Centro Europeo de Másteres y Posgrados

Máster en Aplicaciones de Inteligencia Artificial en Sanidad (2022-2023)

Trabajo Fin de Máster

“ APLICACIÓN DE REDES NEURONALES CONVOLUCIONALES PARA EL DIAGNÓSTICO DE CÁNCER DE PULMÓN A PARTIR DE IMÁGENES HISTOLÓGICAS ”

Autor: Núria Puig Segura

Tutor: Marta Dolcet Negre

Fecha: 3 de diciembre de 2023

AGRADECIMIENTOS

Quisiera agradecer en primer lugar a la doctora Marta Dolcet Negre, tutora del trabajo, por su orientación, sus conocimientos y su comprensión.

Agradezco también a mis tutoras académicas, Maider Cabezas García y Patricia Agudo, por acompañarme y asesorarme a lo largo del máster.

A todos los profesores que he tenido, por sus enseñanzas y por estar siempre dispuestos a resolver cualquier duda.

A Bryan Miguel Perez Ruchat, por interesarse en el proyecto desde el primer momento y por sus siempre valiosos consejos.

A mis amigos y familiares, por haberme escuchado y aconsejado siempre que lo he necesitado.

Sin todas vuestras aportaciones, este proyecto no habría sido posible.

ABREVIATURAS UTILIZADAS

2D: dos dimensiones

CPU: Unidad Central de Procesamiento (del inglés *Central Processing Unit*).

CNN: Red neuronal convolucional (del inglés *Convolutional Neural Network*).

GB: gigabyte.

GPU: Unidad de Procesamiento Gráfico (del inglés *Graphics Processing Unit*).

HSV: Espacio de color basado en matiz, saturación y valor (del inglés *Hue, Saturation, Value*).

IA: Inteligencia Artificial.

RAM: Memoria de Acceso Aleatorio (del inglés *Random Access Memory*).

ReLU: Unidad Lineal Rectificada (del inglés *Rectified Linear Unit*)

RGB: Modelo aditivo de colores rojo, verde, azul (del inglés *Red, Green, Blue*).

ÍNDICE

INTRODUCCIÓN

El cáncer de pulmón es uno de los tipos de cáncer más diagnosticados alrededor del mundo y, en muchos países, es el que provoca una mayor mortalidad. En 2022 fue el segundo cáncer más diagnosticado en España, representando un 12,2% de todos los cánceres diagnosticados en dicho país[1]. En Estados Unidos es también el segundo tipo de cáncer más diagnosticado, aunque es el que provoca una mayor mortalidad. Concretamente, se sabe que en el año 2020 el cáncer de pulmón provocó en dicho país más muertes que el cáncer de pecho, el de colon y el de próstata combinados[2]. En el Reino Unido, datos recogidos entre el 2017 y el 2019 indican que el cáncer de pulmón fue el tercer cáncer más diagnosticado pero, una vez más, fue el que más mortalidad presentó, con un 21% de todos los fallecidos por cáncer en el país durante ese período[3].

Una de las principales causas de la alta mortalidad del cáncer de pulmón es su diagnóstico tardío[2]. Hay dos motivos principales que explican el diagnóstico tardío del cáncer de pulmón. El primero es el desconocimiento de los síntomas de esta enfermedad y su coincidencia con los de muchos otros padecimientos. El segundo es la tardanza en realizar las pruebas médicas necesarias para comprobar si, en efecto, el paciente padece o no un cáncer de pulmón. Además, los pacientes suelen ser referidos a distintos expertos médicos, que a menudo les piden que repitan varias veces las mismas pruebas. Hay numerosos casos en los que los pacientes llegan a verse con hasta cuatro especialistas, hecho que dilata notablemente el proceso de diagnóstico[4]. Por otra parte, los servicios sanitarios tienen una sobrecarga de pacientes a los que atender, de modo que las listas de espera para algunas pruebas son muy largas. Este factor también está aumentando la tardanza en el diagnóstico[5].

El efecto que un diagnóstico tardío tiene sobre los pacientes de cáncer de pulmón está aún en discusión. Algunos estudios apuntan a que dicha duración no tiene una relación directa con la tasa de supervivencia de los pacientes[6]. Sin embargo, otros estudios denotan relación entre la espera (tanto para diagnóstico cómo para tratamiento) y un crecimiento extra del tumor, que puede complicar el tratamiento. Esto es especialmente relevante en aquellos cánceres de pulmón que se originan a partir de células escamosas, ya que progresan todavía más rápido que otros cánceres. Concretamente, hasta un 21% de los pacientes analizados en algunos estudios se volvieron *incurables* mientras esperaban tratamiento, llegando en ocasiones a un aumento del área transversal

del tumor de un 373% [7].Lo que sí se observa claramente es que una mayor espera antes del diagnóstico afecta el estado psicológico del paciente, empeorando su bienestar general[8].
Además, el cáncer de pulmón tiene impactos económicos muy importantes, y se prevé que éstos sean cada vez mayores debido al progresivo aumento de consultas y pruebas asociadas a su diagnóstico y tratamiento[2] .

Otro problema a considerar es la baja sensibilidad de una prueba a menudo usada para comprobar si un paciente tiene cáncer de pulmón, los rayos X. Los rayos X tienen una alta probabilidad de retornar falsos negativos cuando se usan para diagnosticar cáncer de pulmón, hecho que alarga el tiempo de diagnóstico[5]. En un estudio[5], los rayos X dieron un falso negativo a 12 pacientes con cáncer de pulmón que tardaron, de media, 161 días en ser diagnosticados tras haber acudido al médico por primera vez. Esta cifra contrasta muchísimo con los 27 días que, de media, tardaron en ser diagnosticados los otros 72 pacientes.

Una alternativa más fiable para el diagnóstico de cáncer de pulmón es la observación de muestras histológicas. Además, este método también permite conocer el tipo de cáncer de pulmón en cada caso, así como el estadio de desarrollo de la enfermedad[9]. Aun así, este método también utiliza muchos recursos, tanto humanos como económicos, y está sujeto a un tiempo de espera muy influenciado por el estado de sobrecarga actual de gran parte de los centros hospitalarios. Una buena manera de reducir el costo del diagnóstico basado en imágenes histológicas sería usar la Inteligencia Artificial para clasificar dichas imágenes.

La Inteligencia Artificial (IA) es un campo de la informática que busca desarrollar productos de *software* capaces de resolver tareas altamente complejas, que hasta la llegada de estas tecnologías se consideraban imposibles de automatizar. La principal característica que permite a los productos de Inteligencia Artificial resolver problemas tan complejos es su capacidad de procesar cantidades masivas de datos y extraer información de los mismos[10].

La IA ha demostrado tener un gran potencial en la medicina, y se aplica actualmente en campos muy variados, desde el descubrimiento de nuevos fármacos, hasta la mejora en la atención de los pacientes y la eficiencia en el uso de recursos[10]. Entre todas sus aplicaciones, destaca su uso para diagnóstico a partir de imagen médica, puesto que ha demostrado resultados equivalentes a los de

los profesionales sanitarios en varios casos de implementación. Varios de esos casos de éxito, además, corresponden a tareas de diagnóstico de cáncer[11].

Entre los muchos algoritmos de Inteligencia Artificial que podrían usarse, las redes neuronales destacan especialmente en tareas de análisis de imágenes. Concretamente, las redes neuronales convolucionales (*CNN*) son las que han demostrado mejores resultados en clasificación de imágenes.

Se propone, en consecuencia, aplicar las redes neuronales convolucionales para apoyo al diagnóstico de cáncer de pulmón a partir de imágenes histológicas.

OBJETIVOS GENERALES Y ESPECÍFICOS

El objetivo general del proyecto es comparar el desempeño de varias redes neuronales distintas sobre un problema de clasificación de imágenes histológicas de cáncer de pulmón, teniendo también en cuenta la importancia de la explicabilidad en Inteligencia Artificial e identificando aspectos clave para las estrategias de implementación de estas tecnologías.

Con el fin de cumplir con el objetivo general, se fijan los siguientes objetivos específicos:

1. Desarrollo desde cero de dos redes neuronales convolucionales con arquitecturas distintas para la clasificación de imágenes. Aplicación sobre el problema específico del proyecto y refinamiento de los parámetros para su optimización.

2. Aplicación de las redes *InceptionV3* y *ResNet101* para clasificación de las imágenes del problema mediante *Transfer Learning*. Refinamiento de los parámetros mediante *Fine Tuning*.

3. Aplicación de técnicas para visualizar las zonas de las imágenes que causan mayor activación en cada red neuronal. Específicamente, se trabaja con *Saliency Maps* y *Class Activation Maps*.

4. Comparación y comentario de los resultados obtenidos con las cuatro redes utilizadas. Se considerarán las métricas *accuracy, recall, precision* y *f1*. También se compararán las visualizaciones de la activación neuronal de las redes, así como la complejidad de las mismas.

5. Determinación y discusión de los puntos más importantes a considerar para la implementación de una solución de Inteligencia Artificial de este tipo, y comentario de posibles problemas que habría que resolver.

6. Previsión del impacto de dicha implementación y de las gestiones básicas que deberían llevarse a cabo como consecuencia de la misma.

CONSIDERACIONES TÉCNICAS

Todos los procedimientos descritos en este proyecto han sido realizados en un ordenador con un procesador Intel(R) Core(TM) i7-11700KF, una memoria RAM de 32,0 GB, y una GPU GeForce RTX 3060 con 12,0 GB de memoria, usando el sistema operativo *Ubuntu 22.04.3 LTS*[12]. En consecuencia, todos los procesos se han visto limitados por las características técnicas del equipo.

Durante todo el desarrollo del proyecto se trabajará con Python[13] en su versión 3.10.12.

Además, se usará una *Random Seed* al inicio de cada *script* para que los resultados de las redes neuronales sean replicables. Para ello, se han usado la función *seed* de la librería *Numpy* y la función *set_random_seed* de la librería *TensorFlow.*

CONOCIMIENTOS PREVIOS

Redes neuronales

Una red neuronal es un método en Inteligencia Artificial que funciona en base a la interconexión de muchos nodos y es capaz de aprender a procesar e interpretar datos complejos. Dichos nodos, también llamados neuronas, se organizan en capas de distintos tipos que cumplen una serie de papeles para poder resumir toda la información que recibe la red neuronal y producir algún tipo de resultado. Las neuronas de una misma capa llevan a cabo la misma operación sobre la información que reciben de otras capas y, a continuación, transmiten sus resultados a las neuronas de las siguientes capa. Por lo tanto, las diferentes capas de una red neuronal están conectadas entre sí y funcionan de manera conjunta.

Las capas de las redes neuronales se organizan básicamente en tres grupos. En primer lugar se encuentra la capa de entrada, que recibe la información a analizar, la procesa, y la transmite a la siguiente capa. Sin la capa de entrada, la red neuronal sería incapaz de recibir información del exterior, por lo que no podría aplicarse sobre ningún problema. A continuación, la información pasa por una serie de capas conocidas como capas ocultas, dónde va procesándose cada vez más.

Finalmente la capa de salida devuelve el resultado de toda la extracción de información que se ha hecho gracias a todas las capas anteriores. La capa final tendrá un número distitno de nodos dependiendo del formato que haya que tener el resultado[14].

Las capas ocultas tienen también un número variable de nodos, que establecen diferentes conexiones entre ellas. Cada conexión entre dos nodos tiene asociado un valor llamado *weigth* (peso) que representa la manera en que uno de esos nodos afecta al otro. Un nodo, por ejemplo, puede fomentar la activación de otro con el que está conectado, y además hacerlo en mayor o menor medida, reflejándose esa variación en el valor del *weigth*[15]. Es decir, un nodo puede *inducir* a otro a aumentar o disminuir su activación. Es justamente por ese funcionamiento por el que también se conoce a dichos nodos con el nombre de neuronas, puesto que recuerdan a la capacidad de las neuronas de nuestro sistema nervioso de modularse entre sí.

Una vez comprendido el concepto de *weigth* se puede ir un poco más allá para decir que, en realidad, las redes neuronales buscan optimizar sus diferentes *weigths* para modular la activación de los nodos y encontrar una combinación que permita resolver el problema al que se enfrenta de la mejor manera posible[15].

Para ello, las redes neuronales pasan por un proceso conocido como *entrenamiento*, durante el cuál reciben la información que han de analizar acompañada de la predicción correcta. Dado que en este proceso la red neuronal conoce la respuesta que ha de aprender a producir, puede revisar qué neuronas permiten llegar a dicho resultado, y les asignará mayores *weigths*. Por otro lado, dismunirá la activación de neuronas que no sean tan útiles para la tarea en cuestión. Es lo que se conoce como *backpropagation*[15].

