Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Análisis de los mapas de activación para el diagnóstico explicado de lesiones de la piel

Autor: Ignacio Trujillo Artillo

Tutor: María del Carmen Serrano Gotarredona
y Begoña Acha Piñero

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías de Telecomunicación

Análisis de los mapas de activación para el diagnóstico explicado de lesiones de la piel

Autor:
Ignacio Trujillo Artillo

Tutor:
Maria del Carmen Serrano Gotarredona y Begoña Acha Piñero
Catedrática de Universidad

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo Fin de Grado: Análisis de los mapas de activación para el diagnóstico explicado de lesiones de la piel

Autor: Ignacio Trujillo Artillo

Tutor: Maria del Carmen Serrano Gotarredona
y Begoña Acha Piñero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Uno de mis poemas favoritos dice así.

"Cuando emprendas tu viaje a Itaca
pide que el camino sea largo,
lleno de aventuras, lleno de experiencias
Cuando emprendas tu viaje a Itaca
pide que el camino sea largo,
lleno de aventuras, lleno de experiencias.
No temas a los lestrigones ni a los cíclopes
ni al colérico Poseidón,
seres tales jamás hallarás en tu camino,
si tu pensar es elevado, si selecta
es la emoción que toca tu espíritu y tu cuerpo.
Ni a los lestrigones ni a los cíclopes
ni al salvaje Poseidón encontrarás,
si no los llevas dentro de tu alma,
si no los yergue tu alma ante ti.

Pide que el camino sea largo.
Que muchas sean las mañanas de verano
en que llegues -¡con qué placer y alegría!-
a puertos nunca vistos antes.
Detente en los emporios de Fenicia
y hazte con hermosas mercancías,
nácar y coral, ámbar y ébano
y toda suerte de perfumes sensuales,
cuantos más abundantes perfumes sensuales puedas.
Ve a muchas ciudades egipcias
a aprender, a aprender de sus sabios.

Ten siempre a Itaca en tu mente.
Llegar allí es tu destino.
Mas no apresures nunca el viaje.
Mejor que dure muchos años
y atracar, viejo ya, en la isla,
enriquecido de cuanto ganaste en el camino
sin aguantar a que Itaca te enriquezca.

Itaca te brindó tan hermoso viaje.
Sin ella no habrías emprendido el camino.
Pero no tiene ya nada que darte.
Aunque la halles pobre, Itaca no te ha engañado.
Así, sabio como te has vuelto, con tanta experiencia,
entenderás ya qué significan las Itacas"

Konstantino Kavafis

Mi Ítaca durante estos 4 años ha sido terminar el grado de ingeniería de las tecnologías de telecomunicaciones y cuando miro el camino me doy cuenta de la importancia que ha tenido. El camino ha estado lleno de experiencias y personas que me han llevado a estar aquí. Por eso este proyecto es gracias a ellos.

En primer lugar, debo agradecer a mi familia porque siempre son un apoyo fundamental y siempre están pendiente de todos los detalles. Tambien a todos mis compañeros a los que algunos ya puedo denominar mis amigos pues sin ellos el viaje a Ítaca hubiera sido más difícil .No pueden faltar esos amigos de la infancia ni Conchi que han estado pendiente de todo y siempre que necesitaba algo.

Por último, agradecer a todos los profesores que he tenido durante estos 4 cursos y en especial a Carmen y Begoña por toda la ayuda y paciencia que me han brindado en este proyecto. La verdad que ha sido una suerte poder terminar con este proyecto de la mano de ellas.

Ignacio Trujillo Artillo

Sevilla, 2023

Resumen

El melanoma es un cáncer de piel, y la detección temprana a través de la dermatoscopia es fundamental para mejorar la supervivencia. El uso de redes neuronales y el análisis de imágenes dermatoscópicas ayuda a proporcionar diagnósticos más precisos, lo que contribuye a un mejor manejo de la enfermedad y una reducción considerable de la tasa de mortalidad.

En este trabajo se realiza un algoritmo en python usando redes neuronales profundas (VGG-16) para la clasificación de lesiones de piel entre melanomas (maligna) y no melanoma (benigna). Para ello se usarán imágenes dermatoscópicas de la base de datos de ISIC 2020 para entrenar la red. Además, dará a la salida un mapa de activación de las zonas de la imagen que tienen mayor peso en la decisión del diagnóstico. Esto supone una explicación del diagnóstico obtenido además de una validación del funcionamiento del método.

Con esto se pretende ayudar a los especialistas a realizar sus tareas debido al gran número de casos que están apareciendo en estos momentos.

Abstract

Melanoma is a skin cancer, and early detection through dermoscopy is critical to improve survival. The use of neural networks and dermoscopic image analysis helps to provide more accurate diagnoses, which contributes to better disease management and a considerable reduction in the mortality rate.

In this work, a Python algorithm using deep neural networks (VGG-16) is developed for the classification of skin lesions into melanoma (malignant) and non-melanoma (benign). For this, dermoscopic images from the ISIC 2020 database will be used to train the network. In addition, it will give the output an activation map of the areas of the image that have the greatest weight in the diagnostic decision. This involves an explanation of the diagnosis obtained as well as a validation of the performance of the method.

This is intended to help specialists perform their tasks due to the large number of cases that are appearing at the moment.

~Traducido by Deepl~

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 PROBLEMA CLÍNICO Y MOTIVACIÓN

El melanoma, término utilizado para referirse a los tumores melánicos o pigmentados, representa una mayoría de los cánceres entre los europeos. Esto se debe en gran medida a la exposición intermitente de radiación ultravioleta y las quemaduras solares en la infancia. [1]

Según la IARC (Centro Internacional de Investigaciones sobre el Cáncer) en 2020 se diagnosticaron 325.000 casos en todo el mundo, de los cuales fueros 174.000 hombres y 151.000 mujeres. Además, se registraron 57.000 muertes debido a los tumores melánicos. Asimismo, añaden que en los próximos 20 años la cifra de casos aumentará en un 50% y la mortalidad en un 67% en el mismo rango de tiempo.[2]

Aunque España tiene una tasa inferior al resto de Europa el melanoma sigue siendo una causa importante de cáncer y por lo tanto su estudio es de gran importancia [1]. Cabe añadir que la detección temprana es crítica ya que la tasa de supervivencia estimada a 5 años para el melanoma fue superior al 99 % cuando se detectó en una etapa temprana, pero se redujo a alrededor del 14 % cuando se detectó en una etapa posterior.[3]

Para la detección de lesiones de piel se unas la dermatoscopia, que es una técnica de diagnóstico no invasiva, para la obtención de este tipo de imágenes se utiliza el dermatoscopio. El dermatoscopio puede visualizar estructuras dérmicas profundas que normalmente no se pueden ver a simple vista. El diagnóstico se basa en estas estructuras, también conocidas como parámetros o criterios dermatoscópicos.[4]

La dermatoscopia ayuda al diagnóstico temprano de lesiones potencialmente malignas, especialmente el melanoma maligno, y mejora la precisión diagnóstica de las lesiones hiperpigmentadas.[4]

Sin embargo, existen distintas estructuras que se diagnostican con dermatoscopia.

- Nevus: Son lunares que se producen por la acumulación de melanocitos (células que dan color a la piel), son tumores benignos.[5]
- Léntigo solar: son lesiones de la piel que aparecen por su envejecimiento debido a la exposición a la radiación ultravioleta.[6]
- Máculas cafés con leche son lesiones de la piel que se debe un aumento de células madre y de hepatocitos, produciendo una hiperpigmentación de la zona.[7]
- Keratosis seborreica: es una lesión de la piel que produce una hiperpigmentación de la zona. Este tipo de lesión no tiene una causa clara, aunque se cree que es hereditario y deberse a la radiación solar.[8]

Debido a la gran variedad de lesiones y a un aumento de los casos en los últimos años se ha desarrollado un método de ayuda al diagnóstico usando redes neuronales mediante un aprendizaje supervisado. Para ello se ha usado la base de datos del reto ISIC 2020 (Skin Lesion Analysis Towards Melanoma Detection) [9]. Esta base de datos contiene 33.126 imágenes dermatoscopias para el entrenamiento identificadas como malignas y benignas.

Además, para intentar aclarar los resultados de dicho método, se aporta un mapa de activación sobre la imagen dermatoscopica que indica la zona en la que se ha fijado para determinar el diagnóstico. Esto es un método utilizado a modo de explicación de los resultados, de esta manera se puede comprobar el funcionamiento del método.

2 ESTADO DEL ARTE

Los algoritmos de aprendizaje profundo que se basan en redes neuronales convolucionales (CNN) tienen sus raíces en el campo de la visión por computadora y se han convertido en una de las arquitecturas más utilizadas y exitosas en el aprendizaje profundo. Aunque hubo trabajos pioneros en la década de 1980, las CNN alcanzaron un gran éxito en las tareas de reconocimiento visual cuando ImageNet consigue avances en hardware, grandes conjuntos de datos y técnicas de entrenamiento eficientes.[10]

En el año 2012, la creación innovadora de Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton en la competencia ImageNet, también conocida como "AlexNet", fue un hito en el progreso de las CNN. Una arquitectura profunda con múltiples capas convolucionales y capas completamente conectadas fue introducida por AlexNet, que demostró un desempeño excepcional en la clasificación de imágenes a gran escala.[11]

Desde entonces se han desarrollado distintas arquitecturas de CNN. Entre ellas se encuentran VGG, que debe su popularidad a su simplicidad y efectividad, consiguiendo una tasa de error del 7.3% y 7.5% en los retos de ImageNet2012 y 2013 [12], GoogLeNet conocida como Inception obtiene unas tasas de error del 6.67% [13], ResNet introduce un aprendizaje residual facilitando el entrenamiento de las redes consiguiendo una tasa de error de 3,57% [14] y DenseNet, esta introduce conexiones densas lo que supone recibir información de todas las capas anteriores de esta manera consiguen una tasa de error del 6.15%.

Por otro lado, se desarrolla una técnica denominada transfer learning que consiste en poder entrenar una red según un modelo pre-entrenado con un conjunto grande de datos sacando características generales útiles en muchos casos[15]. Esta técnica reduce el tiempo de entrenamiento ya que reduce el coste computacional al no entrenar un modelo desde cero. Además, se consigue un mejor rendimiento cuando no se dispone de una gran base datos al haber aprendido características generales a partir de otros datos.

Las CNN se han utilizado en gran medida para el análisis de imágenes médicas, utilizándose para calificar lesiones cutáneas, como el melanoma. Sin embargo, no en todo lo que se ha hecho en este campo se han usado CNN.

Gola Isasi, B. García Zapirain, A. Méndez Zorrilla [16] presenta una aplicación para el diagnóstico no invasivo de melanomas utilizando la regla ABCD y algoritmos de procesamiento de imágenes basados en reconocimiento de patrones. La regla ABCD es un método muy utilizado por los médicos dermatólogos para evaluar las lesiones cutáneas determinando si es o no un melanoma. Este método se denomina así debido a que las características buscadas son Asimetría, Borde irregular, Color variado, Diámetro. Para ello diseñan un sistema de diagnóstico asistido basado en reconocimiento de patrones.

E. Nasr-Esfahani et al. [17] Proponen un método basado en CNN para analizar automáticamente imágenes clínicas de lesiones cutáneas y clasificarlas como melanoma o no melanoma. El estudio describe cómo se recopilaron imágenes clínicas con casos de melanoma y no melanoma, así como los procedimientos de preprocesamiento utilizados para mejorar las imágenes para su análisis. Luego, utilizan estas imágenes para entrenar un modelo de CNN, optimizando su rendimiento a través de una CNN con 2 capas de convolución y 2 capas Max pooling, una después de cada capa de convolución y terminan con una capa Densa. E. Nasr-Esfahani et al. comparan su método con otros autores obteniendo los mejores valores de accuracy, NPV y PPV y valores muy buenos en sensibility y specificity.

