Proyecto Final

Visión por Computador

Brain Tumor Detection

Juan José Herrera Aranda - juanjoha@correo.ugr.es
Silvia Moreno Barroso - e.silviabm98@go.ugr.es

23 de enero de 2022

Índice

1. Introducción

Un tumor cerebral es un crecimiento descontrolado de células derivadas de componentes cerebrales (tumores primarios) o de células tumorales localizadas en otras áreas del organismo (metástasis). Se considera una de las enfermedades más agresivas, tanto en niños como en adultos. Los tumores cerebrales representan del 85 al 90 por ciento de todos los tumores primarios del sistema nervioso central. Cada año, alrededor de 11.700 personas son diagnosticadas con un tumor cerebral. Los tumores cerebrales se clasifican en: tumor benigno, tumor maligno, tumor pituitario, etc.

Se debe implementar un tratamiento, una planificación y un diagnóstico precisos adecuados para mejorar la esperanza de vida de los pacientes. La mejor técnica para detectar tumores cerebrales es la resonancia magnética (MRI). Los escaneos generan una gran cantidad de datos en las imágenes y estas imágenes son examinadas por el radiólogo. La aplicación de técnicas de clasificación automatizada que utilizan Machine Learning e Inteligencia Artificial ha demostrado consistentemente una mayor precisión que la clasificación manual. Por lo tanto, proponer un sistema que realice detección y clasificación mediante el uso de algoritmos de aprendizaje profundo utilizando Convolution Neural Network (CNN) y Transfer Learning (TL) sería útil para los médicos de todo el mundo.

Los objetivos principales a abordar son dos: el primero es, dado un paciente, obtener una predicción con el máximo nivel de precisión posible sobre si tiene o no cáncer. El segundo es que bajo el supuesto de que el paciente tiene cáncer, clasificarlo en los tres tipos de tumores presentes. Se presenta pues, dos problemas de clasificación, el primero es de clasificación binaria y el segundo de clasificación multiclase.

2. Datos

2.1. Base de Datos

El trabajo se ha desarrollado usando la base de datos *Brain Tumor Classification(MRI)* disponible en kaggle en el siguiente enlace (Base de Datos). Se tienen 4 clases, *glioma_tumor* (826 imgs. para entrenamiento y 100 para test), *meningioma_tumor* (822 imgs. para entrenamiento y 115 para test), *pituitary_tumor* (827 imgs. para entrenamiento y 74 para test) y *no_tumor* (395 imgs. para entrenamiento y 105 para test) . Las tres primeras referidas a tres tipos de cánceres y la última a pacientes sin cáncer. La base de datos consta de dos directorios, uno para las imágenes de resonancia magnéticas de entrenamiento y otros para las de test. Dentro de cada directorio cuelgan 4 subdirectorios, uno para cada clase. Las imágenes se presentan en formato tricanal (RGB) y con distintos tamaños.

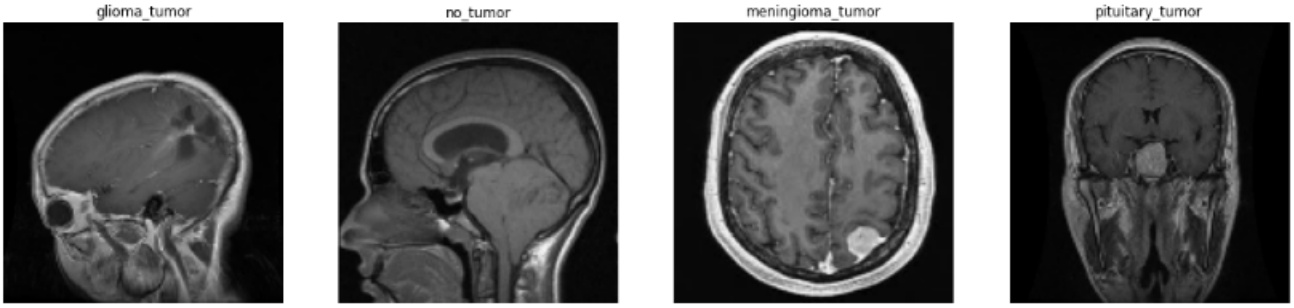

Figura 1: Imagenes correspondientes a cada una de las cuatro clases presentes en el dataset. Fuente

2.2. Preprocesado de imágenes

Para el problema de clasificación binaria, se ha denominado a la clase positiva (denotada como 1) a la clase con cáncer y a la negativa (denotada como 0) a la otra. Haciendo primeramente un estudio sobre el desbalanceo de clases se observa en la Fig (2) un claro desbalanceo en el conjunto de entrenamiento. Por otra parte, en el conjunto de test se tiene un mayor balanceo entre éstas, en cambio, al agrupar las clases con cáncer se observa también cierto desbalanceo.

Puesto que ambos conjuntos (test y entrenamiento) están desbalanceados se puede llegar a considerar que este desbalanceo no merece tratamiento. Sin embargo, al hacer el cociente del numero de instancias sin cáncer frente a las que tienen cáncer obtenemos una razón para el conjunto de entrenamiento de $\frac{395}{2475} \approx 0{,}159$ frente a $\frac{105}{289} \approx 0{,}36$ del conjunto de test. Como no guardan las mismas proporciones, se ha optado por hacer **oversampling** en el conjunto de imágenes sin cáncer, así pues, se han añadido 505 imágenes sintéticas producidas con Keras usando la clase *ImagenDataGenerator* con el objetivo de igualar estas razones. Dichas imágenes sintéticas corresponden a reflexiones horizontales, verticales y a rotaciones de las originales.

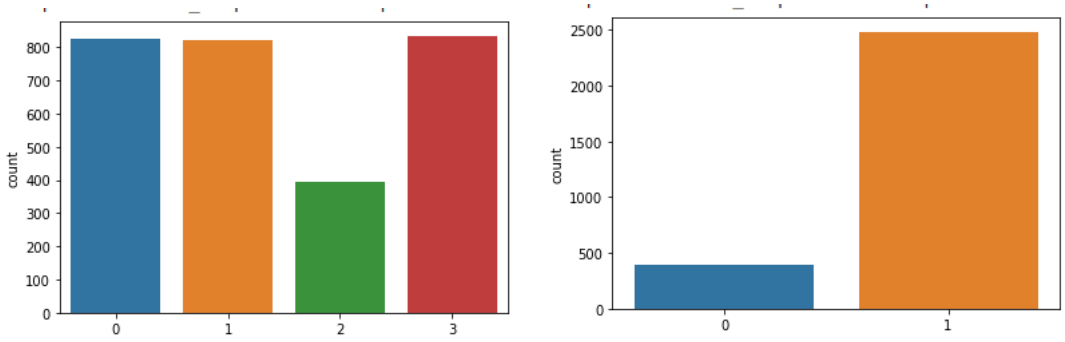

Figura 2: Histogramas referidos al conjunto de entrenamiento. Muestran el número de instancias presentes en las clases. A izquierda se muestran todas las clases, la clase 0 ,1 y 3 corresponden a cáncer y la clase 2 a pacientes sin cáncer. En la imagen de la derecha se han agrupado las clases de cáncer (1) y se muestra frente a las que no tienen cáncer (0)

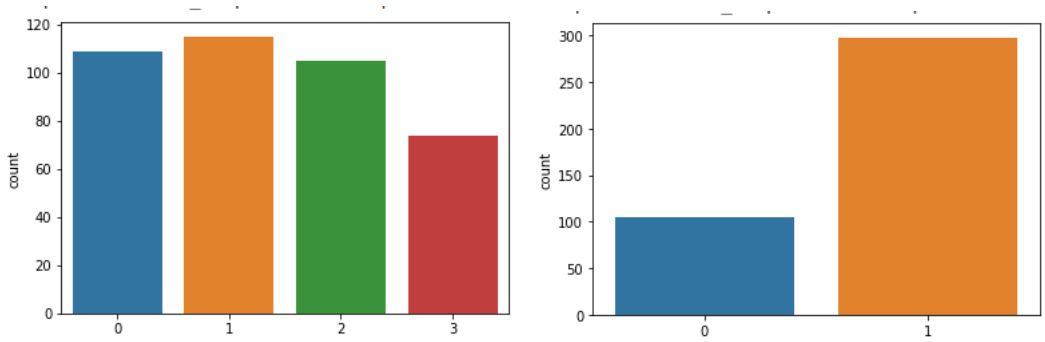

Figura 3: Histogramas referidos al conjunto de test. Muestran a la izquierda el número de instancias que presenta cada clase. A la derecha se han agrupado las clases con cáncer (0, 1 y 3 y se ha denotado como 1) y se muestran junto a la que no lo tienen (denotadas como 0).

