensidad dentro del cerebro. Como último paso expande la máscara dónde el método encuentra tejido craneal.

3.1.4. HWA (Hybrid Watershed Algorithm)

El método HWA **[7]** se parece mucho al BSE puesto que también se basa en la segmentación de cráneo por diferencias de intensidad. Este tipo de algoritmos se centran en segmentar imágenes por componentes conectadas, entendiendo como desconectadas dos segmentos con un gradiente de intensidad entre los dos. Hace la suposición de que hay conectividad total de la materia blanca del cerebro y, por tanto, divide los vóxeles de tejido cerebral de los exteriores. Por último aplica un modelo deformable para encontrar la máscara del cerebro.

3.1.5. 3dSkullStrip

El método 3dSkullStrip **[5]** se basa en la técnica BET, añadiendo un preprocesado para eliminar artefactos o impurezas de la imagen que puedan dañar a la clasificación. Después aplica una superficie esférica que envuelve el cerebro de forma parecida a cómo lo hace el BET y, por último, crea la máscara cerebral.

3.1.6. RoBEx (Robust Brain Extraction)

El método RoBEx **[7]** se basa en el uso de árboles aleatorios de decisión. Para llegar a la clasificación le aplica a la información de entrada varios procesos. Primero normaliza la intensidad y aplica una corrección de media de la señal, colocando la media en cero. Esta información procesada entra en un árbol aleatorio de decisión el cual está entrenado para identificar vóxeles de la superficie cerebral. Esta clasificación de la superficie define un volumen cuya intensidad representa la

probabilidad que un vóxel esté colocado en la frontera. Esta salida se mejora permitiendo una pequeña deformación del modelo para encajar con la frontera real.

Una de las grandes aplicaciones y ventajas de este método es su robustez con sujetos con tumores y en la segmentación de tumores.

3.1.7. CNN

El método CNN **[4]** es el más parecido al nuestro ya que se compone de una red neuronal convolucional 3D. Esta red consiste en siete capas convolucionales ocultas y una capa de salida softmax, para hacer la clasificación. La entrada de este modelo es de 53^3 vóxeles. La estructura aplicada en este método sigue esta tabla:

CNN architecture details.

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8
Filter size	4	5	5	5	5	5	5	1
Number of filters	16	24	28	34	42	50	50	2
Layer input size [voxels^3]	65	31	27	23	19	15	11	7
Layer output size [voxels^3]	31	27	23	19	15	11	7	7

Tabla 1 – Arquitectura CNN

En este caso se aplicó un modelo de 2-fold en entrenamiento. Este método es el más interesante de comparar con nuestro modelo dado a sus similitudes con nuestro modelo.

4. Metodología y desarrollo del proyecto

4.1. Descripción del proyecto

El proyecto que ha sido llevado a cabo ha investigado el problema de la extracción de cráneo utilizando redes neuronales convolucionales 3D. El uso de tres dimensiones es interesante dado el tipo de datos de entrada del cual partimos, Imágenes de Resonancia Magnética (MRI), las cuales son volúmenes de información 3D. Estas imágenes tienen la particularidad de ser volúmenes en 3D, teniendo vóxeles (extensión del píxel en tres dimensiones) de información.

Hasta la fecha típicamente se han aplicado sistemas más clásicos de clasificación para solventar el problema de la extracción de cráneo. La mayoría de casos de redes utilizadas para esta aplicación han sido redes neuronales convolucionales 2D, puesto que están mucho más documentadas y han sido muy utilizadas por la comunidad científica, facilitando la implemetanción y aplicación de las mismas en otros campos. El problema de estas redes bidimensionales es que no explotan las dependencias en profundidad que pueden llegar a tener estos volúmenes.

4.2. Entorno de programación

En este caso se ha utilizado Python como lenguaje principal de programación. Python es el lenguaje más utilizado dentro de la comunidad científica de Machine Learning. Este lenguaje tiene ya implementadas diferentes librerías y frameworks que ayudan mucho en la creación y prueba de todo tipo de redes. Además Python ha demostrado ser un lenguaje sencillo, potente y eficiente en este tipo de programación dónde los tiempos computacionales son tan altos.

También se ha hecho uso del entorno Theano, especializado en programación de redes neuronales convolucionales utilizando el framework de Keras, como ayuda o suplemento adicional a las librerías ya implementadas para Deep Learning y Machine Learning en general. La elección de este entorno y framework ha venido dada por su sencillez de aplicación.

4.3. Arquitectura u_net

La arquitectura analizada e implementada en todas las pruebas es una versión de la denominada u_net **[8]**. Esta arquitectura se basa en un modelo para datos 3D **[9]** de 32 capas, dónde 30 son capas ocultas. Esta arquitectura se basa en ir reduciendo el tamaño de la entrada mediante poolings (añadiendo capas convolucionales) y haciendo concatenaciones entre los datos de la red.

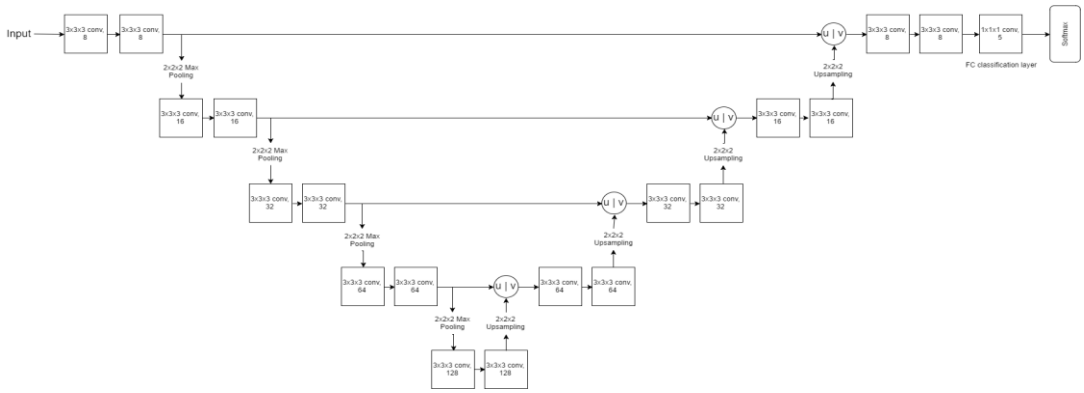


Figura 12 – U_net

En la tabla siguiente se puede observar la distribución de las capas en esta arquitectura:

	Convolucionales						Max-Pooling	UpSampling	Concate	Softmax
Núm. Capas	4	4	4	4	2	1	4	4	4	1
Núm. Filtros	8	16	32	64	128	5	1	1	1	1
Tamaño	3x3x3	3x3x3	3x3x3	3x3x3	3x3x3	1x1x1	2x2x2	2x2x2		

Tabla 2 – Estructura de capas u_net

Como se puede ver utiliza la mayoría de las capas más comunes en las redes neuronales convolucionales, exceptuando la capa de concatenado. Esta capa concatena espacialmente los dos flujos entrantes de información, colocándolos en profundidad uno detrás de otro.

Esta arquitectura tiene una restricción importante, cualquier dimensión de la entrada debe ser un múltiplo de 32 dado a que si no lo fuera, las divisiones que crean los poolings no serían enteras y podría provocar problemas dentro de la red. Por tanto, esta red no acepta ninguna entrada dónde alguna de sus dimensiones sea menor a 32.

4.4. Base de datos LPBA40

Para el entrenamiento y testeo de la red se aplicó como entrada de datos los sujetos de la base LPBA40. Esta base de datos contiene 40 sujetos sanos con imágenes de resonancia magnética de sus cerebros. Esta base de datos contiene tanto las imágenes como las etiquetas de la máscara del cerebro de las mismas para poder entrenar la red. Las dimensiones de cada sujeto son de 272x144x272 vóxeles. Las etiquetas fueron creadas manualmente por expertos en segmentación de materia gris y blanca.

La elección de la base de datos LPBA40 es dada al uso de ella en un artículo científico dónde se aplica una 3D-CNN a la misma aplicación. De esta forma se facilita la comparación de este sistema con los aplicados en estas publicaciones.

4.5. Muestreo

Para entrenar la red se ha de definir un patrón de muestreo e introducción de información en la red. Puede haber muchos tipos de muestreos diferentes (como por ejemplo podría ser el introducir en la red todo el volumen para llevar a cabo un entrenamiento).

En el caso del entrenamiento aplicado se ha implementado un sistema de muestreo por parches. Este muestreo implica tener dimensiones de entrada más pequeñas que las del volumen real ya que, consiste en crear pequeños volúmenes de una dimensión deseada y tratarlos como entrada. Para esto se han de definir cuantos parches equivalen a pasar una imagen y cómo se escogen estos parches. En el caso de este proyecto, diez parches de un mismo volumen equivalen al volumen entero.

Como se ha comentado, otra decisión importante es cómo escoger los parches de cada imagen. Para ello el sistema se ha basado en equilibrar la distribución de probabilidades de clase en cada imagen. Esta decisión viene dada a que en el caso de tener imágenes muy poco equilibradas (por ejemplo con dos clases, una con 90% de aparición y la otra con el 10% restante) la red tendería a cometer errores de clasificación en las clases con menor aparición. Este comportamiento puede afectar bastante en el caso de la extracción de cráneo, dado que el cerebro es la clase menos probable y sin aplicar el método explicado posteriormente la red podría como clase no cerebro algunas zonas del cerebro, creando agujeros inexistentes.

El método aplicado para distribuir más uniformemente las clases consiste en coger parches centrados en un vóxel que tiene equiprobabilidad de ser de una clase u otra.

Utilizando los sujetos de la base de datos LPBA40 sin aplicar esta probabilidad se puede calcular que, en media, hay un 84.6% de clase NO BRAIN y un 15.4% de clase BRAIN. En cambio al aplicar este proceso las probabilidades se equilibran hasta llegar a un 61.6% y un 39.4% respectivamente. Con este método no hay que excederse en equilibrar la probabilidad de aparición de las clases puesto que si llegásemos a un 50%/50% la red podría llegar a crear artefactos en la zona mayoritaria.

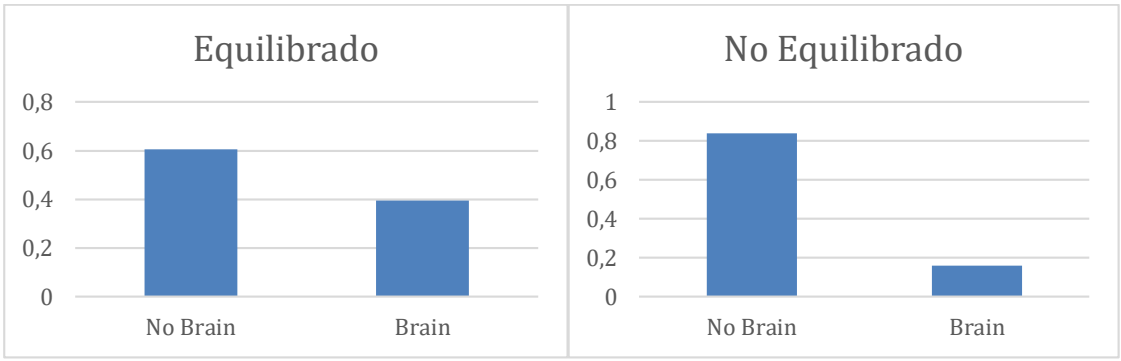


Figura 13 – Distribución real

Figura 14 – Distribución equilibrada

4.6. Aplicación

4.6.1. Dimensión de entrada variante

Para probar la red se efectuaron diferentes entrenamientos, para comprobar cuales daban mejores resultados. Para empezar se aplicaron dos tamaños de parche, observando cómo afectaba esto a los resultados en validación.

Para ello se aplicaron parches de 32x32x32 y de 64x64x64. En esta primera etapa se hizo una partición de 30 sujetos para el entrenamiento y 10 para el testeo. En ambos casos se utilizaron 35 épocas siempre escogiendo el valor de los pesos que dieran mejores resultados y parando la ejecución en caso de que en seis épocas no mejoraran las métricas.