El proceso de entrenamiento se estructura en subprocesos llamados *epochs* (épocas). En cada época, la red neuronal recibe un subconjunto de los datos de entrenamiento, evalúa las conexiones neuronales que mejor permiten llegar a las predicciones deseadas, y finalmente actualiza los *weigths* en consecuencia. Así, va realizando épocas y mejorando progresivamente su resultado hasta haber hecho un número determinado de épocas o haber llegado a cierto desempeño (según convengamos en el momento de ejecutarla).

Una vez la red neuronal ha sido entrenada, tiene los *weigths* optimizados para el tipo de problema con el que se trabaja, de manera que será capaz de predecir imágenes en el contexto de ese problema sin que se le facilite el resultado de antemano. Y, justamente, es aquí donde radica la potencia de las redes neuronales, puesto que una vez entrenadas pueden aplicarse una y otra vez, sobre imágenes para las que no conocemos el resultado[14,15].

Tipos de capas

Entre las capas más características y comunes en las redes neuronales convolucionales encontramos las capas convolucionales, las capas de *pooling*, las capas de normalización y las capas densas. Además, también se usan muy a menudo las capas *Flatten, Dropout*, y de activación.

Capas convolucionales

Las capas convolucionales aplican operaciones matemáticas llamadas convoluciones sobre una zona determinada de la imagen, hecho que les permite detectar patrones y extraer información de dicha imagen. Las capas convolucionales se caracterizan por el número de filtros (llamados *kernels*) que aplican sobre la imagen y por el tamaño de los mismos[16].

La salida de esta capa son mapas de activación, aunque estos serán distintos dependiendo del tipo de capa convolucional escogido. La elección dependerá del tipo de datos de entrada y del proceso que se quiera realizar. Al tratar imágenes, en este proyecto se usan siempre capas convolucionales de dos dimensiones, aunque de dos tipos distintos.

El primer tipo de capa convolucional que se usa es la capa *Conv2D*, que lleva a cabo convoluciones tradicionales. Esto significa que se crea un *kernel* que avanza por la imagen y opera sobre ella. Como las imágenes de entrada son en RGB, tienen tres canales; por lo que su tamaño es, en realidad, de 350x350x3. En consecuencia, el *kernel* que ha de llevar a cabo la convolución también ha de tener esa tercera dimensión. La principal implicación de que el *kernel* tenga esa tercera dimensión es que cada vez que realice una convolución llevará a cabo muchas operaciones matemáticas[17].

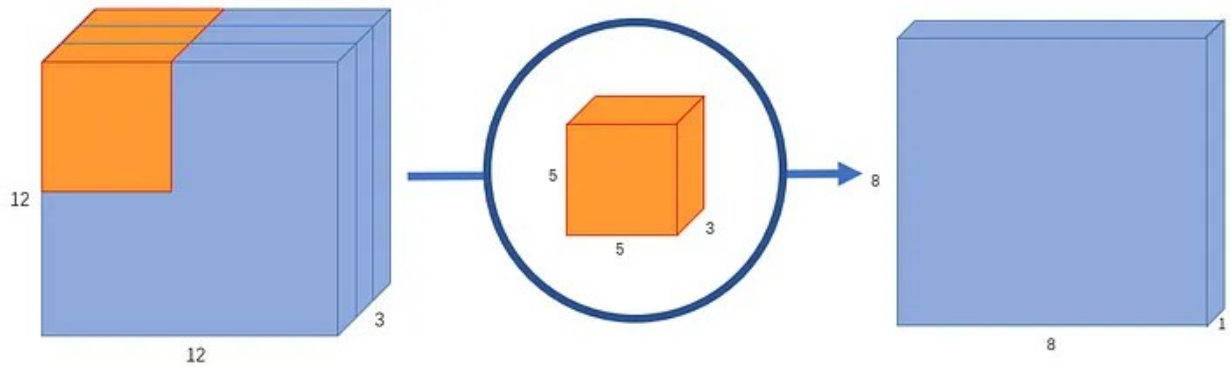

Figura 1. Convolución tradicional. *Wang, C. (2018).*

El segundo tipo de convolución que se usa, la *SeparableConv2D*, consigue aplicar también una convolución pero sigue un procedimiento distinto que le permite reducir drásticamente el número de operaciones matemáticas a realizar[17].

Las capas *SeparableConv2D* llevan a cabo, en realidad, dos convoluciones distintas. En primer lugar, realizan una convolución conocida como *depthwise separable convolution*, que actúa independientemente sobre cada uno de los canales del *input.* Dado que las imágenes con las que se trata en este problema son RGB, la *depthwise spatial convolution* aplicará, por cada imagen analizada, tres convoluciones (una para cada canal del RGB). A continuación, se realiza otra convolución conocida como *pointwise conolution*, que une las salidas de las convoluciones específicas para cada canal de *input* [16, 17]

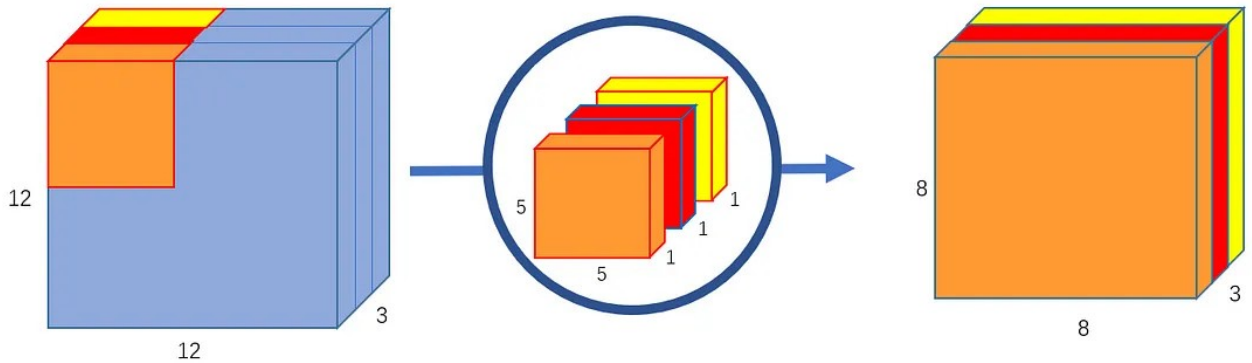

Figura 2. Ejemplo de una *depthwise separable convolution*. *Wang, C. (2018).*

Al funcionar de esta manera, la *depthwise separable convolution* transforma las imágenes menos veces, y necesita llevar a cabo menos operaciones. En consecuencia, la red neuronal tendrá menos parámetros, y será capaz de procesar más información por unidad de tiempo[17].

Capas de pooling

Las capas de *pooling* se usan para reducir progresivamente el tamaño de los mapas de activación generados en las capas convolucionales. Lleva asociada una pérdida de información, aunque su uso tiene beneficios al reducir el *overfitting* y disminuir los requerimientos técnicos de la ejecución del modelo. Disminuye la información de cada canal, por lo que la siguiente capa convolucional tiene menos parámetros a analizar, pero no afecta al número de canales, que se mantiene[16].

Para realizar el *pooling* se usan también filtros de un tamaño determinado, que se aplican sobre la imagen según un *stride.* El *stride* define en qué grado se desplaza el filtro entre cada ejecución, de manera que reduzca en mayor o menor medida la cantidad de información[18].

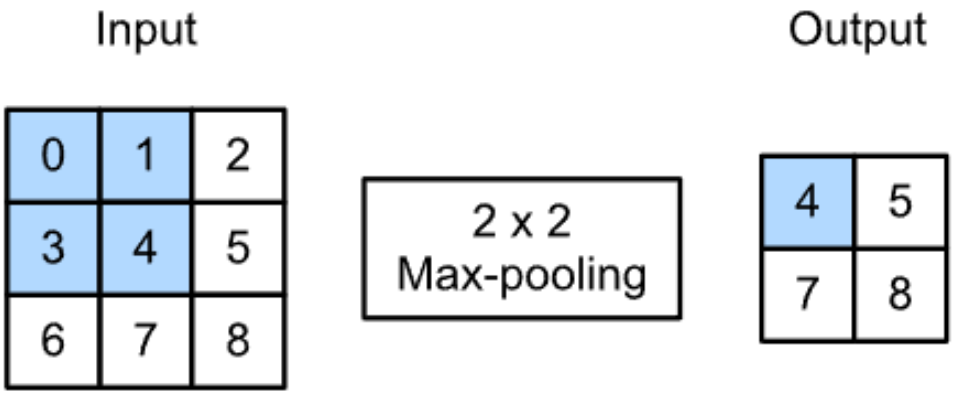

Figura 3. MaxPooling con filtro de 2x2. *Dive into Deep Learning (s.f.)*

Hay distintos tipos de *pooling.* En el proyecto se usan el *Max Pooling*, que calcula y guarda el máximo de los elementos que evalúa cada vez; y el *Average Pooling,* que calcula y guarda el promedio de los elementos evaluados.

Capas de normalización

Las capas de normalización aplican operaciones sobre imágenes para normalizarlas, con el objetivo de mantener ciertos parámetros en determinados rangos de valores. En el proyecto se usa la capa *BatchNormalization*, que aplica normalizaciones sobre lotes de imágenes con el objetivo de mantener el promedio cercano a cero y la desviación estándar cercana a uno[16].

Capas densas

Las capas densas son capas de neuronas totalmente conectadas (*fully-connected*), por lo que reciben información de todas las neuronas de la capa anterior. Estas capas se utilizan después de todas las capas convolucionales para seguir sintetizando la información y llegar a reducir el *output* hasta un número relativamente reducido de neuronas[16] Normalmente, en clasificación de imágenes se reduce el número de neuronas mediante varias capas densas hasta quedar solamente una neurona por clase del problema de clasificación.

Capas Flatten

La capa *Flatten* se utiliza para redimensionar los datos que recibe como input[16]. En este proyecto se ha usado sobre la información bidimensional que devuelven las capas convolucionales y de *MaxPooling,* para redimensionarla y obtener una lista de una sola dimensión que puede entonces pasarse a las capas densas. Una vez llega a las capas densas, la información será sintetizada.

Capas Dropout

La capa *Dropout* se utiliza para desactivar ciertas neuronas aleatoriamente en cada *step* del entrenamiento, en una proporción determinada que se le indica como argumento. Al inactivar aleatoriamente algunas neuronas, se fuerza a que las demás trabajen más en la tarea a resolver, por lo que la red consigue generalizar mejor[16]. Al irse inactivando distintas neuronas en distintos *steps*, y ser el proceso aleatorio, se disminuye el *overfitting* ya que se evita que los resultados del modelo estén basados en una pocas neuronas, especialmente en las capas menos profundas.

Capas de activación

Las capas de activación aplican una función de activación sobre los datos que reciben como *output*, modificando algunos o todos los valores. Las funciones de activación también pueden usarse como

argumentos en otro tipo de capas, llevando a cabo el mismo procedimiento[16]. El resultado de la capa de activación depende directamente de la función de activación que aplican. En el proyecto se han usado dos funciones de activación distintas, la *ReLU* y la *Softmax*.

La función *ReLU* se aplica para romper la linealidad de los datos, puesto que convierte todos los valores negativos del *input* en ceros, mientras que mantiene el valor de los positivos. Esta transformación es útil para reducir el *overfitting*.

La función *Softmax* por otra parte, aplica una función de tipo sigmoide, de manera que devuelve siempre un valor entre cero y uno. Cuánto más pequeño sea el valor de *input*, más cercano a cero será el valor que devolverá la función *Softmax*; y cuánto mayor sea el valor de *input*, más cercano a uno el valor que devolverá la función *Softmax*. Es muy útil aplicar esta función en la última capa de la red neuronal para obtener como resultado un valor entre cero y uno para cada clase del problema de clasificación, que representa la probabilidad de la imagen analizada por la red neuronal de pertenecer a cada una de las clases en cuestión[16].

Concepto de ruido o *noise*

El concepto de ruido se aplica a las imagenes para indicar la variación en el valor de ciertos píxeles, sin ser ello una representación fiel de la realidad. Por lo tanto, implica una pérdida de información, además de una cierta distorsión de la imagen. El ruido puede producirse por muchos procesos, algunos de los que están ligados, por ejemplo, al propio momento de tomar la imagen, o a la conversión de las imágenes de un formato a otro[19].

Una imagen con una gran cantidad de ruido pierde mucha calidad, por lo que su interpretabilidad por parte de cualquier modelo puede verse comprometida hasta el punto de provocar predicciones equivocadas. Es por ello que en esos casos se aplican técnicas de restauración durante el preprocesado. En el caso del *dataset* con el que se ha trabajado no se detecta rastro alguno de ruido sino que se dispone de imágenes de alta calidad. Esto es una gran noticia para conseguir que las redes neuronales tengan un buen desempeño sobre el mencionado *dataset*, pero al mismo tiempo puede causar un sobre ajuste (*overfitting)* que dificultaría la generalización, es decir, la aplicación de los modelos sobre imágenes de cáncer de pulmón provenientes de otras bases de datos.

