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

Redes neuronales convolucionales para la determinación de la profundidad del melanoma

Autor: Eduardo Rubio Camacho
Tutoras: María del Carmen Serrano Gotarredona
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 las Tecnologías de Telecomunicación

Redes neuronales convolucionales para la determinación de la profundidad del melanoma

Autor:
Eduardo Rubio Camacho

Tutoras:
María del Carmen Serrano Gotarredona
Catedrática de Universidad

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: Redes neuronales convolucionales para la determinación de la profundidad del melanoma

Autor: Eduardo Rubio Camacho

Tutoras: María del Carmen Serrano Gotarredona

Begoña Acha Piñero

El tribunal nombrado para juzgar el Trabajo 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

Mis más sinceros agradecimientos a mi familia, a quienes me han acompañado en mi día a día, a quienes pese a la distancia siempre están conmigo y a quienes ya no están, pero siempre estarán.

Gracias con todo mi corazón.

Resumen

El melanoma, tumor maligno de piel, en la última década ha sufrido un notable aumento de su incidencia en todo el mundo, siendo el noveno tipo de cáncer más frecuente en mujeres y el onceavo en hombres.

Este tumor es el cáncer de piel más mortal, debido a su tendencia a la metástasis y a la falta de tratamientos eficaces en etapas avanzadas del tumor. Por lo que es imprescindible la detección temprana de esta lesión, ya que es la única manera de combatir contra su avance.

Las tareas de clasificación mediante inteligencia artificial son una herramienta revolucionaria que está en auge en nuestros días, por lo que debemos de usar esta herramienta para conseguir una fusión entre tecnología y medicina.

Con este trabajo se pretende investigar sobre la clasificación del melanoma según su profundidad, ya que es el indicador de pronóstico más relevante para esta lesión.

Para abordar esta tarea se ha hecho uso de herramientas de aprendizaje automático, mediante Redes Neuronales Convolucionales (CNN) capaces de detectar patrones en imágenes dermatoscópicas para concluir un diagnóstico temprano de las lesiones.

Como resultado de este proyecto, se han llegado a resultados alentadores que podrían allanar camino hacia la detección temprana del estado del melanoma.

Abstract

Melanoma, a malignant skin tumor, has undergone a notable increase in incidence worldwide in the last decade, being the ninth most frequent type of cancer in women and the eleventh in men.

This tumor is the most deadly skin cancer, due to its tendency to metastasize and the lack of effective treatments in advanced stages of the tumor. Therefore, early detection of this lesion is essential, as it is the only way to combat its progression.

The classification tasks through artificial intelligence are a revolutionary tool that is booming nowadays, so we must use this tool to achieve a fusion between technology and medicine.

The aim of this work is to investigate the classification of melanoma according to its depth, since it is the most relevant prognostic indicator for this lesion.

To address this task, use has been made of machine learning tools, using Convolutional Neural Networks (CNN) capable of detecting patterns in dermoscopic images to conclude an early diagnosis of the lesions.

Encouraging results have been reached that could pave the way towards early detection of melanoma status.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

dataset	Conjuntos de datos con los que se entrena la red y validan los resultados
ANN	*Artificial Neural Network* (Red Neuronal Artificial)
CNN	*Convolutional Neural Network* (Red Neuronal Convolucional)
TP	*True Positive* (Verdadero Positivo)
TN	*True Negative* (Verdadero Negativo)
FP	*False Positive* (Falso Positivo)
FN	*False Negative* (Falso Negativo)
TPR	*True Positive Rate* (Tasa de Verdaderos Positivos)
TNR	*True Negative Rate* (Tasa de Verdaderos Negativos)
ROC	*Reveiver Operating Characteristic* (Característica Operativa del Receptor)
AUC	*Area Under Curve* (Área Bajo la Curva ROC)

1 INTRODUCCIÓN

Aprendizaje automático, aprendizaje profundo, inteligencia artificial, son términos que se han apropiado de nuestro día a día, que han producido un cambio de paradigma a la hora de afrontar problemas. Estas herramientas pueden dar soluciones óptimas a problemas de procesamiento de lenguaje natural, optimización, clasificación y agrupación de datos, diagnóstico médico…

El problema a solucionar por este proyecto es la clasificación de muestras de melanoma según su profundidad y como herramienta para encontrar una solución óptima hemos elegido Inteligencia Artificial, en concreto Aprendizaje Profundo, que según ciertos estudios, que comentaremos posteriormente, tiene un futuro esperanzador en este campo.

En este apartado realizaremos una breve introducción a la Inteligencia Artificial, así como a los objetivos que se quieren alcanzar con este proyecto, dejando un apartado completo para el análisis médico del melanoma.

1.1 Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo

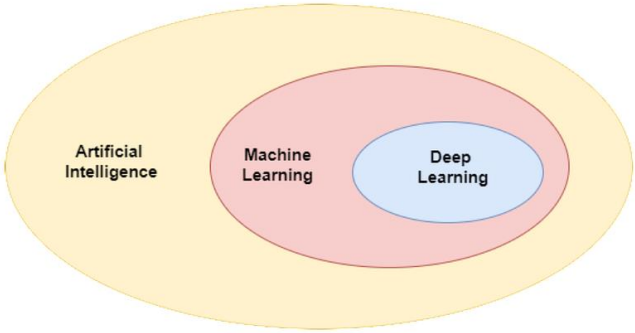

Figura 1–1. Artificial Inteligence, Machine Learning, Deep Learning [1].

La inteligencia artificial (*Artificial Inteligence*) surgió en la década de 1950 y su definición más concisa sería la automatización de tareas. La inteligencia artificial es una solución óptima para problemas bien definidos con reglas claras y estrictas, pero no puede hacer frente a todo lo que salga de este margen.

La clasificación de señales, de patrones dentro de una imagen o problemas más complejos y menos claros, dieron lugar al aprendizaje automático (*Machine Learning*).

El aprendizaje automático se basa en un sistema que recibe un dato de entrada con su salida esperada, sin indicar explícitamente las reglas que debe de seguir para llegar a dicha salida a partir del dato de entrada.

Estos sistemas no se programan, se entrenan. Se le suministran muchos ejemplos, de manear que encuentra patrones estadísticos mediante los cuales crean reglas para conseguir automatizar estas tareas. Aprenden mediante a mejorar en una tarea mediante la experiencia.

El aprendizaje profundo (*Deep Learning*) hace hincapié en el aprendizaje de patrones mediante capas sucesivas, en las que capa a capa se van aprendiendo características más complejas de los datos. La 'profundidad' hace referencia a esta secuencia de capa tras capa, llegando a hablarse de profundidad como el número de capas que conforman el modelo.

La diferencia principal entre Aprendizaje Automático y Aprendizaje Profundo es que en Aprendizaje Automático la extracción de características de los datos ha de hacerse manualmente, mientras que en el

Aprendizaje Profundo esa extracción de característica se realiza de manera automáticamente por la red.

1.2 Objetivos

El objetivo principal de este proyecto es obtener resultados favorables en la predicción del estado del melanoma mediante su profundidad mediante distintas soluciones basadas en Redes Neuronales Convolucionales (CNN).

Además, tenemos otros objetivos, no menos importantes, por cumplir mediante la realización del proyecto son los siguientes:

- **Comprender la importancia del diagnóstico temprano de lesiones malignas de la piel como el melanoma.**
- **Conocer el estado del arte del diagnóstico del melanoma mediante técnicas de distintas análisis de imágenes.**
- **Comprender la teoría del funcionamiento de las Redes Neuronales Artificiales (ANN) y las Redes Neuronales Convolucionales (CNN).**
- **Aplicar el aprendizaje por transferencia para conseguir resultados favorables sobre conjuntos de datos limitados.**
- **Aprender cómo llevar a cabo el proceso de entrenamiento de modelos ANN y CNN utilizando los recursos disponibles en entornos de entrenamiento online.**
- **Procesar y analizar los resultados obtenidos para evaluar el rendimiento de los modelos.**

2 ANÁLISIS MÉDICO

En este apartado se pretende realizar una breve introducción al melanoma, así como a la técnica de evaluación conocida como dermatoscopia, mediante la cual se han obtenido las imágenes que conforman el *dataset* de este proyecto. Se pretende hacer hincapié en la importancia de la profundidad de la lesión en el diagnóstico de la gravedad de esta, mostrando diversos indicadores de pronóstico que se basan en la profundidad para cuantificar la gravedad de la lesión.

2.1 La piel

La piel está dividida en tres capas diferenciadas, de exterior a interior: epidermis, dermis e hipodermis.

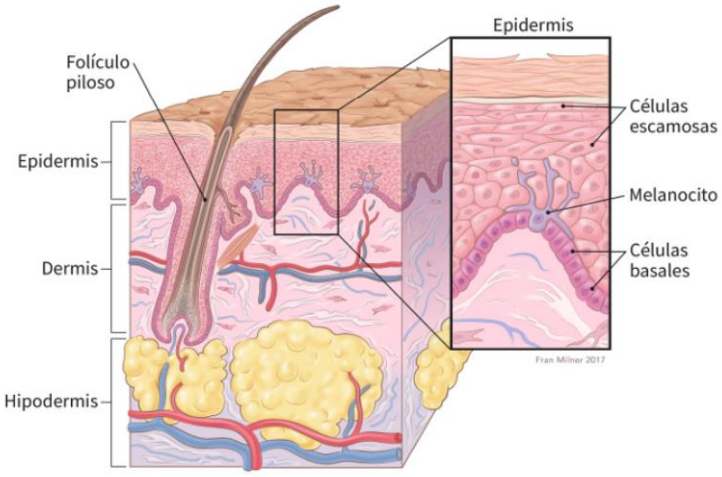

Figura 2–1. Epidermis, dermis e hipodermis [2].

La capa más externa de la piel, la epidermis, es la más propensa a la aparición de la mayoría de los cánceres de piel.

La epidermis está constituida por tres tipos principales de células:

- **Células escamosas**: Estas células están situadas en la parte más externa de la epidermis. Éstas se desprenden, siendo remplazadas por células que son generadas en capas más profundas.
- **Células basales**: Estas células están situadas en la parte interna de la epidermis. Éstas se dividen constantemente para remplazar a las células escamosas. Se aplanan y convierten en las anteriores a medida que avanzan hacía partes más externas de esta capa.
- **Melanocitos**: Productores de melanina, pigmento marrón responsable del bronceado de la piel, además de otorgar de un efecto protector contra los rayos ultravioletas del sol.

Esta capa está separada de su capa contigua, la dermis, mediante la membrana basal.

2.2 Melanoma

El melanoma es un tipo de cáncer de piel o tumor originado en los melanocitos, de los cuales obtienen su color oscuro tan característico. Aunque ciertos melanomas no producen melanina, produciendo un color rosado e incluso blanco.

Este tumor es conocido como el cáncer de piel más peligroso debido a su rapidez de crecimiento, como a su gran capacidad de metástasis. Su temprano diagnóstico es crucial para evitar su propagación y poder recibir tratamiento adecuado.

La profundidad de la lesión es un factor determinante y crítico en el diagnóstico, ya que está directamente relacionado con la gravedad del tumor.