Kaur R et al.[18] utilizan una red neuronal convolucional profunda para la clasificación de melanoma. Esta red neuronal es capaz de extraer características relevantes de las imágenes dermatoscópicas y aprender patrones distintivos asociados con el melanoma. Para ello han usado la base de datos de ISIC 2016, 2017 y 2020 que incluye tanto casos de melanoma como de lesiones benignas. Se propone el uso de capas convolucionales y capas completamente conectadas en la red neuronal para realizar la clasificación, obteniendo una accuracy de 0.81, 0,88 y 0,9 para cada base de datos.

Rashid j et al [19]. utilizan una red VGGnet o ResNet y una transferencia de aprendizaje de modelos de Deep Learning pre-entrenados de MobileNetV2. Esta red se entrena con la base de datos de ISIC 2020 [9] adaptándolo así a la detección de melanomas. De esta manera han obtenido una Accuracy del 98,2%.

2.1 Diagnostico explicado

Por otro lado, en el aprendizaje profundo, las redes neuronales pueden ayudar mucho con las tareas de clasificación, pero con frecuencia no están claras sobre cómo toman decisiones. Para clarificar la toma de decisiones existen 3 ramas distintas[20]. Explicaciones textuales, explicaciones basadas en ejemplos y explicaciones visuales que están siendo las más utilizadas últimamente.

Las explicaciones textuales proporcionan desde descriptores de características sencillos hasta un informe médico completo [20]. Para ello extraen información de los objetos presentes en las imágenes relacionándolas entre ellos y posteriormente se pasan por plantillas de frases predefinidas para crear los descriptores textuales [21].

Las explicaciones basadas en ejemplos muestran ejemplos relacionados de pacientes diagnosticados con anterioridad [20]. Para ello se utilizan redes convolucionales que detectan relaciones entre las lesiones encontradas, de esta manera se obtienen relaciones significativas entre las lesiones de las imágenes [22].

Shen et al [23]. Shen se centra en la interpretación de las decisiones de la red neuronal. Se sugieren métodos de visualización y análisis para determinar qué características son pertinentes para la clasificación de malignidad. Esto ayuda a los médicos a comprender mejor las predicciones del modelo.

Yan et al [24]. entrenaron un modelo en el estudio utilizando explicaciones basadas en datos como el tamaño de la lesión, su ubicación anatómica y las coordenadas relativas del cuerpo. Estas explicaciones ayudaron a comprender las conexiones y organizar los hallazgos de las imágenes radiológicas en la base de datos de lesiones.

Los mapas de activación de importancia se calculan utilizando Grad-CAM el gradiente de salida de la red neuronal con respecto a las características de la capa convolucional. Estos mapas de activación muestran qué partes de la imagen son útiles para predecir la red neuronal. Grad-CAM se puede aplicar retroactivamente a modelos preexistentes sin necesidad de modificar la arquitectura de la red, a diferencia de otras técnicas que requieren estos cambios.[25]

Para crear explicaciones visuales Hilbert et al.[26] utilizan métodos como mecanismos de atención o enfoques basados en gradientes. Estos métodos destacan las áreas de la imagen que son más importantes para el proceso de toma de decisiones del modelo. Los médicos e investigadores pueden identificar visualmente las áreas de interés y obtener una mejor comprensión de las decisiones del modelo al superponer estas explicaciones visuales a las imágenes originales.

2.2 Aumento de datos

Por otro lado, en el campo de la medicina existe un gran problema debido a la escasez de imágenes. Gracias a algunos “challenges” como los que propone ISIC se ha conseguido obtener bases de datos públicas con cierto número de imágenes. Aún asi existen mecanismos para entrenar una CNN con pocas imágenes.

Z. Hussain et al.[27] abordan el problema de la escasez de datos en el campo de la clasificación de imágenes médicas y sugieren una variedad de métodos de ampliación de datos diferentes. Estas técnicas parten de la manipulación de imágenes originales mediante cambios de escala, recorte, rotación y volteo.

Ferhat Bozkurt [28] para aumentar la diversidad y el tamaño del conjunto de datos de entrenamiento en Clasificación de lesiones cutáneas en imágenes dermatoscópicas, utiliza métodos de ampliación de datos. Esto permite que el modelo aprenda características más sólidas y representativas. Se pueden obtener imágenes aumentadas mediante la rotación, la escala, el volteo y la adición de ruido. Esto ayuda a abordar el problema de tener pocos datos anotados en el campo médico.

3 MACHINE LEARNING

El Machine Learning (aprendizaje profundo en español) es un subcampo de las Tecnologías de la Información y Comunicaciones,que consiste en la construcción de algoritmos para la resolución de problemas mediante la recopilación de conjuntos de datos .[29]

Tenemos definiciones generales: [30]

“El campo de estudio que proporciona a las computadoras la capacidad de aprender sin ser programadas explícitamente.” Arthur Samuel, 1959

“Se dice que un programa de computadora aprende a partir de la experiencia E con respecto a alguna tarea T y alguna medida de rendimiento P, si su rendimiento en T, medido por P, mejora con la experiencia E.” Tom Mitchell, 1997.

Los sistemas de machine learning se pueden clasificar como aprendizaje supervisado, semisupervisado, no supervisado o aprendizaje por refuerzo.

3.1 Aprendizaje Supervisado

Los datos de entrenamiento proporcionados al algoritmo para el aprendizaje supervisado incluyen las soluciones deseadas, conocidas como etiquetas.[30].

Un ejemplo muy usado para explicar el aprendizaje supervisado es la clasificación entre correos de spam y no spam en el que tendrías las etiquetas [spam, no-spam] y tomando el vector de características a la entrada le asocia una de las etiquetas.[29,30]

Algunos de los algoritmos de aprendizaje supervisado más importante son:

La **regresión lineal**, es un algoritmo de aprendizaje de regresión que consiste en la búsqueda de la función que relaciona una variable dependiente -que puede ser continua- y una serie de variables independientes -las características, muy utilizado que aprende un modelo que es una combinación lineal de las características de entrada.[29]

La **regresión logística** es similar a la lineal y calcula la relación estimada entre dos variables.[29]

Un **árbol de características** es un grafo acíclico que se usa para tomar decisiones examinando una característica del vector en cada rama del grafo.[29]

Support Vector Machines, da a las etiquetas un valor y considera cada vector un punto en el espacio, después dibuja un plano que separa los puntos con etiquetas con valores diferentes.[29]

3.2 Aprendizaje no supervisado

En el aprendizaje no supervisado los datos proporcionados no están etiquetados si no que el algoritmo aprende por sí solo.

Algunos de los algoritmos de aprendizaje supervisado más importante son:

K-means es un algoritmo de tipo clustering, que consiste en calcular la distancia de cada vector de características al centroide de cada clase y le asignamos el centroide más cercano[29]

DBSCAN es un algoritmo de tipo clustering que se basa en densidad, elige un punto arbitrario

seleccionándolo para el clúster 1 y busca todos sus vecinos en una distancia épsilon. Si supera un número N se añaden al cluster y se hace lo mismo para cada punto añadido al cluster 1.[29]

3.3 Aprendizaje Semisupervisado

El aprendizaje semisupervisado tiene un conjunto de datos etiquetados y otros no etiquetados. El objetivo es que utilizando ambos datos sea capaz de aprender sin necesidad de más datos etiquetados[29]

3.4 Aprendizaje por refuerzo

El aprendizaje por refuerzo no recibe datos etiquetados, debe de aprender por sí mismo recibiendo refuerzos negativos o positivos según las clasificaciones que va haciendo [29].

3.5 Redes Neuronales

Las redes neuronales son un caso de Maching lerning supervisado que tienen entre la entrada y la salida capas ocultas. El objetivo principal de una red neuronal es realizar una clasificación mediante cálculos con el conjunto de datos recibidos a la entrada.[31]

Las redes neuronales pueden ser profundas (DNN) o poco profundas. Aunque no exista un límite claro se suele considerar una DNN cuando tiene más de 5 capas. Para el proceso de entrenamiento se necesita ajustar los pesos y sesgos para minimizar las diferencias entre las predicciones y la realidad. Las DNN son capaces de reconocer patrones complejos, sin embargo, necesitan una gran potencia computacional, pero las GPUs han acelerado el proceso.[31]

3.5.1 Tipos de redes

Existen distintos tipos de redes neuronales

- Redes Generativas antagónicas (GAN) : El generador y el discriminador son los dos componentes principales de esta. El generador crea instancias de datos como imágenes o música a partir de números aleatorios. El discriminador evalúa estas instancias y les da una probabilidad de 0 a 1. El objetivo de la GAN es capacitar al generador para que produzca instancias que, según el discriminador, sean distintas de los datos reales. [31]
- Redes neuronales recurrentes (RNN): estas redes permiten que los datos fluyan en cualquier dirección. La idea fundamental de las RNN es trabajar con información secuencial. Debido a que realizan la misma tarea para cada elemento de una secuencia y se basan en los cálculos anteriores, las RNN se denominan "recurrentes". [31]
- Redes neuronales convolucionales (CNN): Estas redes al aumentar el número de capas de una red neuronal, permiten modelar operaciones más complejas. La idea principal de las CNNs es aplicar una convolución a la imagen en una vecindad de nodos donde el filtro utilizado es un factor multiplicativo del valor del nodo, como 0.5. [31]

3.6 VGG-16

Una de las primeras en aparecer fue VGG-16, una red neuronal convolucional diseñada para tareas de reconocimiento de imágenes a gran escala. Se presentó por Simonyan y Zisserman en 2014. [12]

Este modelo tiene 16 capas consiguiendo una precisión del 92.7% usando la base de datos de ImageNet.

Andrew Zisserman y Karen Simonyan presentaron este modelo en el ImageNet Challenge de 2014 produciendo grandes cambios en lo que había hasta el momento, el campo receptivo era de 3x3 con un paso de un píxel frente al 11x11con paso de 4 pixeles de AlexNet. El uso de capas de entrada más pequeñas produce que más capas no lineales de activación acompañen a las capas de convolución. Permitiendo que se optimicen las funciones de decisión y se tenga una conversión más rápida de la red.[33]

3.6.1 Arquitectura

La VGG-16 consta de 13 capas convolucionales, 5 Max Pooling y 3 capas densas en total esto sumaría 21 capas, pero solo 16 capas pueden aprender, es decir, tienen pesos.

La entrada es una imagen de 224x224 con sus tres planos de color. Todas las capas convolucionales constan de filtros 3x3 y función de activación relu. y el Max Pooling de ventanas 2x2 y un paso de 2 píxeles.

La estructura es de la siguiente manera, la Conv-1 que consta 2 convolucionales de 64 filtros + Max Pooling, la Conv-2 que consta de 2 convolucionales de 128 filtros + Max Pooling, la Conv-3 que consta 3 convolucionales de 256 filtros + Max Pooling, la Conv-4 y la Conv-5 que constan 3 convolucionales de 512 filtros + Max Pooling. Por último, pasa por las 3 capas densas (totalmente conectadas) las 2 primeras tienen 4096 canales y la última clasifica en 1000 categorías por lo que tiene 1000 canales 1 por cada categoría.[33]

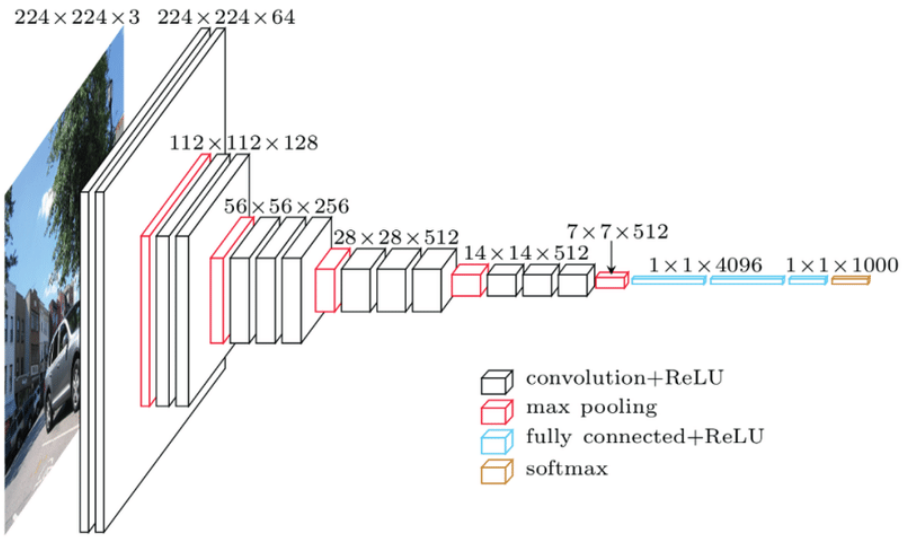


Figura 3.1. Arquitectura de la red VGGNet-16 [26]

4 METODOLOGÍA

Para llevar a cabo este proyecto se ha utilizado una red neuronal convolucional, en concreto la VGG-16. Esta red fue propuesta por K. Simonyan y A. Zisserman en 2014 para clasificación de imágenes a gran escala. Este tipo de redes es posible gracias a repositorios públicos como ImageNet.