Para mejorar la capacidad de generalización de las redes neuronales convolucionales, es posible aplicar una cierta cantidad de ruido sobre las imágenes que analizan, calibrando la cantidad de pérdida de información adecuada para no comprometer el desempeño del modelo. Al introducir ruido, se dificulta levemente la tarea del modelo, de manera que este tiene que *esforzarse* más y acaba siendo más robusto, en el sentido de que sabe enfrentarse a una mayor variabilidad en las imágenes, o a *datasets* de menor calidad. Éste hecho es muy positivo, puesto que muchos *datasets* de imagen médica disponibles actualmente son colecciones digitalizadas de imágenes tomadas de manera analógica durante años o décadas, con la consecuente variabilidad y posible pérdida de calidad. Además, muchos *datasets* de referencia son, en realidad, conjuntos de colecciones de imágenes provenientes de distintos hospitales u organizaciones, hecho que añade aún más variabilidad al problema y aumenta el interés de que la red sea capaz de generalizar.

En las imágenes pueden darse distintos tipos de ruido. Uno de los más conocidos es el comunmente llamado *salt and pepper*, que provoca la aparición de píxeles claros en zonas oscuras de la imagen y de píxeles oscuros en zonas claras. En el caso de este proyecto, se ha decidido aplicar otro tipo de ruido, el de tipo gaussiano. Éste se caracteriza porque su función de densidad de probabilidad es igual a la distribución de Gauss o distribución normal. Por ende, el ruido que se aplica a las imágenes sigue una distribución normal. Para lograr aplicar este ruido a las imágenes, se ha aplicado una capa de tipo *GaussianNoise*[16].

Concepto de *Transfer Learning*

El *Transfer Learning*, o aprendizaje por transferencia, permite utilizar un modelo ya entrenado para resolver un problema específico y aplicarlo a otro del mismo tipo. Típicamente, los modelos que se usan para *Transfer Learning* han sido exhaustivamente entrenados con grandes *datasets*, por lo que tienen un desempeño muy notable en la resolución de ciertos problemas particulares. Para poder servirse de lo que estos algoritmos han *aprendido* al ser entrenados, se realiza un procedimiento conocido como *feature extraction,* o extracción de parámetros, que luego permite inicializar dichos parámetros en valores no aleatorios, según qué valores han demostrado ser los más efectivos durante el entrenamiento en cuestión. Al ser inicializadas de esa manera, las redes retienen gran parte de su capacidad de realizar una tarea específica, pudiendo entonces transferirse ese *conocimiento* a tareas similares.

En el contexto de este proyecto, el tipo de problema a solucionar es la clasificación de imágenes, por lo que se ha aplicado *Transfer Learning* a partir de dos redes neuronales convolucionales que, como ya se ha comentado, son los modelos más capaces de realizar este tipo de tarea. El *Transfer Learning* es una técnica de gran interés en el caso de las redes neuronales, principalmente por dos motivos.

El primer motivo es la gran cantidad de recursos que ahorran. Al inicializar los pesos de manera no aleatoria, se reducen drásticamente los recursos y el tiempo de entrenamiento necesarios para que el modelo sea capaz de generar buenos resultados. Esto se debe a que al aplicar *Transfer Learning*, la red neuronal ya no tiene que aprender desde cero a reconocer patrones o identificar objetos en las imágenes, sino que solo tiene que aprender a reconocer las clases específicas al nuevo problema sobre el que es utilizada[20]. Dicho de otro modo, el modelo retiene la capacidad genérica de analizar imágenes y detectar información relevante en ellas, y solamente necesita reconocer qué información es de interés en el nuevo contexto. En consecuencia, el modelo tiene que realizar un número menor de operaciones - de acciones - para llegar al desempeño deseado.

El segundo motivo es la mejora de los resultados cuando el tamaño del *dataset* a analizar es relativamente pequeño. Al ser las redes neuronales normalmente pre entrenadas con grandísimas cantidades de datos, son capaces de optimizar mucho sus parámetros y refinar en consecuencia su desempeño. Por ello, después pueden aplicarse sobre *datasets* mucho más pequeños y conseguir buenas predicciones, al no tener que aprender la tarea de clasificar imágenes desde cero. Esto resulta muy ventajoso en el mundo de la imagen médica, donde los datos de los que se dispone pueden ser bastante limitados. Además, al poder funcionar con menos imágenes, no es necesario tener tantos ordenadores como serían necesarios para un entrenamiento desde cero. Tampoco se usará tanta memoria, por lo que se podrá trabajar con tecnología menos costosa.

En el caso que se trata en este proyecto, el dataset no es extremadamente pequeño, por lo que una red neuronal simple puede aprender la tarea de clasificarlas desde cero; pero tampoco es especialmente grande, por lo que aprovecharse de esta ventaja del *Transfer Learning* probablemente permita aún mejores resultados[20].

DATOS DE ENTRADA

Los datos que se usarán para el desarrollo del algoritmo forman parte de un conjunto de datos conocido como *Lung and Colon Cancer Histopathological Image Dataset (LC25000)*[21], que contiene imágenes histológicas de cáncer de pulmón y de colon, aunque solamente van a usarse aquellas correspondientes al pulmón.

El dataset fue construido gracias a la colaboración de *The Veterans Health Administration*[22]. Todas las imágenes fueron sacadas con una cámara Leica Microscope MC190 HD Camera conectada a un microscopio Olympus BX41 con un objetivo x60. Se usó también el software Leica Acquire 9072 en un ordenador Apple MacBook Pro de 2012 que funcionaba con macOS v10.13. Además, todas fueron validadas correctamente, además de anonimizadas[21].

Todas las imágenes tienen ya un cierto pre procesado que se les aplicó en el momento en el que se generó el *dataset*. Concretamente, son imágenes a color, que se procesaron para que tuviesen una resolución de 768 x 768 píxeles y fueron guardadas en formato jpeg. Además, la parte del dataset correspondiente al cáncer de pulmón está basado en 750 imágenes: 250 imágenes de tejido pulmonar benigno, 250 imágenes de adenocarcinomas pulmonares y 250 imágenes de carcinomas pulmonares de células escamosas. Gracias al *Data Augmentation*, se consiguió que el dataset tuviese 5000 imágenes de cada una de las tres clases en cuestión. Según las explicaciones de los creadores del *dataset*, inicialmente, se cortaron las imágenes desde su tamaño original de 1024 x 768 píxeles; dejándolas todas en 768 x 768 píxeles. A continuación, se realizó el *Data Augmentation* llevando a cabo rotaciones a la izquierda y a la derecha (de hasta 25 grados, probabilidad 1.0) así cómo giros horizontales y verticales (probabilidad 0.5)[21].

Lo más probable es que el *dataset* sea entonces suficientemente grande para entrenar nuestro algoritmo, puesto que contiene 15000 imágenes. Si aun así no lo fuera, se podrían crear aún más imágenes aplicando otras técnicas de *Data Augmentation* cómo el *scaling* para hacer *zoom in* o *zoom out* en distintas partes de las imágenes; o el *translating* para mover el objeto fotografiado dentro del marco de la imágen.

En caso de que se prevea usar el modelo para clasificar también imágenes tomadas con distintos microscopios o cámaras, podría ser de utilidad hacer *Data Augmentation* variando el brillo y el contraste de las imágenes para asegurarse de incluir posibles variaciones en estos parámetros que fácilmente variarán si se comparan imágenes tomadas con distintos dispositivos.

PREPROCESADO DE LOS DATOS

Para el pre procesado de los datos se han definido distintas funciones, de manera que según la red neuronal que se aplique se usarán unas u otras.

Por una parte, se ha definido la función *general_preprocessing(),* llamada así porque es la única que se usará para las cuatro redes neuronales. En ella se preparan las imágenes para poder usarlas; leyéndolas, descodificandolas en los tres canales del RGB y modificando su tamaño a 350 x 350 píxeles.

Por otra parte, se han definido cuatro funciones de pre procesado más, una para cada una de las cuatro redes neuronales que se usan en el proyecto. Las dos primeras simplemente llaman a la función *general_preprocessing()* y luego aplican el pre procesado recomendado para las redes que se usarán mediante *Transfer Learning*. Las otras dos funciones de pre procesado son de diseño propio, y se asocian a las dos redes neuronales que se construyen desde cero en este proyecto. En consecuencia, son funciones que se han ido perfeccionando al mismo tiempo que se mejoraba la arquitectura de sus respectivas redes neuronales y a medida que se comprobaba qué opciones mejoraban los resultados y cuáles no.

Una de estas funciones de diseño propio es *preprocessing_simpleConv(),* que se usará con la red de diseño propio *SimpleConv*. Esta función llama primeramente a *general_preprocessing(),* para aplicar el pre procesado general. A continuación, convierte el espacio de la imagen de RGB a HSV, para permitir que se aplique una normalización en el tercer canal, el de *value*, puesto que se ha observado que las imágenes tienen mucha variabilidad en este aspecto. Entonces se aplica una normalización no lineal al canal correspondiente al *value* de la imagen, y posteriormente se vuelven a unir los distintos canales de la misma.

A continuación, se reconvierte el espacio de color de la imagen de HSV a RGB. Finalmente se escalan los datos dividiendo el valor de cada píxel entre 255, para pasar de un rango entre 0 y 255 a uno entre 0 y 1. Esto es importante porque mantendrá el valor de los pesos de las redes neuronales en un rango menor, de manera que se evitarán posibles problemas de *vanishing* / *exploding gradient.*

Inicialmente se trató de ecualizar las imágenes, pero al hacerlo las métricas de la red neuronal disminuían considerablemente. Se observaron imágenes a las que se había aplicado esta técnica y se constató que algunas partes de tejido pulmonar pasaban a tener un color rojo saturado, muy similar al que toman los glóbulos rojos con la tinción de hematoxilina-eosina que tienen las muestras. Se consideró que este hecho, sumado a la variabilidad de la iluminación con la que habían sido tomadas las imágenes podía ser el motivo por el que la ecualización no funcionaba. En consecuencia, se estimó que normalizar las imágenes sería probablemente más útil. Se intentó primeramente aplicar una normalización lineal, pero ello tampoco permitió mejorar el desempeño de la red neuronal. Finalmente, se aplicó una normalización no lineal que sí mejoró el desempeño de la red neuronal, y es por ello que esta técnica es la que se usa en el pre procesado.

La otra función de diseño propio es *preprocessing_efficient(),* que se usará con la red de diseño propio *Efficient*. Esta función realiza el mismo proceso que *preprocessing_simpleConv(),* puesto que después de probar distintas alternativas, las soluciones que generaban los mejores resultados en la *SimpleConv* también lo hacen en la *Efficient.* La única diferencia es que los parámetros usados para la normalización no lineal que se usan en ambos casos son ligeramente distintos.

Aún llevando a cabo estas dos funciones prácticamente el mismo pre procesado, se ha decidido definirlas como dos funciones distintas para explicitar que se ha buscado optimizar el pre procesado para cada caso, y que después de probar distintas alternativas se ha llegado a un mismo resultado. Además, si en el futuro se quisiese seguir probando distintos pre procesados, es mejor para ello contar con dos funciones diferentes que poder modificar individualmente.

Finalmente, se ha constatado que el *dataset* está balanceado, puesto que tiene exactamente el mismo número de imágenes de cada clase. Por lo tanto, no ha sido necesario realizar ningún ajuste en este sentido.

ESQUEMA DE VALIDACIÓN

Se ha decidido incluir un 80% de las imágenes del *dataset* en el conjunto de entrenamiento (*training set*), un 10% de las mismas en el conjunto de validación (*validation set)* y el 10% restante en el conjunto de testeo (*testing set*). Es decir, 12000 imágenes se utilizan para entrenar el modelo, 1500 las usa internamente el modelo para optimizar parámetros (validación), y otras 1500 se usan para testearlo posteriormente. Para evitar cualquier desbalanceo, las divisiones se han hecho de tal manera que los tres conjuntos estén representados por las tres clases del problema de clasificación en proporciones iguales. Para ello, se usa el argumento *stratify* de la función *train_test_split* incluida en la librería *sklearn.*

Una vez el modelo ha acabado de entrenar, se usa para generar predicciones a partir de las imágenes del *testing set*, que son totalmente nuevas para él. En consecuencia, comparar estas predicciones (*y_pred*) con las predicciones reales para las mismas (*y_test*) permite estimar el desempeño que tendría cada modelo al encontrarse con imágenes que no haya visto durante su entrenamiento.