4.6.1.1. Parches de tamaño 32x32x32

Este caso es el que peores resultados da. Esto puede ser por la dimensión de entrada, ya que al ser pequeña crea una entrada en la red que no se asemeja para nada con la imagen de un cerebro, creando problemas en la clasificación correcta de los sujetos.

A continuación se muestran dos sujetos diferentes utilizando este entrenamiento. Para conseguir una diferenciación visual se ha dibujado en azul la máscara de las etiquetas y en blanco la predicción llevada a cabo por la red.

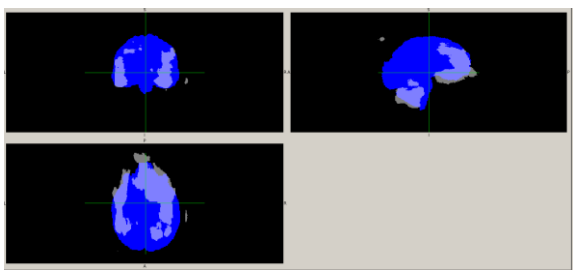


Figura 15 – Sujeto 3 (32^3)

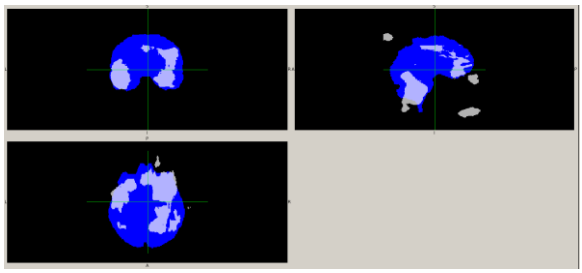


Figura 16 – Sujeto 19 (32^3)

Como se puede observar, la clasificación es bastante mala en todos los cortes del cerebro. Es por ello que este método se descartó y se implementaron otros más precisos y fiables.

4.6.1.2. Entreno simple de una red con parches de 64x64x64

Para intentar mejorar este fenómeno se aplicaron unos parches de entrada más grandes en el entrenamiento, mejorando así la información de entrada de capa parche para la red.

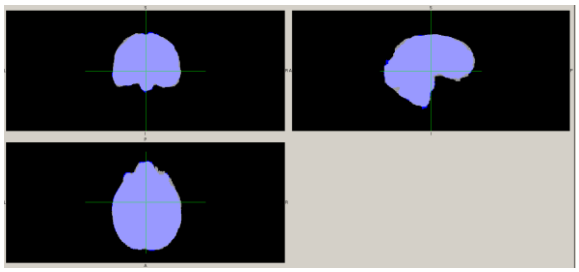


Figura 17 – Sujeto 3 (simple)

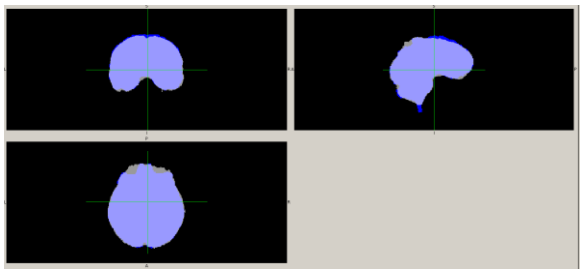


Figura 18 – Sujeto 19 (simple)

En este caso visualmente vemos una gran mejora respecto al uso de parches de 32x32x32. Este método visualmente da muy buenos resultados.

A continuación se adjuntan los resultados de las métricas en entrenamiento y validación respecto las épocas:

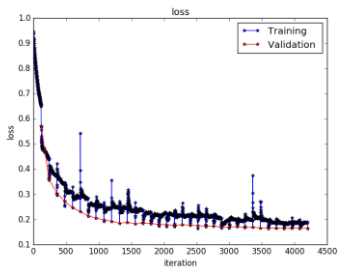


Figura 19 – Gráfica Loss (simple)

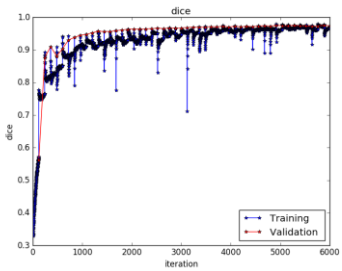


Figura 20 – Gráfica Dice (simple)

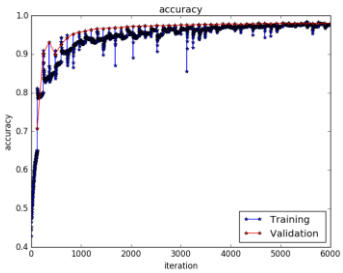


Figura 21 – Gráfica Accuracy (simple)

A continuación se adjuntan los valores numéricos de las métricas utilizadas en el entrenamiento en su última época usando los sujetos de validación:

	Simple
Dice	0,968
Loss	0,153
Accuracy	0,98

Tabla 3 – Métricas Train (simple)

4.6.2. K-Fold

Otro método que se aplicó fue el conocido como k-fold. Este método se suele hacer cuando se dispone de bases de datos pequeñas. Como la base de datos LPBA40 consta de solamente 40 sujetos, se decidió el aplicar este método para comprobar sus resultados. Este método consiste en dividir la base de datos en k particiones dónde una será para test y las restantes para entrenamiento. Después de crear estas particiones itera respecto las particiones y entrena la red K veces, cambiando en cada entrenamiento la partición de test por otra partición hasta acabar con las k redes. A partir de estas cuatro redes entrenadas se extraen los resultados en test y se hace la media. Este proceso consigue utilizar toda la base de datos tanto para entrenar como para testear.

En el caso de este proyecto se utiliza un 4-fold, teniendo que entrenar 4 redes distintas y para promediar los resultados. Esta técnica nos dará unos valores más reales del funcionamiento de la red puesto que tendrá en cuenta diferentes resultados de diferentes entrenamientos distintos usando la misma base de datos.