4.1 Tipo de red utilizada

La red propuesta consta de 22 capas, entre las que hay capas de entrada, convolucionales, MaxPooling, GlobalAveragePooling, y Densas.

La capa de entrada consiste en un preprocesamiento de la imagen, entra la imagen en este caso de 256x256x3 en formato RGB y se le resta a cada píxel el valor RGB promedio.

Las capas convolucionales usan un filtro de convolución de tamaño 3x3, en estas capas filtran la imagen manteniendo la información esencial. El uso de filtros 3x3 frente a otros mayores se debe a una reducción del trabajo computacional además de aprender características más complejas y especificas ya que con los filtros mayores las características son más generales al abarcar mayor número de píxeles.

Las capas MaxPooling siguen a un grupo de capas convolucionales con el objetivo de reducir el número de parámetros de los mapas de características creados además de reducir el sobreajuste . Para esta capa se utilizan filtros 2x2 con un paso de 2.

Tras el uso de bloques de las dos capas anteriores se usa una GlobalAveragePooling que realiza una media de todos los filtros de características obteniendo un vector.

Por último, se aplican 2 capas densas, estas son capas totalmente conectadas, la primera de ellas utiliza una función de activación ReLU, esta iguala a 0 todo valor de entrada negativo dejando los positivos iguales. La siguiente capa densa es de tipo softmax que da a su salida las probabilidades de pertenecer a cada una de las clases.[12,30-31]

La red propuesta consta de una capa de entrada seguida de dos bloques compuestos por dos capas convolucionales y un MaxPooling. Posteriormente le siguen 3 bloques compuestos por 3 capas convolucionales y un MaxPooling y, por último, una capa GlobalAveragePooling segida de una densa con función de activación relu en 20 clases y una densa con función de activación softmax en 2 clases (maligno y benigno).

4.2 Transfer Learning

Debido a la escases de datos es muy complicado poder entrenar una red desde cero se parte de una red entrenada previamente. Esta técnica se denomina transfer learning y nos permite importar los parámetros como punto inicial de nuestra red y entrenarla posteriormente con los datos específicos [19].

En la red propuesta se ha importado los pesos de ImageNet, en este caso se utilizan los pesos de todas las capas menos las últimas 5 que se entrenan con el conjunto de datos obtenidos de la BB.DD ISIC 2020[9].

4.3 Aumento de Datos

Ante la escases de datos otra de las técnicas a utilizar es el data Augmentation que consiste en realizar transformaciones sencillas al conjunto de datos disponible para el entrenamiento.

En este caso hemos realizado distintos tipos de transformaciones únicamente al conjunto de datos de entrenamiento de casos malignos, ya que era bastante inferior al número de casos Benignos (584 frente a 32542) . Las transformaciones llevadas a cabo han sido distintos ángulos de rotación (20º, 35º y 120º) y un rango de zoom de 0´5, obteniendo un total de 1227.

Gracias a la ampliación del conjunto de datos se puede entrenar la red obteniendo mayor precisión ya que no se observará la misma imagen en las distintas epochs [34] .

4.4 Diagnostico explicado

Ante la posibilidad de fallo de la red se ha llevado a cabo un diagnostico explicado. Esto consiste en implementar explicaciones visuales de la decisión. Para ello se a utilizado Grand-CAM, un método que utiliza los gradientes con respecto a las activaciones de la capa de convolución y posteriormente se promedian dependiendo de la importancia. Por último, se superpone a la imagen original.

De esta manera obtenemos las zonas más importante para la toma de decisión de la red pudiendo comprobar si la red toma las decisiones de manera aleatoria o ha aprendido las características importantes en la toma de decisiones [25].

5 IMPLEMENTACIÓN

Para la implementación de este proyecto se ha optado por el lenguaje de programación Python debido a la gran facilidad en su programación y ser uno de los lenguajes más utilizados actualmente. Para ello se ha basado en la implementación de las librerías Keras y Tensorflow.

Keras es una interfaz de programación de aplicaciones (API) de alto nivel excelentemente diseñada y fácil de usar para entrenar, evaluar y ejecutar redes neuronales. TensorFlow es una librería que permite crear y entrenar redes neuronales para detectar patrones de pensamiento y razonamiento humanos.

Para el desarrollo del algoritmo se ha utilizado Google colab que es una herramienta de Google para programar on-line que permite el uso gratuito de las GPU´s de Google.

5.1 Implementación del código

Para la implementación del código se han importado las siguientes librerías

```
import numpy as np
import os
import cv2
import PIL.Image as Image
import keras.backend as K
from keras.models import *
from keras.callbacks import *
from keras.applications import vgg16
from keras.models import Sequential,Model
from keras.callbacks import ReduceLROnPlateau,ModelCheckpoint,EarlyStopping
import keras
from keras.preprocessing.image import ImageDataGenerator
from  keras.layers  import  Dense  ,concatenate,  Conv2D,Flatten  ,  InputLayer,
ZeroPadding2D, Activation, MaxPooling2D,Dropout, GlobalAveragePooling2D, Input
from tensorflow.compat.v1 import disable_eager_execution
from keras.saving.legacy.save import load_model
```

Figura 5.1. Librerías Utilizadas

La librería numpy permite cálculos numéricos y análisis de datos, la librería os permite acceder al sistema operativo, la librería cv2 permite trabajar con imágenes, la librería permite la edición de imágenes. Además de estas librerías se importan distintas funcionalidades de Keras.

Para trabajar con las imágenes de la BB.DD de ISIC 2020 se han subido a drive y se accede a ellas mediante la librería drive, que te da acceso a la cuenta de Google drive donde se encuentran las imágenes y os.chdir selecciona el directorio en el que se va a trabajar.

```
from google.colab import drive
drive.mount('/content/drive')
os.chdir('/content/drive/My Drive/IMAGENES')
```

Figura 5.2 Conexión con Google drive

Posteriormente se leen todas las imágenes de sus respectivos directorios utilizando os.listdir se crea

una lista con los archivos del directorio pasado como argumento y se convierte en un array mediante np.asarray.

```
path_benign='Benign'
lista_ficheros1=os.listdir(path_benign)
path_malignant='Malignant'
lista_ficheros2=os.listdir(path_malignant)
np.random.shuffle(lista_ficheros1)
np.random.shuffle(lista_ficheros2)
lista_1_2 = np.asarray(lista_ficheros1)
lista_2_2 = np.asarray(lista_ficheros2)
```

Figura 5.3. Creación de lista de imágenes

Después se divide el número de imágenes en distintos grupos, el 70% se utiliza para entrenamiento, el 20% para validación y un 10% para test. Para ello se guarda la cantidad de imágenes de cada tipo calculando el porcentaje de la longitud total de la lista (devuelta por la función len).

```
num_train1=round(0.7*len(lista_ficheros1))
num_val1=round(0.2*len(lista_ficheros1))
num_test1=round(0.1*len(lista_ficheros1))

num_train2=round(0.7*len(lista_ficheros2))
num_val2=round(0.2*len(lista_ficheros2))
num_test2=round(0.1*len(lista_ficheros2))
```

Figura 5.4 División en grupos de imágenes

Se inicializan las variables donde se van a almacenar las imágenes y la verdad de referencia.

```
test_x  = []
train_x = []
val_x   = []
test_y  = np.zeros((num_test,2))
train_y = np.zeros((num_train,2))
val_y   = np.zeros((num_val,2))
```

Figura 5.5. Inicialización de variables

Se realiza un bucle for para leer todas las imágenes, para ello se usa la librería PIL. Cada imagen se almacena en la variable correspondiente el 70% de las primeras imágenes en train, el 20% siguiente en val y el último 10% en test. Además, en train_y, val_y y test_y se almacena [1,0] que indica que son benignas.

```
for i,imagen in enumerate(lista_ficheros1):
  I=Image.open(os.path.join(path_benign,imagen))
  #I=np.asarray(I)
  I=I.resize((256,256))
  if i <num_test1:
    test_x.append(np.asarray(I))
    test_y[i,:]=[1,0]
  if num_test1<=i<num_test1+num_val1:
    val_x.append(np.asarray(I))
    val_y[i-num_test1,:]=[1,0]
  if num_test1+num_val1<=i:
    train_x.append(np.asarray(I))
    train_y[i-(num_test1+num_val1),:]=[1,0]
```

Figura 5.6. Apertura de las imágenes

Se repite el mismo bucle para lista_ficheros2, cambiando la verdad de referencia a [0,1] y los índices de las matrices que empezarían en la posición siguiente al último dato almacenado de lista_ficheros1.

5.2 Red Propuesta

La red se ha utilizado la VGG-16

Mediante keras.optimizers.Adam se utiliza para ajustar los pesos y sesgos de la red neuronal durante el entrenamiento, con la intención de mejorar la precisión del modelo. keras.optimizers.SGD se utiliza para ajustar los pesos y sesgos de la red neuronal durante el entrenamiento utilizando el algoritmo de Descenso de Gradiente Estocástico.

```
adamgrad=keras.optimizers.Adam(learning_rate=0.001, beta_1=0.09,
beta_2=0.8, epsilon=1e-8, decay=0.0, amsgrad=True)
opt = keras.optimizers.SGD(lr=0.001)
```

Figura 5.7. Optimización de los pesos

Posteriormente como se va a usar la red VGG-16 se carga su arquitectura y se cargan los pesos de ImageNet.

```
vgg=vgg16.VGG16(include_top=False,weights='imagenet',input_shape=input_shape)
```

Figura 5.8. Importación de la red

Se Asignan la salida de la red vgg a la variable x para poder trabajar con ella.

```
x = vgg.output
```

Figura 5.9. Salida de la red

Se congelan los pesos de todas las capas menos las últimas 5 durante el entrenamiento, de esta manera se evita que se actualicen los pesos de las primeras capas durante el entrenamiento conservando los pre-entrenados.

```
for nn,layer in enumerate(vgg.layers):
    layer.trainable=False
    if(nn>=(len(vgg.layers)-5)):
        layer.trainable=True
```

Figura 5.10. Se congelan los pesos

Se crea una capa AveragePooling 2D a la salida de la red (x), esta capa calcula la media de los elementos de estudio de una imagen. Después se crea una capa densa con función de activación Relu y 20 salidas y por último se crea otra capa densa úon 2 salidas y activación softmax .Finalmente se añaden todas estas capas al modelo.

```
x = GlobalAveragePooling2D(name='avg_pool1')(x)
z = Dense(20,activation = 'relu') (x)
w = Dense(2,activation = 'softmax') (z)
model = Model(inputs=vgg.input, outputs=w)
```

Figura 5.11. Añadimos 3 capas a la red.

Esta instrucción indica que se guardan los pesos en un archivo hdf5 durante el entrenamiento cuando se obtenga la mejor métrica.

```
mcp_save = ModelCheckpoint('.mdl_wts.hdf5', save_best_only=True,
monitor='val_loss', mode='min')
```

Figura 5.12. Se guardan los pesos.

Por último, se compila el modelo usando la función de pérdidas 'binary_crossentropy', un optimizador SGD y métricas de ‘accuracy’ y AUC.

```
model.compile(loss='binary_crossentropy',optimizer='sgd',
metrics=['accuracy',keras.metrics.AUC()])
```

Figura 5.13. se compila el modelo.