Para el cálculo de error se usará la *categorical crossentropy*. Las métricas que se consideran relevantes para la comparación de las redes neuronales son *accuracy*, *precission, recall* y *f1-score*. La *accuracy* se calcula dividiendo el número correcto de predicciones entre el número total de predicciones. En consecuencia, representa la proporción de predicciones que es correcta. La *precision* se calcula dividiendo el número de verdaderos positivos entre la suma de verdaderos positivos y falsos positivos. Es decir, la *precision* indica la proporción en qué los datos clasificados para una clase determinada forman realmente parte de dicha clase. El *recall* se calcula dividiendo el número de verdaderos positivos entre la suma de verdaderos positivos y falsos negativos. Por ende, el *recall* representa la proporción de positivos reales de cada clase que el algoritmo clasifica correctamente. Finalmente, *f1* representa la media armónica de la *precision* y el *recall*, y permite tener una visión general de la corrección de la clasificación[23].

Además, se comparará también el número de parámetros de los modelos. Esta métricas clave durante el proceso de diseño y desarrollo dado que es directamente proporcional a la capacidad computacional necesaria para entrenar y validar el modelo. También afectará al producto final ya que, a más parámetros, mayor tamaño tendrá el mismo, y más memoria será necesaria para cargarlo.

Es igualmente esperable que un modelo con más parámetros necesite más tiempo para generar una predicción, aunque esto no depende únicamente del número de parámetros. También es muy influyente en este caso la arquitectura del modelo.

CONSIDERACIONES COMUNES

Dado que se busca comparar el desempeño de varias redes neuronales en una tarea determinada, es crucial mantener constantes ciertos procedimientos para evitar que añadan variabilidad de más. Por ello, se usan los mismos *callbacks* para el entrenamiento de todas las redes neuronales, tal y como se detalla más adelante.

Además, la división del *dataset* de imágenes en *subsets* de entrenamiento, validación y testeo se ha hecho de igual manera para todas las redes neuronales, asegurando que todas ellas *ven* exactamente las mismas imágenes en el momento de entrenamiento y validación; y que las predicciones finales que se usan para valorar su desempeño también sean sobre el mismo grupo de imágenes. Ésto es muy importante para saber que cualquier diferencia que pueda encontrarse al comparar los resultados de las predicciones se debe a la capacidad de extraer e interpretar información de las propias redes. Si se encontrasen con distintos conjuntos de entrenamiento, validación y testeo, podrían aparecer diferencias provocadas por ese hecho.

También se ha usado un *batch size* de 32 en todas las redes neuronales ya que distintos estudios apuntan a que es uno de los *batch size* que mejores resultados proporciona manteniendo un tiempo de ejecución reducido[24, 25].

Callbacks usados

Los *callbacks* son funciones que se pasan como argumento a otra función para que esta las ejecute con el objetivo de completar alguna acción o rutina.

Para la ejecución de las redes neuronales de este proyecto se han usado tres *callbacks* distintos, que se han nombrado *model_checkpoint_callback*, *learning_rate_scheduler* y *early_stop.*

Por una parte, *model_checkpoint_callback* genera un documento en el que se guardan los *weigths* del entrenamiento de una red neuronal con el fin de poder usarlos posteriormente sin tener que repetir el entrenamiento, ahorrando en consecuencia muchos recursos. Para los objetivos de este proyecto, solo se han guardado los *weigths* correspondientes a la mejor época de entrenamiento según la *accuracy* obtenida para el conjunto de validación (*val_accuracy).*

Por otra parte, l*earning_rate_scheduler* ha sido utilizado para modificar el *learning rate* de las redes neuronales conforme realizan las distintas épocas. El *learning rate* es un parámetro que determina el tamaño de los *steps* que se realizan en cada iteración para encontrar el mínimo de la función de pérdida. Es decir, el *learning rate* determina la magnitud en la que se modifican los *weigths* de las redes neuronales en cada *step*. Gracias a este *callback* es posible que el *learning rate* varíe a lo largo de las épocas. Concretamente, se ha diseñado un l*earning_rate_scheduler* que determina un *learning rate* de 1×10^{-4} para las primeras 19 épocas de entrenamiento, mientras que a partir de la época número 20, el *learning rate* disminuye progresivamente. Esta disminución progresiva del *leraning rate* está pensada para ayudar al modelo a seguir aprendiendo, cada vez de detalles más pequeños.

Finalmente, el *early_stop* afecta a la cantidad de épocas de entrenamiento que realiza el modelo. Concretamente, limita el número de épocas que se realizan después de aquella en la que se obtenga el mejor resultado hasta el momento. Se ha usado este *callback* para limitar a 10 las épocas que se pueden realizar después de un cierto valor de *val_accuracy*. Si durante esas 10 épocas no se logra mejorar la *accuracy* en validación, la red neuronal deja de entrenar. Si ese parámetro logra mejorarse, sigue ejecutando hasta otro límite de 10 épocas, momento en el que se vuelve a parar si no se mejora aún más, y así repetidamente.

Estos tres *callbacks* han sido usados para el entrenamiento de las cuatro redes neuronales que se usan en el proyecto. Además, también se han usado los tres en el *Fine Tuning* de las redes *InceptionV3* y *ResNet50*, aunque en ambos casos el *learning rate* se ha iniciado en 1×10^{-5}, por lo que el *Fine Tuning* se ha llevado a cabo con un *learning rate* menor que el primer entrenamiento. Con ello se pretende hacer un ajuste más fino, más al detalle, de los *weigths* de las redes neuronales, que podría llevar a un aumento en el desempeño.

DESARROLLO

El código usado en este proyecto está distribuido en distintas *notebooks* y ficheros reunidos en una carpeta a la que es posible acceder a través del enlace que se ha adjuntado en la entrega. En esa misma carpeta se encuentra el fichero *README.md*, en el que se detalla la organización del código en los distintos archivos mencionados.

Diseño y aplicación de *CNN*s propias

Red SimpleConv

Se ha diseñado una red neuronal convolucional con una arquitectura bastante simple, que sin embargo consigue muy buenos resultados en la clasificación de las imágenes mencionadas. Específicamente, y en la versión que actualmente demuestra mejor desempeño, dicha red inicia con un *Input Layer* y, a continuación, alterna diversas capas del tipo *Conv2D, MaxPooling, BatchNormalization,* y *Dropout.* La red es completamente secuencial, lo que significa que una capa siempre está debajo de la anterior y se ejecuta a partir de los *outputs* de la misma.

Las capas de tipo *Conv2D* tienen *kernels* de dimensión 3 x 3, y se les han asignado un número de filtros creciente. Después de estas capas, tiene una de tipo *Flatten*, seguida de varias repeticiones de capas *Dense, ReLU y Dropout* consecutivas. Las capas *Dense* van disminuyendo en número de neuronas, hasta tener 3 en la última capa. Esas tres neuronas finales se encargan de poder identificar una de las clases a las que pueden pertenecer las imágenes, puesto que son tres clases excluyentes: adenocarcinoma, tejido benigno, o carcinoma de células escamosas. Después de esa última *Dense* de tres neuronas, se aplica una capa *Softmax* que permite obtener, para cada imagen, la probabilidad de que corresponda a cada una de las clases del problema. Por lo tanto, no solo permite conocer la predicción del modelo, sino también la certeza con la que se realiza la predicción.

Red Efficient

Posteriormente, se ha diseñado una segunda red neuronal convolucional con ciertos cambios respecto a la más simple que buscan hacerla más eficiente. La arquitectura de esta segunda red inicia también con un *Input Layer*, pero después alterna diversos bloques de capas del tipo

SeparableConv2D, MaxPooling, BatchNormalization, Gaussian Noise y *Dropout.* Al igual que *SimpleConv,* esta red también es secuencial. Por lo tanto, la principal diferencia entre ellas es que las capas de convolución que utiliza *Efficient* son de otro tipo, y que además usa la capa *Gaussian Noise,* que no estaba presente en la arquitectura de la *SimpleConv.*

Como ya se ha mencionado, la *SeparableConv2D* es una capa que permite obtener los mismos resultados que una capa *Conv2D*, pero utilizando menos recursos, por lo que aplicarla en esta red neuronal hace que tenga considerablemente menos parámetros que las otras tres.

La capa *GaussianNoise,* por otra parte, introduce una cierta cantidad de ruido de tipo gaussiano en las imágenes. La cantidad de ruido que se aplica se ha ajustado para mejorar la generalización sin afectar negativamente a la capacidad de predicción de la red. Teóricamente, esto le permitiría a la red adaptarse mejor a imágenes de otros *datasets*, puesto que es ahora un modelo con mejor generalización, y ello le facilitaría el análisis de la variabilidad que provocaría la introducción de imágenes de otros *datasets.*

Después de estas capas, tiene una de tipo *Flatten*, seguida de varias repeticiones de capas *Dense, ReLU y Dropout* consecutivas. Las capas *Dense* van disminuyendo en número de neuronas, hasta tener 3 en la última capa, con el mismo propósito que en el caso de la red *SimpleConv.* Al igual que esta, la red *Efficient* también finaliza con una capa *Softmax* que permite obtener como resultado la probabilidad de que una imagen pertenezca a cada una de las clases posibles.

Aplicación de CNNs ya existentes

Las redes neuronales que se han usado mediante Transfer Learning son *Inception V3* y *ResNet50.* Las dos fueron diseñadas para tareas de clasificación de imágenes, y demuestran una capacidad predictiva excelente hasta en los *datasets* más complejos. Inicialmente, se planteó usar la *ResNet101* en lugar de la *ResNet50*, puesto que la primera es una versión con una arquitectura más profunda que se estimaba que tenía potencial de producir mejores resultados. Su uso no ha sido posible finalmente, dado a limitaciones ligadas a los requerimientos de memoria de la misma, por lo que se ha optado por la *ResNet50*, que sigue siendo un modelo muy potente. Aún así, la *ResNet50* también tiene unas necesidades muy altas en cuanto a memoria se refiere, y no ha sido posible

entrenarla en la *GPU* del ordenador, sino que se ha tenido que usar la *CPU*. En consecuencia, entrenarla ha sido un procedimiento mucho más lento de lo que habría sido usando la *GPU*.

InceptionV3 y *ResNet50* son dos redes neuronales convolucionales que fueron entrenadas para la clasificación de imágenes de un mismo *dataset,* el *ImageNet.* El hecho de que hayan sido entrenadas con el mismo *dataset* disminuye la variabilidad no deseada que se introduce en el análisis, y que podría llevar a conclusiones erróneas en el momento de analizar y comparar resultados.

Además, es posible cargar en su arquitectura los *weigths* correspondientes a dicho entrenamiento para poder aplicar métodos de *Transfer Learning* sobre otros problemas de clasificación. Por todo ello, se ha decidido usarlas en este proyecto.

InceptionV3

La *InceptionV3* es la tercera versión de una red neuronal convolucional llamada *Inception.* Esta red, y todas sus versiones, ha sido diseñada por un equipo de Google. Fue inicialmente desarrollada en el contexto del *ImageNet Large-Scale Visual Recognition Challenge 2014*, un concurso de diseño de algoritmos para la clasificación de imágenes que tuvo lugar en 2014[26].

En el momento de su presentación, bajo el nombre *GoogleLeNet*, la red *Inception* consiguió generar los mejores resultados hasta aquel momento en tareas de clasificación sobre un *dataset* conocido como *ImageNet.* Este *dataset* contenía en el momento del desarrollo de *Inception* más de 14 millones de imágenes anotadas manualmente según la presencia o ausencia de 1000 objetos diferentes, que equivaldrían a las clases del problema de clasificación[27].

La arquitectura de esta red neuronal inicia con una capa de *input*, y está basada en repeticiones de capas del tipo *Conv2D, MaxPooling, AveragePooling, BatchNormalization, Activation,* y *Concatenate.* Después de las repeticiones de estas capas, se utilizan capas *Dense* y *Dropout,* además de finalizar típicamente con una *Softmax*. En este caso, a diferencia de las dos redes anteriores, las capas no usan únicamente el *output* de una capa anterior, sino que en ocasiones toman información de varias capas a la vez.