De aquí en adelante es un preprocesado común para ambos problemas. La mayoría de las imágenes de la base de datos contiene áreas negras y espacios no deseados, produciendo peores clasificaciones. Por tanto, es necesario recortar las imágenes y eliminar las zonas no deseadas para así usar la información útil de cada imagen. Se usa el método de recortado en [1] y [2] el cual usas el cálculo de extremos. Los pasos para recortar se pueden observar en la figura (Fig.4). Primero se carga la imagen, después se le aplica thresholding para convertirla en una imagen binaria. También se aplican operaciones de dilatado y erosión para eliminar el ruido. Después se selecciona los contornos más largos de la imagen tras aplicarle el threshold y se calculan los cuatro puntos extremos, esto es, el extremo por arriba, por abajado, por la derecha y por la izquierda. Finalmente se recorta la imagen usando la información de contorno y los extremos.

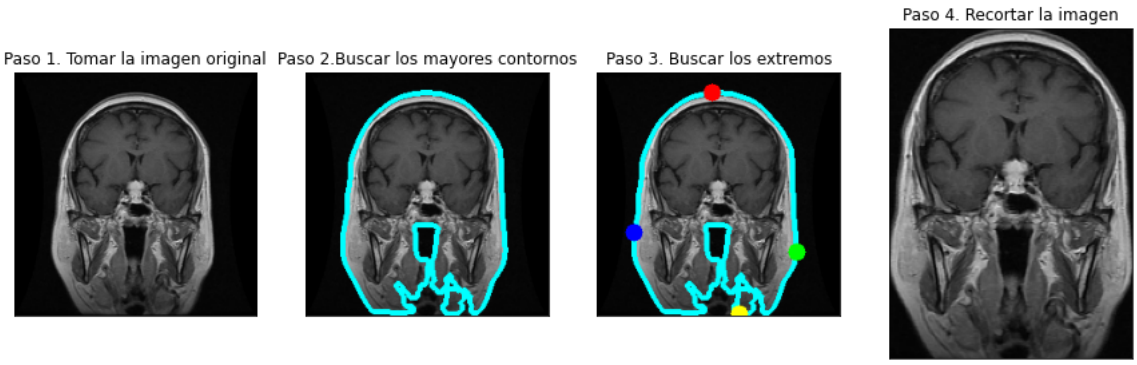

Figura 4: Proceso de recorte de la imagen. Se observa en la primera imagen la presencia de un área negra alrededor del cráneo la cual es eliminada en el último paso tras aplicarle todo el proceso de recortado.

Tras lo anterior, las dimensiones de las imágenes son claramente distintas entre sí, por lo que se procede con un redimensionamiento usando interpolación bicúvica por lo expuesto en [2] ya que crea una curva más suave que otros métodos de interpolación, como por ejemplo, el bilineal, y es mejor opción para las imágenes MR porque hay una mayor cantidad de ruido a través de los filos.

Finalmente, como se va a usar un modelo de DensetNet121 pre-entrenado en ImageNet se va a aplicar el preprocesado sobre éstas imágenes usando la función *preprocess_input* disponible en Keras para DenseNet121. Otros modelos

2.3. Detalles de implementación

Tras la lectura de datos y el preprocesado, las clases están guardadas de forma contigua de tal manera que primero están las instancias de la clase 0, seguidas de la clase 1, clase 2 y clase 3. Entonces, para una mayor diversidad se ha barajado tanto el conjunto de entrenamiento como el de test usando la semilla *123456*. Por otro lado, para poder ejecutar los modelos correctamente se han vectorizado las etiquetas, de tal manera que cada clase se representa con un vector de ceros y unos. Por ejemplo, para el problema multiclase, la clase 2 corresponde con el vector $[0, 1, 0]$ puesto que solo hay tres clases.

3. Fundamentos teóricos

Se ha empleado la red DenseNet121 como arquitectura principal en el trabajo. Esta decisión se ha tomado debido a que muchas investigaciones recientes han demostrado que si una red convolucional contiene conexiones más cortas entre las capas cercanas a la entrada y las capas cercanas a la salida, la red convolucional se puede entrenar de manera más profunda, más precisa y más eficiente. Por tanto, se decidió usar la red DenseNet para resolver el problema.

En base a artículos de investigación donde se compara el uso de otras redes en el diagnostico de tumores cerebrales se obtiene que las redes densas(D-CNN) han obtenido resultado bastante buenos (Fig. 5).

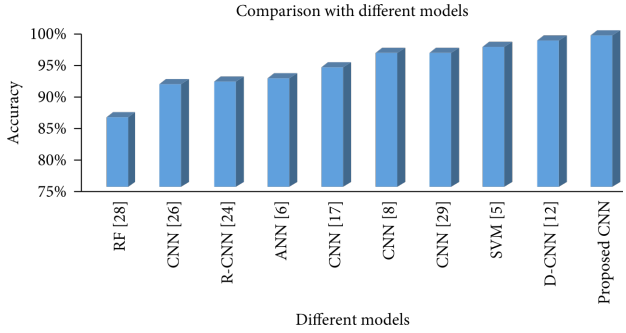

Figura 5: Comparativa realizada con diferentes modelos en problemas de diagnóstico de tumores cerebrales.

La red convolucional densa (DenseNet) conecta cada capa con cada una de las otras capas de una manera feed-forward. Mientras que las redes convolucionales tradicionales con L capas tienen L conexiones -una entre cada capa y su capa siguiente- DenseNet tiene L(L+1)/ 2 conexiones directas. Para cada capa, los mapas de características de todas las capas anteriores se utilizan como entradas, y sus propios mapas de características se utilizan como entrada en todas las capas posteriores. Las redes densas tienen varias ventajas convincentes: alivian el problema de 'vanishing gradients', refuerzan la propagación de características, fomentan la reutilización de características y reducen sustancialmente el número de parámetros. El efecto contraintuitivo de este patrón de conectividad densa es que requiere menos parámetros que las redes convolucionales tradicionales, ya que no es necesario volver a aprender los mapas de características redundantes. Las capas de DenseNet son muy estrechas (por ejemplo, 12 mapas de características por capa), añadiendo sólo un pequeño conjunto de mapas de características al 'conocimiento colectivo' de la red y manteniendo el resto de mapas de características sin cambios, y el clasificador final toma una decisión basada en todos los mapas de características de la red.

Además de una mayor eficiencia de los parámetros, una de las grandes ventajas de las redes densas es la mejora del flujo de información y de los gradientes en toda la red, lo que facilita su entrenamiento. Cada capa tiene acceso directo a los gradientes de la función de pérdida y a la señal de entrada original, lo que da lugar a una supervisión profunda implícita. Esto ayuda al entrenamiento de arquitecturas de red más profundas. Además, también se observa que las conexiones densas tienen un efecto regularizador, que reduce el sobreajuste en tareas con tamaños de conjuntos de entrenamiento más pequeños, como es el caso del dataset con el que se trabaja aquí. La concatenación de mapas de características aprendidos por diferentes capas aumenta la variación en la entrada de las capas siguientes y mejora la eficiencia. Esto constituye una diferencia importante entre las DenseNets y las ResNets. En comparación con las redes Inception, que también concatenan características de diferentes capas, las DenseNets son más simples y eficientes.

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figura 6: Arquitecturas DenseNet para ImageNet.

Cada capa añade k mapas de características propias. La tasa de crecimiento regula la cantidad de información nueva que cada capa aporta al estado global. Una vez escrito, se puede acceder al estado global desde cualquier punto de la red y, a diferencia de las arquitecturas de red tradicionales, no es necesario replicarlo de capa en capa.