Una vez terminada la red mostramos todas las capas con el comando sumary.

```
model.summary()
```

```
 Layer (type)                                Output Shape
Param #
=================================================================
 input_1_Cc (InputLayer)     [(None, 256, 256, 3)]              0

 block1_conv1_Cc (Conv2D)    (None, 256, 256, 64)      1792

 block1_conv2_Cc (Conv2D)    (None, 256, 256, 64)      36928

 block1_pool_Cc (MaxPooling2  (None, 128, 128, 64)     0
 D)

 block2_conv1_Cc (Conv2D)    (None, 128, 128, 128)     73856

 block2_conv2_Cc (Conv2D)    (None, 128, 128, 128)     147584

 block2_pool_Cc (MaxPooling2  (None, 64, 64, 128)          0
 D)

 block3_conv1_Cc (Conv2D)    (None, 64, 64, 256)       295168

 block3_conv2_Cc (Conv2D)    (None, 64, 64, 256)       590080

 block3_conv3_Cc (Conv2D)    (None, 64, 64, 256)       590080

 block3_pool_Cc (MaxPooling2  (None, 32, 32, 256)      0
 D)

 block4_conv1_Cc (Conv2D)    (None, 32, 32, 512)       1180160

 block4_conv2_Cc (Conv2D)    (None, 32, 32, 512)       2359808

 block4_conv3_Cc (Conv2D)    (None, 32, 32, 512)       2359808

 block4_pool_Cc (MaxPooling2  (None, 16, 16, 512)      0
 D)

 block5_conv1_Cc (Conv2D)    (None, 16, 16, 512)       2359808

 block5_conv2_Cc (Conv2D)    (None, 16, 16, 512)       2359808

 block5_conv3_Cc (Conv2D)    (None, 16, 16, 512)       2359808

 block5_pool_Cc (MaxPooling2  (None, 8, 8, 512)             0
 D)

 avg_pool1 (GlobalAveragePoo  (None, 512)                     0
 ling2D)

 dense (Dense)               (None, 20)
10260

 dense_1 (Dense)             (None, 2)
42

=================================================================
Total params: 14,724,990
Trainable params: 7,089,726
Non-trainable params: 7,635,264
_________________________________________________________________
```

Figura 5.14. Capas de la red.

5.3 Entrenamiento

Se realiza el entrenamiento del modelo mediante la función model.fit donde utiliza los datos de train_x y train_y como entrada y los datos para validación de val_x y val_y. Para el entrenamiento se han determinado 23 epochs lo que significa que los datos pasan por la red 23 veces para entrenar la red.

```
for iteri in range(1):
  modelo_history= model.fit(train_x, train_y,validation_data=(val_x,
val y),epochs=23, batch size=32,)
```

Figura 5.15. Entrenamiento del modelo

5.4 Validación del algoritmo

Para calcular el rendimiento del algoritmo hay que introducir distintos parámetros.

- TP (True Positive): se considera como verdadero positivo cuando el algoritmo define un positivo (maligno) y la ground true también lo define como positivo (maligno).
- TN (True Negative): se considera verdadero negativo cuando el algoritmo y la ground true consideran negativo (benigno).
- FP (False Positive): se considera un falso positivo cuando el algoritmo define un positivo (maligno) y la ground true como negativo (benigno).
- FN (False Negative): se considera un falso negativo cuando el algoritmo define un negativo (benigno) y la ground true como positivo (maligno).

Estos parámetros se usan para definir la matriz de confusión. La matriz de confusión se dedica a comparar los valores predichos por el algoritmo con los reales.

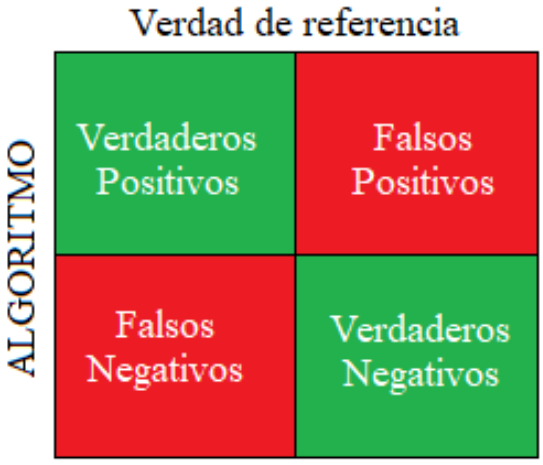


Figura 5.16. Matriz de confusión

Para obtener estos parámetros hemos creado una función que hemos llamado matriz de confusión a la que pasamos como parámetros dos arrays con valores 0 o 1 dependiendo si es maligno o benigno respectivamente y se hace una and entre los datos de entrada para obtener los parámetros.

```
def confusion_matriz(test_pred,test_set_y):
  tam = len(test_pred)
  tabla_conteo = np.zeros((1,4)).astype(int)
  for i in range(tam):
    if((test_pred[i] and test_set_y[i])):
      #verdadero positivo
      tabla_conteo[0,0]=tabla_conteo[0,0]+1
    if((not(test_pred[i]) and not(test_set_y[i]))):
      #verdadero negativo
      tabla_conteo[0,1]=tabla_conteo[0,1]+1
    if((test_pred[i] and not(test_set_y[i]))):
      #falso positivo
      tabla_conteo[0,2]=tabla_conteo[0,2]+1
    if((not(test_pred[i]) and test_set_y[i])):
      #falso negativo
      tabla_conteo[0,3]=tabla_conteo[0,3]+1
  #print(tabla_conteo)
  return tabla_conteo
```

Figura 5.17. Cálculo de la matriz de confusión

A partir de estos parámetros se pueden calcular distintas métricas que informan del funcionamiento del algoritmo:

- Sensibility: indica el porcentaje de enfermos que han recibido un diagnóstico positivo (maligno) correcto.

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

- Specificity indica el porcentaje de enfermos que han recibido un diagnóstico negativo (benigno) correcto.

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

- Valor Predictivo Positivo: indica el porcentaje de que el paciente acierte con un diagnóstico positivo (maligno).

$$VPP = \frac{TP}{TP + FP}$$

- Accuracy: indica el porcentaje de que el paciente acierte con un diagnóstico.

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

Para calcular estas métricas se ha creado una función llamada calculo_valores que recibe como parámetro la tabla con los TP,TN,FP y FN y calcula las distintas métricas con las fórmulas ya mostradas.

```
def calculo_valores(tabla_conteo):
    tabla_conteo = tabla_conteo
    a,b = tabla_conteo.shape

    precision = np.zeros(a) #tp+tn/tp+tn+fp+fn
    vpp       = np.zeros(a) #tp/fp+tp
    especifici= np.zeros(a) #tn/tn+fp
    sensibilid= np.zeros(a) #tp/tp+fn
    total = sum(tabla_conteo[0,:])
    for i in range(a):
        precision[i] = (tabla_conteo[i,0]+tabla_conteo[i,1])/total
        vpp[i]       =
tabla_conteo[i,0]/(tabla_conteo[i,0]+tabla_conteo[i,2])
        especifici[i]=
tabla_conteo[i,1]/(tabla_conteo[i,1]+tabla_conteo[i,2])
        sensibilid[i]=
tabla_conteo[i,0]/(tabla_conteo[i,0]+tabla_conteo[i,3])
    # print("Precision",precision, "\n",
    #       "VPP",vpp, "\n",
    #       "Especificidad",especifici, "\n",
    #       "Sensibilidad",sensibilid, "\n",)
    return precision,vpp,especifici,sensibilid,tabla_conteo
```

Figura 5.18. Cálculos de las medidas de rendimiento

5.5 Mapa de activación

Por último, se realiza una función que usando los pesos de salida de la última capa realiza un mapa de activación mostrando las zonas de la imagen más determinantes en la resolución del diagnóstico. Esta función toma como parámetros de entrada el modelo de la red, la ruta de la imagen de entrada y la ruta de salida de la imagen.

Después carga las imágenes y obtiene los pesos de la última capa del modelo y con eso Calcula el mapa de activación de clase utilizando los pesos y las salidas de la capa de convolución final. Además, imprime las predicciones de la imagen e indica si el resultado es maligno o benigno. Finalmente guarda la imagen original con el mapa de activación superpuesto en la ruta indicada como parámetro.

```
def visualize_class_activation_map(model_path, img_path,
output_path):
  model=load_model(model_path)
  original_img=Image.open(os.path.join(img_path))
  original_img=original_img.resize((256,256))
  w,h,_=np.shape(original_img)

  #reshape to the network imput shape (3,w,h)


  img=np.array([np.transpose(np.float32(original_img),(0,1,2))])

  #get the 512 imput wight to the softmax
  class_weight = model.layers[-1].get_weights()[0]
  final_conv_layer = get_output_layer(model, 'block5_pool_Cc')
  get_output = K.function([model.layers[0].input],
[final_conv_layer.output, model.layers[-1].output])
  [conv_outputs, predictions] = get_output([img])
  conv_outputs = conv_outputs[0, :, :, :]
  conv_outputs.shape
  conv_outputs=np.array([np.transpose(np.float32(conv_outputs),(
2,1,0))])
  conv_outputs = conv_outputs[0, :, :, :]

  #Create the class activation map.
  cam = np.zeros(dtype = np.float32, shape =
conv_outputs.shape[1:4])
  for i, w in enumerate(class_weight[:, 1]):
        cam += w * conv_outputs[i, :, :]
  print ("predictions", predictions)
  PREDIC= np.argmax(predictions,1)
  if PREDIC == 0:
    print('BENIGNO')
  else :
    print('MALIGNO')
  cam /= np.max(cam)
  cam = cv2.resize(cam, (256, 256))
  heatmap = cv2.applyColorMap(np.uint8(255*cam),
cv2.COLORMAP_JET)
  heatmap[np.where(cam < 0.2)] = 0
  img = heatmap*0.5 + original_img
  image =  Image.fromarray(np.uint8(img))
  image.save(output_path)
```

Figura 5.19. Cálculo del mapa de activación

5.6 Data Augmentation

Para mejorar los resultados se realiza un aumento de la BB.DD, en este caso de los melanomas (ficheros de datos malignos). Para ello se ha creado un programa aparte que procesa sólo las imágenes malignas de la misma manera que en el programa principal, de manera que se separan en 3 conjuntos train ,test y val. Cuando se han separado los 3 grupos se usa la función ImageDataGenerator para crear 3 tipos de transformaciones al conjunto de datos.

```
datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.5)
datagen1 = ImageDataGenerator(
    rotation_range=35,
    zoom_range=0.5)
datagen2 = ImageDataGenerator(
    rotation_range=120,
    zoom_range=0.5)
```

Figura 5.20. Transformaciones de las imágenes

Una vez se han creado los 3 tipos de transformaciones entrenamos el modelo con las imágenes del grupo train.

```
datagen.fit(train_x)
datagen1.fit(train_x)
datagen2.fit(train_x)
```

Figura 5.21. Entrenamiento de las transformaciones

Posteriormente se realizan las transformaciones mediante la función Flow y se crea un bucle para guardar las imágenes transformadas en un directorio nuevo.

```
from PIL import Image
iterator = datagen.flow(train_x, train_y, batch_size=32)
for i, (batch_x, batch_y) in enumerate(iterator):
    # Guardar las imágenes en el directorio
    for j in range(batch_x.shape[0]):
        image =  Image.fromarray(np.uint8(batch_x[j]))
        label = batch_y[j]
        filename = f"/content/drive/My
Drive/IMAGENES/Malignant2/DATA_{i * iterator.batch_size +
j}.png"
        image.save(filename)
        #cv2.imwrite(filename, image)
        #save_image(image, filename)  # Guardar la imagen
utilizando la función save_image() adecuada
    # Salir del bucle después de guardar todas las imágenes
necesarias
    if i == len(train_x) // 32:
        break
```

Figura 5.22. Cálculo y guardado de las transformaciones.

Una vez se han creado las nuevas imágenes se ejecuta el método con la BB.DD ampliada.

6 RESULTADOS

En este apartado se van a mostrar los resultados obtenidos con el método explicado anteriormente con la BB.DD original de ISIC 2020[9] .