En este apartado se utilizaron más épocas de entrenamiento, para ver si esto podía mejorar de alguna forma las métricas y la clasificación de la red. En este caso se utilizaron 50 épocas en cambio de 35, alargando su tiempo computacional. A continuación se añade un sujeto test referente a cada fold:

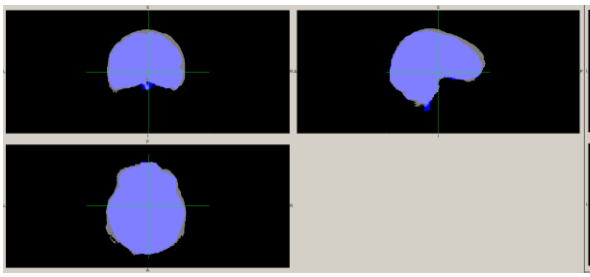


Figura 22 – Sujeto 3 (Fold 1)

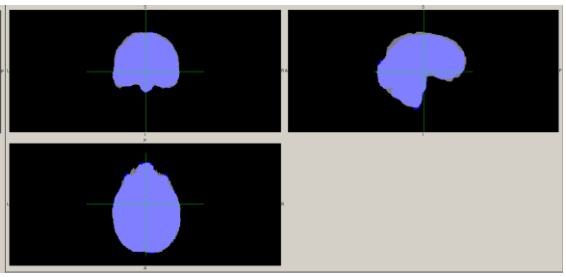


Figura 23 – Sujeto 19 (Fold 2)

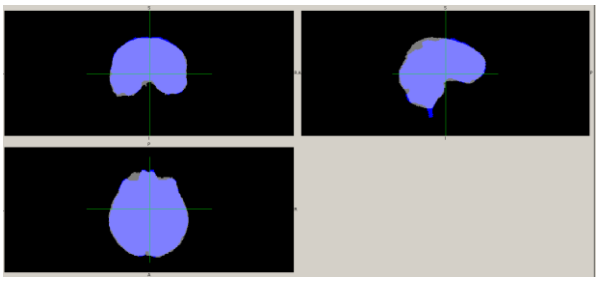


Figura 24 – Sujeto 26 (Fold 3)

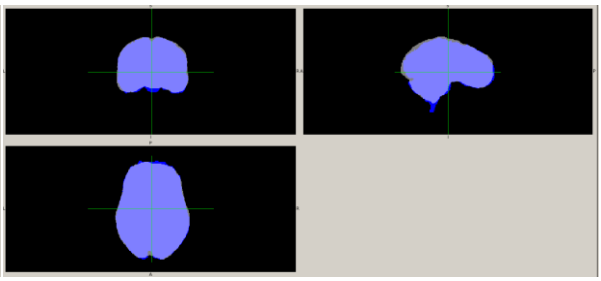


Figura 25 – Sujeto 34 (Fold 4)

Las diferentes particiones creadas para aplicar con el método k-fold contenían los siguientes sujetos:

	Sujetos Train	Sujetos Validación
Fold 1	S10 a S40	S01 a S10
Fold 2	S01 a S10 y S20 a S40	S10 a S20
Fold 3	S01 a S20 y S30 a S40	S20 a S30
Fold 4	S01 a S30	S30 a S40

Tabla 4 – Particiones K-fold

Arriba se han representado cuatro casos distintos (un caso para cada entrenamiento diferente de la red utilizando una partición diferente). El sujeto representado para cada uno de los entrenamientos es un sujeto que pertenece a la partición de validación en cada caso.

Esta técnica ha dado peores resultados visualmente que el entreno simple anteriormente comentado. Como se ha comentado antes, esta técnica da unos resultados más reales del funcionamiento de la red al tener diferentes resultados con distintas particiones promediados.

A continuación se adjuntan la evolución de las métricas de Loss, Accuracy y Dice para el entreno de cada Fold.

Fold 1

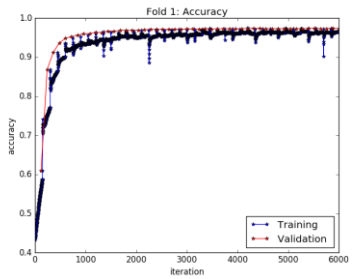


Figura 26 – Accuracy (Fold 1)

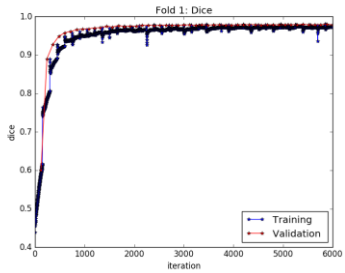
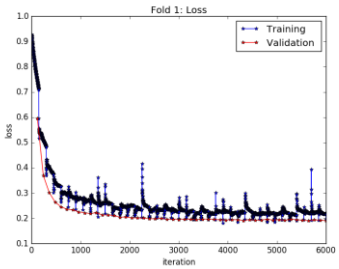


Figura 27 – Dice (Fold 1)



28 –Loss (Fold 1)

Fold 2

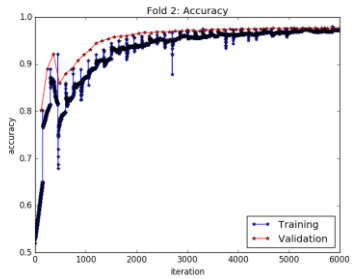


Figura 29 – Accuracy (Fold 2)

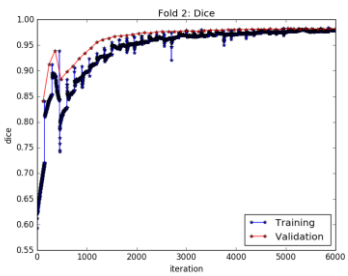


Figura 30 – Dice (Fold 2)

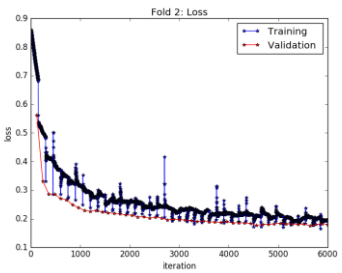


Figura 31 – Loss (Fold 2)