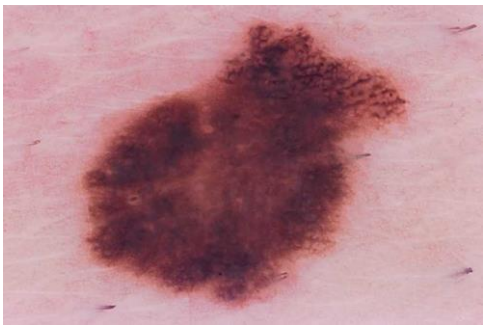

Figura 2–2. Melanoma [3].

2.3 Dermatoscopia

La dermatoscopia es una técnica no invasiva de evaluación de lesiones en la piel que, mediante un instrumento de explotación conocido como dermatoscopio, usa aumento y luz para visualizar estructuras en la epidermis y en la dermis superficial que de otra manera no serían visibles en una rutina a simple vista.

Esta técnica es una alternativa para métodos de evaluación de lesiones cutáneas invasivos, como la biopsia, que se basa en la toma de una muestra de tejido de la lesión para su posterior análisis.

Se ha demostrado que la dermatoscopia aumenta la sensibilidad y la especificidad del diagnóstico frente a examinaciones a simple vista [4].

Mediante esta técnica se ha conformado el *dataset* con el que se ha entrenado nuestra inteligencia artificial, ya que los ejemplos muestran rasgos más acentuados y distintivos. Esto facilitará el aprendizaje para poder realizar distinciones entre las distintas clases.

2.4 Características en las que se fija un dermatólogo

Existen algoritmos que se usan en dermatología para el diagnóstico de melanoma de manera simple mediante el uso de imágenes dermatoscópicas de las lesiones.

Estos algoritmos establecen una puntuación en el análisis de lesiones melanocíticas en función de las características, a tener en cuenta según el algoritmo, que presenten. En función a dicha puntuación, se determina con precisión si se trata de una lesión benigna o melanoma.

A continuación, se listan los algoritmos más conocidos y las características que tienen en consideración de las lesiones [5].

2.4.1 Regla ABCD

En la regla ABCD [6] las siglas ABCD hacen referencia a las iniciales de las características a analizar de la lesión: Asimetría, Borde, Color y Diferencia estructural.

En la siguiente tabla se muestra cada característica con su rango posible de valores y su factor mediante el que se pondera su peso en la puntuación:

Característica	Rango	Factor
Asimetría	0 – 2	1,3
Borde	0 – 8	0,1
Color	1 – 6	0,5
Diferencia estructural	1 – 6	0,5

Tabla 2-1. Puntuación regla ABCD.

La puntuación se calcula como la suma de la puntuación dentro del rango de cada característica por su factor correspondiente para todas las características:

$$P_T = \sum_{i=1}^{4} p_i f_i \qquad (1\text{-}1)$$

Donde:

- P_T : Puntuación Total.
- p_i : Puntuación para la característica i-ésima.
- f_i : Factor de ponderación de la característica i-ésima.

Se considerará melanoma si la puntuación total es mayor a igual a 5,75.

2.4.2 Lista de los siete puntos

El método de la lista de los siete puntos [7] consigue una puntuación dándole diferente importancia a características que suelen estar presentes en el melanoma.

A continuación, se muestra las características asociada a su puntuación:

Característica	Puntuación
Red pigmentada atípica	2
Velo azul blanquecino	2
Patrón vascular atípico	2
Estrías irregulares	1
Estructuras de regresión	1
Pigmentación irregular	1
Glóbulos irregulares	1

Tabla 2-2. Puntuación lista de los siete puntos.

Si la puntuación obtenida es mayor a 2, se considera melanoma.

2.4.3 Lista de los tres puntos

El método de la lista de los tres puntos [8] es parecido al anterior, la lista de los siete puntos, pero menos exhaustivo.

Siendo tres las características determinantes para determinar si es melanoma: estructuras asimétricas, red pigmentada atípica y estructuras azul blanquecinas.

Característica
Estructura asimétrica
Red pigmentada atípica
Estructura azul blanquecinas

Tabla 2-3. Características lista de los tres puntos.

En este caso si tiene presente más de una de estas características se considerará una lesión maligna.

2.4.4 Método de Menzies

El método de Menzies [9] valora once características de la lesión, divididas entre características malignas y benignas.

En la siguiente tabla se muestra la división de características entre malignas y benignas:

Características malignas	Características benignas
Simetría	Velo azul blanquecino
Monocromía	Múltiples puntos oscuros
	Pseudópodos
	Retículo radial
	Distribución radial
	Despigmentación cicatriz
	Puntos oscuros periféricos
	Múltiples colores
	Múltiples puntos grisáceos

Tabla 2-4. Características método de Menzies.

Para que la lesión no sea considerada como melanoma:

- No deberá de estar presente ninguna de las características malignas.
- Deberá de estar presente al menos una característica benigna.

En caso contrario la lesión se considerará melanoma.

2.4.5 Algoritmo de CASH

En el algoritmo de CASH [10], las siglas CASH hacen referencia a la iniciales de las características a examinar de la lesión: Color, desorden Arquitectural, Simetría y Homogeneidad.

Según la presencia y estado de dichas características en la lesión se da la siguiente puntuación:

Característica	Estado	Puntuación
Color	Marrón claro, marrón oscuro, negro, rojo, blanco, azul	1 – 2 – 3 – 4 – 5 – 6
Desorden Arquitectural	No existe, moderado, marcado	0 – 1 – 2
Simetría	Simetría de color, forma y contorno, simetría de color, forma o contorno, sin simetría	0 – 1 – 2
Homogeneidad	Red, glóbulos, pseudópodos, velo azul blanquecino, estructuras de regresión, manchas, vasos sanguíneos polimorfos	1 – 2 – 3 – 4 – 5 – 6 – 7

Tabla 2-5. Puntuación algoritmo CASH.

En la tabla, se representan las características, sus posibles estados con sus correspondientes puntuaciones.

Para un puntuación superior a siete, se considera melanoma.

2.5 Indicadores de pronóstico

En el estudio del melanoma y de su gravedad se utilizan distintos indicadores de pronóstico para una

cuantificación común a la gravedad del estado del tumor.

Estos indicadores tienen en común que determinan la gravedad del tumor en base a la profundidad de incisión en la piel.

2.5.1 Índice de Breslow

El índice de Breslow se reconoce como el indicador de pronóstico más útil [11]. Contempla la división de tres gravedades según distintos umbrales de profundidad, medido desde el estrato granuloso hasta las células malignas más profundas que invaden la dermis.

En la siguiente tabla se muestran los tres niveles de gravedad según el índice de Breslow:

Riesgo	Profundidad (mm)
Bajo	< 0,8
Intermedio	0,8 – 4
Elevado	> 4

Tabla 2-6. Índice de Breslow.

2.5.2 Niveles de invasión de Clark

Los niveles de invasión de Clark se basan también en la profundidad del melanoma en la piel. A diferencia del anterior, no se recoge la medida de la profundidad en milímetros, sino hasta que capa de la piel se prolonga.

Esta medida está más relacionada con los distintos lugares de la piel donde aparezca el melanoma, ya que las capas de la piel tienen distintos grosor dependiendo de la zona que estemos tratando.

Los niveles de invasión de Clark son cinco y se dividen como se muestra en la siguiente tabla:

Niveles	Profundidad
Nivel I	In situ
Nivel II	Invasión de dermis papilar
Nivel III	Expansión de la dermis papilar
Nivel IV	Invasión dermis reticular
Nivel V	Invasión tejido celular subcutáneo

Tabla 2-7. Niveles de invasión de Clark.

2.5.3 Etapas del melanoma

El estado del melanoma fue categorizado, por G. Argenziano et al. [12], en tres etapas distintas diferenciadas en profundidad.

Esta categorización hace referencia a la gravedad del tumor en función de la profundidad de incisión en la piel.

En la siguiente tabla se muestra las diferentes etapas con su profundidad asociada:

Etapas	Profundidad (mm)
Etapa I	< 0.76
Etapa II	0.76 – 1.5
Etapa III	> 1.5

Tabla 2-8. Etapas del melanoma.

Las imágenes que conforman el *dataset* utilizado están categorizadas según este indicador, por lo que la red neuronal realizará la clasificación de las lesiones entre estas tres etapas distintas del melanoma.

3 ESTADO DEL ARTE

Con este apartado se pretende exponer el estado actual del diagnóstico de la profundidad del melanoma, mediante un estudio real realizado a profesionales con experiencia en dermatoscopia, así como distintos trabajos basados en técnicas de análisis de imágenes enfocados en predecir la profundidad de la lesión.

3.1 Porcentaje de acierto en diagnóstico

En 2022 se realizó un estudio internacional [13] en el que se evaluó la exactitud de profesionales con experiencia en dermatoscopia. Una parte de dicho estudio consistía en que los profesionales estimaran el grosor de Breslow de los melanomas basándose en imágenes dermatoscópicas.

Este estudio se realizó sobre un conjunto de muestras de melanoma de un total de 1456 imágenes de dermatoscopia de melanomas, de las cuales 788 pertenecían a la categoría de in situ, 474 a melanoma invasivo con grosor menor o igual de 1,0 mm y 194 a melanoma invasivo con grosor mayor a 1,0 mm.

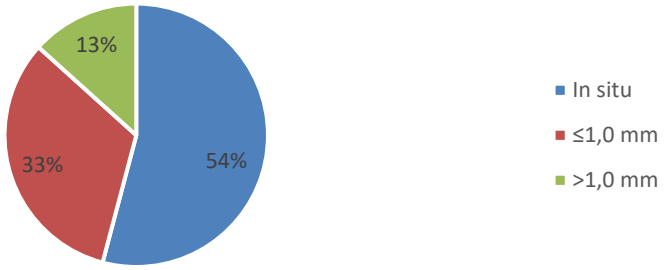

Figura 3–1. Porcentaje de muestras estudio diagnóstico.

Finalmente, 438 profesionales realizaron 22.314 predicciones sobre la profundidad de las muestras, obteniendo los siguientes resultados:

Índice de Breslow	Precisión
≤1,0 mm[1]	85,4% - 86,4%
>1,0 mm	69.2% - 72.5%

Tabla 3-1. Precisión estudio diagnóstico.

3.2 Trabajos relacionados

Existen distintos trabajos basados en técnicas de análisis de imágenes para la clasificación y predicción de la profundidad del melanoma.

A continuación, se exponen dichos trabajos, así como los resultados obtenidos en cada uno de ellos.

[1] Dentro de este índice se incluyen también los melanomas de la categoría in situ.

3.2.1 Rubegni et al. (2010)

Rubegni et al. realizan un análisis dermatoscópico digital (ADD) en el que, mediante un software propietario específico, logran la extracción de 49 características distintas del melanoma [14].

Este estudio se realiza sobre 141 muestras dermatoscópicas de distintos melanomas pertenecientes a una base de datos privada, con el fin de predecir la profundidad antes de realizar la cirugía.

Se realiza una distinción de las muestras en dos clases:

- Melanomas finos, de menos de 1 mm de profundidad.
- Melanomas gruesos, de más de 1 mm de profundidad.

Se realizó una matriz de clasificación en la que serían evaluadas las características extraídas de cada muestra para predecir su clasificación final.