Sea x_0 una imagen que se pasa a través de una red convolucional. La red consta de L capas, cada una de las cuales implementa una transformación no-lineal $H_l(\cdot)$ donde el subíndice l hace referencia al número de capa. $H_l(\cdot)$ puede ser una función compuesta de operaciones como BatchNormalization, ReLu, Pooling o convoluciones. Se denota a la salida de la l^{th} capa como x_l. En las redes densas se tiene que la l^{th} capa recibe el mapa de caracteríastias de todas las capas anteriores, $x_0, ..., x_{l-1}$ como entrada, entonces:

$$x_l = H_l([x_0, ..., x_{l-1}]) \tag{1}$$

donde $[x_0, ..., x_{l-1}]$ es la concatenación de los mapas de características producidos en las capas $0, ..., l-1$ visto como un único tensor. Esto es a lo que se refiere la conectividad densa y de ahí el nombre de DenseNet. De manera más visual, se puede apreciar en la figura (Fig. 7) la concatenación de cada salida de las capas previas hacia las siguientes.

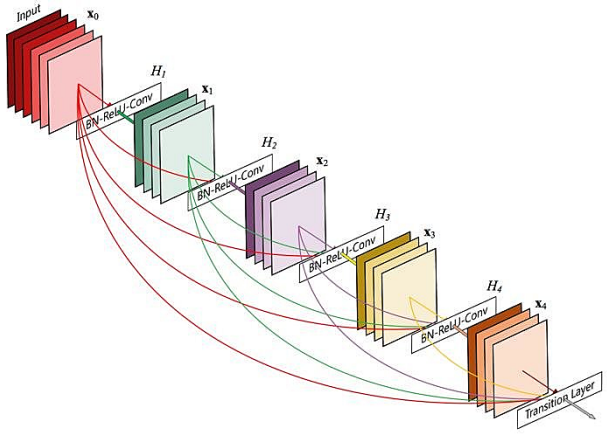

Figura 7: Bloque denso de 5 capas con una tasa de crecimiento k=4. Cada capa toma como entrada todos los valores de características de los anteriores.

4. Metodología

4.1. Aspectos técnicos e hiperparámetros

Para la clasificación multiclase se ha usado la **función de entropía cruzada categórica** como función de pérdida, la cual se define mediante la expresión $loss = -\sum_{i=1}^{n} y_i log\hat{y}_i$ (donde n es el tamaño de salida, y_i es la etiqueta i-ésima e $\hat{y}_i$ es la salida i-ésima del modelo), por ser útil para tareas de clasificación multiclase donde una imagen tiene una sola etiqueta de entre un conjunto de varias. Para la clasificación binaria se ha usado como función de pérdida la llamada **función de entropía cruzada binaria** que es análoga a la anterior pero adaptada a clasificación binaria.

El **optimizador** usado para ambos problemas ha sido Adam porque como se expone en un estudio realizado en [3]: *los algoritmos más recientes como Adam, están construidos en base a sus predecesores, por tanto podremos esperar que su rendimiento sea superior.* Se ha usado como learning_rate un valor de 10^{-3} como en [4] por presentar buenos resultados.

En cuanto al **proceso de validación experimental** decir que se ha aplicado la técnica de cross-validation dividiendo el conjunto de entrenamiento en tres subconjuntos: entrenamiento, validación y test. El conjunto de test se corresponde con un 20 % de los ejemplos de entrenamiento. El conjunto de validación se corresponde al 10 % del 80 % restante y finalmente el conjunto de entrenamiento que son el resto. El número de *folds* empleados ha sido tres debido a las limitaciones de RAM de Google Colab. Dicho proceso de validación experimental se usará para comparar los modelos entre sí para posteriormente hacer una selección de los modelos, entrenarlos con todos los datos exceptuando un 10 % para validación y evaluarlos con los datos de test.

El **tamaño del batch** empleado ha sido 32. La elección se ha realizado por un compromiso entre velocidad de entrenamiento (mayores tamaños del batch se traducen en entrenamiento más rápidos) y requerimientos computacionales. Además, un tamaño de batch muy grande afecta negativamente a la calidad del modelo. El número de **épocas** ha sido 40 porque un número mayor de épocas implicaría mayor tiempo de cómputo y además, como se emplea la técnica de **early stopping** en el entrenamiento, casi nunca se llega a alcanzar dicho valor y en los casos en los que lo hace, los resultados en validación son lo suficientemente buenos como para inferir que no hace suponer seguir entrenando el modelo. El valor de parámetro *patience* de early stopping ha sido 7, ya que un valor mayor producía en los modelos un mayor sobreajuste.

4.2. Métricas

Para este trabajo se han empleado numerosas métricas, sobre todo en el problema de clasificación binaria y el por qué de ellas se expone en apartados siguientes. En dicho problema la clase positiva es la clase que presenta cáncer y la negativa la que no lo presenta. Así pues, se han usado los *verdaderos positivos* (TP), que son las instancias que pertenecen a la clase positiva y se clasifican como tal, los *verdaderos negativos* (TN) que es análoga a la anterior pero con la clase negativa, los *Falsos Negativos* (FN) que son instancias pertenecientes a la clase positiva pero que se clasifican como negativos y los *Falsos Positivos* (FP), que son instancias pertenecientes a la clase negativa y que se clasifican como positivas.

Dichas métricas no aportan mucha información por separado, pero a partir de operaciones algebraicas con éstas métricas se obtienen otras que sí que aportan información más significativa. Estas otras métricas son:

Nombre	Descripción	Fórmula
Accuracy	Proporción del número de predicciones correctas entre el total de predicciones.	$Acc = \frac{TP+TN}{TN+TP+FP+FN}$
True Positive Rate, Recall o Sensitivity (TPR)	Predicciones positivas correctas entre el número total de predicciones positivas.	$TPR = \frac{TP}{TP+FN}$
True Negative Rate (TNR) o Specificity	Prediciones negativas correctas entre el número total de negativas.	$TNR = \frac{TN}{TN+FP}$
False Positive Rate (FPR)	Predicciones positivas incorrectas entre el número total de negativas.	$FPR = \frac{FN}{TP+FN}$
Área bajo la curva ROC (AUC)	Pa curva roc es una representación bidimensional del rendimiento de un clasificador y muestra el compromiso entre beneficio (True Positive) y coste (False Positive). El valor máximo que se puede tomar es 1 e indica que el clasificador es perfecto con esta métrica.	$AUC = \frac{1+TPR-FPR}{2}$
F1-score (F1)	Pedia armónica de PPV y TPR y penaliza más los errores al clasificar ejemplos positivos (FN) que los errores al clasificar ejemplos negativos (FP)	$F1 = \frac{PPV*TPR}{PPV+TPR}$
Gmean	Media geométrica, de forma similar al F1-score, penaliza más los errores al clasificar ejemplos positivos (FN) que los errores al clasificar ejemplos negativos (FP).	$\sqrt{PPV * TPR}$

5. Modelos

En esta sección se presentan los modelos realizados tanto para clasificación binaria como multiclase. Los cuatro primeros son comunes para ambos problemas y solo se diferencian en la capa de salida, pues para clasificación binaria habrá dos neuronas mientras que para el otro problema habrá tres.

5.1. Modelo 1: Modelo Simple

Se parte de la arquitectura DenseNet121 con los pesos de ImageNet eliminando la capa totalmente conectada de la parte superior y usando la capa de AveragePooling. Se indica también que los pesos son no entrenables para que de este modo solo se entrene las capas que se añaden a continuación del bloque denso y especificadas en la tabla 1.

Layers	Output Size	DenseNet-121
Convolution	112 × 112	
Pooling	56 × 56	
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	
	28 × 28	
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	
	14 × 14	
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Transition Layer (3)	14 × 14	
	7 × 7	
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$
Classification Layer	1 × 1	
		fully-connected,

Figura 8: Esquema de DenseNet121

Layers	Output Size
BatchNormalization	7x7x1024
Activation Relu	7x7x1024
AvgPooling	1x1x1024
Linear	2

Tabla 1: Capas añadidas a la arquitectura DenseNet121 anteriormente comentada

5.2. Modelo 2: Añadir varias capas FC

Anteriormente solo se ha añadido una capa totalmente conectada, ahora se plantea la opción de si sería interesante añadir más capas FC al modelo. En [4], [5] se detalla entre otras cosas la utilidad del uso de este modelo con 3 capas totalmente conectadas: la primera con 1024 neuronas,la segunda con 512 neuronas y finalmente la última con 2 ó 3 neuronas (dependiendo del problema). Además de las anteriores también se añadieron capas de normalización por lotes, dropout incorporando también funciones de activación. Las capas añadidas quedan presentes en la tabla 2. Cabe destacar que se ha eliminado tanto en este como en los siguientes modelos

la capa de AvgPooling.