Fold 3

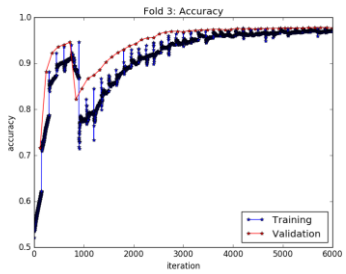


Figura 32 – Accuracy (Fold 3)

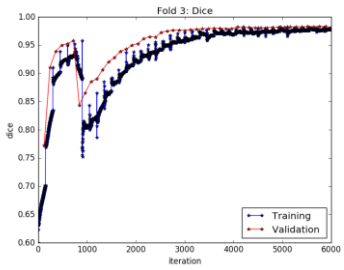


Figura 33 – Dice (Fold 3)

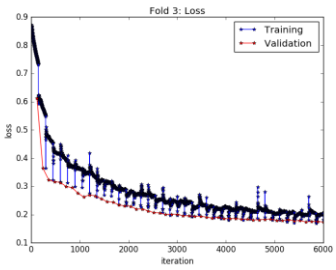


Figura 34 – Loss (Fold 3)

Fold 4

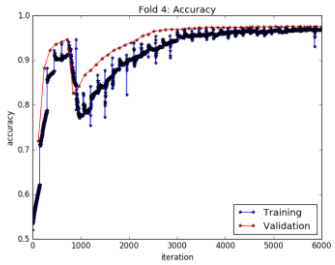


Figura 35 – Accuracy (Fold 4)

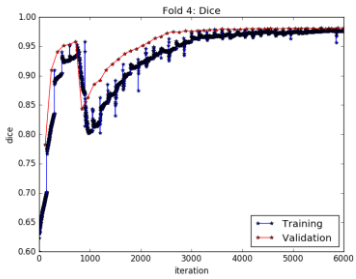


Figura 36 – Dice (Fold 4)

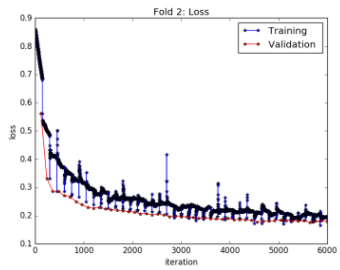


Figura 37 – Loss (Fold 4)

Como se puede observar en las gráficas todas siguen una tendencia buena acorde con los resultados teóricos de entrenamiento de una red. Al final del entreno se calculan unos valores de las métricas en validación representados en la siguiente tabla:

	Fold 1	Fold 2	Fold 3	Fold 4	Mean
Dice	0,979	0,9822	0,9817	0,9814	0,98103
Loss	0,191	0,1796	0,1739	0,1869	0,18275
Accuracy	0,973	0,9774	0,9775	0,9759	0,976

Tabla 4 – Métricas Validación (K-Fold)

La última columna es la columna de la media aritmética de cada métrica de cada fold. Esta columna es la más interesante dado a que tiene en cuenta todos los resultados, siendo los valores que más se asemejan al funcionamiento real de la red.

4.6.3. Data augmentation

Otra técnica que se ha aplicado a la base de datos para mejorar el entrenamiento es la conocida como Data augmentation. Este proceso se basa en hacer más grande la base de datos a partir de cambios de posición de los vóxeles de los sujetos (en general giros o rotaciones de cada sujeto). De esta forma, al tener una base de datos mayor, se pueden conseguir mejores resultados de clasificación.

Para aplicar este proceso es importante que haya una cierta simetría en la información de los vóxeles de cada sujeto puesto que los nuevos sujetos creados al aplicar el proceso deben tener una lógica para la aplicación. En el caso del proyecto, se trata con imágenes de resonancia magnética de cerebros, dónde hay simetría respecto un corte vertical central. Aprovechando este hecho se aplica una inversión desde este plano vertical. A continuación se pone como ejemplo la MRI de un sujeto y, a su lado, el nuevo sujeto creado con esta técnica:

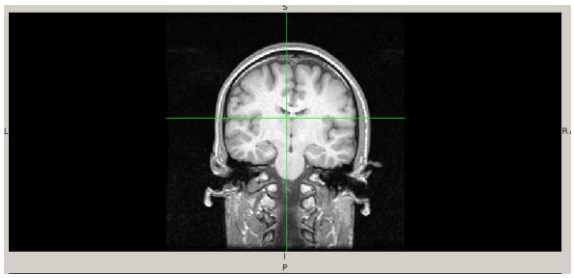


Figura 38 – MRI S01

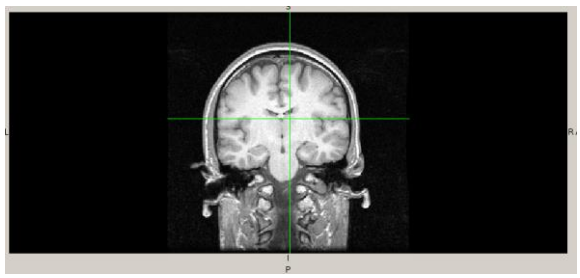


Figura 39 – MRI S01 invertido

Para llevar el entrenamiento de esta red se han utilizado 40 sujetos + 40 nuevos sujetos (invertidos). De aquí 60 se han utilizado para entrenar y 20 para testear. Además se han utilizado parches de 64x64x64 y 50 Épocas.

A continuación se añade la clasificación del cerebro de dos sujetos:

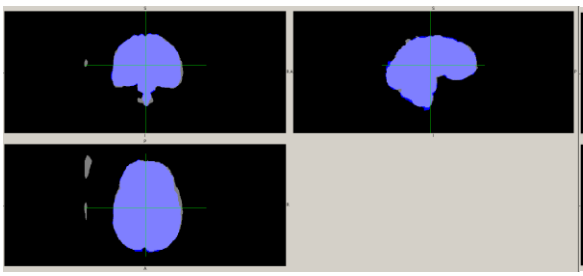


Figura 40 – S03 (DA)

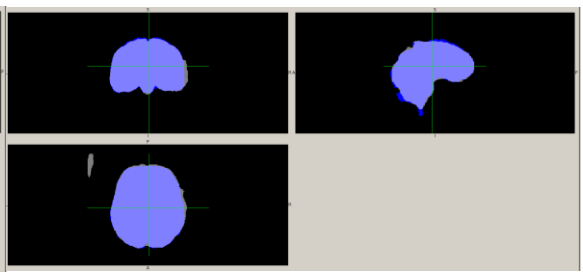


Figura 41 – S19 (DA)

Se puede observar que la clasificación crea unos artefactos (por tanto clasifica como cerebro) a la izquierda del volumen. Aunque esta clasificación es un problema, se puede observar que la clasificación del tejido cerebral tiene bastantes buenos resultados visualmente.