Los resultados obtenidos en este estudio son muy favorables en la predicción de la profundidad:

- Para melanomas finos se consiguió un acierto de 97 de 108 muestras, equivalente a una precisión del 89,8%.
- Para melanomas gruesos se consiguió un acierto de 25 de 33 muestras, equivalente a una precisión del 86,5%

En conclusión, se lograron identificar correctamente 122 de 141 muestras, lo que equivale a una exactitud de acierto total del 86,52%.

3.2.2 Aurora Sáez et al. (2016)

Aurora Sáez et al. realizan un estudio para predecir la profundidad del melanoma mediante un método que combina regresión logística y redes neuronales artificiales [15].

Realiza una extracción de características de las muestras que se basan en hallazgos clínicos que se relacionan con la profundidad de la lesión.

Para la clasificación usan un método de regresión logística basado en un hibrido de modelo lineal y modelo de red neuronal producto-unidad, conocido en inglés como *Logistic regression using Initial variables and Product Units* (LIPU) [16].

En el estudio se realizan una clasificación binaria y una clasificación multiclase en base a la profundidad del melanoma:

- La clasificación binaria diferenciará entre melanoma delgado y grueso.
- La clasificación multiclase diferenciará entre melanoma delgado, intermedio y grueso.

Esta clasificación se basa en las etapas del melanoma [12] que detallamos anteriormente.

De manera que la clasificación binaria identifica melanoma delgado como la etapa I (menor a 0,76 mm) y melanoma grueso engloba tanto la etapa II como la etapa III (mayor o igual a 0,76 mm).

Mientras que la clasificación multiclase se ciñe a la definición de etapas del melanoma, correspondiendo melanoma delgado a etapa I, melanoma intermedio a etapa II y melanoma grueso a etapa III.

El *dataset* utilizado para la realización del estudio corresponde al conjunto de 250 imágenes dermatoscópicas de *Interactive Atlas of Dermoscopy* [3], que se descompone en:

- 167 imágenes de melanoma pertenecientes a la etapa I.
- 54 imágenes de melanoma correspondientes a la etapa II.
- 29 imágenes de melanoma correspondientes a la etapa II.

Los resultados obtenidos finalmente son favorables:

- Para clasificación binaria se obtiene una precisión total del 77,6%.
- Para clasificación multiclase se obtiene una exactitud total del 68,4%.

3.2.3 Jaworek-Korjakowska et al. (2019)

Jaworek-Korjakowska et al. realizan un estudio donde en el que predicen la profundidad del melanoma mediante redes neuronales convolucionales (CNN) [17].

El estudio se centra en la clasificación multiclase entre las distintas etapas del melanoma [12] descritas anteriormente.

El método escogido en este estudio radica en una solución de *Deep Learning*, mediante una arquitectura CNN VVG-19 [18], con su salida conectada a una red neuronal ANN *fully-connected* para realizar la clasificación de las muestras entre las tres etapas del melanoma.

El entrenamiento de la red se realiza con imágenes pertenecientes a *Interactive Atlas of Dermoscopy* [3], con un número de ejemplos bastante reducido con solo 244 imágenes de dermatoscopia del melanoma:

- Etapa I: 142 ejemplos.
- Etapa II: 46 ejemplos.
- Etapa III: 55 ejemplos.

Estos ejemplos fueron pasados por un preprocesamiento para eliminar características sin valor para la red de las imágenes como: líneas, burbujas de aire, pelos, marco negro…

Los resultados obtenidos finalmente son bastante favorables:

- Para la clasificación de las muestras de etapa I se obtiene una precisión del 86,9%.
- Para la clasificación de las muestras de etapa II se obtiene una precisión del 85,5%.
- Para la clasificación de las muestras de etapa I se obtiene una precisión del 89,3%.
- La exactitud total en la clasificación es del 87,2%.

4 MÉTODO

EN este apartado se pretende establecer una base teórica común sobre las redes neuronales artificiales (*Artificial Neural Network, ANN*) y las redes neuronales artificiales convolucionales (*Convolutional Neural Network, CNN*), de su funcionamiento, los distintos elementos que las componen y su base matemática. Por último, se presentan los modelos o arquitecturas elegidos para la realización del trabajo.

4.1 Redes neuronales artificiales

Una red neuronal artificial (ANN) es un modelo matemático y computacional que se utiliza para relacionar entradas con salidas. Uno de sus usos es la clasificación de datos previamente etiquetados.

A grandes rasgos, una red neuronal, mediante cierto aprendizaje o entrenamiento es capaz de conseguir aproximar con cierta precisión la función:

$$f(x) = y \qquad (1\text{-}2)$$

Donde:

- x : dato de entrada.
- $f(\bullet)$: función aproximada mediante el aprendizaje.
- y : valor real de la etiqueta o de salida.

4.1.1 La neurona

Una red neuronal está formada por la interconexión de neuronas. La neurona de define como el elemento fundamental y más básico de una red neuronal.

En el caso general, la neurona consta de entradas, que son a su vez las salidas de otras neuronas. Su funcionamiento se basa en realizar cierta operación o procesamiento sobre dichas entradas y generar así su propia salida.

Estos son los tres componentes principales de la neurona: las entradas, la operación que realiza sobre ellas y la salida, producto del procesamiento de las entradas.

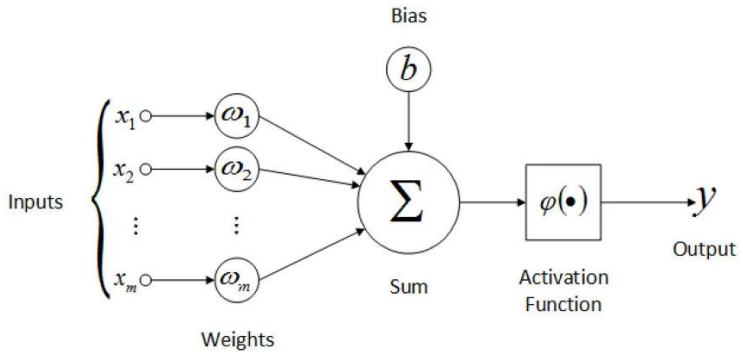

Figura 4–1. Arquitectura básica de una neurona [19].

El procesamiento que realiza la neurona se divide en dos funcionalidades: función de entrada y función de

activación.

A continuación, se muestra de forma matemática la operación entrada-salida que se realiza para el caso general de la neurona i-ésima[2]:

$$y_i = \varphi(\sum_{j=1}^{m} \omega_j^i x_j^i + b) \tag{1-3}$$

Donde:

- y_i : salida de la neurona i .
- $\varphi(\bullet)$: función de activación.
- x_j^i : entrada desde la salida de la neurona j hacia la neurona i .
- ω_j^i : peso asociado a la entrada x_j^i .
- b : valor umbral de la función de activación, conocido como sesgo, o del inglés *bias*.

4.1.1.1 Función de entrada

La función de entrada se basa en la combinación de las distintas entradas de la neurona para dar como resultado un único valor. Cada entrada está ponderada por un cierto peso que hace referencia a la importancia del valor de dicha entrada en la combinación con las demás entradas.

Para el caso general, algunas de las funciones de entrada más utilizadas para combinar los valores de entrada de la neurona son la siguientes [20]:

- Función suma ponderada:

$$f(x) = \sum_{j=1}^{n} \omega_j^i x_j \tag{1-4}$$

- Función máximo:

$$f(x) = \max(\omega_j^i x_j) \tag{1-5}$$

- Función mínimo:

$$f(x) = \min(\omega_j^i x_j) \tag{1-6}$$

La utilización de una función u otra está relacionada con el problema y los datos concretos. Aunque, la suma ponderada suele ser la más utilizada.

4.1.1.2 Función de activación

La función de activación se aplica al resultado de la función de entrada para dar lugar a la salida de la neurona.

Deben ser funciones no lineales para poder aproximar cualquier tipo de función. Gracias a esta característica, se consigue así aumentar la capacidad de la red para detectar patrones y relaciones complejas de los datos.

Algunas de las funciones de activación más utilizadas son [20]:

- Función sigmoide:

[2] La función de entrada representada en el ejemplo es la función suma ponderada.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1-7}$$

- Tangente hiperbólica:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1-8}$$

- Función rectificadora (*Rectified Lineal Unit, ReLU*):

$$\mathrm{Re}\,LU(x) = \max(0, x) \tag{1-9}$$

4.1.2 Arquitectura de una red neuronal

Una red neuronal está organizada en distintas capas: una capa de entrada, una o más capas ocultas y una capa de salida.

Estas redes se conocen como *totalmente conectadas*, en inglés *fully connected*, debido a que una neurona de una capa está conectada a todas las neuronas de la anterior y siguiente capa.

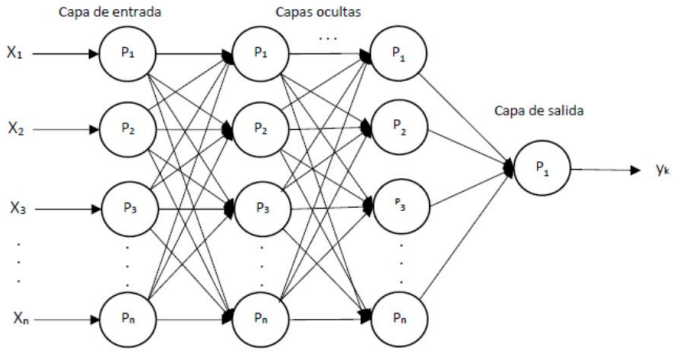

Figura 4–2. Arquitectura de capas red neuronal [21].

4.1.2.1 Capa de entrada

Las neuronas situadas en la capa de entrada representan el valor de las distintas muestras de los datos. Definen la entrada de los datos a la red y el valor de su salida dependen íntegramente de estos.

En el caso general, podemos decir que existen tantas neuronas como características muestren los datos de entrada a la red.

4.1.2.2 Capa oculta

En el caso general, las redes poseen más de una capa oculta, lo que se conoce como redes multicapa.

La capacidad de procesamiento, de predicción y de clasificación está estrechamente relacionada con el número de neuronas y el número de capas ocultas.

En las capas más superficiales se produce un procesamiento de los datos más básico y a medida que se va profundizando en sus capas el procesamiento de los datos acaba siendo más exhaustivo, fijándose más detalladamente en los detalles que caracterizan los datos.

Aunque, a priori, pueda parecer que aumentando el número de neuronas y de capas tendríamos un red que sea más precisa en su clasificación, debemos tener en cuenta que un aumento excesivo de ambas podría llevar a la red a la sobre especialización en el proceso de entrenamiento.

4.1.2.3 Capa de salida

En función del tipo de clasificación el número de neuronas en la capa de salida será diferente:

- Para clasificación binaria es necesario solo una neurona en la capa de salida, la cual es capaz de definir 1 o 0 en función de una clase u otra.
- Para clasificación multiclase es necesario tantas neuronas como clases participen en la clasificación, correspondiendo cada neurona a una clase específica.

La función de activación más utilizadas en las neuronas de la capa de salida depende también del tipo de clasificación:

- Para clasificación binaria se suele usar la función de activación *Sigmoide*. Debido a que su rango de clasificación se encuentra entre 0 y 1, el valor de la salida de la neurona estará también entre 0 y 1. En general, un valor superior a 0,5 indicará la pertenencia a una clase e inferior a 0,5 a su opuesta.
- Para clasificación multiclase se suele usar la función de activación *Softmax*. Debido a que la suma de los resultados de las neuronas es 1, lo que permite trasladar directamente el valor de la salida de la neurona como la probabilidad de pertenecer a dicha clase.