Layer No.	Layer Type	Kernel size	I-O dimension	I-O channels
1	BatchNormalization	-	7—7	1024—1024
2	Activation Relu	-	7—7	1024—1024
3	Flatten	-	7—50176	1024—1
3	Linear	-	50176—1024	-
4	BatchNormalization	-	1024—1024	-
5	Dropout	-	1024—1024	-
6	Linear	-	1024—512	-
7	BatchNormalization	-	512—512	-
8	Dropout	-	512—512	-
9	Linear	-	512—2	-

Tabla 2: Capas añadidas tras la últimas capa de DensetNet121 y que conforman la arquitectura del modelo 2.

El número de neuronas de las capas densas va disminuyendo y se introduce pocos nodos en las capas para que el número de parámetros crezca poco en comparación al modelo 1. Inevitablemente se tendrá que calcular más parámetros en este modelo pero se pretende que el número de parámetros a entrenar no aumente mucho con respecto al modelo 1.

La segunda idea de este modelo ha sido introducir capas Dropout con una probabilidad del 40 %-50 %. Lo que se consigue con esta técnica es que ninguna neurona memorice parte de la entrada que es precisamente lo que sucede cuando hay sobreajuste. Esta capa se encargará de desactivar aleatoriamente un porcentaje de neuronas de acuerdo a la probabilidad de descarte previamente definida con el objetivo de hacer que la red se vuelva menos sensible a los pesos específicos de las neuronas. Esto a su vez da como resultado una red con una mayor capacidad de generalización y es menos probable que sobreajuste los datos de entrenamiento.

La tercera idea ha sido usar una capa de BatchNormalization la cual acelera el proceso de entrenamiento y suaviza la función de pérdida, por tanto obtiene mejoras bastante considerables.

5.3. Modelo 3: Añadir un bloque VGG

VGG permite aumentar la profundidad de una red sin aumentar de forma explosiva el número de parámetros ya que permite sustituir todas las convoluciones por convoluciones 3x3. Partiendo de las ventajas que tiene VGGNet se ha decidido introducir en la red un bloque VGG. Inicialmente se tiene la arquitectura de la red DenseNet121, entrenada para ImageNet y quitándo la capa totalmente conectada de la parte superior y la capa de AveragePooling. A continuación, se añade un bloque VGG y una capa totalmente conectada con 2 ó 3 neuronas (dependiendo del problema) junto con capas previas de Dropout y BatchNormalization.

Se programó una función llamada **vgg_block** a la que se le pasa la capa de entrada, el número de filtros y el número de convoluciones que se pretende hacer en el bloque. A partir de ahí, construye un bloque VGG formado por tantas capas de convoluciones 2D como se especifiquen, el tamaño del kernel es 3x3, se realiza padding y la función de activación es ReLu. Tras una capa de MaxPooling finaliza el bloque VGG. Por tanto la arquitectura final sería la red DenseNet con los cambios comentados en el párrafo anterior más las capas de la tabla **??**

Layer No.	Layer Type	Kernel size	I-O dimension	I-O channels
1	Conv2D	3	7—7	1024—64
2	Conv2D	3	7—7	64—64
3	MaxPooling	2	7—3	64—64
4	Flatten	-	3—576	64—1
5	BatchNormalization	-	576—576	
6	Dropout	-	576—576	
7	Linear	-	576—2	

Tabla 3: Capas añadidas al final de DenseNet121 habiendo eliminado previamente la última capa de totalmente comentada y la capa de AvgPooling.

Añadiendo varios módulos VGG se ha comprobado que se obtenían resultados análogos pero se producía un aumento en el número de parámetros a aprender.

5.4. Modelo 4: Añadir dos módulos Inception

Antes de que surgiera la red Inception, la investigación intentaba ver cómo hacer modelos más profundos, porque la forma más sencilla de mejorar el rendimiento de las redes neuronales profundas es aumentando su tamaño, pero algunos de los problemas que enfrentan los investigadores eran: El problema de la desaparición del gradiente, las redes profundas son propensas al sobreajuste, coste en cálculo de filtros de dimensión mayor en capas convolucionales y la complejidad sobre la elección del tamaño del filtro en las capas de convolución. Comentar que para este último problema, la elección de un kernel más grande se usa para que la información se distribuya más globalmente mientras que la elección de un kernel más pequeño hace que la información se distribuya de manera local.

Más adelante, se propuso una arquitectura de red neuronal convolucional profunda llamada Inception, que ganó ILSVRC 2014. La característica más importante de Inception es que a través de un diseño, la red ha aumentaba la profundidad y el ancho bajo la premisa de que la cantidad de cálculo no cambia, mejorando así el uso de los recursos informáticos por parte de la red profunda. La arquitectura de red de Inception está diseñada en base a los principios de Hebbian y la intuición de procesamiento a múltiples escalas, lo que optimiza la arquitectura de Inception.

Después de descubrir el gran aporte de esta arquitectura se decidió introducir módulos Inception a la red DenseNet121, entrenada para ImageNet, quitando la capa totalmente conectada de la parte superior y la capa de AveragePooling.

Se programó una función llamada **inception_block** que recibe como argumento la capa inicial y una serie números de filtros para cada convolución a aplicar. Se llamará dos veces a la función con objeto de introducir dos bloques Inception al modelo DenseNet121.

Primero se añadimos un bloque Inception donde se van a realizar de forma paralela:

- 64 convoluciones 1x1
- 96 convoluciones 1x1 seguidas de 128 convoluciones 3x3
- 16 convoluciones 1x1 seguidas de 32 convoluciones 5x5
- MaxPooling 3x3 seguido de 32 convoluciones 1x1

A continuación, se concatena las cuatro capas anteriores y se añade el siguiente bloque Inception donde se realiza de forma paralela:

- 128 convoluciones 1x1
- 128 convoluciones 1x1 seguidas de 192 convoluciones 3x3
- 32 convoluciones 1x1 seguidas de 96 convoluciones 5x5
- MaxPooling 3x3 seguido de 64 convoluciones 1x1

Una vez se concatenen las cuatro capas anteriores, se añaden dos capas densas, la primera de 512 neuronas y la segunda con dos neuronas. Además, entre estas capas se han usado las capas Dropout y BatchNormalization.