6.1 Puntos de Partida

ISIC contiene la mayor colección publica de imágenes, pertenece a la sociedad internacional de imágenes digitales de la piel (ISDIS). Esta sociedad crea esta base de datos publica con la intención de mejorar el diagnósticos de melanomas. Desde 2016 ISIC propone desafíos anuales, en los cuales la complejidad y participación ha ido creciendo. Para este trabajo se ha usado la BB.DD de reto ISIC 2020 [9] que consta de 33126 imágenes. De esta BBDD se ha tomado los 584 casos malignos y 1000 casos aleatorios de los benignos.

Durante la implementación del código usado para la obtención de los resultados de ha aplicado un transfer learning de los pesos de imagnet. Además, se divide el conjunto de imágenes en tres grupos train, test, y val cada uno tiene el 70%, 20% y 10% de las imágenes respectivamente.

Posteriormente se mostrarán los resultados tras haber aplicado el Data Augmentation sobre las 409 imágenes del grupo train obteniendo un total de 1811 imágenes malignas y se añaden 811 imágenes benignas aleatorias de la BB.DD original a las 1000 ya utilizadas antes, para tener una cantidad de datos balanceados (misma cantidad de ambos tipos).

Por último, se usará la BB.DD de ISIC 2017 [35] para mostrar los resultados del mapa de activación ya que dicha BB.DD cuenta con la Ground True (verdad de referencia) de la segmentación además de el diagnostico entre maligno y benigno. Gracias a esta segmentación podemos concretar si la activación se está realizando correctamente.

6.2 Resultados Sin Data Augmentation

En la gráfica de training loss and accuricy (Figura 5.1) podemos observar como en las 30 epochs que se han elegido se llega a un accuracy de 1 y unas pérdidas prácticamente de 0 por lo tanto el modelo tiene un buen entrenamiento ya que está obteniendo valores óptimos.

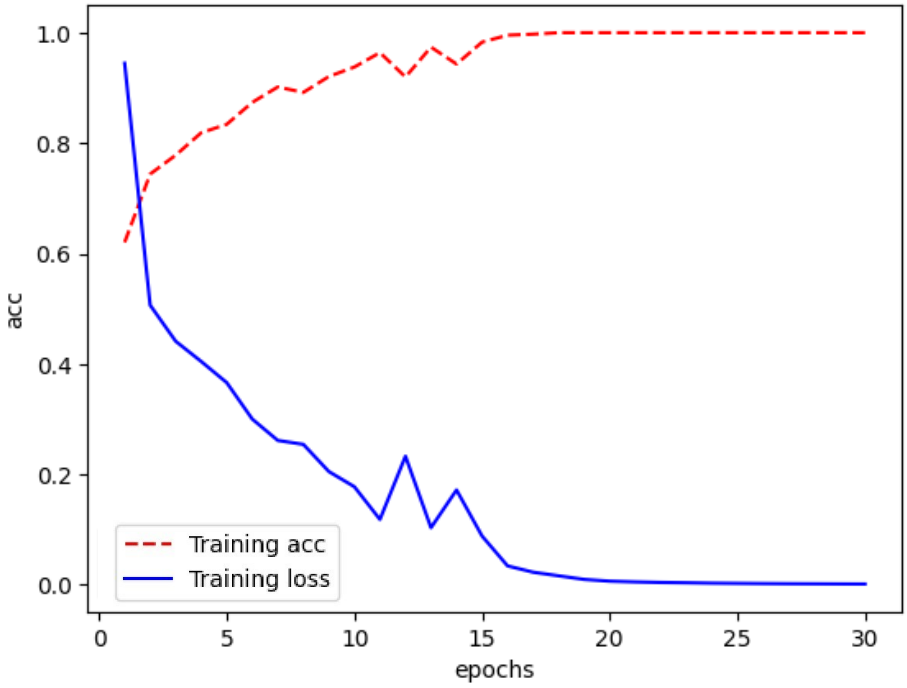


Figura 6.1 . Perdidas y accuracy en 30 epochs de entrenamiento

En la curva roc podemos observar que la auc (que mide el área bajo la curva) se acerca a 1 (que significa que el diagnóstico es perfecto) por lo tanto podemos observar que los resultados son razonablemente buenos.

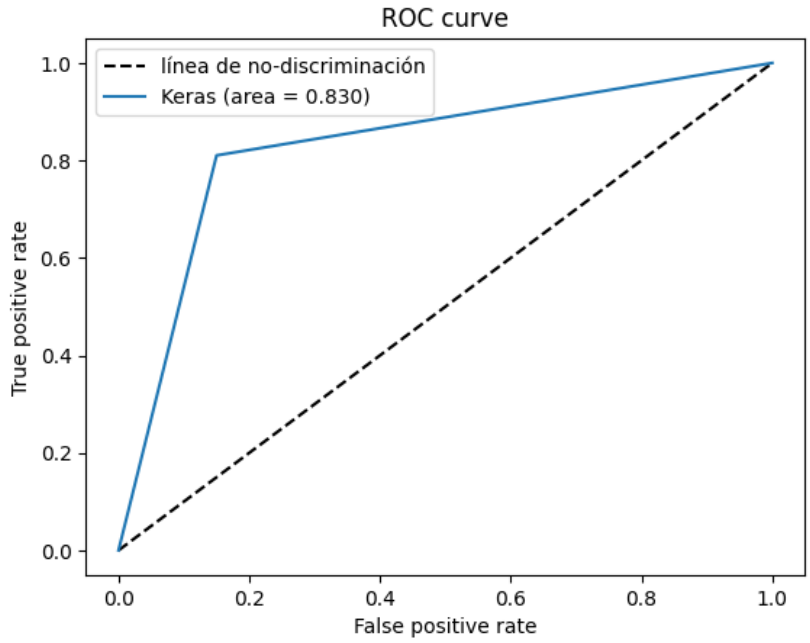


Figura 6.2 . Curva ROC

Con este entrenamiento y el conjunto de imágenes descrito con anterioridad se han obtenido los siguientes valores.

Tabla 6.1. Parámetros de rendimiento

Parámetros	
True Positive	47
True Negative	85
False Positive	15
False Negative	11

Tabla 6.2. Métricas de rendimiento

Parámetros	
Sensibility	81,03%
Especificity	85,00%
Accuracy	83,54%
PPV	75,81%

6.3 Resultados Con Data Augmentation

En la gráfica de training loss and accuricy después del data Augmentation (Figura 5.2) podemos observar como también en 30 epochs que se llega a un accuracy de 1 y unas pérdidas prácticamente de 0 por lo que también tiene un buen entrenamiento al obtener valores óptimos.

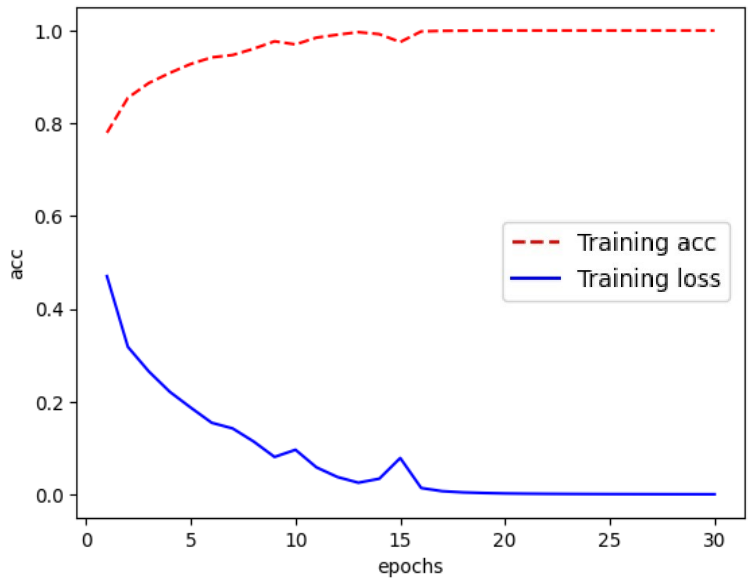


Figura 6.3. Pérdidas y accuracy en 30 epochs de entrenamiento después del data Augmentation

Tras el entrenamiento se han obtenido los siguientes valores.

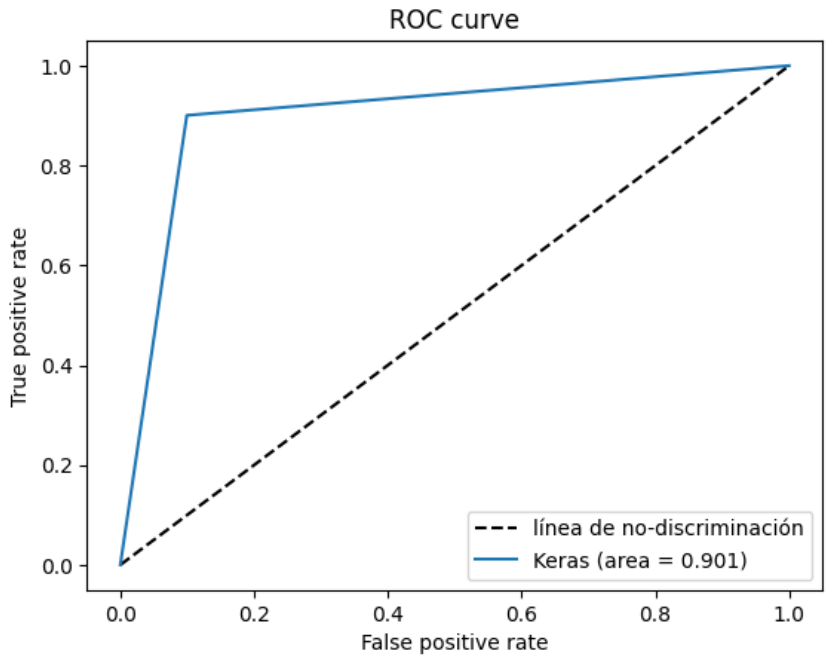


Figura 6.4 . Curva ROC con Data Augmentation

Como se puede observar los resultados mejoran notablemente, ya que la AUC pasa a ser 0.901 frente a 0.83 sin el data Augmentation y todos los parámetros de medida superan el 90%.

Tabla 6.3. Parámetros de rendimiento con data Augmentation

Parámetros	
True Positive	163
True Negative	163
False Positive	18
False Negative	18

Tabla 6.4. Métricas de rendimiento con data Augmentation

Parámetros	
Sensibility	90,06%
Especificity	90,06%
Accuracy	90,06%

PPV	90,06%

Como podemos observar se obtienen mejores valores tras el data Augmentation ya que la red ha obtenido más imágenes y por tanto puede caracterizar de mejor manera las lesiones.

6.4 Mapa de Activación sin Data Augmentation

Para obtener estos resultados se ha creado un código que realiza un solapamiento de la ground true con la imagen resultado de la función visualize_class_activation_map (nombrada y explicada anteriormente). La superposición se realiza de manera que nos muestra solo la zona que pertenece a la lesión y el resto de la imagen en negro, de esta manera se puede comprobar si la zona activada cae dentro de la zona lesionada.