4.1.2.4 Sentido de propagación de la información

Según el sentido de la propagación de información de capa a capa podemos definir dos tipologías de redes: Red Neuronal Directa (*Feedforward Neural Network*, FNN) y Red Neuronal Recurrente (*Recurrent Neural* Network, RNN)

- En las redes de tipo FNN el sentido de la información tiene lugar desde las capas más externas hacia las capas internas de la red.
- En las redes de tipo RNN el sentido de la información existen bucles entre sus conexiones, de manera que la salida de una neurona sirve como entrada en la siguiente iteración. Esto las hace especialmente idóneas para la predicción de patrones y dependencias temporales.

4.1.3 Entrenamiento de una red neuronal

El funcionamiento de una neurona viene determinado por[3]:

$$y = \varphi(\omega x + b) \tag{1-10}$$

Donde:

- x : entrada de la neurona.
- ω : peso asociado a la entrada.
- $\varphi(\bullet)$: función de activación.
- b : valor umbral de la función de activación, conocido como sesgo, o del inglés *bias*.

El proceso de entrenamiento de la neurona consiste en la optimización de los pesos de cada entrada, así como del sesgo.

4.1.3.1 Función de coste

La función de coste determina el error que existe entre el valor de la predicción de nuestra red y el valor real esperado, para así optimizar los parámetros de la red.

[3] Simplificación de la ecuación 1-3 para una única neurona.

Existen dos definiciones a diferenciar según el número de muestras o ejemplos que tengamos en cuenta:

- Función de pérdida (*Loss*): Función definida para un ejemplo concreto.
- Función de coste (*Cost*): Función definida para una colección de ejemplos.

Existen diferentes funciones de costes, de las cuales las más conocidas son las siguientes.

4.1.3.1.1 Error cuadrático medio

Para regresión lineal, tenemos la función de coste *Error Cuadrático Medio* (*Mean Square Error, MSE*).

A continuación, se muestra la representación matemática de esta función para un número m de muestras:

$$J(\omega,b)=\frac{1}{m}\sum_{i=1}^{m}(f(x^i)-y^i)^2 \qquad (1\text{-}11)$$

Donde:

- x^i : valor de la muestra i-ésima.
- y^i : valor de la salida esperado, valor de la etiqueta.
- $f(\bullet)$: función aproximada de la red mediante entrenamiento.

4.1.3.1.2 Entropía cruzada

Para regresión logística, la función de coste más utilizada en problemas de clasificación es la función *Entropía Cruzada*, o en inglés *Cross-Entropy* [22].

Para clasificación binaria sobre un número de muestras m , *Binary Cross-Entropy*[4]:

$$J(\omega,b)=\frac{1}{m}(\sum_{i=1}^{m}y^i\log(f(x^i))+(1-y^i)\log(1-f(x^i))) \qquad (1\text{-}12)$$

Para clasificación multiclase sobre un número de muestras m , *Categorical Cross-Entropy:*

$$J(\omega,b)=\frac{1}{m}(\sum_{i=1}^{m}\sum_{k=1}^{k}y_k^i\log(f^k(x^i))+(1-y_k^i)\log(1-f^k(x^i))) \qquad (1\text{-}13)$$

Esta última función es igual que la anterior a diferencia de que se itera entre todas las clases k que tengan lugar en la clasificación multiclase.

4.1.3.2 Método de descenso por gradiente

El método de descenso por gradiente es un algoritmo matemático de búsqueda de mínimos para un función dada.

En lo relativo al entrenamiento de redes neuronales, consiste en encontrar el valor de los pesos y el sesgo que minimizan el valor de la función de coste.

4.1.3.2.1 Base matemática

En términos generales, el algoritmo hace uso del gradiente para encontrar la dirección de descenso más pronunciada y moverse en el sentido negativo, que disminuye el valor de la función.

Aquí entra en función el hiperparámetro factor de aprendizaje, en inglés *learning* rate, que define la magnitud

[4] Donde el valor esperado solo puede ser 0 o 1.

con la que se realiza el cambio en los parámetros, pesos y sesgo, para cada iteración del algoritmo.

La actualización de los pesos se define matemáticamente con la siguiente expresión:

$$\omega_{i+1} = \omega_i - p\frac{\Delta J(\omega, b)}{\Delta \omega} \qquad (1\text{-}14)$$

Donde:

- ω_i : valor del peso actual.
- p : hiperparámetro *learning rate*.
- $J(W)$: función de coste.
- $\frac{\Delta \bullet}{\Delta \omega}$: derivada parcial con respecto del peso.
- ω_{i+1} : actualización del nuevo peso, resultado del descenso por gradiente.

Al igual que para los pesos, también se realiza la actualización de los sesgos como se indica a continuación:

$$b_{i+1} = b_i + p\frac{\Delta J(\omega, b)}{\Delta b} \qquad (1\text{-}15)$$

Donde:

- b_i : valor del sesgo actual.
- p : hiperparámetro *learning rate*.
- $J(W)$: función de coste.
- $\frac{\Delta \bullet}{\Delta b}$: derivada parcial con respecto del sesgo.
- b_{i+1} : actualización del nuevo sesgo, resultado del descenso por gradiente.

La actualización de todos los parámetros de la red se produce en cada iteración del algoritmo. De esta forma, iteración a iteración, el algoritmo de descenso por gradiente se va acercando a su objetivo, el mínimo absoluto de la función de coste.

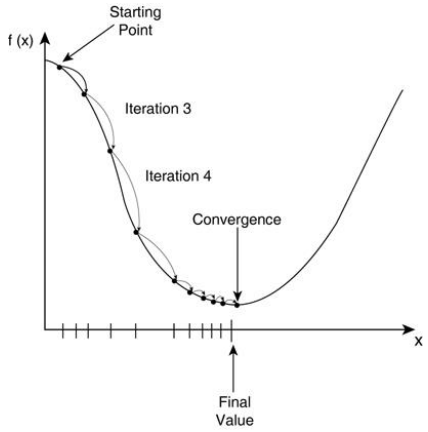

Figura 4–3. Iteraciones descenso por gradiente [23].

4.1.3.2.2 Métodos de descenso por gradiente

La diferencia entre los distintos métodos de descenso por gradiente radica en la manera en la actualizan los

parámetros de la red.

Algunos de los métodos de descenso por gradiente más conocidos son los siguientes [20]:

- Descenso de Gradiente Estocástico, en inglés *Stochastic Gradient Descent (SGD)*.
- Propagación Cuadrática Media, en inglés *Root Mean Square Propagation (RMSProp)*.
- Estimación Adaptativa de Momento, en inglés *Adaptative Moment Estimation (Adam)*.

4.1.3.3 Retropropagación

El método de retropropagación es el algoritmo mediante el cual se produce el entrenamiento de toda la red, la actualización de todos sus parámetros, de cada una de las neuronas.

El método de descenso por gradiente utiliza la retropropagación para actualizar todos los parámetros de la red.

Se define como época, a la siguiente iteración sobre los conjunto de muestras o datos con el que se quiere entrenar la red:

- Alimentación de la red con un conjunto de datos, esta calcula sus respectivas predicciones en función del valor de sus parámetros
- Cálculo del valor de la función de coste
- Cálculo de los gradientes y errores en la capa de salida
- Propagación de gradientes y errores desde la capa de salida hasta la capa de entrada
- Actualización de los pesos de la red

Esta iteración o época se repite en múltiples ocasiones, de manera que por cada iteración la red actualiza sus parámetros, converge y se ajusta a esos datos con los que se está alimentando, aprendiendo a clasificar, dando predicciones más precisas.

4.1.4 Conjunto de datos

Con conjunto de datos se hace referencia a los datos con los que entrena la red. También se conoce como conjunto de ejemplos o de muestras, siendo más conocido como *dataset* en inglés.

Este *dataset* se suele dividir en tres conjuntos diferenciados:

- Conjunto de entrenamiento: conjunto de datos con el que se entrena la red y esta aprende. 60% - 80%.
- Conjunto de validación: conjunto de datos con el que se mide el rendimiento de la red. Es una medida sesgada, ya que el entrenamiento de la red está influenciado por los datos del rendimiento de este conjunto. 20% - 10%.
- Conjunto de test: conjunto de datos con el que se mide el rendimiento real y final de la red. 20% - 10%.

El porcentaje de división indicado es el comúnmente usado para realizar la división, aunque cada subconjunto dependerá del problema en concreto que se esté enfrentando y del número de ejemplos que se disponen.

4.1.5 Problema de rendimiento

El problema de rendimiento común a todos los entrenamientos que pueda tener una red neuronal artificial se conoce como *Overfitting*, o sobre entrenamiento.

Esto ocurre cuando la red se sobre ajusta a los datos con los que se ha entrenado, siendo incapaz acertar y determinar la predicción correcta cuando se presenta un ejemplo que nunca ha visto, que esté fuera del conjunto de entrenamiento.

Este problema provoca un alto rendimiento con el conjunto de entrenamiento, pero un mal rendimiento con el conjunto de test.

4.1.5.1 Tratamiento de datos

Existen diversas técnicas de tratamiento de datos para solucionar el problema de *Overfitting*.

- Simplificación de datos: Esta técnica se realiza sobre el conjunto de muestras. Se reduce el número de características de los datos, para que la red no se fije en los pequeños detalles y se fije en las características determinantes de cada clasificación.
- Regularización: Esta técnica se realiza sobre la red a entrenar. Se mantiene el número de las características de los datos, pero se reducen los valores de los parámetros de la red. De esta manera, la red no tiene la "complejidad" como para distinguir características menos relevantes, centrándose en las características más relevantes de las muestras para distinguir entre una clase u otra.

4.1.5.2 Soluciones a problemas de rendimiento

4.1.5.2.1 Parada temprana

La parada temprana, en inglés *Early Stopping*, se basa en la monitorización del error de validación, o valor de la función de coste.

A medida que entrenamos la red con el conjunto de datos de entrenamiento, el error de la red en dicho conjunto irá descendiendo, así como el error del conjunto de validación. Aunque, por lo general, en un determinado momento, el error del conjunto de entramiento sigue disminuyendo, pero el error de validación aumenta.

Esta característica muestra como la red se está sobre ajustando, por lo que nos quedamos con el estado de la red en ese punto, ya que representa el estado óptimo de la red en el entrenamiento.

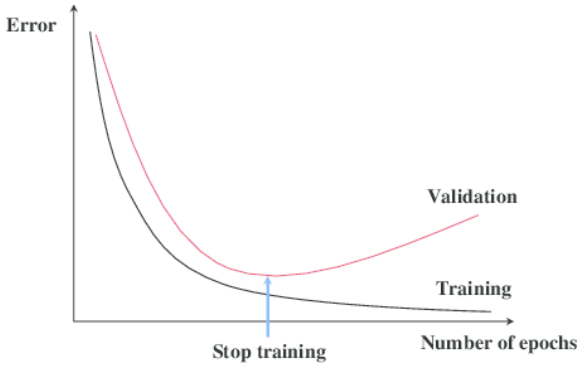

Figura 4–4. Método de *Early Stopping* [24].