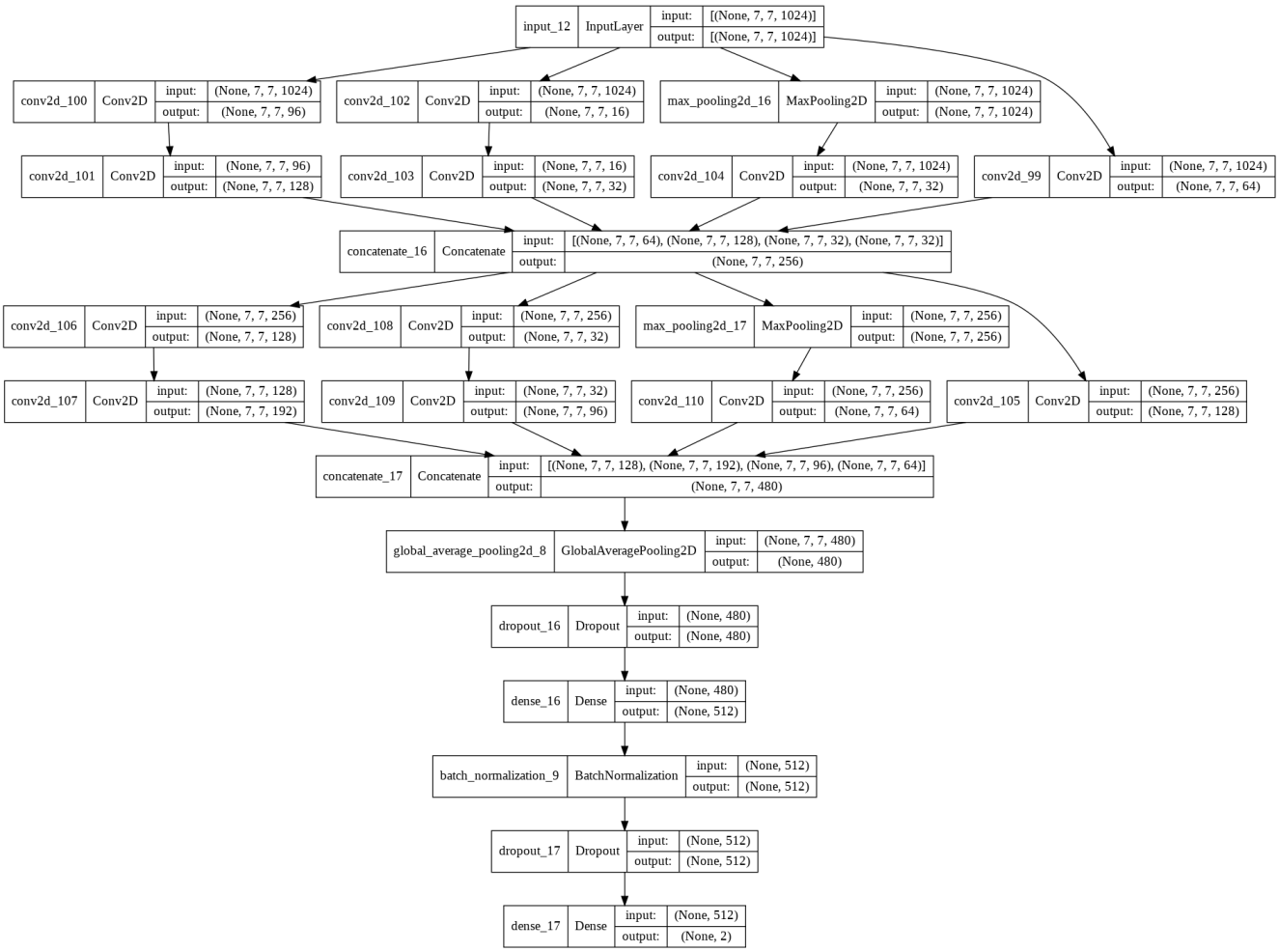

Figura 9: Bloques Inception en la arquitecutra.

5.5. Ensembles

La idea de realizar un ensemble de varios modelos se ha tomado de [7]. Se ha definido una función llamada *ensemble* que se encarga de agrupar las salidas de los distintos modelos que se le pasan como argumentos a la función haciendo la media de dichas salidas. Para ello se hace uso de la capa *Average* de Keras. Esta capa se encarga de hacer un promedio de la lista de tensores que recibe como argumento y devuelve un único tensor, que será la salida del modelo Ensemble.

Se parte de los modelos, preentrenados con ImageNet, disponibles en Keras y se elimina de cada modelo la última capa fully-connected, así como la capa de Average Pooling. Se introducen tres capas densas de 1024,512 y 3 neuronas justo después de una capa Flatten. Además, se introducen, entre las capas densas, capas de DropOut y de BatchNormalization. La estructura de las capas que se añaden al modelo base proporcionado por Keras es la misma que se usó en el Modelo 2, la cual proporcionaba buenos resultados.

Se han realizado tres ensembles distintos tomando diferentes modelos base con objeto de intentar conseguir resultados que mejoren a los modelos previamente definidos: el *primer ensemble* agrupa DenseNet121 y VGG19, el *segundo ensemble* usa DensetNet121, VGG19 e IncceptionV3 y el *tercer ensemble* incluye a DensetNet121, VGG19 y ResNet101.

La implementación se lleva a cabo siguiento el artículo [9].

5.6. Ajuste Fino

Para llevar a cabo un ajuste fino de los modelos se ha implementado la función **fine_tuning**. Esta realiza, en primer lugar, un entrenamiento de las últimas capas añadidas a la red, esto es, se congelan las capas del modelo base proporcionado por Keras y se entrenan las últimas capas introducidas que no fueron preentrenadas con ImageNet. Una vez hecho esto se descongelan las capas del modelo base y se entrena nuevamente toda la red sobre el conjunto de datos con el que se está trabajando. Para ello, el número de épocas y learning rate inicial seleccionados es menor, en concreto de 20 y 10^{-4} respectivamente, con el objetivo de evitar el sobreajuste y un cambio brusco de los pesos, lo cual haría perder la ventaja de que la red esté preentrenada. También se disminuye el tamaño de batch, de 32 a 16, para evitar errores del tipo OOM (out of memory).

Se realizará un ajuste fino de los modelos 2, 4 y el ensemble que incluye DensetNet121, VGG19 y ResNet101, que, como se verá en el apartado de resultados, son los que presentan un mejor rendimiento para el problema de clasificación multiclase abordado.

6. Discusión de los resultados

6.1. Resultados

En este apartado se presentarán los resultados, tanto para el proceso de validación experimental como para la evaluación de los modelos usando los datos de test de los dos problemas. Algo llamativo de los resultados es que en los modelos hay el mismo número de falsos negativos que de falsos positivos, provocando que el número de verdaderos negativos y verdaderos positivos también sea el mismo. No es muy común que esto pase, y el código se ha revisado minuciosamente para ver si fuera un fallo del mismo. Tras descartar esta posibilidad se considera que ha sido algo esporádico que depende de la base de datos y otros factores como el preprocesado de los datos. Las tablas 4 y 5 corresponden al proceso de validación experimental usando cross-validation sobre los modelos de la clasificación binaria. La tabla 6 corresponde a los resultados usando el conjunto de test para validar los modelos también sobre el problema de clasificación binaria. Análogamente, las tablas 7 y 8 son las correspondientes al proceso de validación experimental y la tabla 9 la correspondiente a la evaluación con el conjunto de test, todas del problema de clasificación binaria. Además se incluyen gráficas que muestran la evolución del entrenamiento de los modelos.

6.1.1. Clasificación Binaria

Sin Oversampling	**Modelo 1**	**Modelo 2**	**ModeloInception**	**ModeloVGG**
loss	0,0779	0,1027	0,0965	0,0917
accuracy	0,9722	0,9767	0,9760	0,9694
auc	0,9923	0,9865	0,9922	0,9881
precision	0,9722	0,9767	0,9760	0,9694
recall	0,9722	0,9767	0,9760	0,9694
F1-score	0,9722	0,9767	0,9760	0,9694
Gmean	0,9722	0,9767	0,9760	0,9694
FNR	0,0278	0,0233	0,0240	0,0306
FPR	0,0278	0,0233	0,0240	0,0306
TNR	0,9722	0,9767	0,9760	0,9694

Tabla 4: Tabla que registra los valores de las métricas obtenidos con cada modelo en la clasificación binaria. En las ejecuciones no se ha hecho OverSampling sobre la clase sin tumor.

Oversampling	Modelo 1	Modelo 2	Modelo 4	Modelo 3
loss	0,1213	0,0942	0,1193	0,1152
accuracy	0,9560	0,9704	0,9559	0,9598
auc	0,9881	0,9864	0,9843	0,9885
precision	0,9560	0,9704	0,9559	0,9598
recall	0,9560	0,9704	0,9559	0,9598
F1-score	0,9560	0,9704	0,9559	0,9598
Gmean	0,9560	0,9704	0,9559	0,9598
FNR	0,0440	0,0296	0,0440	0,0402
FPR	0,0440	0,0296	0,0440	0,0402
TNR	0,9560	0,9704	0,9559	0,9598

Tabla 5: Tabla que registra los valores de las métricas obtenidas con cada modelo en la clasificación binaria. En las ejecuciones se ha hecho OverSampling sobre la clase sin tumor.

TEST	Modelo 2	Modelo 4
loss	0,4848	0,2376
accuracy	0,9156	0,9107
auc	0,9228	0,9625
precision	0,9156	0,9107
recall	0,9156	0,9107
F1-score	0,9156	0,9107
Gmean	0,9156	0,9107
FNR	0,0844	0,0893
FPR	0,0844	0,0893
TNR	0,9156	0,9107

Tabla 6: Resultados de aplicar el conjunto de test a dos de los modelos sin Oversampling del problema de clasificación binaria. Se ha entrenado con el 90 % de los datos de entrenamiento, se ha validado con el 10 % restante y finalmente se ha evaluado en el conjunto de test.