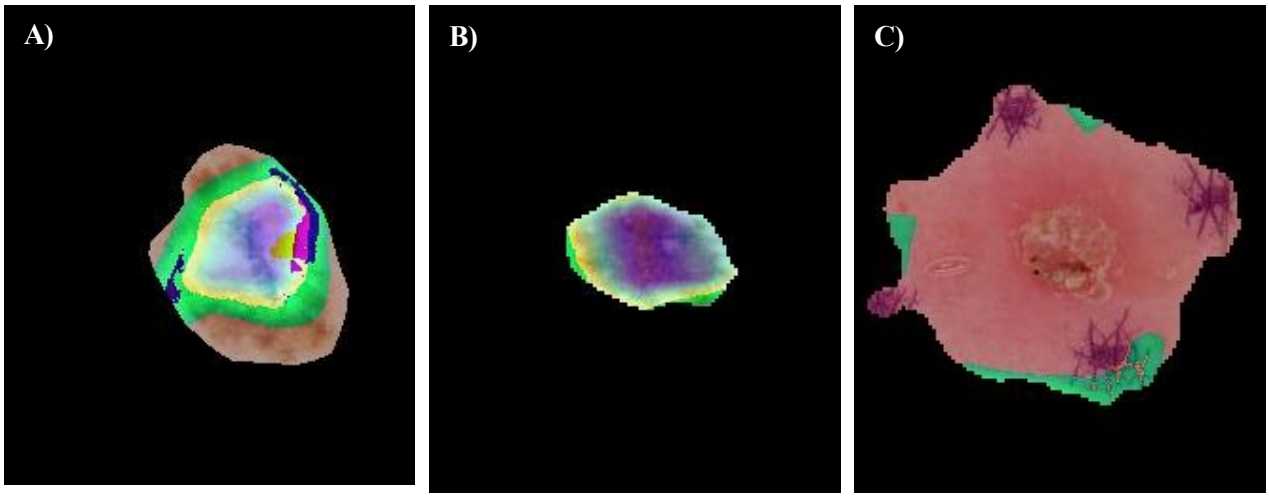
Para la BB.DD de ISIC 2017 [35] se ha calculado el diagnostico que ha supuesto un accuracy de un 76.67% y además se ha comprobado si se active dentro de la zona lesionada, siendo ligeramente superior las imágenes en las que la zona activada coincide con la zona lesionada.

Tabla 6.5. Parámetros de rendimiento del mapa de activación

Parámetros	
Coinciden zonas lesionada y activada	88
No coinciden zonas lesionada y activada	62

De los 62 en los que no coinciden ambas zonas 35 son imágenes mal diagnosticadas, mientras que 27 son imágenes bien diagnosticadas.

Algunos de los resultados de la superposición con la segmentación despues del data Augmentation son los siguientes.



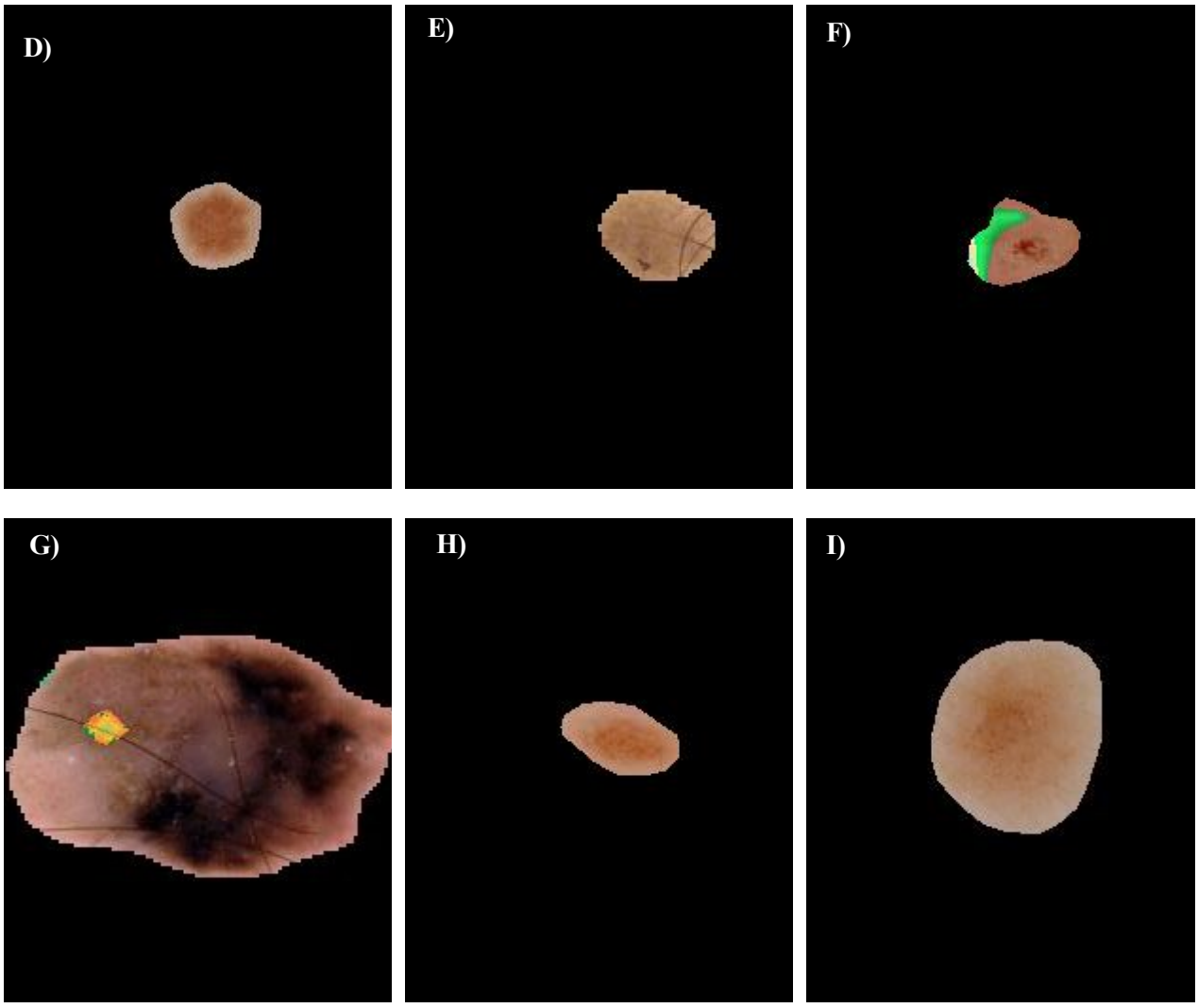


Figura 6.5. ejemplos solución mapas de activación. A) ISIC_0012400; B) ISIC_0012201; C) ISIC_0013127; D) ISIC_0003462 ; E) ISIC_0014616; F) ISIC_0012204; G) ISIC_0014946; H) ISIC_0007528; I) ISIC_0006914.

Como se observa en la figura 5.3 podemos encontrar activaciones de la zona completa o parcial y existen imágenes en las que el mapa de activación no es correcto.

6.5 Mapa de Activacion con Data Augmentation

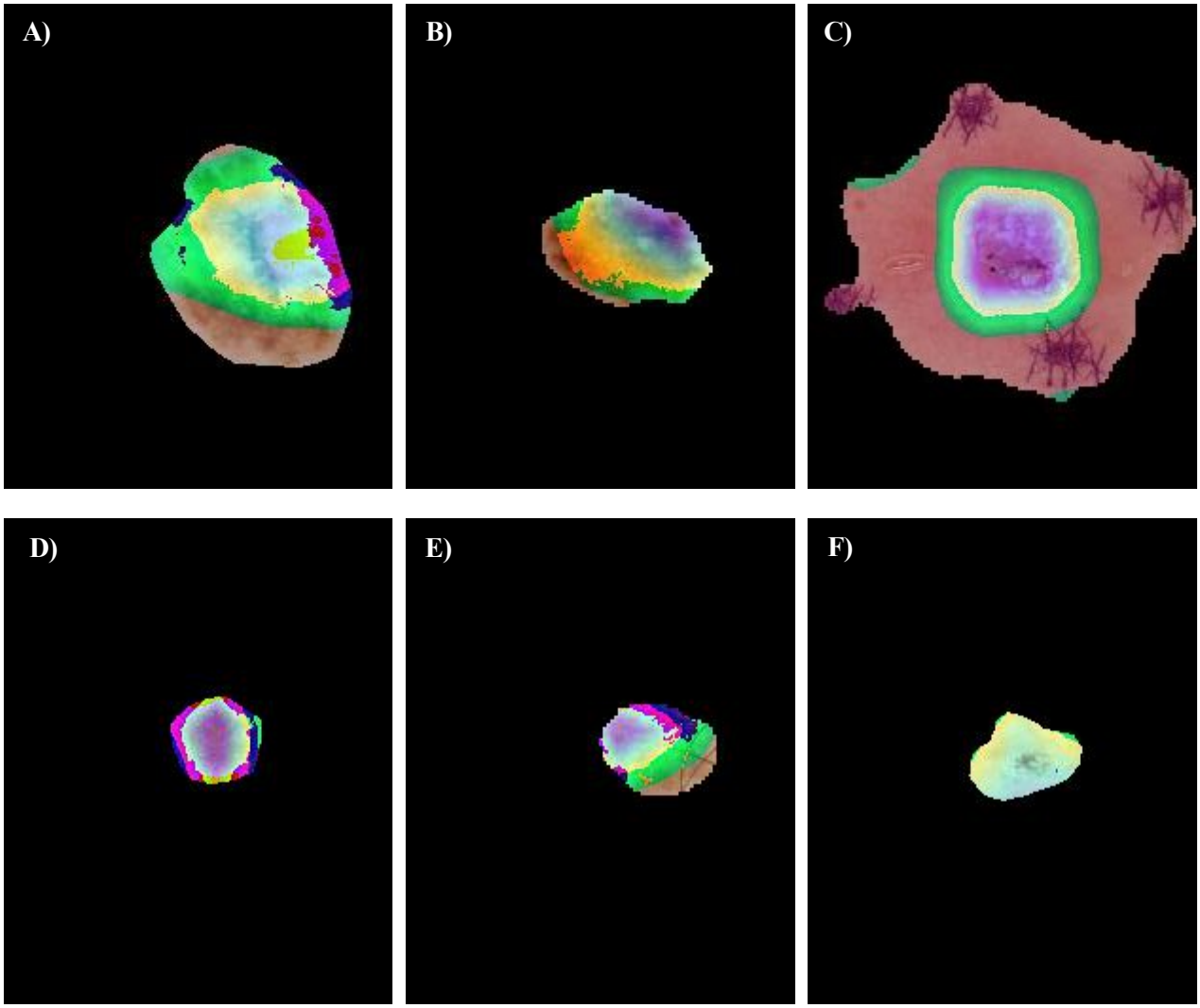
Los resultados obtenidos con la BB.DD de ISIC 2017 [35] usando el modelo entrenado tras el data Augmentation han supuesto un accuracy de un 93,33% que supone un aumento considerable en el acierto de los casos. Por otro lado la coincidencia entre las zonas lesionadas y activada aumenta notablemente.

De los 32 casos en los que la activación no se realiza correctamente 20 son imagenes mal diagnosticadas mientras que 12 son imágenes bien diagnosticadas.

Tabla 6.6 Parámetros de rendimiento del mapa de activación

Parámetros	
Coinciden zonas lesionada y activada	118
No coinciden zonas lesionada y activada	32

Algunos de los resultados de la superposición con la segmentación después del data Augmentation son los siguientes.



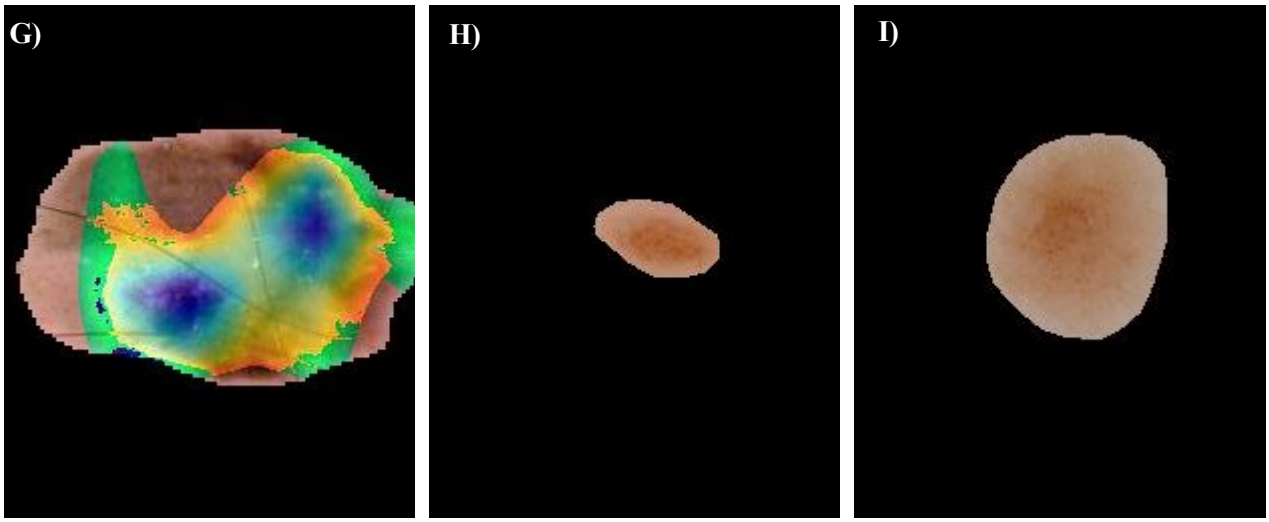


Figura 6.6 ejemplos solución mapas de activación después del data activation. A) ISIC_0012400; B) ISIC_0012201; C) ISIC_0013127; D) ISIC_0003462; E) ISIC_0014616; F) ISIC_0012204.; G) ISIC_0014946; H) ISIC_0007528; I) ISIC_0006914.