A continuación se añaden la evolución de las métricas en entrenamiento:

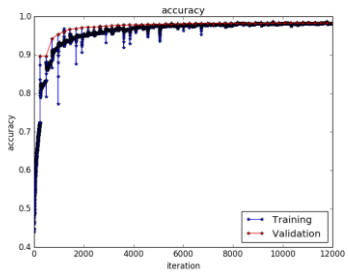


Figura 42 – Accuracy (DA)

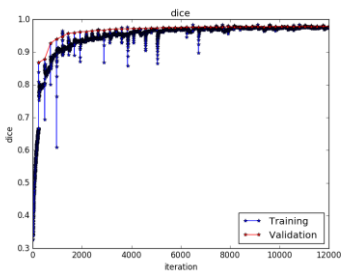


Figura 43 – Dice (DA)

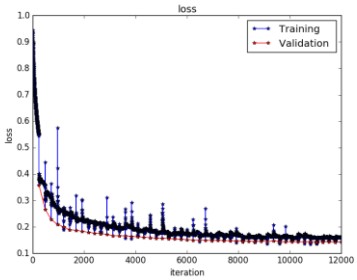


Figura 44 – Loss (DA)

Como se puede observar, la tendencia de las curvas es correcta. Es importante recalcar que en estos casos tenemos el doble de iteraciones que en los casos anteriores (12000 iteraciones) dado a que tenemos el doble de sujetos. Se puede ver que a partir de épocas tempranas ya se llega a unos valores de las métricas buenos.

A continuación se va a adjuntar una tabla sobre las métricas de validación calculadas al acabar el entrenamiento de la red:

	Data Augmentation
Dice	0,9788
Loss	0,143
Accuracy	0,9839

Tabla 7 – Métricas en validación (DA)

Se pueden observar los mejores valores en validación comparando con los otros dos métodos. Este método da buenos resultados pero ha creado artefactos incorrectos en la clasificación. Aun así, dado que estos artefactos están bastante alejados del cerebro se podrían eliminar con un post procesado del volumen.

4.6.4. Resumen de resultados en test

En este apartado se comentaran los resultados numéricos en el testeo de cada entrenamiento explicado en los apartados anteriores. Puesto que los resultados visibles en el caso de entrada de 32x32x32 son muy malos no se añaden a la tabla comparativa de los métodos. Para hacer esta comparación se pondrán los valores medios de Dice, Accuracy, Loss y Distancia de Hausdorff.

Como ya se ha comentado antes los resultados utilizando solamente el entreno simple de una red con parches de entrada de 64x64x64 son algo mejores que los que se aplican con el k-fold. Esto puede ser visible en la clasificación del sujeto puesto que los resultados con k-fold tienen más trozos mal clasificados que el primer método.

A continuación se adjunta la tabla con la media de las métricas evaluando todos los sujetos de la base de datos LPBA40 con cada método. En el caso 4-Fold, se evalúan todos los sujetos con cada red entrenada y a estos resultados se les hace un promedio:

	Simple	4-Fold	Data Augmentation
Loss	0,1498	0,1652	0,1517
Accuracy	0,992	0,9436	0,9887
Dice	0,966	0,91	0,9521
H.Distance	23,14	61,25	100,17

Tabla 7 – Métricas Simple vs 4-fold vs Data Augmentation

Como se puede observar el método que mejores resultados ha dado ha sido el de entrenar una sola red usando parches de 64x64x64 y sin Data Augmentation. Las métricas son bastante parecidas para data augmentation pero algo peores, esto puede ser debido a la incorrecta clasificación de los artefactos que se han visto en los sujetos anteriores. Por último está el método 4-fold con 40 sujetos en la base de datos que, aunque sea el que peores resultados da, es menos dependiente de la partición concreta entre test y entrenamiento que se haya hecho de la base de datos, puesto que se ha hecho un promedio entre cuatro redes entrenadas con diferentes particiones. Por tanto, el método 4-fold da una visión de las métricas más realista de la red.

5. Resultados

5.1. Comparativa con el estado del arte

En este apartado se compararán los resultados obtenidos por el entrenamiento del sistema usando k-fold implementado con algunos de los métodos utilizados en el estado del arte por la comunidad científica. La selección de este método de entrenamiento viene dada a que es el que mejor ilustra el comportamiento real de la red, siendo el más fiable para llevar a cabo esta comparación. Es importante recalcar que el sistema entrenado en este proyecto sólo ha utilizado la base de datos LPBA40 como datos tanto de entrenamiento como de test y los demás métodos han utilizado fusiones de otras bases de datos junto con esta, las cuales serán referenciadas junto con el método en concreto.

Las diferentes bases de datos con las que han sido implementados los otros métodos de clasificación son: ISBR y Oasis.

- **ISBR:** Una base de datos que consta de 18 sujetos sanos. Las etiquetas fueron creadas manualmente por expertos en segmentación de materia gris y blanca.
- **Oasis:** Una base de datos que consta de 77 sujetos, 57 sanos y 20 con algún tipo de demencia. Las etiquetas fueron creadas por un método automático, no tan preciso como en la base de datos anterior.

El hecho de que los métodos implementados en el estado del arte hayan utilizado una base de datos con sujetos con tumores o patologías complica algo la clasificación, haciendo esos sistemas bastante más robustos.

Para llevar a cabo estas comparaciones se deberán utilizar las tablas de métricas proporcionadas en los artículos científicos y calcularlas para nuestro sistema.