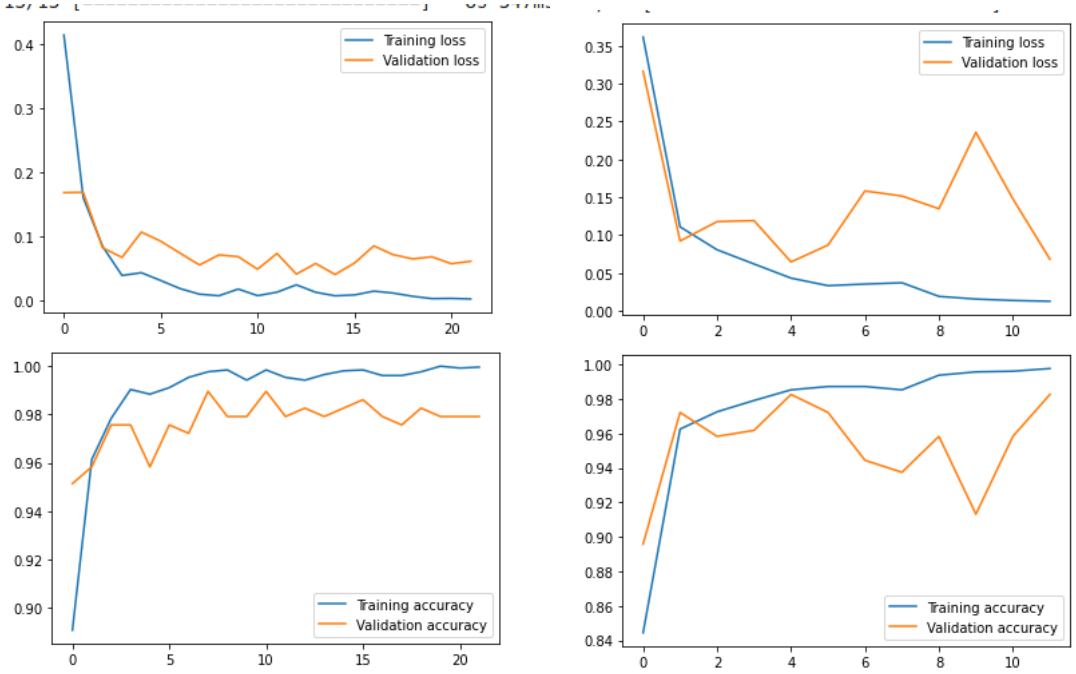

Figura 10: Las gráficas de la parte superior se corresponden a las curvas de la función de pérdida en el conjunto de entrenamiento y de validación. Las gráficas de la parte inferior se corresponden a las curvas de accuracy en el entrenamiento y en la validación. Las dos de la parte izquierda corresponden al modelo 2 y las dos de la derecha al modelo 4.

6.1.2. Clasificación Multiclase

Clasif. Mult.	**Modelo 1**	**Modelo 2**	**Modelo 3**	**Modelo 4**
loss	0.2244	0.1766	0.2834	0.2360
accuracy	0.8675	0.9073	0.8711	0.8816
	Ensemble 1	**Ensemble 2**	**Ensemble 3**	
loss	0,1760	0,1715	0,1546	
accuracy	0,9150	0,9162	0,9267	

Tabla 7: Resultados del proceso experimental en la Clasificación Multiclase

Fine Tuning	**Modelo 2**	**Modelo 4**	**Ensemble 3**
loss	0,3689	0,2429	0,1399
accuracy	0,9094	0,9175	0,9617

Tabla 8: Resultados tras aplicar un ajuste fino a los modelos 2, 4 y al ensemble 3 en la clasificación multiclase.

Test	**Modelo2**	**Modelo 4**	**Ensemble 3**	**Fine T. Model 4**	**Fine T. Ensemble 3**
loss	1,6304	1,297	1,9063	2,1222	1.9240
accuracy	0,7583	0,7953	0,7181	0,7281	0.7483

Tabla 9: Resultados en la evaluación con el conjunto de test del problema de clasificación multiclase

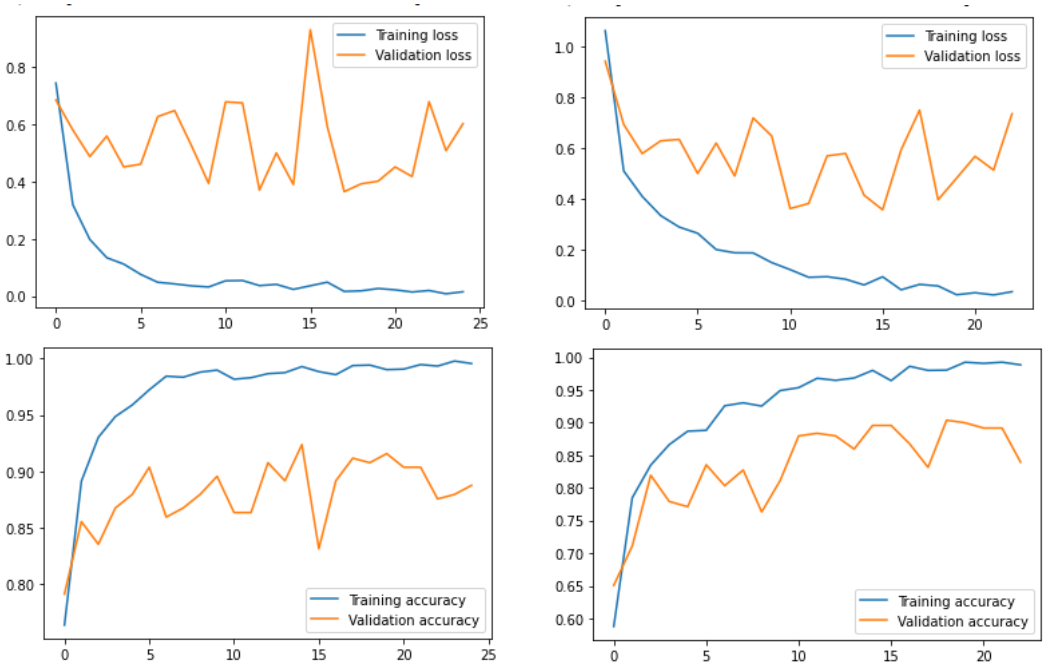

Figura 11: Las gráficas de la parte superior se corresponden a las curvas de la función de pérdida de entrenamiento y validación. Las gráficas de la parte inferior se corresponden a las curvas de accuracy en el entrenamiento y en la validación. Las dos de la parte izquierda corresponden al modelo 2 y las dos de la derecha al modelo 4

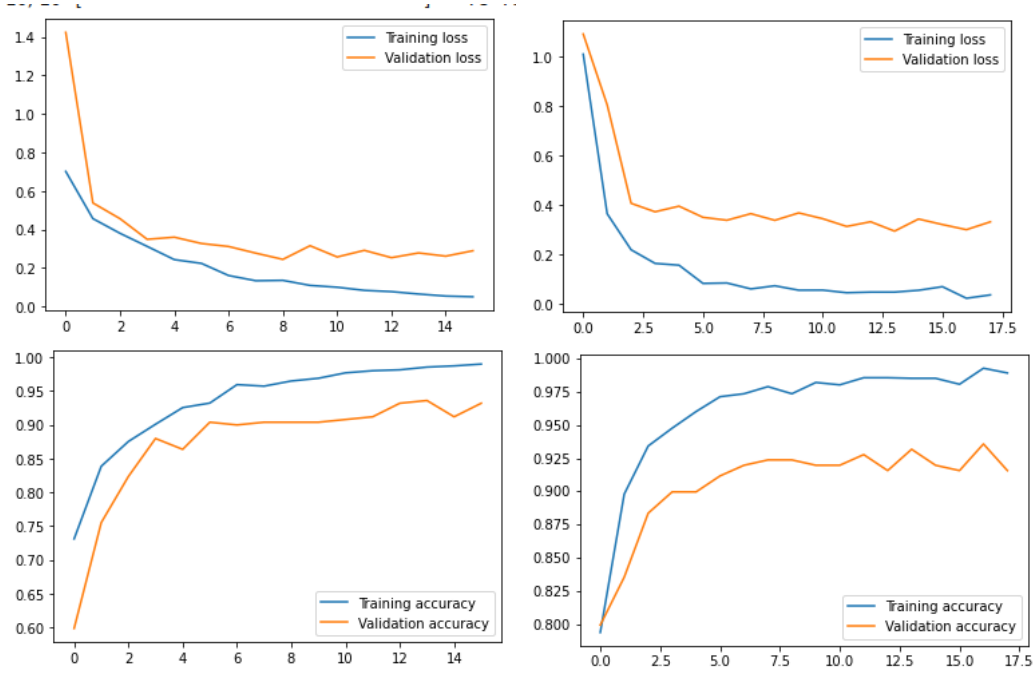

Figura 12: Las gráficas de la parte superior se corresponden a las curvas de la función de pérdida de entrenamiento y validación. Las gráficas de la parte inferior se corresponden a las curvas de accuracy en el entrenamiento y en la validación. Las dos de la parte izquierda corresponden al Ensemble 3 y las dos de la derecha al fine tuning del modelo 4.