4.1.5.2.2 Dilución

El fundamento de la técnica de regularización Dilución, en inglés *Dropout*, reside en omitir, de manera aleatoria, neuronas en el proceso de entrenamiento.

Con esta técnica se consigue que la red desarrolle una representación más redundante de la distinción entre clases, mediante neuronas distintas a las que se han omitido en ese determinado momento.

La probabilidad de poner a cero la neurona es un hiperparámetro, o parámetro configurable, de la red a ajustar dependiendo del problema concreto al que nos enfrentemos.

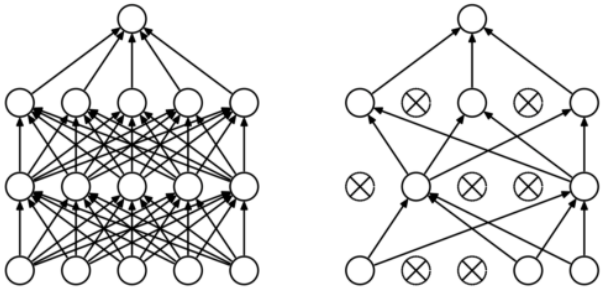

Figura 4–5. Red antes y después de aplicar *Dropout* [25].

4.1.5.2.3 Aumentado de datos

El aumentado de datos es una técnica que se utiliza para aumentar el tamaño del *dataset*, añadiendo diversidad al conjunto, mediante transformaciones o variaciones que se realizan sobre el *dataset* original.

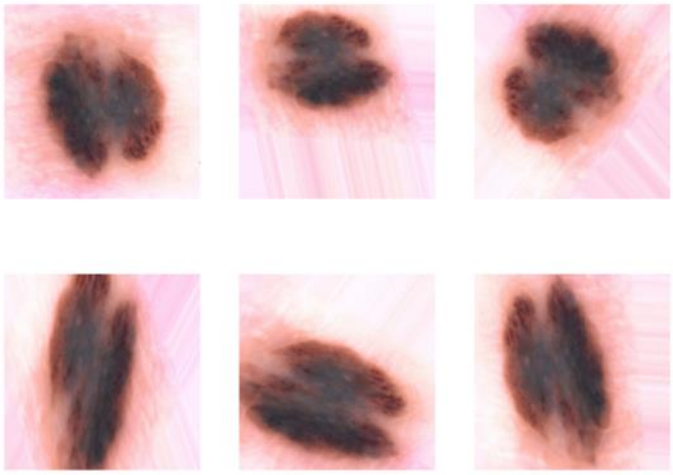

Figura 4–6. Aumentado sobre melanoma del *dataset* [3].

En la Figura 4-6 tenemos la dermatoscopia original de un melanoma, en la esquina superior izquierda. El resto de las imágenes son producto de realizar transformaciones aleatorias de rotación, desplazamiento, estiramiento, acercamiento y alejamiento.

4.2 Redes neuronales convolucionales

Las redes neuronales convolucionales, en inglés *Convolutional Neural Network (CNN)*, son un tipo de red neuronal artificial con una arquitectura diseñada para el procesamiento de imágenes, entre otros, así como la detección de patrones en éstas.

Todos los conceptos implicados en las redes neuronales artificiales típicas (ANN) son aplicados para este tipo de redes: funciones de coste, funciones de activación, entrenamiento, problemas de rendimiento…

4.2.1 Arquitectura de una red neuronal convolucional

La arquitectura de una red neuronal convolucional consta de distintas capas: capa de entrada, capa de convolución, capa de activación, capa de agrupación y capa de aplanamiento.

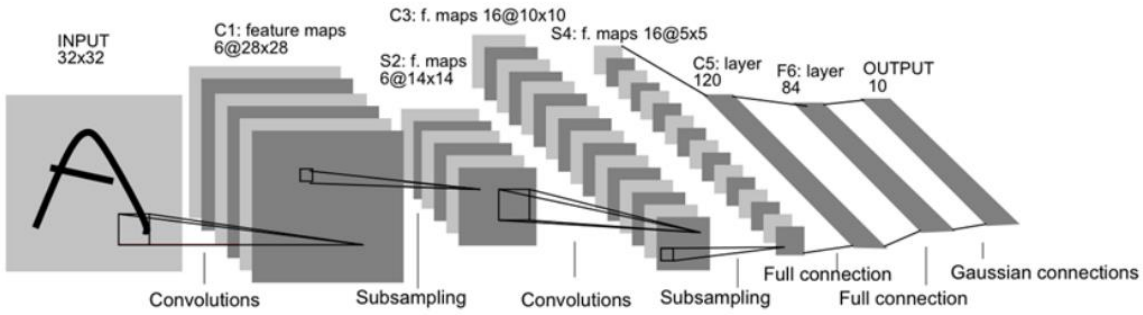

Figura 4–7. Arquitectura red neuronal modelo: LeNet-5 [26].

La capa de entrada representa los datos, representados de manera matricial, pueden ser secuencias de datos como: imágenes, señales y sucesos temporales.

La salida de esta red convolucional sirve de entrada para una red neuronal típica, que hará la función de clasificador entre las distintas clases que representen las distintas muestras de entrada.

4.2.1.1 Capa de convolución

La capa de convolución es la encargada de extraer patrones o características de los datos de entrada.

Los datos de entrada se presentan como una matriz. En el caso concreto de una imagen a color, estaríamos frente a una matriz de tres dimensiones: alto, ancho y profundidad. Donde la profundidad hace referencia a los tres canales de la imagen: rojo, verde y azul (RGB).

4.2.1.1.1 Filtro

El elemento principal de esta capa es el filtro, matriz, o también conocido como *kernel*, que es el encargado en realizar la operación de convolución sobre la entrada para así conseguir la extracción de las características.

El resultado de aplicar el filtro a los datos de entrada es otra matriz, conocida como matriz de características, que a su vez servirá como entrada a la siguiente capa convolucional.

Para un matriz de entrada de dimensión: alto, ancho y profundidad, el filtro a aplicar deberá tener, necesariamente la misma profundidad.

En el ejemplo de la imagen de entrada RGB, los filtros que se le apliquen a dicha imagen, necesariamente deberán tener profundidad tres.

4.2.1.1.2 La neurona

En general, cada neurona de la red tiene asociado un filtro, como entrada un mapa de características y como salida un mapa de características.

En el aprendizaje de la red se actualizan los valores del filtro asociados a cada neurona, ya que funcionan como los pesos de las redes neuronales artificiales típicas.

Existen tantas neuronas como posiciones a las que se aplica el filtro a los datos de entrada. Es decir, cada neurona de una capa se encarga de aplicar la operación de convolución a una región de la matriz de entrada y existen tantas neuronas como en regiones se divida la matriz de entrada.

Todas las neuronas de una capa comparten los pesos o valores de sus filtros, de esta manera extraen el mismo patrón independientemente de la región de la imagen en la que se encuentre. Esto también optimiza el entrenamiento, ya que solo se realiza la operación de actualización de los pesos de un filtro para toda una capa.

4.2.1.1.3 Operación de convolución

La operación de convolución no es más que el producto escalar de la matriz de entrada y el filtro, dando como resultado un elemento del mapa de características de salida.

Esta operación se realiza tantas veces como desplazamientos pueda realizar el filtro sobre la matriz de entrada.

En la siguiente imagen se muestra cómo se realiza paso a paso la operación de convolución. Podemos ver como el filtro, comenzando en la esquina superior izquierda de la matriz de entrada de dimensión 7x7, puede realizar cinco desplazamientos a su derecha y cinco desplazamientos hacia abajo, dando una matriz resultado de dimensión 5x5, que contiene veinticinco elementos, tantos como desplazamientos ha realizado el filtro.

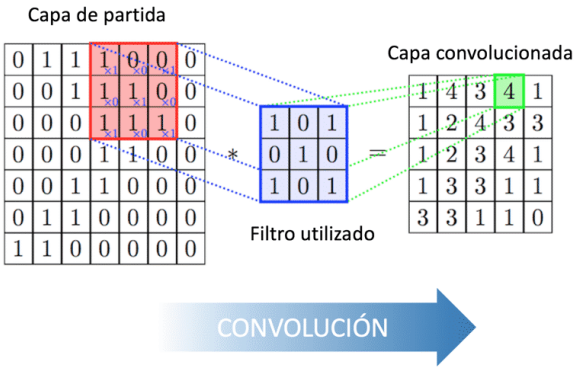

Figura 4–8. Operación de convolución [27].

Como ejemplo, se puede aplicar a la imagen RGB un filtro que tuviese una dimensión 5x5x3. Este filtro se descompondrá en una matriz 5x5 que realizará la operación de convolución al canal asociado a rojo, otra matriz de 5x5 que realizará la operación al canal asociado al verde y otra matriz de 5x5 asociada al canal azul.

El resultado de aplicar un filtro de profundidad tres, a una imagen de profundidad tres, será un mapa de características de profundidad tres.

A la salida habrá tantos mapas de características como filtros se apliquen a la entrada.

Es importante destacar que la operación de convolución es derivable, ya que esta característica es determinante en el entrenamiento de la red. Como en cualquier otra red neuronal, el método de entrenamiento es el descenso por gradiente, algoritmo que necesita de la característica de derivabilidad para optimizar la red.

4.2.1.2 Capa de activación

La capa de activación se suele colocar siempre después de la capa de convolución.

Al igual que en la redes neuronales típicas, esta capa aporta no linealidad al modelo, debido a que la convolución realizada no deja de ser una operación lineal.

La función de activación *ReLU* es la más usada en esta capa, ya que hace más rápido el entrenamiento, frente a funciones como la función *Tanh* o la función *Sigmoid* [28].

4.2.1.3 Capa de agrupación

El funcionamiento de la capa de agrupación, o en inglés *Pooling layer*, se basa en reducir la dimensionalidad del mapa de características obtenido después de realizar la operación de convolución.

Esta operación es crucial a la hora de optimizar el modelo, ya que disminuye los parámetros a entrenar del mismo, además de ayudar a controlar el sobre ajuste.

Su fin es el de resumir la información del mapa de características, realizando cierta operación sobre un subconjunto o región de este, logrando así reducir la dimensión.

Existen diversos tipos de capas de agrupación, dos de las más usadas son las siguientes:

- *Max-pooling*: obtiene como resultado el máximo de los elementos que se están evaluando.
- *Average-pooling*: obtiene como resultado la media de los elementos que se están evaluando.

Esta capa se suele intercalar después de varias sucesiones de capas de convolución y capas de activación.

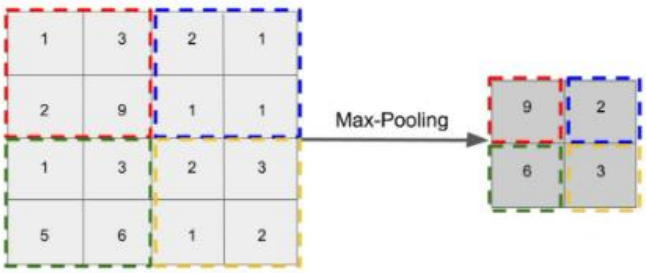

Figura 4–9. Ejemplo de agrupación *Max-pooling* [20].