	Dice	Sensitivity	Specificity
4-fold	0,91	0,98	0,86

Tabla 8 – Resultados implementación

	Dice	Sensitivity	Specificity
CNN	**95.77 (±0.01)**	94.25 (±0.03)	**99.36 (±0.003)**
BEaST	89.96 (±0.12)	87.89 (±0.12)	98.54 (±0.01)
BET	93.05 (±0.03)	95.47 (±0.05)	97.71 (±0.02)
Robex	95.30 (±0.02)	95.99 (±0.03)	98.81 (±0.01)
3dSkullStrip	92.34 (±0.04)	91.88 (±0.07)	98.45 (±0.01)
HWA	91.37 (±0.03)	**99.06 (±0.01)**	95.88 (±0.01)
BSE	78.77 (±0.10)	95.57 (±0.06)	85.12 (±0.12)

Tabla 9 – Resultados Estado del arte

Comparando los métodos del estado del arte entre sí podemos observar que los peores para la aplicación de Skull Stripping serían BSE y BEaST en promedio. El método CNN es mejor en Dice y Specificity, siendo el que mejor clasifica las dos clases en promedio. HWA, en cambio, es el mejor en Sensitivity, siendo el que mejor clasifica la clase positiva (cerebro).

Como se puede observar en las tablas de resultados, el sistema implementado en términos de Dice se encuentra en un valor parecido al método HWA, superando a BSE y BEaST. En términos de Sensitivity sólo es superado por HWA, siendo el segundo mejor de los métodos. Por último, en términos de Specificity tiene un valor bastante bajo, sólo per encima de BSE, siendo el segundo peor en esta métrica.

Además de estos resultados numéricos también se dispone en el artículo que habla sobre estos métodos **[4]** unas gráficas que representan los resultados de Dice para cada una de las bases de datos anteriormente comentadas. En nuestro caso vamos a comparar los resultados en la base LPBA40 para comparar con el sistema implementado.

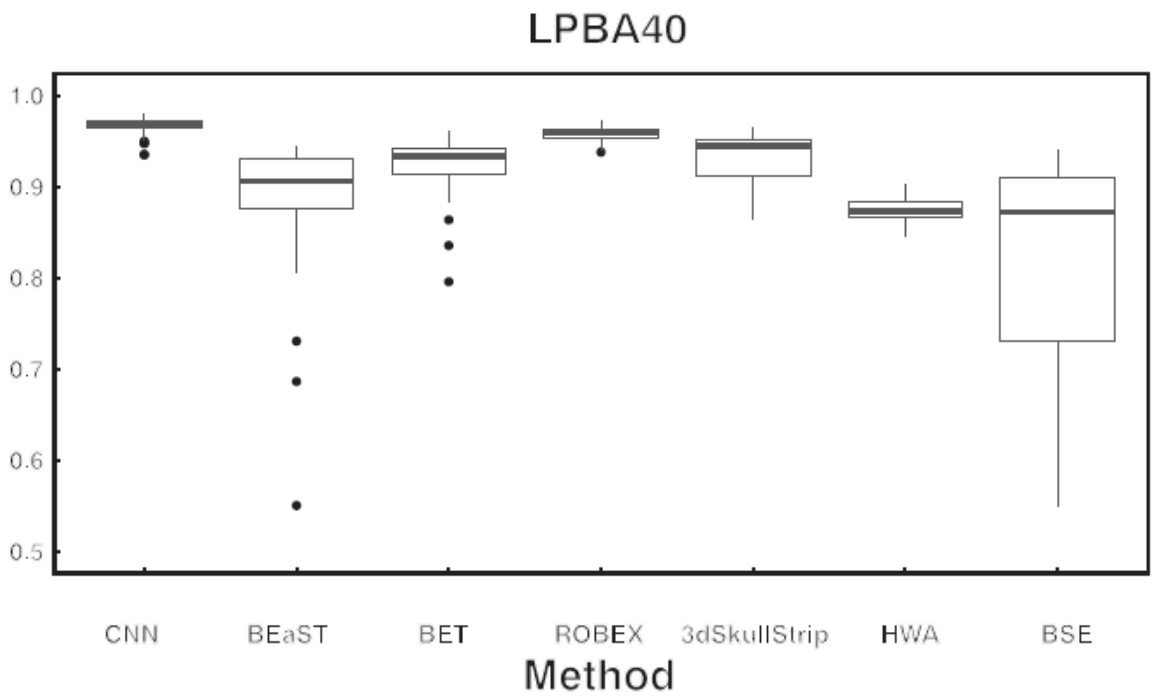


Figura 45 – Dice estado del arte

Podemos observar que las métricas de Dice en este tipo de base de datos son algo mejores que en el promedio de todas (información de la tabla 8). Esto puede ser dado a que la base de datos LPBA40 tiene sujetos sanos, los cuales son más simples de clasificar que los enfermos con tumores o con patologías cerebrales.

Como se ha comentado anteriormente, es interesante el comparar el sistema implementado con el llamado CNN ya que hace uso de 3D CNN. El sistema CNN utiliza como base de datos de entrenamiento las tres bases de datos comentadas (LPBA40, Oasis e ISBR). Esta es una diferencia importante con el sistema implementado dado que la base de datos LPBA40 sólo contiene sujetos sanos y, en cambio, Oasis también incluye algunos casos de sujetos con patologías.

La diferencia principal entre el sistema implementado y el método CNN es la arquitectura. CNN basa su diseño en una cadena de pasos convolucionales cuando el sistema implementado se basa en una estructura de concatenaciones de Poolings y Upsamplings. Es por ello que una opción de mejora del sistema sería el eliminar niveles de pooling y upsampling a la red para comprobar si hay mejoras en el resultado.

El sistema implementado en el proyecto tiene buenos resultados puesto que consigue unas métricas muy buenas para un sistema de clasificación. En comparación con los métodos del estado del arte, pero, es algo inferior en prestaciones, sobre todo comparándolo con los que mejores resultados tiene como podrían ser CNN o Robex. Aun así es un sistema con potencial y posibilidades de mejora.