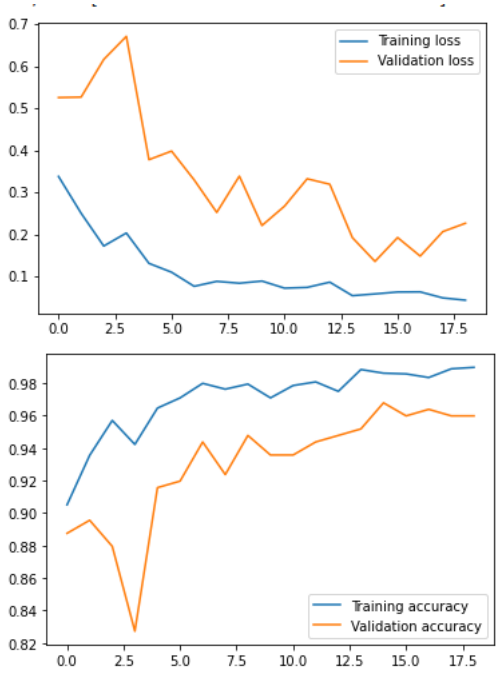

Figura 13: La gráfica de la parte superior se corresponde a las curvas de la función de pérdida de entrenamiento y validación. La gráfica de la parte inferior se corresponden a las curvas de accuracy en el entrenamiento y en la validación. Ambas correspondiente al modelo Ensemble 3 con ajuste fino.

6.2. Análisis de los resultados

Se presenta en esta subsección un análisis de los resultados obtenidos en el proceso experimental usando validación cruzada con el objetivo de seleccionar los mejores modelos para evaluarlos en el test. Primero se comentará la clasificación binaria y después la clasificación no binaria.

La **clasificación binaria** trata de predecir si el paciente tiene o no cáncer. Se han realizado cuatro modelos para este problema y cada uno se ha ejecutado dos veces: una primera vez teniendo en cuenta el Oversampling y otra sin él porque se intuyó que podía afectar de un modo u otro a los resultados, los cuales quedan registrados en las tablas 4 y 5. Como se tienen distintas métricas se observa que un mismo modelo puede presentar el mejor resultado para una pero no para todas. Así pues, para la ejeccución en la que no se hizo oversampling se tiene que el modelo 1 es el que mejor índice auc presenta. El modelo 2 es el que mejor accuracy, precision, recall, F1-score, FNR, TNR Gmean y TNR presenta . El modelo Inception y VGG no presenta en ninguna métrica el mejor resultado con respecto al resto. En cuanto a la ejecución usando oversampling se puede apreciar que los resultados son algo peores en términos generales aunque con pequeñas mejoras en cuanto a modelos individuales. El modelo 2 sigue siendo el que mejores resultados presenta en accuray con un valor de 0.9704 frente a 0.9767 de la otra ejecución, donde se aprecia un sutil empeoramiento. En cambio, el índice auc mejora en el modelo 4 pasando de 0.9881 sin usar oversampling a 0.9885 usando dicho procesado. Pero en términos globales, el valor del índice auc empeora, el mayor valor sin usar oversampling es de 0.9923 frente a 0.9885 usándolo. Análogo para otras métricas.

Haciendo una comparación global de los modelos, se observa que los mejores resultados se producen en los datos en los que no se aplica oversampling a la clase no_tumor en el conjunto de entrenamiento. Usando este conjunto de datos se ve que los que mejores resultados aportan son los modelos más simples, esto es, el modelo 1 y 2, siendo los modelos complejos, el 3 y 4 los peores. Para la evaluación en el conjunto de test se han cogido dos modelos, el que mejor

resultados ha aportado de entre los dos más simples y de entre los dos más complejos, es decir, el modelo 2 y el 4.

En la tabla 6 se aprecian los resultados de evaluar en el conjunto de test los modelos anteriormente comentados. Se observa que los mejores resultados siguen siendo por parte del modelo 2, aunque ambos modelos los presentan muy similares. Los resultados son peores que en el proceso de validación experimental, teniendo un índice AUC el modelo 2 de de 0.9228 frente a 0.99. Con esto se puede ver que el modelo no generaliza tan bien como debiera, aún así, el índice AUC es muy próximo a 1 lo que indica un buen rendimiento por parte del clasificador. Lo anterior también es apreciable con el resto de métricas y en el otro modelo.

En la figura 10 se pueden apreciar tanto las curvas de la función de pérdida como las curvas del accuracy en el entrenamiento y en la validación. En las dos gráficas del modelo 2 se puede observar que las curvas correspondientes al entrenamiento y la validación están ligeramente alejadas entre sí y se produce un poco de sobreajuste. En las dos gráficas del modelo 4 se observa que las curvas correspondientes al conjunto de validación no presentan una tendencia estable. En las primera épocas la función de perdida en la validación desciende y conforme avanzan las épocas cambia a una tendencia creciente para después decrecer.

Por otro lado, en el proceso experimental de la **clasificación multiclase** se tienen 10 modelos. Los Modelos i, con i=1,2,3,4, corresponden a DensetNet con modificaciones en las últimas capas. Los modelos Ensemble i, con i = 1,2,3, son distintos ensembles realizados en los que todos tienen en común a DensetNet. Finalmente se tiene un ajuste fino sobre los modelos 2,3 y Ensemble 3. Los resultados en términos generales son en todos superiores a 0.85 de accuracy e inferiores a 0.97. Sin tener en cuenta los ajustes finos, se tiene que los que mejores resultados presentan son los Ensembles, siendo el mejor el tercero con un valor de 0.92 de accuracy, y después de él está el segundo ensemble y finalmente el primero, todos ellos con una accuracy por encima de los modelos 1, 2, 3 y 4. El primer ajuste fino, realizado sobre el modelo 2, presenta unos resultados por debajo de los tres Ensembles. El segundo ajuste fino, realizado sobre el modelo 4, presenta un accuracy por encima del primer ensemble pero inferior a los dos ultimos. Finalmente el ajuste fino realizado sobre el ensemble 3 es el que mejores resultados presenta con un acuraccy de 0.9617, el mejor hasta el momento.

Tras el proceso de validación experimental se han evaluado el modelo 2, el modelo 4, el Ensemble 3 y el fine tuning con el modelo 4 y con el Ensemble 3 en el conjunto de test ya que han sido los que mejores resultados han presentado. En la tabla 9 se pueden ver los resultados de dicha evaluación. Se tiene que el modelo que mejores resultados presenta es el 4 seguido del modelo 2, después irían el ajuste fino del ensemble 3 y del modelo 4, siendo los resultados del ensemble 3 los peores. En términos globales el accuracy está por encima del 70 % pero por debajo del 80 %. Los modelos no presentan buenos resultados pues apenas predicen correctamente tres cuartos de las predicciones totales.

Las figuras 11, 12 y 13 muestran las curvas de entrenamiento y validación tanto para la función de pérdida como para el accuracy. A simple vista se observa en las gŕaficas del modelo 2 unas claras evidencias de sobreajuste. El accuracy en el entrenamiento pasa a valer prácticamente 1 en las últimas épocas distanciándose bastante con respecto a la curva de validación. Análogo para la función de pérdida pero en vez de valer 1, vale 0, pues el entrenamiento se ajusta totalmente a los datos. Algo similar ocurre con el modelo 4 pero no tan pronunciado como en el modelo 2. En las curvas del Ensemble 3 se puede ver que también hay sobreajuste pero en mucha menor medida que en los anteriores. En cuanto a los dos ajustes finos, comentar que también sufren de sobreajuste pero no de manera tan drástica como los dos primeros.