4.2.1.4 Capa de aplanamiento

La capa de aplanamiento, en inglés *Flattening Layer*¸ se utiliza para interconectar la red neuronal convolucional (CNN) con la red neuronal artificial (ANN) o clasificador.

Su funcionamiento radica en realizar una operación que convierte el mapa de características resultante de la CNN, con cierta dimensión, en un vector o matriz unidimensional, para así poder interconectarlo con la red ANN.

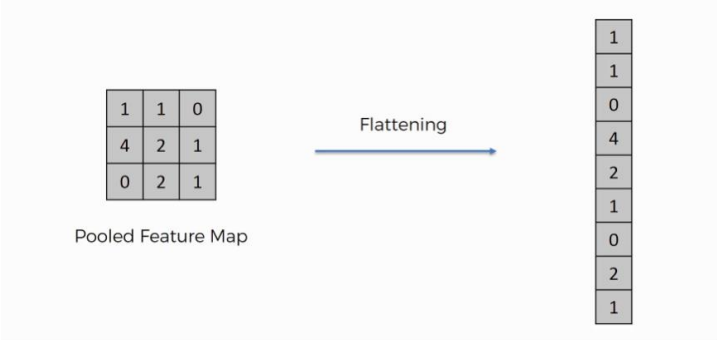

Figura 4–10. Ejemplo de aplanamiento [29].

Como se puede apreciar en la Figura 4-10, simplemente se despieza la matriz en un vector unidimensional.

4.3 Redes empleadas

Con este apartado se pretende exponer la arquitecturas de redes neuronales convolucionales (CNN) escogidas para la realización del proyecto.

Hacemos uso de modelos predefinidos para partir desde una base sólida y optimizada en cuestión de arquitectura, lo que permite un mayor rendimiento, así como una optimización de los recursos.

Las arquitecturas escogidas son las siguientes: VGG16, VGG19, Resnet50V2 e InceptionV3.

4.3.1 Aprendizaje por transferencia

El Aprendizaje por Transferencia, en inglés *Transfer Learning*, utiliza características aprendidas por un modelo frente a un problema para dar solución a otro problema distinto.

Además, se hace uso del Ajuste Fino, en inglés *Fine Tuning*, que se basa en reutilizar una red preentrenada (*Transfer Learning*), congelando ciertas capas de la red para que mantengan sus pesos, normalmente las primeras de la red que han aprendido características más básicas de las imágenes, y se entrenan las últimas capas de la red, para que aprendan características específicas del problema concreto que abordan.

Esto es determinante para nuestro proyecto, ya que el *dataset* del que disponemos está muy limitado en número de imágenes perteneciente a cada clase.

4.3.2 VGG16 y VGG19

Ambas arquitecturas, VGG16 y VGG19, se presentaron en el ImageNet Challenge 2014 [30], en el que lograron obtener el primer y segundo puesto en las tareas de clasificación y detección [18].

Estas arquitecturas usan filtros de dimensión 3x3, lo que llevó a una mejora significative de los modelos que se habían presentado previamente.

La diferencia entre una arquitectura y otra radica en el número de capas:

- VGG16 consta de 13 capas convolucionales con capa de activación *ReLU* y 3 capas de neuronas totalmente conectas con capa de activación *ReLU*. (Columna D de Figura 4-11/12).
- VGG19 consta de 16 capas convolucionales con capa de activación *ReLU* y 3 capas de neuronas totalmente conectadas con capa de activación *ReLU*. (Columna E de Figura 4-11/12).

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figura 4–11. Desgloses de arquitecturas VGG16 y VGG19 [18].

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Figura 4–12. Número de parámetros (millones) de arquitecturas VGG16 y VGG19 [18].

4.3.2.1 Capas del modelo a entrenar

Este apartado solo contempla las capas de la red neuronal convolucional (CNN), las capas convoluciones, que serán entrenadas o congeladas. El clasificador que va a la salida de esta red, las capas de la red neuronal artificial (ANN) *fully-connected*, siempre tendrán que ser entrenadas.

El número óptimo de capas a entrenar para los modelos son los siguientes para nuestro *dataset*[5]:

- VGG16: últimas 4 capas de la CNN. Comprende las capas *maxpool* y 3xconv3-512.
- VGG19: últimas 5 capas de la CNN. Comprende las capas *maxpool* y 4xconv3-512.

4.3.3 Resnet50V2

Resnet50V2 [31] es un tipo de red neuronal convolucional (CNN) que pertenece a la familia de redes residuales profundas [32].

Este modelo implementa conexiones residuales que facilita la propagación de la información entre las distintas capas de la red. En la que una capa de red obtiene el mapa de características producido por la capa anterior y también el mapa de la anterior a esta última.

[5] Este número de capas a entrenar ha sido producto de un estudio previo de cada arquitectura, además de prueba y error de diferentes números de capas para encontrar el óptimo del entrenamiento.

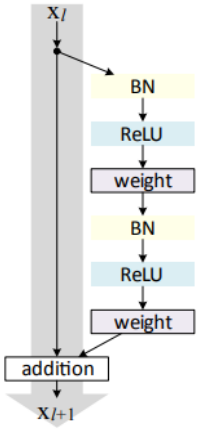

Figura 4–13. Conexión residual [31].

Esta arquitectura está compuesta por un total de 103 capas entre capas de convolución, activación, *pooling* y capas de neuronas totalmente conectadas. Concretamente, Resnet50V2, está compuesta, como se indica en su nombre, por 50 capas convolucionales.

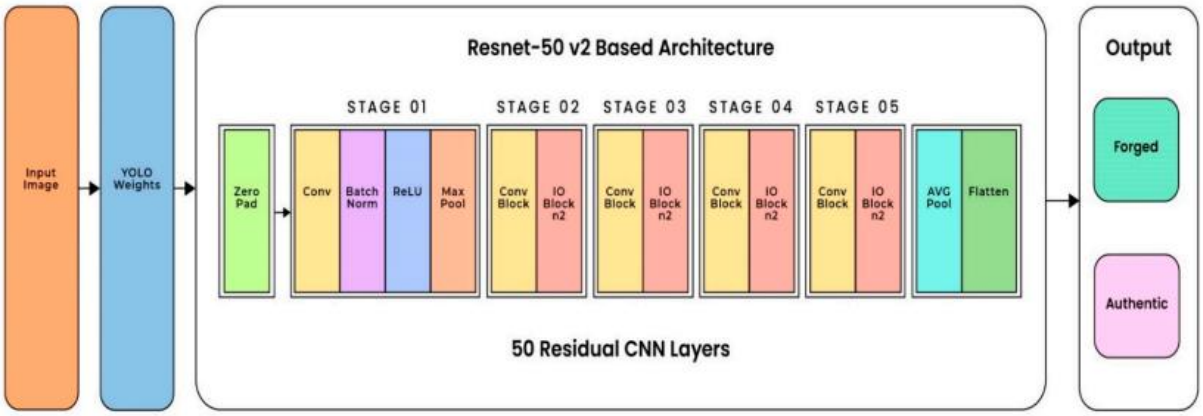

Figura 4–14. Arquitectura Resnet50V2[6] [33].

El número de parámetros o pesos que pueden ser entrenados en esta arquitectura es igual a 25,6 millones.

4.3.3.1 Capas del modelo a entrenar

El número óptimo de capas a entrenar de este modelo para nuestro *dataset*:

- Últimas 36 capas, que hacen referencia al quinto y último bloque de convoluciones.

4.3.4 InceptionV3

InceptionV3 [34] fue desarrollada por investigadores de Google en 2015 como evolución de la arquitectura GoogleNet [34].

Esta arquitectura está compuesta por capas de convolución con filtros de diferentes tamaños, capas *Average-pooling*, capas *Max-pooling*, capas *Dropout* y capas de neuronas totalmente conectadas. La función de coste utilizada por este modelo para clasificación multiclase es *Categorical Cross-Entropy.*

[6] YOLO (*You Only Look Once*): Referencia a que , en el caso concreto del artículo donde hemos obtenido la imagen de la arquitectura, los pesos que se usan son pertenecientes a un preentrenamiento de detección de imágenes en tiempo real.

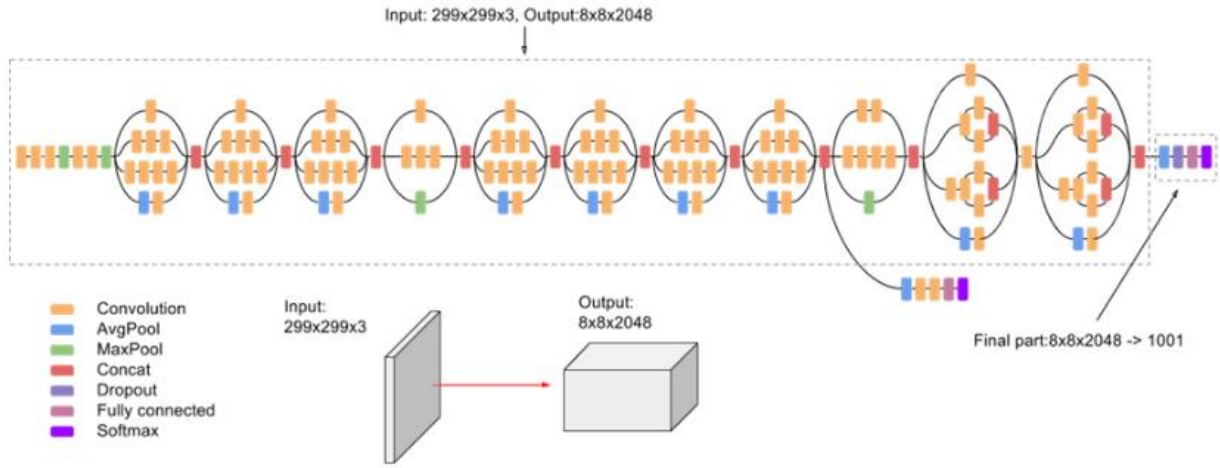

Figura 4–15. Arquitectura alto nivel InceptionV3 [35].

4.3.4.1 Capas del modelo a entrenar

El número óptimo de capas a entrenar de este modelo para nuestro *dataset*:

- Últimas 62 capas, que hacen referencia a los dos últimos bloques de convoluciones del modelo.

4.3.5 Comparación entre redes empleadas

En la siguiente tabla se realiza una comparación entre la exactitud frente al *dataset* de ImageNet [30], el número total de parámetros a entrenar y la profundidad en capas de cada modelo:

Modelo	Top-1 *Accuracy*	Top-5 *Accuracy*	Parámetros	Capas
VGG16	71,3%	90,1%	138.4M	16
VGG19	71,3%	90,0%	143,7M	19
Resnet50V2	76,0%	93,0%	25,6M	103
InceptionV3	77,9%	93,7%	23,9M	189

Tabla 4-1. Comparación entre redes [36].

5 IMPLEMENTACIÓN

PYTHON es el lenguaje de programación mediante el cual se ha llevado la implementación del entrenamiento, como la visualización de resultados de las redes neuronales convolucionales elegidas para dar solución al problema de clasificación de profundidad en el melanoma.