Lo que ocurre en esta red es que está organizada en módulos, que ejecutan distintas operaciones sobre un mismo *input,* generando varios *outputs*. A continuación, esos *outputs* correspondientes a un mismo *input* se combinan y se pasan como *input* al siguiente módulo, tal y como se puede observar en la **Figura 4.**

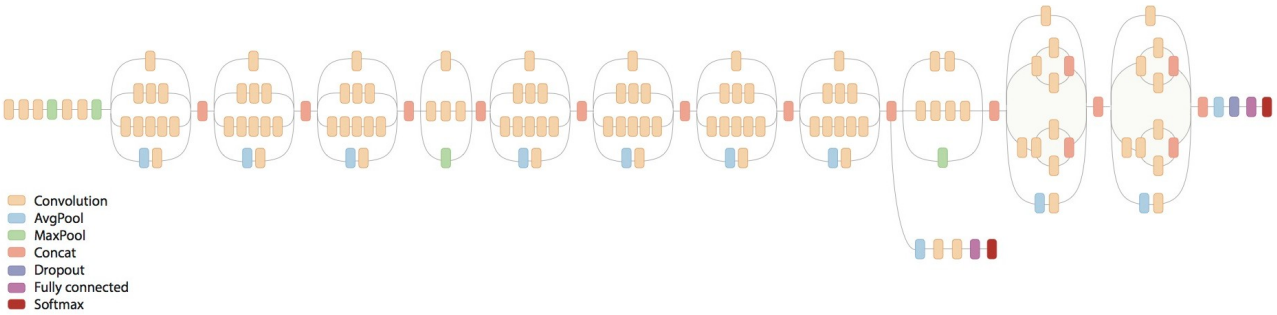

Figura 4. Arquitectura de la red *InceptionV3.* *Hackaday (2017)*

En el momento en el que se diseñó la primera versión de la red *Inception*, esta arquitectura en módulos resultó un avance extraordinario en la clasificación de imágenes de grandes *datasets*, puesto que la única opción conocida hasta entonces era añadir más profundidad a las redes secuenciales. El problema es que a partir de cierta profundidad no se conseguían mejores predicciones, ya que las redes así construidas eran incapaces de extraer información adecuadamente. Además, al aplicar varias operaciones sobre un mismo *input,* la red *Inception* tiene mejor capacidad de tratar bases de datos con mucha variabilidad[28].

Para poder usarla sobre el conjunto de datos con el que se trabaja, se ha tenido que modificar ligeramente. Las capas colocadas después de la última capa convolucional no se han utilizado, y en su lugar se ha implementado una capa *Flatten,* seguida de seis bloques de capas *Dense* con sus respectivas activaciones. Cada capa *Dense,* además, tiene menos neuronas que la anterior, de modo que sintetiza cada vez más la información. Las cinco primeras *Dense* tienen una función *ReLU* como activación, para romper la linealidad. La última capa *Dense* tiene solamente tres neuronas, una por cada clase que se quiere predecir, además de una función *Softmax* como activación. Estas dos características permiten obtener como *output* las probabilidades de que una imagen pertenezca a cada una de las tres clases presentes en el dataset.

ResNet50

La *ResNet50* es una versión de la *ResNet* que consta de 50 capas, 48 de las cuales son convoluciones. Fue desarrollada por un equipo de Microsoft un año después que la primera *Inception*, y nuevamente como parte del concurso *ImageNet Large-Scale Visual Recognition Challenge,* en su convocatoria de 2015. La primera versión de *ResNet* consiguió los mejores resultados en la clasificación sobre el *dataset ImageNet* al ser presentada en el mencionado concurso[29].

La arquitectura de la *ResNet50* se caracteriza por estar estructurada en bloques conocidos como *Residual Blocks,* ilustrados en la **Figura 5**. Estos bloques toman un *input* determinado y lo pasan por una serie de capas para analizar la información recibida. Una vez se llega al final del bloque, se suman los resultados obtenidos de la serie de capas con la información que se había recibido como *input* de manera que se obtiene información similar al *input,* pero con ciertas modificaciones. El resultado de un *Residual Block* pasa entonces al siguiente bloque.

De esta manera, se pierde menos información al aplicar los *layers* incluidos en cada *Residual Block,* y la red puede tener mucha más profundidad sin comprometer su capacidad de predicción.

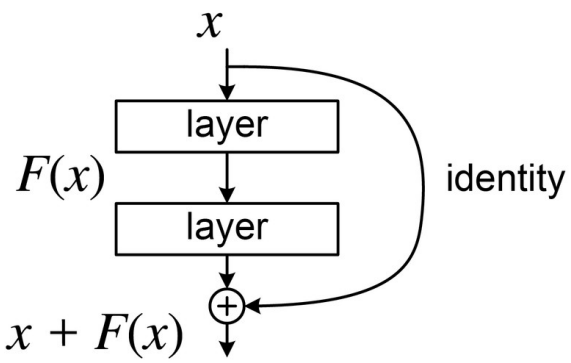

Figura 5. Estructura de un *Residual Block.* *He, K. et al. (2019)*

Para poder usarla sobre el conjunto de datos con el que se trabaja, se ha modificado de la misma manera que la *InceptionV3.*

Explicabilidad de los modelos

En algunos casos, es difícil estar seguro de la información en la que se basan los modelos de inteligencia artificial para realizar sus predicciones[10]. Las redes neuronales son uno de los algoritmos en los que puede ser complicado comprender ese proceso para realizar predicciones concretas, especialmente dado al procesos de abstracción o de síntesis de datos complejos. En dichos casos, es muy importante el concepto de la Inteligencia Artificial explicable.

La Inteligencia Artificial explicable pone énfasis en la importancia de comprender los procesos mediante los que los modelos producen predicciones. Dado que en este proyecto se trata con datos de carácter médico con el objetivo de producir predicciones de diagnóstico de cáncer, se considera muy relevante la implementación de alguna técnica para aumentar la explicabilidad de los modelos aplicados. Para facilitar la comprensión de los resultados se han aplicado dos técnicas de visualización de la activación neuronal, los *Saliency Maps* y los *Class Activation Maps*.

Saliency Maps

Los Saliency Maps representan los gradientes generados durante el proceso de *backpropagation* sobre el *input* de la red que, en este caso, serían las imágenes. Para ello, se representan los píxeles de la imagen con valores relacionados con esos gradientes. Al tener lugar esa diferenciación en toda la imagen, es posible observar las zonas más características de la misma, que también son las que más contribuyen a la predicción final del modelo. Por lo tanto, esta técnica se centra en representar las zonas con información relevante para poder generar una predicción sobre una imagen en particular. Los *Saliency Maps* son una técnica muy utilizada para visualizar los resultados de todo tipo de redes neuronales, entre ellas las convolucionales, y permite aumentar la explicabilidad de estos modelos de inteligencia artificial[30].

Como los *Saliency Maps* intentan representar la importancia de cada píxel, es relativamente probable que la imagen resultante tenga mucho ruido, al ser algunos píxeles identificados como responsables de mucha activación de la red neuronal, mientras que otros contiguos pueden ser identificados como causantes de muy poca activación. Esto puede explicarse porque al aplicar esta técnica se infiere la importancia de los píxeles a partir de información muy sintetizada[30].

Para implementar los *Saliency Maps* se ha definido la función *draw_saliency()*. A esta función hay que indicarle, mediante argumentos, la ruta dónde encontrar la imagen a partir de la que hay que elaborar el *Saliency Map* (*image_path*), la red neuronal que se usará para realizar la predicción (*model*), el preprocesado que se debe aplicar a la imagen (*pre_fn*) y la ruta en la que se debe guardar el *Saliency Map* (*sal_path*).

Lo primero que hace la función es aplicar el pre procesado indicado a la imagen seleccionada. Esa imagen, puede entenderse como una matriz de tamaño (350, 350, 3), puesto que tiene 350 x 350 píxeles y está codificada en tres canales. Las redes neuronales, sin embargo, funcionan a partir de un *batch* de imágenes, por lo que el siguiente paso es transformar la imagen a un tamaño de (1, 350, 350, 3) mediante la función *expand_dims* de *tensorFlow.* Al añadir esa dimensión se está indicando que tenemos un *batch* de una sola imagen. A continuación se usa la función *GradientTape()* de *tensorFlow* para poder conocer los resultados de las derivadas que se calculan durante el *backpropagation,* al usar el modelo para predecir la clase de la imagen seleccionada. A partir de esa información se genera el *Saliency Map*, que se superpone a la imagen seleccionada para una mayor comprensión de los resultados. Finalmente, se guarda el mapa generado en la ruta indicada en *sal_path.*

Class Activation Maps

Los *Class Activation Maps* son un tipo de *Saliency Maps* diseñado especialmente para modelos de clasificación de imágenes. La característica que los diferencia de otros *Saliency Maps* es que los *Class Activation Maps* toman en cuenta la clase a la que la imagen pertenece para determinar la información que es relevante según la misma. Por lo tanto, los *Class Activation Maps* determinan las zonas de una imágen que más representativas son de una clase específica[30].

Para generar los *Class Activation Maps* se ha definido la función *drawCAM()*, que está basada en el código de documentación de *Keras* correspondiente a *grad_cam*[31]. A la fnción *drawCAM()* hay que indicarle, mediante argumentos, la ruta de la imagen seleccionada (*img_path*), el tamaño de la misma (*size*), el modelo a partir del que generar los *Class Activation Maps* (*model*), una *key (model_key),* la ruta en la que guardar el mapa resultante (*cam_path*), el valor del parámetro *alpha* (*alpha*) y el pre procesado que se debe aplicar a la imágen (*pre_fn*).

Dentro de la función *drawCAM()* encontramos, en primer lugar, un diccionario que contiene la posición de la última capa convolucional en cada una de las redes neuronales con las que se trabaja. Es la *key* para este diccionario (*model_key*) la que se pasa como argumento a *drawCAM()*, para obtener la posición de la última capa convolucional según la red neuronal con la que se trabaje en cada momento. Ha sido necesario crear este diccionario porque los nombres de las capas neuronales cambian si las redes se ejecutan varias veces consecutivas, por lo que la única manera de saber que se accede siempre a la capa correcta es hacerlo a través de su posición.

Lo primero que hace *drawCAM()* es aplicar el pre procesado necesario en cada caso a la imágen de entrada, además de redimensionarla por los mismos motivos que en la función *draw_saliency*, y siguiendo el mismo procedimiento que en la misma. A continuación se usa *GradientTape()* para observar los gradientes de la ejecución de uno de los modelos, y se genera un *heatmap* que refleja las zonas que el modelo en cuestión considera más representativas de cada clase. En otras palabras, muestra la zona de la imagen en la que el modelo se ha fijado más para hacer la predicción por identificarla como el área de la imagen más característica de la clase predecida.

Finalmente, se realizan una serie de transformaciones sobre el heatmap que permitirán superponerlo a la imagen que se ha usado como *input* para generar el *Class Activation Map.*

Generación de matrices de Saliency Maps y Class Activation Maps

Para que los mapas de las distintas redes fuesen comparables entre ellos, se han aplicado sobre una misma imagen de cada una de las clases del problema. La imagen usada para generar los mapas de cada clase se ha elegido de manera aleatoria, generando un número entero entre 1 y 5000 gracias a la función *randint()* de la librería *random*.

Una vez escogidas las imágenes a utilizar, se han usado varios bucles para crear los mapas, uno para los *Saliency Maps* y el otro para los *Class Activation Maps*. Cada bucle genera uno de los tipos de mapa de cada imagen seleccionada según la ejecución de una de las redes. Además, el bucle en cuestión también guarda los mapas de cada tipo en un directorio específico, indicando en el nombre del archivo la clase que representan y la red neuronal con la que se ha generado el mapa. Por lo tanto, se tiene un directorio con todos los *Saliency Maps* y otro directorio con todos los *Class Activation Maps.*

Finalmente se generan las matrices de mapas que se presentan como resultados. Hay una matriz para *Saliency Maps* y otra para los *Class Activation Maps.* Son matrices de tres columnas y cinco filas. En cada columna de una matriz se observan imágenes correspondientes a una de las clases del problema (adenocarcinoma, tejido benigno, carcinoma de células escamosas). En la primera fila, se observan las imágenes seleccionadas para generar los mapas de cada clase. El resto de las filas corresponden a los mapas generados a partir de cada una de las redes neuronales aplicadas (*SimpleConv, Efficient, InceptionV3, ResNet50*).

RESULTADOS Y DISCUSIÓN

Para evaluar la efectividad de los modelos se ha usado el *classification_report* de *sklearn*, con el que se obtienen los valores de *precision, recall* y *f1-score* de cada modelo para cada clase. Estos resultados pueden consultarse en las **Tablas 1. a 4.** Además, en la **Tabla 5.** se incluye el número de parámetros de cada modelo, la *validation accuracy* más alta a la que han llegado en su entrenamiento; y los valores promedio de *precision, recall* y *f1-score*.