6. Presupuesto

En esta sección se analizará de forma teórica cual sería el presupuesto del proyecto para poderse llevar a cabo contando el material utilizado para trabajar y las horas empleadas en el desarrollo entero del proyecto.

Puesto que el producto final esperado del proyecto es un software capaz de hacer la separación del cerebro del cráneo, los costes materiales que se le introducirán serán licencias de los programas y librerías necesarios.

El lenguaje de programación utilizado ha sido Python y todas las librerías son de código abierto, por tanto, no incrementan el presupuesto mínimo final del proyecto.

Como IDE de programación en Python ha sido utilizado el programa PyCharm de JetBrains cuya licencia profesional asciende a 20€/mes. Dado que el proyecto ha durado alrededor de 4 meses, se le suma al presupuesto 80€ en gastos de programación.

Otro gasto importante que añadir en el presupuesto es el del coste de las horas aplicadas en el desarrollo del mismo. Aproximadamente se han destinado unas 350h en el desarrollo de todo el proyecto más 40h de tutorías. Considerando el salario estándar de becario marcado por la Universidad Politécnica de Catalunya de 8€/hora, la cifra asciende a 3120€.

Puesto que este proyecto es de investigación y desde el ámbito académico no se incluyen los costes del servidor de computación utilizado, el cual ha sido prestado por la UPC para llevar a cabo este tipo de actividades académicas.

El presupuesto final asciende a los **3200€** en total, siendo este el coste de producción del proyecto.

7. Conclusiones y futuros desarrollos

Como hemos podido observar en apartados anteriores, nuestro sistema consigue buenos resultados en la extracción del cráneo en sujetos sanos.

En la comparativa directa del sistema se ha podido comprobar que se necesitan ciertas mejoras para llegar a unos resultados comparables a los de los mejores métodos implementados en el estado del arte.

Hay diferentes puntos de desarrollo de los cuales se podría mejorar el sistema para mejorar notablemente su trabajo de clasificación:

- **Uso de otras bases de datos:** Un factor que limita mucho el sistema es el uso de una sola base de datos con sólo 40 sujetos en total. El hecho de que hayan pocos sujetos y muy parecidos entre sí provoca una buena clasificación del sistema para sujetos de la misma base de datos pero quizás no generalice lo suficiente para sujetos de otras bases. Es por ello que una mejora importante sería el usar en el entrenamiento diferentes bases de datos (aumentando el número de sujetos tanto en entrenamiento como en test), ayudando así al cálculo de los pesos óptimos para su generalización.
- **Uso de la distancia Hausdorff como métrica de entreno:** En el entreno se utiliza como métricas el Loss, Accuracy y Dice. Una buena idea sería el introducir como métricas de mejora también la distancia Hausdorff ya que por el tipo de aplicación en la que la red está diseñada, hay una relación directa entre una distancia Hausdorff pequeña y un buen funcionamiento de la red.
- **Hacer algún cambio a la arquitectura:** Durante el proyecto siempre se ha mantenido la estructura de la red, sin hacer ningún cambio en su morfología de capas. Una propuesta podría reducir el número de capas de polling y upsampling y comprobar que resultados se consiguen.

Como se ha ido viendo a lo largo del documento, el uso de una base de datos pequeña ha limitado bastante los resultados y la capacidad de clasificación del sistema. Es por ello que siempre el poder utilizar una cantidad mayor de datos para el entreno de la red siempre será beneficioso.

El uso de CNN en la imagen médica puede ser extrapolable a otras aplicaciones interesantes. El campo del Skull Stripping es uno de los más importantes ya que es la primera técnica necesaria para procesar cerebros mediante MRI. Hay otros estudios que parten de aquí, estudios que se centran en diferentes patologías y tumores tanto en cerebro como en otros órganos.

Hay muchos sistemas implementados con CNN que dan muy buenos resultados en aplicaciones dónde hasta ahora se habían utilizado métodos más clásicos, es por ello que el campo de las redes neuronales convolucionales es un campo inmenso por descubrir y estudiar, del cual se conseguirán cosas que hasta la época nos parecen ciencia ficción.

Bibliografía

[1] Curso Machine Learning y CNN de Standford 2016 Disponible: http://cs231n.github.io/

[2] Stephen M. Smith. *"BET: Brain Extraction Tool".* FMRIB technical Report, 2001

[3] Eskildsen SF, Coupé P, Fonov V, Manjón JV, Leung KK, Guizard N, Wassef SN, Østergaard LR, Collins DL. *"BEaST: brain extraction based on nonlocal segmentation technique."* Alzheimer's Disease Neuroimaging Initiative, 2011

[4] Jens Kleesiek, Gregor Urbana, Alexander Hubert, Daniel Schwarz, Klaus Maier-Hein, Martin Bendszus, Armin Biller. *"Deep MRI brain extraction: A 3D convolutional neural network or skull stripping."* 2016

[5] 3dSkullStrip Method: *https://afni.nimh.nih.gov/pub/dist/doc/program_help/3dSkullStrip.html*

[6] BSE Method: *http://brainsuite.org/processing/surfaceextraction/bse/*

[7] Christine Fennema-Notestine, I. Burak Ozyurt, Camellia P. Clark, Shaunna Morris, Amanda Bischoff-Grethe, Mark W. Bondi, Terry L. Jernigan, Bruce Fischl, Florent Segonne, David W. Shattuck, Richard M. Leahy, David E. Rex, Arthur W. Toga, Kelly H. Zou, Morphometry BIRN, and Gregory G. Brown. "*Quantitative Evaluation of Automated Skull-Stripping Methods Applied to Contemporany and Legacy Images: Effects of Diagnosis, Bias Correction, and Slice Location."* 2006

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, German, 2015.

[9] Adrià Casamitjana, Santi Puch, Asier Aduriz, Elisa Sayrol, and Verónica Vilaplana. *"3D Convolutional Networks for Brain Tumor Segmentation".* Signal Theory and Communications Department, Universitat Polit`ecnica de Catalunya. BarcelonaTech, Spain, 2016.