Como se observa en la figura 5.4 y en comparación con la figura 5.3 existen imágenes en las que el mapa de activación coincide con la zona segmentada tanto antes como después del data Augmentation , imágenes en la que se mejora la zona de activación coincidiendo mayormente con la zona segmentada y imágenes que no se activen ni antes ni después del data Augmentation en la zona segmentada lo que indica que la posibilidad de que el diagnóstico de dichas imágenes sea erróneo es mayor.

7 CONCLUSIONES

El machine learning es una técnica muy usada para diagnóstico por imagen sobre todo tras la aparición de BB.DD de gran tamaño y calidad en ciertos ámbitos, gracias a "challenges" como los de ISIC.

Debido al aumento de casos de lesiones de piel en los últimos años y la importancia de la detección temprana en el caso del melanoma este trabajo pretende realizar una ayuda al diagnóstico, proporcionando un primer diagnóstico, de esta manera que realice una primera criba entre todos los casos que recibe el especialista.

Además, para dar más información proporciona un mapa de activación, de este modo el especialista con ver las zonas activadas puede dar mayor o menor valor al diagnóstico proporcionado por el algoritmo. Además, se puede validar el funcionamiento del algoritmo comprobando como mejora tras el uso de data Augmentation .

Finalmente, después de los resultados obtenidos, se puede decir que se ha cumplido el objetivo marcado en un principio, realizar un diagnóstico y proporcionar como salida las zonas más influyentes en este. Sin embargo, aunque dichos resultados son buenos existe un margen de mejora.

Uno de los puntos a mejorar sería poder contar una cantidad de casos malignos superior a los actuales (584) de manera que sea comparable al número de casos benignos de casos benignos (32542). Aunque realizando un Data Augmentation hemos conseguido mejorar los resultados notablemente, la posibilidad de introducir patrones nuevos incrementa la posibilidad de acierto.

Por otro lado, este trabajo está centrado en la detección, no obstante, se podría añadir la funcionalidad de segmentar la zona lesionada o aumentar la clasificación a otro tipo de lesiones. De esta manera daríamos al especialista una herramienta más versátil y precisa.

REFERENCIAS

[1] A. Alcalá Ramírez Del Puerto, J. C. Hernández-Rodriguez, M. Sendín-Martín, J. Ortiz-Alvarez, J. Conejo-Mir Sánchez, and J. J. Pereyra-Rodriguez, "Skin cancer mortality in Spain: adjusted mortality rates by province and related risk factors," International journal of dermatology, vol. 62, no. 6, pp. 776-782, 2023/06//, 2023.

[2] M. Arnold, D. Singh, M. Laversanne, J. Vignat, S. Vaccarella, F. Meheus, A. E. Cust, E. de Vries, D. C. Whiteman, and F. Bray, "Global Burden of Cutaneous Melanoma in 2020 and Projections to 2040," JAMA Dermatology, vol. 158, no. 5, pp. 495-503, 20

[3] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," Nature, vol. 542, no. 7639, pp. 115-118, 2017/02/01, 2017.

[4] D. Palacios-Martínez, and R. A. Díaz-Alonso, "Dermatoscopia para principiantes(I): características generales," Medicina de Familia. SEMERGEN, vol. 43, no. 3, pp. 216-221, 2017.

[5] "NEVUS MELANOCÍTICOS" , Hospital Universitario Infanta Elena, https://www.hospitalinfantaelena.es/es/cartera-servicios/servicios-medicos/dermatologia.ficheros/40713-NEVUS%20MELANOC%C3%8DTICOS.pdf (12/06/2023)

[6] "Léntigo", redacción médica. https://www.redaccionmedica.com/recursos-salud/diccionario-enfermedades/lentigo (12/06/2023)

[7] "Queratosis seborreica", Mayo Clinic, https://www.mayoclinic.org/es-es/diseases-conditions/seborrheic-keratosis/symptoms-causes/syc-20353878 (12/06/2023)

[8] Suman K. Jha ; Magda D. Méndez , "Cafe Au Lait Macules", Nacional library of Medicine. https://www.ncbi.nlm.nih.gov/books/NBK557492/ (12/06/2023)

[9] Rotemberg, V., Kurtansky, N., Betz-Stablein, B., Caffery, L., Chousakos, E., Codella, N., Combalia, M., Dusza, S., Guitera, P., Gutman, D., Halpern, A., Helba, B., Kittler, H., Kose, K., Langer, S., Lioprys, K., Malvehy, J., Musthaq, S., Nanda, J., Reiter, O., Shih, G., Stratigos, A., Tschandl, P., Weber, J. & Soyer, P. A patient-centric dataset of images and metadata for identifying melanomas using clinical context. Sci Data 8, 34 (2021). https://doi.org/10.1038/s41597-021-00815-z

[10] J. Deng, W. Dong, R. Socher, L. J. Li, L. Kai, and F.-F. Li, "ImageNet: A large-scale hierarchical image database." pp. 248-255. (2015)

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.

[12] K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition.". (2016)

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions." pp. 1-9 sensors (2022)

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition." pp. 770-778. applied sensed (2022)

[15] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436-444, 2015/05/01, 2015.

[16] A. Gola Isasi, B. García Zapirain, and A. Méndez Zorrilla, “Melanomas non-invasive diagnosis application based on the ABCD rule and pattern recognition image processing algorithms,” Computers in Biology and Medicine, vol. 41, no. 9, pp. 742-755, 2011/09/01/, 2011.

[17] E. Nasr-Esfahani, S. Samavi, N. Karimi, S. M. R. Soroushmehr, M. H. Jafari, K. Ward, and K. Najarian, "Melanoma detection by analysis of clinical images using convolutional neural network." pp. 1373-1376.

[18] R. Kaur, H. GholamHosseini, R. Sinha, and M. Lindén, "Melanoma Classification Using a Novel Deep Convolutional Neural Network with Dermoscopic Images," Sensors, 22, 2022].

[19] J. Rashid, M. Ishfaq, G. Ali, M. R. Saeed, M. Hussain, T. Alkhalifah, F. Alturise, and N. Samand, "Skin Cancer Disease Detection Using Transfer Learning Technique," Applied Sciences, 12, 2022].

[20] B. H. M. van der Velden, H. J. Kuijf, K. G. A. Gilhuijs, and M. A. Viergever, “Explainable artificial intelligence (XAI) in deep learning-based medical image analysis,” Medical Image Analysis, vol. 79, pp. 102470, 2022/07/01/, 2022.

[21] X. Zeng, L. Wen, Y. Xu, and C. Ji, “Generating diagnostic report for medical image by high-middle-level visual information incorporation on double deep learning models,” Computer Methods and Programs in Biomedicine, vol. 197, pp. 105700, 2020/12/01/, 2020.

[22] K. Yan, X. Wang, L. Lu, L. Zhang, A. P. Harrison, M. Bagheri, and R. M. Summers, "Deep Lesion Graphs in the Wild: Relationship Learning and Organization of Significant Radiology Image Findings in a Diverse Large-Scale Lesion Database." pp. 9261-9270. Cornell University 2017

[23] S. Shen, S. X. Han, D. R. Aberle, A. A. Bui, and W. Hsu, “An interpretable deep hierarchical semantic convolutional neural network for lung nodule malignancy classification,” Expert Systems with Applications, vol. 128, pp. 84-95, 2019/08/15/, 2019.

[24] K. Yan, X. Wang, L. Lu, L. Zhang, A. P. Harrison, M. Bagheri, and R. M. Summers, "Deep Lesion Graphs in the Wild: Relationship Learning and Organization of Significant Radiology Image Findings in a Diverse Large-Scale Lesion Database." pp. 9261-9270. 2017

[25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization." pp. 618-626. IEEE International Conference on Computer Vision 2017.

[26] A. Hilbert, L. A. Ramos, H. J. A. van Os, S. D. Olabarriaga, M. L. Tolhuisen, M. J. H. Wermer, R. S. Barros, I. van der Schaaf, D. Dippel, Y. B. W. E. M. Roos, W. H. van Zwam, A. J. Yoo, B. J. Emmer, G. J. Lycklama à Nijeholt, A. H. Zwinderman, G. J. Strijkers, C. B. L. M. Majoie, and H. A. Marquering, “Data-efficient deep learning of radiological image data for outcome prediction after endovascular treatment of patients with acute ischemic stroke,” Computers in Biology and Medicine, vol. 115, pp. 103516, 2019/12/01/, 2019.

[27] F. Bozkurt, “Skin lesion classification on dermatoscopic images using effective data augmentation and pre-trained deep learning approach,” Multimedia Tools and Applications, vol. 82, no. 12, pp. 18985-19003, 2023/05/01, 2023.

[28] Z. Hussain, F. Gimenez, D. Yi, and D. Rubin, "Differential Data Augmentation Techniques for Medical Imaging Classification Tasks," AMIA Annu Symp Proc, vol. 2017, pp. 979-984, 2017.

[29] V. K. Vemuri, "The Hundred-Page Machine Learning Book," Journal of Information Technology Case and Application Research, vol. 22, no. 2, pp. 136-138, 2020/04/02, 2020.

[30] A. Géron, "Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems", O'Reilly Media, 2017.

[31] Tutor, "Redes neuronales profundas", Código Fuente. https://www.codigofuente.org/redes-neuronales-profundas-tipos-caracteristicas/ (01/06/2023)

[32] "Understanding VGG16: Concepts, Architecture, and Performance", Datagen. https://datagen.tech/guides/computer-vision/vgg16/ (07/06/2023)

[33] Great Learning, "Everything you need to know about VGG16", médium. https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918 (07/06/2023)

[34] Jordi Torres, "Data Augmentation y Transfer Learning", Jordi TORRES.IA. https://torres.ai/data-augmentation-y-transfer-learning-en-keras-tensorflow/ (03/07/2023)

[35] Codella N, Gutman D, Celebi ME, Helba B, Marchetti MA, Dusza S, Kalloo A, Liopyris K, Mishra N, Kittler H, Halpern A. "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)". arXiv: 1710.05006 [cs.CV]