Tabla 1. Métricas por clase aplicando la red *SimpleConv*:

	Precision	Recall	f1-score
Tejido benigno (1)	0.99	1.0	1.0
Adenocarcinoma (0)	0.99	0.93	0.96
Carcinoma de células escamosas (2)	0.94	0.99	0.96

Tabla 2. Métricas por clase aplicando la red *Efficient*:

	Precision	Recall	f1-score
Tejido benigno (1)	0.98	1.0	0.99
Adenocarcinoma (0)	0.92	0.94	0.93
Carcinoma de células escamosas (2)	0.96	0.92	0.94

Tabla 3. Métricas por clase aplicando la red *InceptionV3*:

	Precision	Recall	f1-score
Tejido benigno (1)	1.0	1.0	1.0
Adenocarcinoma (0)	0.98	0.95	0.96
Carcinoma de células escamosas (2)	0.95	0.98	0.97

Tabla 4. Métricas por clase aplicando la red *ResNet50*:

	Precision	Recall	f1-score
Tejido benigno (1)	1.0	1.0	1.0
Adenocarcinoma (0)	0.99	0.98	0.99
Carcinoma de células escamosas (2)	0.98	0.99	0.99

En primer lugar, se observa que las cuatro redes neuronales tienen muy buen desempeño, puesto que todas logran resultados superiores al 0.90 para todas las métricas y clases. Además, las redes *InceptionV3* y *ResNet50* generan resultados muy destacables, con un 0.95 o más en todas las métricas.

El hecho de que *InceptionV3* y *ResNet50* consigan mejores resultados que *SimpleConv* y *Efficient* se explica por el hecho de que las primeras son redes con arquitecturas más complejas y mejor optimizadas y, en consecuencia, pueden hacer distinciones más finas entre las distintas clases. Además, al haberlas utilizadas junto a los *weigths* que ajustaron al ser entrenadas con *datasets* muy grandes y complejos, tienen una muy alta capacidad de análisis de imágenes.

Las cuatro redes neuronales consiguen métricas ligeramente superiores para las imágenes de tejido benigno (*clase 1)* que para las otras dos clases, aunque sería necesario hacer algún análisis para comprobar si dicha diferencia tiene significación estadística, puesto que es una variación pequeña. En caso de ser así, podría plantearse la opción de aplicarse una red que primero diferencie los pacientes sanos de los enfermos, y pasar los resultados identificados como enfermos a una segunda red entrenada solo para distinguir entre los dos tipos de cáncer.

Tabla 5. Número de parámetros y promedios de las métricas para cada red neuronal

	Nº de parametros	Accuracy	Precision	Recall	f1-score
SimpleConv	224,225,459 (855.35 MB)	0.9647	0.97	0.97	0.97
Efficient	7,329,582 (27.96 MB)	0.9427	0.95	0.95	0.95
InceptionV3	21,802,784 (83.17 MB)	0.9907	0.98	0.98	0.98
ResNet50	150,634,851 (574.63 MB)	0.9953	0.99	0.99	0.99

Los promedios de las métricas son también muy buenos en todos los casos, estando en las cuatro redes por encima del 0.95. Según estos promedios, la *ResNet50* es la red con un mejor desempeño, seguida de la *InceptionV3,* la *SimpleConv* y, finalmente, la *Efficient.* Las redes neuronales que se han diseñado desde cero tienen probablemente un peor desempeño porque su arquitectura es completamente secuencial, por lo que modificarla podría mejorar sus métricas.

Teniendo en cuenta que se está trabajando para diagnosticar cáncer de pulmón, se considerará menos perjudicial diagnosticar cómo enfermo a un paciente sano, en contraposición con diagnosticar a un paciente enfermo cómo sano. Esto se debe a que diagnosticar un paciente enfermo de cáncer cómo sano podría potencialmente provocar que no recibiese tratamiento en ese momento, con un previsible empeoramiento de su caso como resultado. Por contra, diagnosticar un paciente sano como enfermo probablemente tenga poco impacto sobre el mismo, ya que si se le considera enfermo de cáncer algún médico especialista tendrá que revisar su caso, y tendría que ser capaz de darse cuenta de que se trata de un falso positivo.

Es por esto que se considerará mejor opción para el problema particular que se trata a la red neuronal con valores de *Recall* mayores, puesto que esta métrica penaliza la predicción de falsos negativos que queremos evitar. En este contexto, se consideraría que la *ResNet50* es la que mejor soluciona el problema, consiguiendo en promedio un 0.99 de *Recall*, además de no comprometer tampoco las otras métricas. Sin embargo, la red *InceptionV3,* con un *Recall* promedio de 0.98, debería tenerse también en muy buena consideración, dado que una diferencia tan pequeña podría ser únicamente un reflejo de la ejecución que se ha llevado a cabo.

Cabe mencionar que los resultados que se presentan corresponden únicamente a una ejecución por red neuronal. Hubiese sido preferible ejecutar todas las redes un cierto número de veces, a modo de réplicas, para obtener las métricas como promedios de los resultados de todas esas ejecuciones. Trabajar de esa manera hubiese proporcionado resultados más robustos, con más relevancia estadística, puesto que se tendría en cuenta la variabilidad entre ejecuciones. Aún siendo una mejor opción, la realización de réplicas no ha sido posible debido a limitaciones de tiempo y de la capacidad de procesamiento del equipo.

Al ejecutar las redes una sola vez, se ha usado una *seed* para que la aleatoriedad se generase de la misma forma en todas ellas, tratando así de que los resultados fuesen lo más comparables posible. Sin embargo, estos resultados podrían no representar correctamente la capacidad de las redes en cuestión, al representar un solo caso. En resumen, estas métricas son indicadores del desempeño de las redes, pero para poder asegurar cualquier conclusión sería necesario llevar a cabo ejecuciones adicionales a modo de réplicas.

Si nos fijamos, por otra parte, en los parámetros, observamos que la red con un menor número de parámetros es, con diferencia, la *Efficient.* La siguiente en orden creciente de parámetros sería la *InceptionV3,* que tiene 3 veces más parámetros que la *Efficient.* En tercer lugar se coloca la *ResNet50*, con 20 veces más parámetros que la *Efficient.* Finalmente, la red con un mayor número de parámetros es la *SimpleConv,* que tiene 30 veces más parámetros que la *Efficient.*

El hecho de que *Efficient* tenga muchos menos parámetros que las otras redes ha hecho que su entrenamiento fuera notablemente más rápido que en los otros casos. Por ende, ha tenido ventajas en el momento de desarrollarla. Probablemente el número reducido de parámetros de esta red también podría hacer que fuera capaz de realizar predicciones más rápidamente, aunque el tiempo que tarda una red en realizar una serie de predicciones no depende únicamente del número de parámetros que contiene, sino que tiene un papel clave su arquitectura (tanto las capas que la conforman como la estructura en sí misma, que puede, por ejemplo, permitir realizar varias operaciones al mismo tiempo).

Se tiene constancia del tiempo que han tardado las redes neuronales en realizar las predicciones sobre el conjunto de testeo, para lo que la *Efficient* ha necesitado únicamente 2 segundos, frente a los 4 segundos de la *SimpleConv*, los 11 segundos de la *InceptionV3,* o los 53 segundos de la *ResNet50.* Estos datos, si bien pueden hacer sospechar que la *Efficient* es considerablemente más rápida que las demás para llevar a cabo sus predicciones, deberían valorarse más arduamente. La necesidad de dicha comprobación radica en que los tiempos mencionados derivan de ejecuciones que se dieron en distintos momentos, además de que en ocasiones el ordenador estaba realizando varios procesos a la vez, y de que la *ResNet* fue ejecutada en la *CPU* en lugar de la *GPU*. Para poder ser verdaderamente comparables debería calcularse el tiempo de inferencia, asegurándose de que las redes se ejecutan en igualdad de condiciones[32].

Además, al tener *Efficient* un menor número de parámetros, tiene también menos requerimientos técnicos, por lo que no necesitaría equipos informáticos tan potentes como las demás, disminuyendo potencialmente el costo de una hipotética implementación, así como el posterior mantenimiento.

Finalmente, se comparan los Saliency Maps y Class Activation Maps de las matrices generadas, que pueden observarse en las **Figuras 6. y 7**.