6.3. Interpretación de los resultados.

Ahora se pasará a interpretar los resultados anteriores y a esclarecer el uso de las métricas en la clasificación binaria.

En la **clasificación binaria** un *falso negativo* es cuando se predice que el paciente no tiene cáncer pero realmente sí lo tiene. Estos errores tienen repercusiones catastróficas sobre la vida del paciente ya que atentan contra ella. Por lo que el objetivo crucial es disminuir el número de falsos negativos tanto como se pueda. Un *falso* positivo como mucho puede ocasionar un gasto de material, dinero y tiempo por parte de los sanitarios hacia el paciente, ya que, si se detecta que el paciente tiene cáncer pero realmente no lo tiene, es dinero que se gasta el paciente en tratamiento y tiempo perdido por parte de los médicos. Se quiere, pues, un clasificador que sin repercutir o repercutiendo mínimamente en los falsos positivos, obtenga la mínima cantidad posible de falsos negativos. Por otra parte, la clasificación binaria presenta el problema del desbalanceo en la clase negativa, agravando aún más lo anterior, puesto que hay menos ejemplos de entrenamiento en la clase negativa que la positiva. Por este motivo se han usado diferentes métricas que hacen incapié en la importancia de los falsos negativos y se le ha restado importancia al accuracy.

De todos modos, gracias al análisis anterior se puede ver que los resultados en el proceso de validación experimental no son alarmantes pues se ha conseguido que todos los modelos obtengan buenísimos resultados y, en particular, los modelos en los que no se realiza oversampling a los datos presentan mejores métricas. El modelo que menos tasa de falsos negativos tiene es el 2 y se puede ver gracias al TNR y además también es el que mejor tasa de aciertos tiene, por lo que es es el mejor modelo. Además, viendo otras métricas de ese mismo modelo se observa que por ejemplo el F1-score, la cual penaliza más los FN que los FP es el más alto. También, presenta el valor más alto de la Gmean, la cual trata de maximizar los aciertos en ambas clases, de esta manera tampoco se descuidan las predicciones de la clase positiva. Algo que llama la atención es que los modelos que usan módulos inception y VGG no destacan frente a los más simples. Esto puede deberse entre otros factores a que quizás el factor de la complejidad del modelo juegue un papel negativo en los resultados por culpa del sobreajuste.

En cuanto a la interpretación de los resultados de los modelos elegidos para evaluarlos finalmente en el conjunto de test cabe destacar que siguen presentando buenos rendimientos pero no tanto como cabría esperar. La capacidad de generalización no es lo suficientemente buena como se desearía. En la figura 10, en lo que respecta al modelo 4, cabe comentar que la inestabilidad de la curva de validación puede deberse a que el modelo entrena con un número insuficiente de épocas de entrenamiento o que este para demasiado pronto debido al efecto de la técnica *early stopping*. Quizás un aumento de épocas y un mayor valor en el argumento 'patience' hubieran contribuido a mejores resultados.

Para la **clasificación multiclase** las clases están más balanceadas que en el problema anterior, por lo que la métrica accuracy es bastante significativa en cuanto al rendimiento del modelo se refiere. Además, ahora los *falsos negativos* no tienen tanto peso como antes, por lo que solo se ha usado una única métrica. Gracias al análisis de los resultados se sabe que los resultados en test han sido bastante inesperados en comparación con los obtenidos por el proceso de experimentación. Esto se debe fundamentalmente al sobreentrenamiento, explicado también en el análisis. También ocurre que el número de imágenes que se tiene en total no son suficientes como para poder aplicarles técnicas de deep learning como las que se han aplicado. Como trabajo futuro se puede proponer recabar más imágenes etiquetadas con el objeto de

aumentar el dataset. Por otra parte, en este trabajo no se ha cometido Data Snooping y los datos del test no están contaminados, ya que en muchos papers y en trabajos que circulan por la web se ha observado que el conjunto de test es usado también para validación, en cuyo caso se podría sospechar que el conjunto de test está contaminado y los resultados en el test no pueden ser considerados representativos del rendimiento del modelo en un escenario real. En el caso de este trabajo sí se podrían considerar más representativos y más cerca de lo que podría ocurrir en un escenario real debido a que solo se ha usado el conjunto de test para finalmente evaluar el modelo, de ahí que se obtengan peores resultados.

A modo de resumen, se tiene que Fine tuning mejora los resultados del modelo sin fine tuninng, ensemble mejora los de los modelos por separado. El modelo 4 es el que mejores resultados presenta en test siendo el ensemble 3 el que peores resultados ya que es el más complejo y puede generaliza peor al tener sobreajuste .

Por último, comentar que en algunas gráficas, se puede observar al principio del entrenamiento de los modelos que tanto la pérdida como el accuracy en el conjunto de validación son mejores que en el conjunto de entrenamiento. Este hecho es debido a que en un principio el modelo no está ajustado a las datos de entrenamiento, de manera que se produce underfitting y, puesto que el conjunto de validación es más pequeño y simple que el de entrenamiento, se obtienen mejores resultados en el conjunto de validación simplemente por puro azar. A medida que se avanza en el entrenamiento este efecto desaparece y, como es de esperar, los resultados en el conjunto de validación son inferiores a los del conjunto de entrenamiento. Así pues, las primeras épocas podían despreciarse a la hora de analizar los resultados.

7. Conclusión

Los objetivos principales del trabajo eran la creación y ajuste de modelos que primero predigan con la máxima precisión posible si el paciente tiene o no cáncer y, bajo la suposición de que lo tiene, predecir también con precisión qué tipo de cáncer es. Se puede afirmar haber logrado el primer objetivo, aunque el modelo puede ser mejorable. En cambio, esto no ocurre con el segundo, pues el mejor de los modelos realiza predicciones correctas en un 80 % de los casos, siendo un modelo bastante pobre en perspectiva de lo que se quería lograr. Como trabajo futuro se propone recopilar más imágenes del problema para enriquecer el aprendizaje y mejorar estos resultados, usando nuevas arquitecturas y más técnicas para reducir el sobreajuste, siempre y cuando no se contaminen los datos de test.

8. Bibliografía

Referencias

[1] Zhang, X.; Zhou, X.; Lin, M.; Sun, J. *Finding Extreme Points in Contours with OpenCV. In PyImageSearch. Available online: https://www.pyimagesearch.com/2016/04/11/finding-extreme-points-in-contours-with-opencv (accessed on 10 August 2020).*

[2] Kang, Jaeyong & Ullah, Zahid & Gwak, Jeonghwan. (2021), *MRI-Based Brain Tumor Classification Using Ensemble of Deep Features and Machine Learning Classifiers. Sensors. 21. 2222. 10.3390/s21062222.*

[3] Estudio sobre optimizadores *https://velascoluis.medium.com/optimizadores-en-redes-neuronales-profundas-un-enfoque-pr%C3%A1ctico-819b39a3eb5*

[4] Ho T.P., Hoang V.T. (2022) *CNN Parameter Adjustment for Brain Tumor Classification. In: Iyer B., Ghosh D., Balas V.E. (eds) Applied Information Processing Systems. Advances in Intelligent Systems and Computing, vol 1354. Springer, Singapore. https://doi.org/10.1007/978-981-16-2008-9_1*

[5] Polat, O., & Güngen, C. (2021). Classification of brain tumors from MR images using deep transfer learning. The Journal of Supercomputing, 1-17. *https://link.springer.com/content/pdf/10.1007/s11227-020-03572-9.pdf*

[6] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708). *https://arxiv.org/abs/1608.06993*

[7] Inception Network *https://ichi.pro/es/inception-y-versiones-de-inception-network-134685801652483*

[8] VGG and Inception modules *https://machinelearningmastery.com/how-to-implement-major-architecture-innovations-for-convolutional-neural-networks/*

[9] Implementación de Ensemble *https://towardsdatascience.com/ensembling-convnets-using-keras-237d429157eb*