CÓDIGO PYTHON

```
import numpy as np

import os

import cv2

import PIL.Image as Image

import keras.backend as K

from keras.models import *

from keras.callbacks import *

from keras.applications import vgg16

from keras.models import Sequential,Model

from keras.callbacks import ReduceLROnPlateau,ModelCheckpoint,EarlyStopping

import keras

from keras.preprocessing.image import ImageDataGenerator

from keras.layers import Dense ,concatenate, Conv2D,Flatten , InputLayer,
ZeroPadding2D, Activation, MaxPooling2D,Dropout, GlobalAveragePooling2D,
Input

from tensorflow.compat.v1 import disable_eager_execution

from keras.saving.legacy.save import load_model


from google.colab import drive

drive.mount('/content/drive')

os.chdir('/content/drive/My Drive/IMAGENES')


path_benign='Benign'

lista_ficheros1=os.listdir(path_benign)

path_malignant='Malignant'

lista_ficheros2=os.listdir(path_malignant)

np.random.shuffle(lista_ficheros1)

np.random.shuffle(lista_ficheros2)

lista_1_2 = np.asarray(lista_ficheros1)

lista_2_2 = np.asarray(lista_ficheros2)


"""'''train, val, test: imágenes originales de entrenamiento, validación y
test'''"""
```

```
train_x=[]

val_x=[]

test_x=[]


"""Train_set_y,  val_set_y,test_set_y: ground_truth de entrenamiento,
validación y test.


70% entrenamiento, 20% validación. 10% tests


"""


num_train1=round(0.7*len(lista_ficheros1))

num_val1=round(0.2*len(lista_ficheros1))

num_test1=round(0.1*len(lista_ficheros1))


num_train2=round(0.7*len(lista_ficheros2))

num_val2=round(0.2*len(lista_ficheros2))

num_test2=round(0.1*len(lista_ficheros2))


print(len(lista_ficheros1),len(lista_ficheros2))


print(num_train1+num_val1+num_test1)

if len(lista_ficheros1)<(num_train1+num_val1+num_test1):

  num_test1=num_test1-(num_train1+num_val1+num_test1)+len(lista_ficheros1)

if len(lista_ficheros1)>(num_train1+num_val1+num_test1):

  num_test1=num_test1+(num_train1+num_val1+num_test1)-len(lista_ficheros1)

print(num_train1)

print(num_val1)

print(num_test1)

print(num_train1+num_val1+num_test1)


print(num_train2+num_val2+num_test2)

if len(lista_ficheros2)<(num_train2+num_val2+num_test2):

  num_test2=num_test2-(num_train2+num_val2+num_test2)+len(lista_ficheros2)
```

```
if len(lista_ficheros2)>(num_train2+num_val2+num_test2):
  num_test2=num_test2+(num_train2+num_val2+num_test2)-len(lista_ficheros2)
print(num_train2)
print(num_val2)
print(num_test2)
print(num_train2+num_val2+num_test2)

num_train=num_train1+num_train2
num_val=num_val1+num_val2
num_test=num_test1+num_test2

"""Para la ground truth:"""

test_y  = np.zeros((num_test,2))
train_y = np.zeros((num_train,2))
val_y   = np.zeros((num_val,2))

for i,imagen in enumerate(lista_ficheros1):
  I=Image.open(os.path.join(path_benign,imagen))
  #I=np.asarray(I)
  I=I.resize((256,256))
  if i <num_test1:
    test_x.append(np.asarray(I))
    test_y[i,:]=[1,0]
  if num_test1<=i<num_test1+num_val1:
    val_x.append(np.asarray(I))
    val_y[i-num_test1,:]=[1,0]
  if num_test1+num_val1<=i:
    train_x.append(np.asarray(I))
    train_y[i-(num_test1+num_val1),:]=[1,0]

"""Ahora los tumores malignos"""

for i,imagen in enumerate(lista_ficheros2):
```

```
  I=Image.open(os.path.join(path_malignant,imagen))
#  I=np.array(I)
  I=I.resize((256,256))
  if i <num_test2:
    test_x.append(np.asarray(I))
    test_y[i+num_test1,:]=[0,1]
  if num_test2<=i<num_test2+num_val2:
    val_x.append(np.asarray(I))
    val_y[i-num_test2+num_val1,:]=[0,1]
  if num_test2+num_val2<=i:
    train_x.append(np.asarray(I))
    train_y[i-(num_test2+num_val2)+num_train1,:]=[0,1]

train_x = np.asarray(train_x)
val_x =  np.asarray(val_x)
test_x = np.asarray(test_x)

print(np.shape(train_x))
print(np.shape(val_x))
print(np.shape(test_x))

train_y=np.asarray(train_y)
val_y=np.asarray(val_y)
test_y=np.asarray(test_y)

print(np.shape(train_y))
print(np.shape(val_y))
print(np.shape(test_y))

input_shape=(256,256,3)

adamgrad=keras.optimizers.Adam(learning_rate=0.001, beta_1=0.09, beta_2=0.8,
epsilon=1e-8, decay=0.0,
              amsgrad=True)
```

```
opt = keras.optimizers.SGD(lr=0.001)

"""batch_size=12"""

vgg=vgg16.VGG16(include_top=False,weights='imagenet',input_shape=input_shape)

for layer in vgg.layers:
    layer._name = layer._name + str('_Cc')

x = vgg.output

"""Fine tuning de las últimas 5 capas"""

for nn,layer in enumerate(vgg.layers):
    layer.trainable=False
    if(nn>=(len(vgg.layers)-5)):
        layer.trainable=True

x = GlobalAveragePooling2D(name='avg_pool1')(x)
z = Dense(20,activation = 'relu') (x)
w = Dense(2,activation = 'softmax') (z)
model = Model(inputs=vgg.input, outputs=w)

"""reduce_lr = ReduceLROnPlateau(monitor='accuracy',
                                 factor=0.5,
                                 patience=2,
                                 verbose=1,
                                 mode='max',
                                 min_lr=0.0000001)

earlyStopping = EarlyStopping(monitor='loss', patience=20, verbose=1,
mode='min')

to save a model or weights (in a checkpoint file)
```

```
"""


mcp_save = ModelCheckpoint('.mdl_wts.hdf5', save_best_only=True,
monitor='val_loss', mode='min')


"""Ahora Manuel volvía a hacer el data augmentation. Pero no se usa y no lo
hemos copiado.


model.compile(loss='binary_crossentropy',optimizer=adamgrad,
metrics=['accuracy',tensorflow.keras.metrics.AUC()])


"""


model.compile(loss='binary_crossentropy',optimizer='sgd',
metrics=['accuracy',keras.metrics.AUC()])

model.summary()


"""Empieza el entrenamiento"""


for iteri in range(1):

  modelo_history = model.fit(train_x, train_y,validation_data=(val_x,
val_y),epochs=30, batch_size=32)


def confusion_matriz(test_pred,test_set_y):

  tam = len(test_pred)

  tabla_conteo = np.zeros((1,4)).astype(int)

  for i in range(tam):

    if((test_pred[i] and test_set_y[i])):

      #verdadero positivo

      tabla_conteo[0,0]=tabla_conteo[0,0]+1

    if((not(test_pred[i]) and not(test_set_y[i]))):

      #verdadero negativo

      tabla_conteo[0,1]=tabla_conteo[0,1]+1

    if((test_pred[i] and not(test_set_y[i]))):

      #falso positivo

      tabla_conteo[0,2]=tabla_conteo[0,2]+1
```

```
    if((not(test_pred[i]) and test_set_y[i])):
      #falso negativo
      tabla_conteo[0,3]=tabla_conteo[0,3]+1
  #print(tabla_conteo)
  return tabla_conteo


def calculo_valores(tabla_conteo):
    tabla_conteo = tabla_conteo
    a,b = tabla_conteo.shape


    precision = np.zeros(a) #tp+tn/tp+tn+fp+fn
    vpp       = np.zeros(a) #tp/fp+tp
    especifici= np.zeros(a) #tn/tn+fp
    sensibilid= np.zeros(a) #tp/tp+fn
    total = sum(tabla_conteo[0,:])
    for i in range(a):
        precision[i] = (tabla_conteo[i,0]+tabla_conteo[i,1])/total
        vpp[i]       =
tabla_conteo[i,0]/(tabla_conteo[i,0]+tabla_conteo[i,2])
        especifici[i]=
tabla_conteo[i,1]/(tabla_conteo[i,1]+tabla_conteo[i,2])
        sensibilid[i]=
tabla_conteo[i,0]/(tabla_conteo[i,0]+tabla_conteo[i,3])
    # print("Precision",precision, "\n",
    #       "VPP",vpp, "\n",
    #       "Especificidad",especifici, "\n",
    #       "Sensibilidad",sensibilid, "\n",)
    return precision,vpp,especifici,sensibilid,tabla_conteo


test_pred = model.predict(test_x)
test_pred1 = np.argmax(test_pred,1)
test_y_1= np.argmax(test_y,1)
#print(test_pred1)  one hot encoding
print(test_pred1)
print(test_y_1)
```

```
cm = confusion_matriz(test_pred1,test_y_1)
calculo_valores(cm)

import matplotlib.pyplot as plt
history_dict = modelo_history.history
print(history_dict.keys())
#dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
acc      = modelo_history.history['accuracy' ]
val_acc  = modelo_history.history['val_accuracy' ]
loss     = modelo_history.history[    'loss' ]
val_loss = modelo_history.history['val_loss' ]

epochs   = range(1,len(acc)+1,1)

plt.plot ( epochs,     acc, 'r--', label='Training acc'  )
plt.plot ( epochs, loss,  'b', label='Training loss')
plt.title ('Training accuracy and loss')
plt.ylabel('acc')
plt.xlabel('epochs')



plt.legend()
plt.figure()

model=load_model("model.keras")
from sklearn.metrics import roc_curve, roc_auc_score
test_pred1=np.argmax(test_pred,1)
test_pred3=np.ones([158,2])
for i, test_ in enumerate(test_pred1):
  if test_==0:
    test_pred3[i,0]=1
    test_pred3[i,1]=0
  if test_==1:
    test_pred3[i,0]=0
```

```
    test_pred3[i,1]=1
auc= roc_auc_score(test_y,test_pred3)
print(auc)
test_y1=np.argmax(test_y,1)
fpr, tpr, thresholds = roc_curve(test_y1,test_pred1)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--', label='línea de no-discriminación')
plt.plot(fpr, tpr, label='Keras (area = {:.3f})'.format(auc))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend()
plt.show()


def get_output_layer(model, layer_name):
    # get the symbolic outputs of each "key" layer (we gave them unique
names).
    layer_dict = dict([(layer.name, layer) for layer in model.layers])
    layer = layer_dict[layer_name]
    return layer


def visualize_class_activation_map(model_path, img_path, output_path,):
  model=load_model(model_path)
  original_img=Image.open(os.path.join(img_path))
  original_img=original_img.resize((256,256))
  w,h,_=np.shape(original_img)


  #reshape to the network imput shape (3,w,h)


  img=np.array([np.transpose(np.float32(original_img),(0,1,2))])


  #get the 512 imput wight to the softmax
  class_weight = model.layers[-1].get_weights()[0]
  final_conv_layer = get_output_layer(model, 'block5_pool_Cc')
```

```
  get_output = K.function([model.layers[0].input], [final_conv_layer.output,
model.layers[-1].output])
  [conv_outputs, predictions] = get_output([img])
  conv_outputs = conv_outputs[0, :, :, :]
  conv_outputs.shape
  conv_outputs=np.array([np.transpose(np.float32(conv_outputs),(2,1,0))])
  conv_outputs = conv_outputs[0, :, :, :]
  #Create the class activation map.
  cam = np.zeros(dtype = np.float32, shape = conv_outputs.shape[1:4])
  for i, w in enumerate(class_weight[:, 1]):
        cam += w * conv_outputs[i, :, :]
  print ("predictions", predictions)
  PREDIC= np.argmax(predictions,1)
  if PREDIC == 0:
    print('BENIGNO')
  else :
    print('MALIGNO')
  cam /= np.max(cam)
  cam = cv2.resize(cam, (256, 256))
  heatmap = cv2.applyColorMap(np.uint8(255*cam), cv2.COLORMAP_JET)
  heatmap[np.where(cam < 0.2)] = 0
  img = heatmap*0.5 + original_img
  img2= np.zeros([256,256,3])
  image =  Image.fromarray(np.uint8(img))
  image.save(output_path)
  image.show()
  f = open ('salida.txt','a')
  z=len(img_path)
  predic=str(PREDIC)
  salida=img_path[z-16:z-4]+','+predic
  f.write(salida)
  f.close()

model.save("model.keras")
```

```
path_result='ENTRADAS'

path_sal='Salidas'

lista_ficheros3=os.listdir(path_result)

for i,imagen1 in enumerate(lista_ficheros3):

  print(os.path.join(path_result,imagen1))

  visualize_class_activation_map("model.keras",
os.path.join(path_result,imagen1) ,  os.path.join(path_sal,imagen1))

  os.path.join(path_result,imagen)


visualize_class_activation_map("model.keras",
'ENTRADAS/ISIC_0013562.jpg','ISIC_0013562.jpg')

visualize_class_activation_map("model_dataAumentation.keras",
'ENTRADAS/ISIC_0013562.jpg','ISIC_0013562.jpg')
```