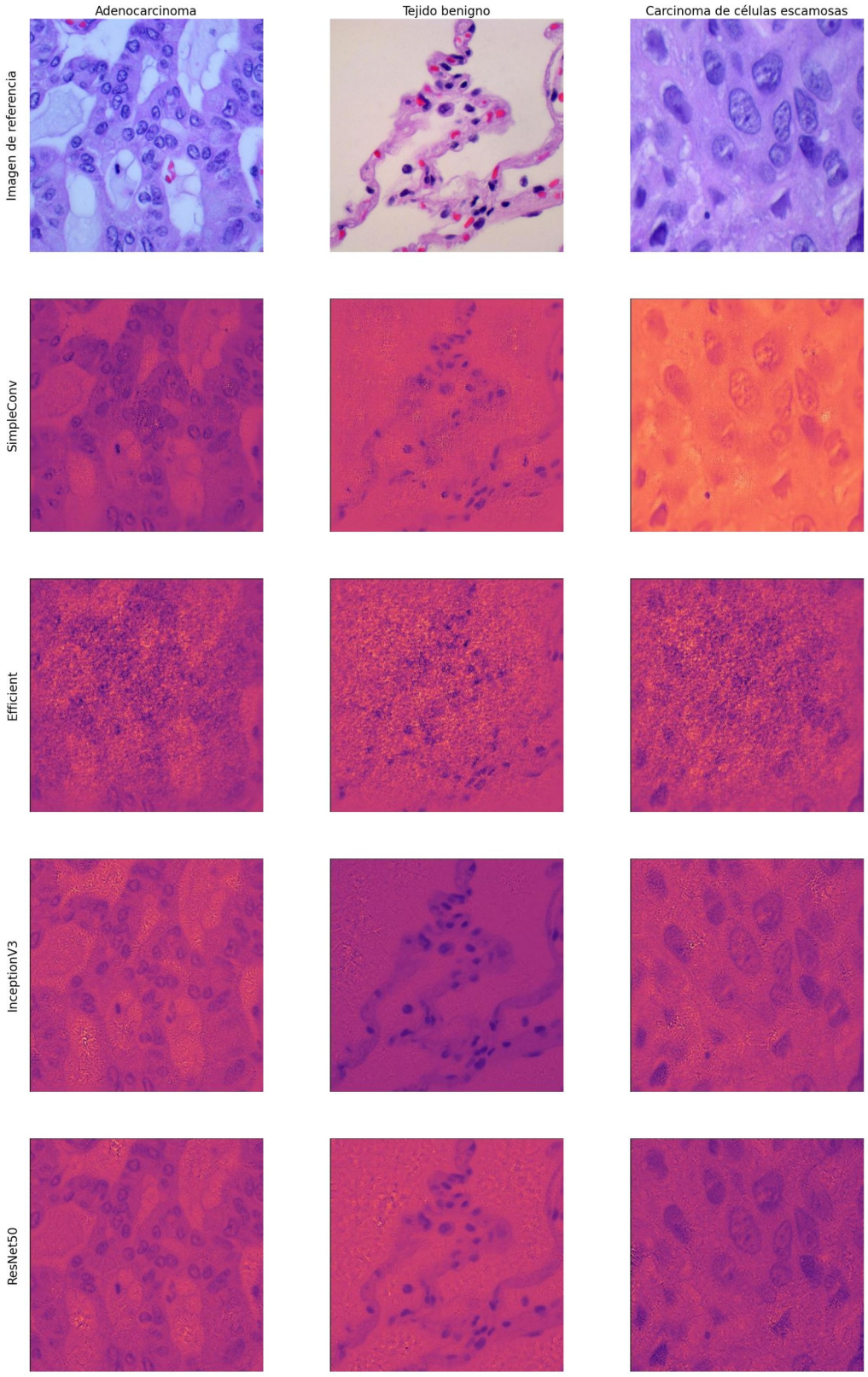

Figura 6. *Saliency Map* de cada red neuronal para las imágenes seleccionadas

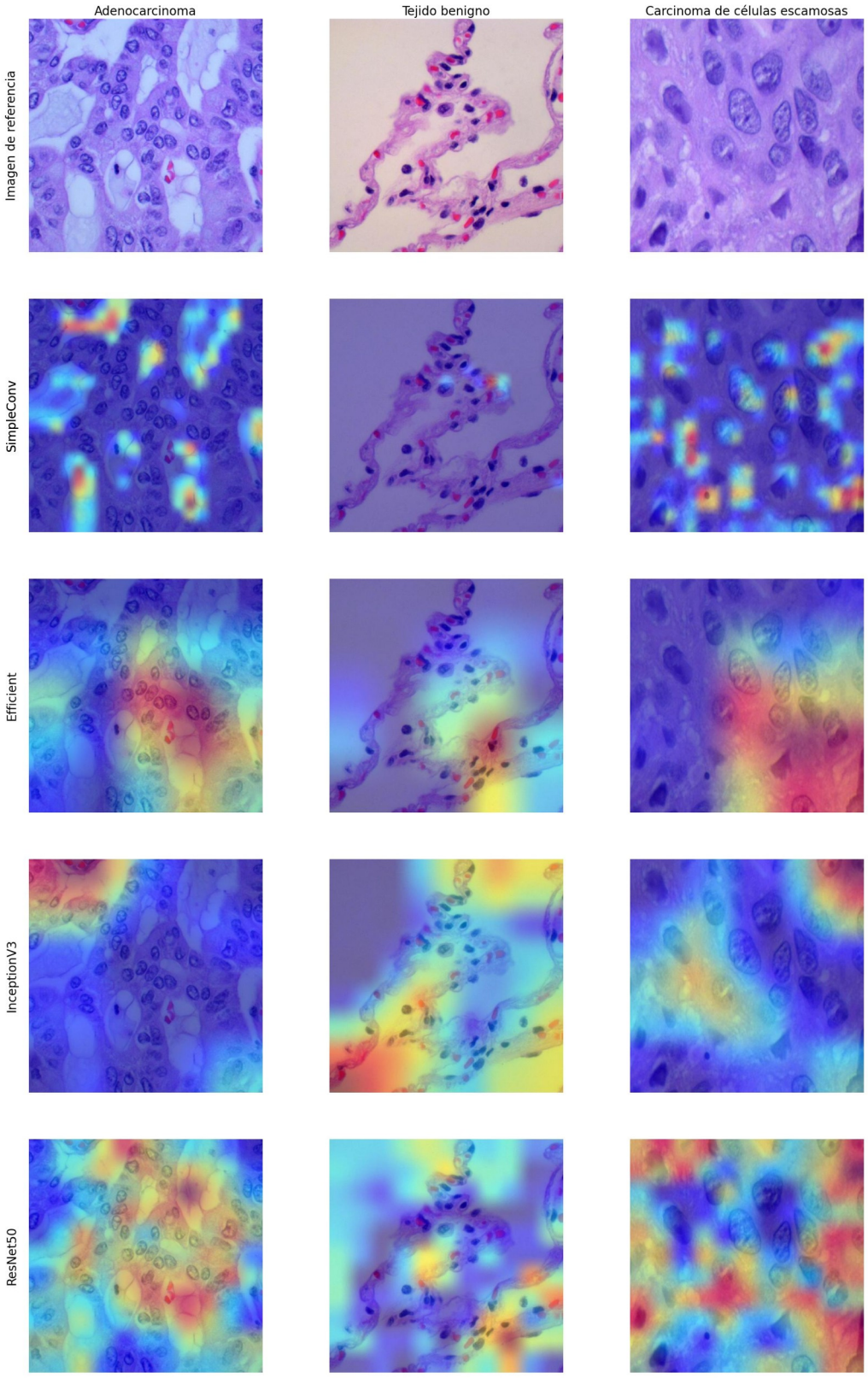

Figura 7. *Class Activation Map* de cada red neuronal para las imágenes seleccionadas

En la **Figura 6.** se muestran los *Saliency Maps* de cada red neuronal para las tres imágenes de la primera fila de la matriz. En el caso de la red *SimpleConv* se observa muy poca activación, especialmente para la imagen de carcinoma de células escamosas. Además, tanto en *SimpleConv* como en *Efficent* se observa que la activación es relativamente similar para las tres clases, mientras que en *InceptionV3* y *ResNet50* parece ser más específica a cada imagen. Esto indicaría que *InceptionV3* y *ResNet50* son más capaces de diferenciar las clases en cuestión, probablemente por haber conseguido analizar detalles más finos.

Esta última idea se ve reforzada al observar la **Figura 7.** , que reúne los *Class Activation Maps* de las redes para las imagenes seleccionadas. En esta matriz se representa la importancia de distintas zonas en las imágenes para la predicción del modelo, según un gradiente entre azul y rojo, donde el azul representa la mínima importancia para la predicción, y el rojo la máxima importancia.

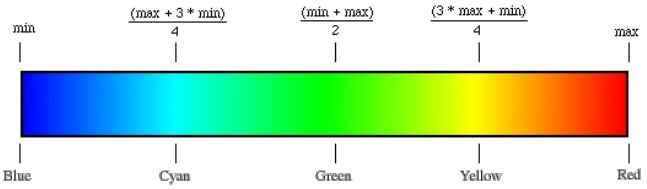

Figura 8. Gradiente de color usado en los Class Activation Maps. *MathWorks (s.f).*

Observando los *Class Activation Maps* se puede ver que la red *SimpleConv* genera mapas bastante diferentes a las otras redes. Para la imágen de adenocarcinoma, parece fijarse en las zonas más claras de la imágen original, ya que marca sus contornos casi exactamente. Para la imágen de tejido benigno, presenta muy poca activación, marcando puntos aislados, sin degradados. Finalmente, para la imagen de carcinoma de células escamosas marca también zonas concretas, aunque de forma menos aislada que en el tejido benigno. Se cree que la diferencia entre los mapas de *SimpleConv* y las otras redes puede deberse a lo poco profundo de su arquitectura, que podría dificultarle la extracción de características.

Las otras tres redes neuronales generan degradados de color más suaves, y en general marcan áreas mayores a las que marca *SimpleConv*. Se observa que estas tres redes están relativamente de acuerdo respecto a algunas de las zonas importantes para las imágenes de tejido benigno, aunque sus mapas para los dos tipos de cáncer difieren más, especialmente los de la *ResNet50* respecto a las otras dos.

Dado que la activación de *SimpleConv* difiere considerablemente de las demás, se sospecha que su capacidad de identificar características relevantes es peor, por lo que su explicabilidad podría ser inferior a la de las otras redes. Por lo tanto, es probable que *Efficient, InceptionV3* y *ResNet* sean mejores opciones, aunque sería de gran importancia contar con la opinión de un experto en la materia que pudiese comentar hasta que punto las zonas que cada red considera importantes para el diagnostico son realmente características de las clases correspondientes.

CONSIDERACIONES DE IMPLEMENTACIÓN

En el caso de que alguna de estas redes neuronales quisiese implementarse como parte de un producto de inteligencia artificial para apoyo en el diagnóstico de cáncer de pulmón, se considera que lo más adecuado sería subirlo a la nube para facilitar su uso a cualquier profesional sanitario que lo requiera, siempre pensando en el máximo beneficio para los pacientes. Se recomendaría que el código del aplicativo se incluyera en un *Docker* para crear una máquina virtual que se podría compartir en la nube a través de los servicios de *Microsoft Azure*.

También se considera recomendable instaurar un sistema de inicio de sesión para que solo los perfiles autorizados para ello puedan acceder al producto. De esta manera, se podría saber en todo momento qué personas tienen acceso a qué información. Una manera sencilla sería aprovechar las credenciales que los profesionales sanitarios usan para acceder al sistema informático propio del hospital, aunque por motivos de seguridad lo más indicado sería evitar que la contraseña fuese la misma. También se recomendaría evitar usar contraseñas que hayan sido filtradas en algún ataque informático. Esto puede comprobarse, por ejemplo, a través del sitio web *haveibeenpwned.com,* en el apartado *Passwords*[33]. Adicionalmente, se recomendaría que se usara un generador de contraseñas para crear contraseñas seguras, además de un gestor de contraseñas para guardarlas de manera segura. Algunos gestores de contraseña incluyen también la generación de contraseñas seguras, por lo que ambas acciones podrían llevarse a cabo a través de una misma aplicación.

Además, también por seguridad se instaría a los usuarios del producto a cambiar su contraseña cada seis meses, además de hacer un cambio extraordinario de contraseña en caso de detectarse algún problema de seguridad informática.

Al estar el producto en la nube, un error de funcionamiento del mismo no afectaría al flujo de trabajo interno y, en consecuencia, el sistema informático del hospital funcionaría igual, aunque sin poder usar el producto en cuestión. Esto es muy positivo, ya que es clave evitar que la implantación de un producto pueda causar fallos en el resto del sistema.

Mantener el producto en la nube tendría un cierto coste económico asociado, que está directamente ligado al tamaño del mismo. Dado que los presupuestos de los centros sanitarios suelen ser ajustados, se aconsejaría implementar un producto que no tenga un tamaño excesivamente grande para así disminuir el coste, siempre que genere resultados satisfactorios. Además, implementar un producto liviano permitiría subirlo a varios servidores por un coste reducido. Esta posibilidad podría ser de interés si quisiese acceder al producto desde distintas regiones del mundo, puesto que garantizaría una disminución del tiempo de latencia al usarlo.

El tiempo de latencia es el tiempo que tardan los datos a transferirse a través de una red[34]. En el caso de utilizar el producto de Inteligencia Artificial subido a la nube, el tiempo de latencia representaría el tiempo que los datos necesitan para viajar del ordenador del hospital al servidor que mantiene el producto en la nube; y viceversa. Subir el producto en varios servidores permitiría al usuario final conectarse a un servidor más cercano, acortando la distancia a recorrer, y permitiendo en consecuencia que el tiempo de latencia disminuya. Al reducir el tiempo de latencia, se reduciría el tiempo que el modelo necesitaría para devolver sus predicciones, de manera que se podría aplicar sobre una cantidad mayor de datos en un mismo intervalo de tiempo.

Si bien es cierto que los modelos que se han aplicado han demostrado una gran capacidad predictiva, su uso se recomienda únicamente como herramientas de apoyo a la toma de decisiones, por lo que se recomienda usarlos siempre con supervisión de profesionales sanitarios. El objetivo principal de la herramienta es que pueda dar una propuesta de diagnóstico basada en su capacidad de aprendizaje, además de indicar al profesional en qué zonas de la imagen se ha centrado para llegar a esa conclusión. Sería entonces tarea del profesional tomar la decisión final, teniendo en cuenta la predicción de la red, pero anteponiendo su criterio y experiencia. Además, si el producto se diseñase para que el profesional pudiese indicarle la decisión que ha tomado en última instancia, sería posible que aprendiese de manera continua.

PREVISIÓN DE IMPACTO

La implementación de un producto de inteligencia artificial basado en alguna de las redes neuronales analizadas en el proyecto tendría el potencial de dar soporte a los profesionales sanitarios en la toma de decisiones, dada su notable capacidad de generar predicciones acertadas. Dicho producto podría complementar el conocimiento del profesional sanitario y evitar que su capacidad de diagnóstico empeore en caso de analizar muchas preparaciones histológicas seguidas, debido a la fatiga visual que ello conlleva.

En caso de desacuerdo entre el producto de IA y el profesional sanitario, la opción de generar mapas de activación para conocer las áreas en las que se basa el producto para elaborar sus predicciones permitirían analizar su veracidad. Se recomendaría usar los *Class Activation Maps* frente a los *Saliency Maps*, al ser más informativos y fáciles de interpretar.

Por otra parte, considerando que todas las redes tienen un desempeño extraordinario en la correcta identificación de tejido benigno, el hipotético producto podría utilizarse como una herramienta de priorización. El producto podría analizar las imágenes y etiquetarlas según la urgencia con la que se requiera que sean revisadas: las imágenes que el producto clasificase como tejido benigno no serían urgentes, mientras que las que clasificase como cáncer serían urgentes. A continuación, el sanitario revisaría primeramente las imágenes etiquetadas como urgentes, puesto que en caso de confirmarse la predicción, los pacientes en cuestión requerirían seguimiento médico inmediato. Inclusive podría implementarse el producto para indicar que las imágenes para las que predice un carcinoma de células escamosas son de una urgencia mayor que las de adenocarcinoma, puesto que los cánceres de células escamosas habitualmente empeoran con más rapidez. En consecuencia, sería lógico priorizar ese cáncer para que los pacientes que lo padezcan reciban atención antes. Finalmente, después de revisar las imágenes urgentes, el médico revisaría las no urgentes, aquellas que el producto considera tejido benigno.

Otra opción es que el producto directamente separe las imágenes que detecta como tejido benigno para que no las llegue a revisar ningún sanitario. Esto podría ser muy positivo, porque aligeraría considerablemente la carga de trabajo de los sanitarios, y se cree factible dada la gran capacidad que estas redes han demostrado para detectar las muestras benignas. Sin embargo, en caso de decidirse

por esta opción se recomendaría hacer controles regulares para asegurarse de que la herramienta mantiene su capacidad de detectar tan bien los casos benignos. Una opción sería recuperar un cierto número de esas imágenes aleatoriamente y pasarlas a un sanitario para que indique qué diagnostico consideraría apropiado. Podría entonces decidirse que si la herramienta sigue diagnosticando correctamente a los benignos en más de un cierto porcentaje (quizá un 98 o 99%) se mantiene el sistema. Si se detectase que su capacidad baja, sin embargo, sería necesario modificar el sistema.