En este apartado se pretende mostrar el desarrollo de la implementación del entrenamiento, así como el procesamiento de los resultados que se ha realizado para cada una de las arquitecturas elegidas.

Ambas implementaciones se han llevado a cabo en la plataforma Google Colab [37]. Esta plataforma permite ejecutar código Python desde el navegador, además de acceso gratuito a GPUs, indispensable para tareas de entrenamiento de redes, debido al alto coste computacional que estas demandan.

El libro *Deep Learning with Python* de *François Chollet* [38] ha sido fundamental para introducción a la desarrollo de redes neuronales y guía imprescindible para la elaboración de este código.

5.1 Entrenamiento

A continuación, se muestra el código desarrollado para la implementación del entrenamiento de los distintos modelos. El código es el mismo para todos los modelos, exceptuando la selección del modelo, así como el número de capas que se deciden entrenar dependiendo del mismo.

La librería principal y fundamental utilizada para realizar esta implementación es *Keras* [39], librería de código abierto en lenguaje Python enfocada en el desarrollo de redes neuronales.

Librerías utilizadas

```
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
from tabulate import tabulate
from keras import metrics
from keras import callbacks
from keras import losses
from keras import optimizers
from keras import models
from keras import layers
from keras.preprocessing.image import ImageDataGenerator
```

Montaje de Drive en entorno de ejecución

```
drive.mount('/content/drive')
```

Montamos nuestro Google Drive en la ruta del entorno de ejecución: /content/drive.

Comandos en entorno de ejecución

```
!cp /content/drive/MyDrive/TFG/dataset.zip /content/
!unzip /content/dataset.zip -d /content/dataset/
```

- Comando 'cp': copia el fichero 'dataset.zip' en la carpeta '/content'. Con este comando conseguimos que nuestro *dataset* se aloje en el entorno de ejecución, disminuyendo el tiempo de entrenamiento. Sin este comando se necesitaría la transferencia de la imagen hacia el entorno de ejecución, previo a la carga de dicha imagen en memoria.
- Comando 'unzip': descomprime el *dataset* en el entorno de ejecución.

Tratamiento de imágenes

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=180,
    width_shift_range=0.25,
    height_shift_range=0.25,
    shear_range=15,
    zoom_range=[0.5, 1.5]
)

test_datagen = ImageDataGenerator(rescale=1./255)
```

ImageDataGenerator genera lotes de imágenes con aumento de datos en tiempo real. Se indica el escalado o normalización de las imágenes, como los parámetros a aplicar en el aumento de las imágenes:

- *Rotation_range*: Rango de grados para rotaciones aleatorias.
- *Width_shift_range*: Rango de cambio de ancho en proporción al total de la imagen.
- *Height_shift_range*: Rango de cambio de altura en proporción al total de la imagen.
- *Shear_range*: Rango de corte en grados. Distorsiona la imagen para recrear ángulos de percepción.
- *Zoom_range*: Rango de aumento o alejado de la imagen. Valor aleatorio entre los dos valores del intervalo.

Solo se realiza el aumento de la imágenes para el conjunto de imágenes de entrenamiento, para el conjunto de validación únicamente se normalizan.

```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    shuffle=True
)

validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=(224, 224),
    batch_size=32,
    shuffle=True
)
```

El método *Flow_from_directory* carga las imágenes del *dataset*. Los parámetros que debemos de indicar a este método son los siguientes:

- Ruta del *dataset*: Esta ruta debe respetar el siguiente árbol de directorios, donde se indica a qué subconjunto y clase pertenece cada imagen.

```
ruta_raiz/
├── train/
│   ├── clase_A/
│   ├── clase_B/
│   └── clase_C/
├── validation/
│   ├── clase_A/
│   ├── clase_B/
│   └── clase_C/
└── test/
    ├── clase_A/
    ├── clase_B/
    └── clase_C/
```

Figura 5–1. Árbol de directorios necesario.

- *Target_size*: Permite redimensionar la dimensión de las imágenes según de alto y de ancho que se indique.
- *Batch*_size: Tamaño de los lotes de imágenes.
- *Shuffle:* Indicar verdadero o falso si se quiere orden aleatorio o alfanumérico, respectivamente.

Cálculo de la partición ponderada del total de imágenes

```
n_samples = Counter(train_generator.labels)
n_total_samples = len(train_generator.labels)


weight_0 = (1/n_samples[0])*(n_total_samples/3.0)
weight_1 = (1/n_samples[1])*(n_total_samples/3.0)
weight_2 = (1/n_samples[2])*(n_total_samples/3.0)


class_weight = {0: weight_0, 1: weight_1, 2: weight_2}
```

Se calcula la partición ponderada del total de imágenes de cada clase.

Con esta partición, más adelante, indicaremos al modelo la ponderación de la función de coste, así como la importancia que deberá darle a las características de un ejemplo en función de la clase a la que pertenezca.

Modelo CNN

```
conv_base = applications.X(weights='imagenet',
                           include_top=False,
                           input_shape=(224, 224, 3)
                           )
```

- *Weights*: se indica que los pesos serán ajustados a los del *dataset* de ImageNet, de manera que siempre partimos de una red preentrenada.
- *Include_top*: Indicamos 'False' para que las imágenes no sufran de ningún preprocesamiento.
- *Input_shape*: Dimensión de los datos de entrada. El '3' indica que la imágenes son RGB.
- *X*: En su lugar indicaremos si la red que queremos entrenar es:
    - vgg.Vgg16
    - vgg.Vgg19
    - Resnet50V2
    - InceptionV3

Capas del modelo a entrenar

```
for i in conv_base.layers[:-X]:
  i.trainable = False

for i in conv_base.layers[-X:]:
  i.trainable = True
```

- X: En su lugar indicaremos el número de capas de la red, empezando por la última, que queremos que sean

entrenadas.

Clasificador – ANN

```
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(3, activation='softmax'))
```

Al modelo elegido se le añaden una red neuronal artificial (ANN) *fully-connected* que hacen de clasificador final entre las 3 clases.

- *Flatten*: aplana el mapa de características final de la CNN para que funcione de entrada para la ANN.
- *Dense*: Una capa de 128 neuronas, precedida de otra capa de 64 neuronas totalmente conectadas con activación ReLU.
- *Dropout*: Capa de dilución con una probabilidad de 0,5 de desactivar neuronas.
- *Dense*: Capa de 3 neuronas que representan las 3 clases de clasificación, con función de activación *Softmax*.

Descenso por gradiente

```
optimizer = optimizers.Adam(
    learning_rate = 0.0001
)
```

Adam es método de descenso por gradiente escogido para todos los modelos, debido a sus resultados, así como a su rapidez en alcanzar valores óptimos en el menor número de lotes.

Ajustamos el hiperparámetro *learning rate* a 0,0001.

Función de coste

```
loss = losses.CategoricalCrossentropy()
```

La función de coste elegida, común para todos los modelos, es la función *Categorical Cross-Entropy*.

Funciones de Callback

El valor monitorizado por las funciones de *Callback* es la función de coste en entrenamiento o la función de coste en validación. Ya que el valor de la función de coste es ponderado según el número de ejemplos que tenga cada clase, es la medida más fiel a la realidad del *dataset*, así como con la que mejores resultados se obtienen.

```
reduce_lr = callbacks.ReduceLROnPlateau(
    monitor='loss',
    factor=0.5,
    patience=4,
    min_lr=0.0000001
    )
```

La función *ReduceLROnPlateu* reduce el valor del hiperparámetro *learning rate* en el caso que la red no mejore el parámetro de validación que se le indique.

- monitor: parámetro de validación que monitoriza la función.
- factor: ponderación que se le aplica al hiperparámetro *learning rate*.
- patience: número de épocas que espera a que mejore el parámetro antes de aplicar el valor de *factor* al hiperparámetro *learning rate*.
- min_lr: valor mínimo que puede llegar a alcanzar el hiperparámetro *learning rate*.

```
checkpoint = callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    save_best_only=True,
    monitor='val_loss',
    mode='min'
    )
```

La función *ModelCheckPoint* guarda el valor de los pesos de la red cada vez que se alcanza la condición que se le indique mediante sus parámetros. Ayuda a controlar el *Overfitting* del modelo, ya que, a lo largo del entrenamiento, se queda con el mejor valor del parámetro de validación que le indiquemos.

- filepath: ruta y nombre del fichero donde se guardan los pesos de la red.
- save_weights_only: *True:* se guardarán únicamente los pesos de la red – *False:* se guardará el modelo al completo.
- save_best_only: *True*: únicamente se guarda el mejor valor – *False*: se guardan todos los valores a medida que vaya mejorando la red.
- monitor: parámetro de validación que monitoriza la función.
- mode: '*max'*: máximo valor del parámetro – *'min'*: mínimo valor del parámetro.

```
early_stopping = callbacks.EarlyStopping(
    monitor='val_loss',
    patience=30,
    mode='min'
    )
```

La función *EarlyStopping* detiene el entrenamiento cuando el parámetro de validación que monitoriza no mejora en el número de épocas que le indiquemos.

- monitor: parámetro de validación que monitoriza la función.

- patience: número de épocas que espera a que mejore el parámetro antes de detener el entrenamiento.
- mode: '*max*': máximo valor del parámetro – *'min'*: mínimo valor del parámetro.

Compilación del modelo

```
model.compile(
    optimizer=optimizer,
    loss=loss,
    metrics=['accuracy'],
    loss_weights=[class_weight[0],class_weight[1],class_weight[2]]
)
```

La función *compile* compila el modelo indicando el método de descenso por gradiente a usar, la función de pérdidas, la métrica que mostrará durante el entrenamiento, además de la función de coste, y aplica la ponderación de la función de coste que explicamos en el apartado: Cálculo de la partición ponderada del total de imágenes.

- optimizer: método de descenso por gradiente, anteriormente definido.
- loss: función de coste anteriormente definida.
- metrics: parámetro de validación que muestra, a parte del valor de la función de coste, durante el entrenamiento.
- loss_weights: Array que indica el peso que atribuye a cada clase en la función de coste, anteriormente calculado.

Entrenamiento