Es evidente que aún con ese control se estaría aceptando cierto error, por lo que un cierto número de personas enfermas se estarían detectando como sanas, pero no hay que olvidar que cualquier método de diagnóstico tiene un cierto error asociado. En el caso de la identificación de tejido benigno se considera que el potencial error es suficientemente bajo en todas las redes como para automatizar su diagnóstico siempre y cuando se realicen comprobaciones de desempeño cada pocos meses.

Para la mayoría de pacientes sanos, se contaría con el beneficio de que el diagnóstico automático para casos de tejido benigno sería muy rápido, evitando las largas esperas que a menudo se dan debido a la saturación de los sistemas sanitarios y que pueden provocar en los pacientes problemas psicológicos como estrés o ansiedad.

Para la mayoría de los pacientes enfermos, tanto la posibilidad de automatizar el diagnóstico de tejido benigno como la de priorizar las preparaciones histológicas en las que el producto detecta algún cáncer agilizaría su proceso de diagnóstico, evitando que su enfermedad avance mientras esperan diagnóstico.

Además, el producto también podría permitir analizar una mayor cantidad de imágenes histológicas en un cierto periodo de tiempo, por lo que permitiría liberar tiempo de los profesionales, que tendrían pues mayor disponibilidad para estar en contacto con los pacientes o llevar a cabo otras tareas.

También podría servir como soporte en la formación de profesionales sanitarios, gracias a la explicabilidad que se ha conseguido con los *Class Activation Maps* y *Saliency Maps*.

Finalmente, también reduciría los costes derivados de la repetición de ciertas pruebas a lo largo del tiempo de espera. Si bien la toma de imágenes para enviarlas al modelo requiere de recursos económicos, a medio y largo plazo se prevé que se amortizaría la inversión gracias al coste económico que se ahorraría con la implementación del producto en cuestión. Además, muchas de las imágenes que se requerirían para el diagnóstico mediante la herramienta que se propone suelen tomarse ya cuando se diagnostica de manera tradicional, así que no necesariamente usaría recursos adicionales a la práctica tradicional.

Se espera que el coste asociado a su implementación y a la formación de los profesionales que tendrán que usarla pueda ser compensado al usar la herramienta debido al gran impacto que se espera que pueda provocar.

CONCLUSIONES

Se ha constatado la potencia de la inteligencia artificial y, concretamente, de las redes neuronales convolucionales, para resolver tareas de clasificación de imágenes. Asimismo, se ha comprendido más profundamente su potencial en el ámbito de la salud. También se ha observado la importancia de la arquitectura de las redes y del proceso de optimización de las mismas, ya que todo ello afecta considerablemente a la capacidad de predicción de estos modelos. Asimismo, se han comprobado las ventajas de usar *Transfer Learning* para aplicar redes pre entrenadas con *datasets* masivos y se ha visto que normalmente producen mejores resultados, aunque la diferencia en el caso específico que se ha abordado no es muy grande a causa de la simplicidad de la tarea de clasificación.

En cuanto a las redes analizadas, se considera que *SimpleConv* es la menos adecuada para implementarla en un producto de Inteligencia Artificial dado su elevado número de parámetros, que aumentaría enormemente el costo de implementación y los requerimientos técnicos necesarios para usarla. Además, su ejecución sería probablemente más lenta que en otros casos. Pese a que genera métricas destacables, otras redes tienen aún mejor desempeño, por lo que *SimpleConv* no ofrece ningún beneficio razonable que pudiera compensar sus inconvenientes.

Las otras tres redes se consideran más aptas para su implementación, puesto que generan métricas destacables y parecen tener una mejor explicabilidad. Todo ello, mientras tienen considerablemente menos parámetros, por lo que también se prevé que sean más rápidas y tengan requerimientos inferiores a *SimpleConv*. Dependiendo del tipo de implementación por la que finalmente se optase, algunas de ellas pueden ser más recomendables que otras.

Si se implementase un producto con el objetivo de automatizar completamente el diagnóstico, sin supervisión alguna por parte de ningún profesional de la salud, sería preferible basarse en la red que mejor desempeño demuestre, aunque ello signifique un mayor gasto económico o un diagnóstico más lento. Según los resultados obtenidos, la *ResNet50* sería la red neuronal más indicada en este caso, aunque se recomendaría llevar a cabo una serie de ejecuciones de las redes a modo de réplicas para evaluar mejor el desempeño de las mismas. Esto es especialmente relevante para poder comparar mejor la *ResNet50* y la *InceptionV3*, puesto que los resultados obtenidos hasta ahora son muy similares para ambas.

Sin embargo, la implementación de un producto sin supervisión alguna no es recomendable, puesto que la opinión y experiencia de los profesionales sanitarios es clave para un correcto diagnóstico. Además, dichos profesionales son los que tratan directamente con el paciente, por lo que podrían disponer de información adicional de importancia para la interpretación de las imágenes. En consecuencia, es mucho más recomendable plantear el producto como una herramienta de apoyo al diagnóstico.

Si se opta por la opción de priorizar las imágenes a revisar por el profesional según su urgencia, se considera que la mejor red a implementar es la *Efficient*, dado que al ser considerablemente más liviana que las demás permitiría una implantación más barata y sería potencialmente capaz de ofrecer predicciones más rápidamente. Esta red tiene unas métricas ligeramente peores que las demás en los casos de adenocarcinoma y carcinoma de células escamosas, pero siguen siendo métricas muy favorables. Se considera adecuado aceptar un desempeño ligeramente peor considerando los beneficios comentados y el hecho de que la decisión última sería tomada por un profesional sanitario. Además, se estima que esta red podría mejorarse si su arquitectura se modificase para no ser completamente secuencial y se aumentase su profundidad, mientras que se mantendrían los beneficios asociados a la misma.

Por otra parte, si se optase por un producto que detectase automáticamente los casos de tejido benigno de manera que el profesional sanitario únicamente analizase las imágenes clasificadas como adenocarcinoma o carcinoma de células escamosas, la red *Efficient* sería también la más adecuada. Esta red tiene una capacidad excelente para identificar los casos de tejido benigno, al igual que las demás, pero como ya se ha comentado, tiene beneficios muy interesantes sobre las mismas.

Según los resultados obtenidos y considerando la importancia del criterio de los profesionales sanitarios, además de todas las ventajas atribuibles a esta opción, se concluye que la mejor alternativa es implementar la red *Efficient* en un producto para clasificar las imágenes histopatológicas y priorizar su observación por parte de un profesional sanitario según la urgencia relacionada a dicha clasificación, con la opción de automatizar completamente la detección de tejido benigno. El producto en cuestión se subiría a la nube y podría accederse desde distintos servidores, siguiendo los procedimientos ya descritos.

BIBLIOGRAFÍA

[1] Sociedad Española de Oncología Médica (2020). *Las cifras del cáncer en España 2020*. https://seom.org/seomcms/images/stories/recursos/Cifras_del_cancer_2020.pdf

[2] American Cancer Society (2023). *Cancer Facts & Figure 2023.* https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2023/2023-cff-special-section-lung-cancer.pdf

[3] Cancer Research UK. https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/lung-cancer#heading-One

[4] Ellis P. M., Vandermeer R. (2011). *Delays in the diagnosis of lung cancer*. J. Thorac. Dis; 3(3),183–188. https://doi.org/10.3978/j.issn.2072-1439.2011.01.01

[5] Bjerager, M., Palshof, T., Dahl, R., Vedsted, P., Olesen, F. (2006). *Delay in diagnosis of lung cancer in general practice.* British Journal of General Practice. 56(532), 863-868. https://pubmed.ncbi.nlm.nih.gov/17132354/

[6] Pita-Fernández, S., Montero-Martinez, C., Pértega-Diaz, S., Verea-Hernando, H. (2003). *Relationship between delayed diagnosis and the degree of invasion and survival in lung cancer.* Journal of Clinical Epidemiology. 56(9), 820-825. https://doi.org/10.1016/S0895-4356(03)00166-5.

[7] O'Rourke, N. Edwards, R. (2000). *Lung Cancer Treatment Waiting Times and Tumour Growth.* Clinical Oncology. 12(3), 141-144. https://doi.org/10.1053/clon.2000.9139.

[8] American Cancer Society. https://www.cancer.org

[9] Collins L. G., Haines C, Perkel R, Enck RE. *Lung cancer: diagnosis and management.* Am Fam Physician. 2007 Jan 1;75(1):56-63. PMID: 17225705.

[10] Aframian, A., Bohr, A., Bohr, H., Botker, J. P., Cobb, J., Cohen, G., Colombo, S., Dutt, R., Gerke, S., Hlávka, J. P., Iranpour, F., Jeddi, Z., Memarzadeh, K., Minssen, T., Mirnezami, R., Tran, K., Ward, T. y Witkowski, E. (2022). *Inteligencia artificial en el ámbito de la salud.* Elsevier. ISBN 978-84-1382-017-0

[11] Ehteshami Bejnordi, B., Veta, M., Johannes van Diest, P., van Ginneken, B., Karssemeijer, N., Litjens, G., van der Laak, J. A. W. M., the CAMELYON16 Consortium, Hermsen, M., Manson, Q. F., Balkenhol, M., Geessink, O., Stathonikos, N., van Dijk, M. C., Bult, P., Beca, F., Beck, A. H., Wang, D., Khosla, A., Gargeya, R., … Venâncio, R. (2017). *Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer.* JAMA, 318(22), 2199–2210. https://doi.org/10.1001/jama.2017.14585

[12] Sobell, M. G. (2015). *A practical guide to Ubuntu Linux*. Pearson Education.

[13] Van Rossum, G., & Drake Jr, F. L. (1995). *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam.

[14] Amazon (2023). *¿Qué es una red neuronal? https://aws.amazon.com/es/what-is/neural-network/*

[15] Chollet, F (2021). *Deep Learning with Python*. Manning. ISBN 978-1-61729-686-4

[16] Keras (s.f.). *Keras layers API.* https://keras.io/api/layers/

[17] Wang, C. *A Basic Introduction to Separable Convolutions* (14 de agosto de 2018). Towards Data Science. https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

[18] Dive into Deep Learning (s.f.). *Pooling.* https://d2l.ai/chapter_convolutional-neural-networks/pooling.html

[19] Brian W. Keelan; Robert E. Cookingham (2002). *Handbook of Image Quality*. CRC Press. ISBN 0-8247-0770-2.

[20] Salehi, A.W.; Khan, S.; Gupta, G.; Alabduallah, B.I.; Almjally, A.; Alsolai, H.; Siddiqui, T.; Mellit, A. *A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope.* Sustainability 2023, 15, 5930. https://doi.org/10.3390/su15075930

[21] Borkowski, A., Bui, M., Thomas, B. Wilson, C., DeLand, L., Mastorides, S. (2019). *Lung and Colon Cancer Histopathological Image Dataset (LC25000).* https://doi.org/10.48550/arXiv.1912.12142

[22] The Veterans Health Administration (VHA). https://www.va.gov/health/

[23] Keras (s.f.) *Metrics*. https://keras.io/api/metrics/

[24] Brownlee, J. *A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size*. (19 de agosto de 2019). Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

[25] Thakur, A. *What's the Optimal Batch Size to Train a Neural Network*? (s.d.) Weights & Biases. https://wandb.ai/ayush-thakur/dl-question-bank/reports/What-s-the-Optimal-Batch-Size-to-Train-a-Neural-Network---VmlldzoyMDkyNDU

[26] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. (2015). *Rethinking the Inception Architecture for Computer Vision* https://arxiv.org/pdf/1512.00567.pdf,%20https://arxiv.org/pdf/1409.0575.pdf

[27] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848

[28] Raj, B. *A Simple Guide to the Versions of the Inception Network* (29 de mayo de 2018). Towards Data Science. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

[29] He, K. Zhang, X., Ren, S., Sun, J. *Deep Residual Learning for Image Recognition* (2019) https://arxiv.org/pdf/1512.03385.pdf

[30] Salehi, M. *A Review of Different Interpretation Methods (Part 1: Saliency Map, CAM, Grad-CAM)* (18 de enero de 2020). Medium. https://mrsalehi.medium.com/a-review-of-different-interpretation-methods-in-deep-learning-part-1-saliency-map-cam-grad-cam-3a34476bc24d

[31] Chollet, F. *Grad-CAM class activation visualization.* (7 de marzo de 2021). Keras. https://keras.io/examples/vision/grad_cam/

[32] Geifman, A. *The Correct Way to Measure Inference Time of Deep Neural Networks* (1 de mayo de 2023). Deci. https://deci.ai/blog/measure-inference-time-deep-neural-networks/

[33] Have I been pwned? https://haveibeenpwned.com/Passwords

[34] Amazon (2023). *¿Qué es la latencia de red?* https://aws.amazon.com/es/what-is/latency/