```
history = model.fit(
    train_generator,
    batch_size=32,
    epochs=100,
    callbacks=[reduce_lr, checkpoint, early_stopping],
    validation_data=validation_generator,
    class_weight=class_weight
)
```

- train_generator: generador de lotes de imágenes de entrenamiento, anteriormente definido.
- batch_size: tamaño de cada lote de imágenes.
- epochs: épocas que durará el entrenamiento.
- callbacks: funciones de *callback*, anteriormente definidas.
- validation_data: generador de lotes de imágenes de validación, anteriormente definido.
- class_weight: importancia que le dará a las características de cada clase según el diccionario que indiquemos, anteriormente calculado.

Gráfica de función de coste y exactitud durante entrenamiento

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

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

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

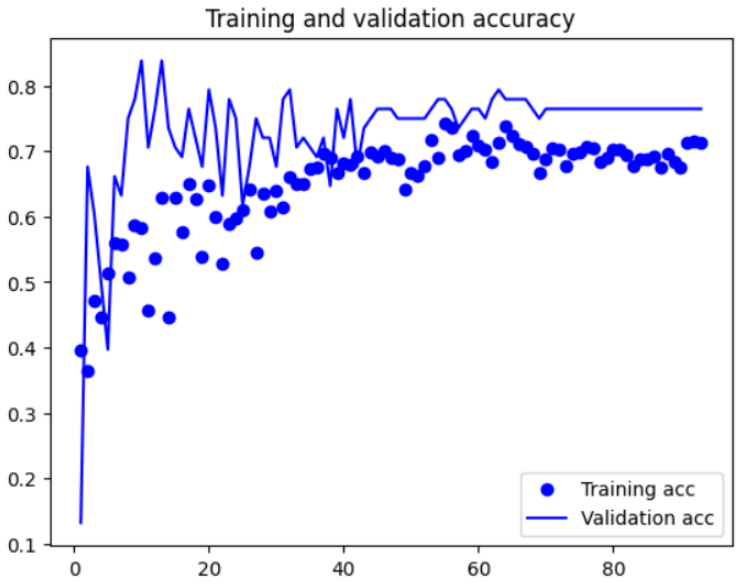

Figura 5–2. Ejemplo gráfica exactitud en entrenamiento.

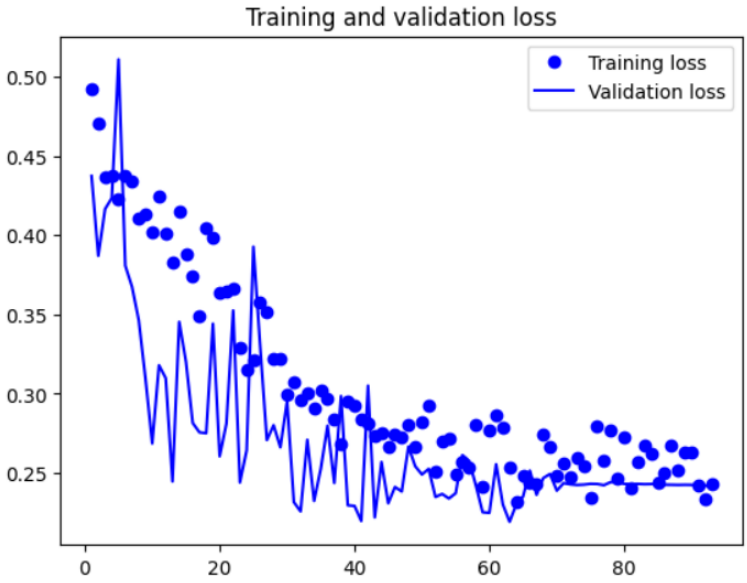

Figura 5–3. Ejemplo gráfica función de coste en entrenamiento.

5.2 Resultados

A continuación, se muestra el código desarrollado para la obtención de resultados con los modelos previamente entrenados. El código es el mismo para todos los modelos, exceptuando el modelo y los pesos de este que se utiliza para realizar las predicciones sobre el subconjunto de test del *dataset*.

La librería principal utilizada para es *Scikit-Learn* [40], librería de código abierto en lenguaje Python enfocada en el análisis de datos.

Librerías utilizadas

```
import matplotlib.pyplot as plt
import numpy as np
from keras import metrics
from keras import callbacks
from keras import losses
from keras import optimizers
from keras import models
from keras import layers
from keras import applications
from google.colab import drive
from collections import Counter
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import ConfusionMatrixDisplay, classification_report
from sklearn.preprocessing import label_binarize
from sklearn.metrics._plot.roc_curve import roc_curve, auc
```

Función curva ROC

```
def ROC(y_pred, y_true):

  fig, ax = plt.subplots(figsize=(8, 6))

  classes = [0,1,2]
  n_classes = ['etapaI', 'etapaII', 'etapaIII']

  y_pred_binarized = label_binarize(y_pred, classes=classes)

  y_true_binarized = label_binarize(y_true, classes=classes)

  fpr = {}
  tpr = {}
  thresh = {}
  roc_auc = dict()

  for i in classes:

    fpr[i], tpr[i], thresh[i] = roc_curve(
        y_true=y_true_binarized[:,i],
        y_score=y_pred_binarized[:,i]
```

```
        )

    roc_auc[i] = auc(fpr[i], tpr[i])

    plt.plot(
        fpr[i],
        tpr[i],
        label='%s (auc=%0.2f)'%(n_classes[i], roc_auc[i])
        )

plt.plot([0,1],[0,1],'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("One-vs-Rest ROC curves")
plt.legend(loc='lower right')
plt.plot()
plt.show()
```

Mediante la función *ROC* que hemos definido, calculamos la Tasa de Falsos Positivos (*FPR*) y la Tasa de Verdaderos Positivos (*TPR*) para cada una de las clases mediante el método *One-vs-Rest*.

Carga dataset y configuración modelo

Como en la implementación del entrenamiento cargamos el dataset y configuramos el modelo de la red, repitiendo los pasos de entrenamiento:

- Montaje de Drive en entorno de ejecución.
- Comandos en entorno de ejecución.
- Tratamiento de imágenes:
    - Solo se importa subconjunto de test.
    - No se hace aumentado de los ejemplos.
    - La opción *shuffle* se marca a *False* para que no se barajen las muestras.
- Cálculo de la partición ponderada del total de imágenes.
- Modelo CNN.
- Capas del modelo a entrenar.
- Capas totalmente conectadas.
- Descenso por gradiente.
- Función de coste.
- Compilación del modelo.

Carga de pesos

```
model.load_weights(ruta_fichero_pesos.h5)
```

Antes de compilar el modelo, se cargan los pesos previamente entrenados.

Evaluación del modelo

```
test_predict = model.predict(test_generator)

test_predict_labels = np.argmax(test_predict, axis=1)
```

Se evalúa el modelo contra el subconjunto de test para que realice sus predicciones y se transforman las probabilidades obtenidas mediante *Softmax* a *One-Hot-Encoded*, donde habrá un 1 en la posición de la clase que haya predicho la red (i.e. [0 1 0]).

Informe por clase

```
report = classification_report(
    y_pred=test_predict_labels,
    y_true=test_generator.labels,
    target_names=['etapaI', 'etapaII', 'etapaIII'],
    digits=4
    )

print(report)
```

La función *classification_report* nos proporciona métricas por cada clase para la evaluación de nuestro modelo: precisión, *recall*, *f1-score* y exactitud.

- y_pred: predicciones del modelo sobre el subconjunto de test.
- y_true: valor real de la clasificación de las muestras del subconjunto de test.
- target_names: nombre de las clases.
- digits: cantidad de decimales a mostrar en los resultados.

	precision	recall	f1-score	support
etapaI	0.9167	0.7857	0.8462	56
etapaII	0.1250	0.1250	0.1250	8
etapaIII	0.2500	0.7500	0.3750	4
accuracy			0.7059	68

Figura 5–4. Métricas por clase.

Matriz de confusión

```
disp_test = ConfusionMatrixDisplay.from_predictions(
    test_generator.labels,
    test_predict_labels,
    labels=[0,1,2],
    display_labels=["etapaI","etapaII","etapaIII"],
    cmap='Blues')
```

La función *from_predictions* genera una matríz de confunsión con las predicciones que haya realizado la red sobre el subconjunto de prueba:

- test_generator.labels: valor real de la clasificación de las muestras del subconjunto de test.
- test_predict_labels: predicciones del modelo sobre el subconjunto de test.
- labels: valor de las etiquetas de las clases.
- display_labels: nombre asociado al valor de las etiquetas.
- cmap: color para la representación de la matriz.

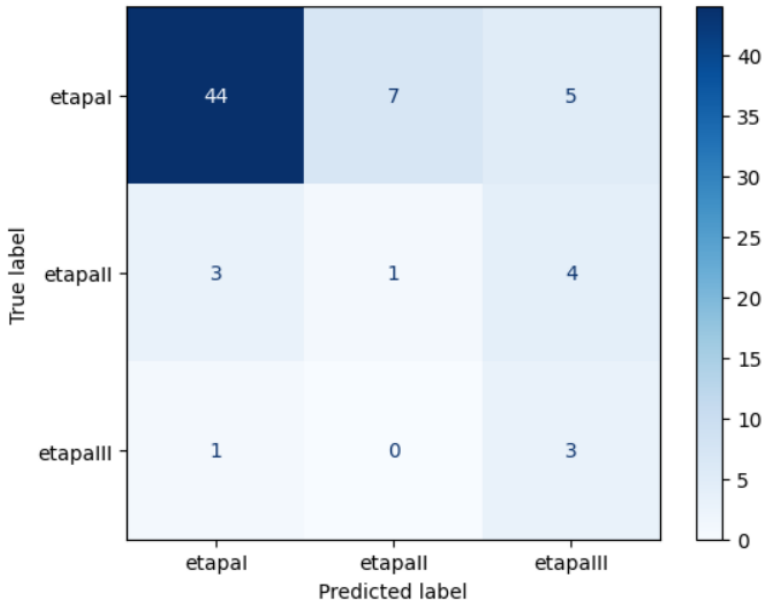

Figura 5–5. Ejemplo matríz de confusión.

Curva ROC y AUC

```
ROC(test_predict_labels, test_generator.labels)
```

A la función previamente definida, debemos de pasarle los valores predichos por el modelo y los valores reales. Esto nos representa una gráfica con las curvas ROC, una por cada clase *One-vs-Rest*, y el AUC para su clase correspondiente.

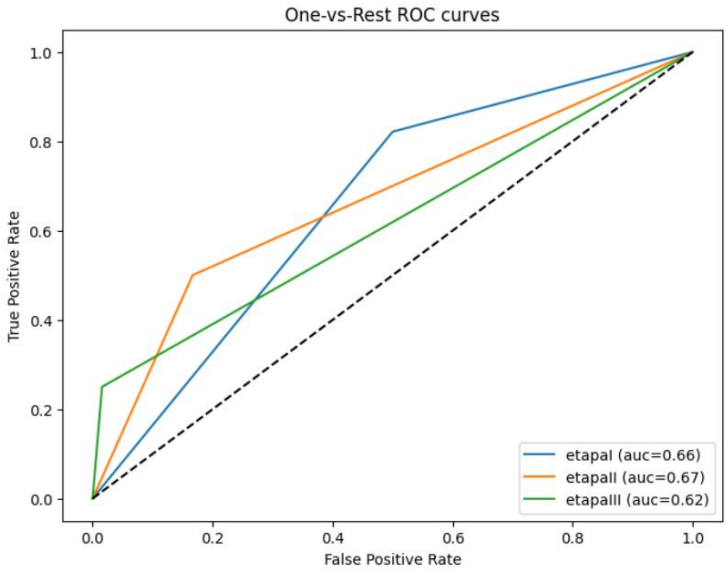

Figura 5–6. Ejemplo curva ROC y AUC.

6 RESULTADOS

CON este apartado se pretende mostrar los orígenes del *dataset* utilizado, así como una introducción a los distintos parámetros de validación que existen para comprobar y analizar la eficiencia de los modelos una vez entrenados. Además, se exponen los resultados obtenidos de los distintos modelos expuestos, una vez entrenados con el *dataset* que hemos conformado.

6.1 Dataset empleado

El *dataset* que hemos conformado para nuestro proyecto consta de muestras pertenecientes a dos *dataset* distintos: *Interactive Atlas of Dermoscopy* y *The International Skin Imaging Collaboration (ISIC)*.

6.1.1 Interactive Atlas of Dermoscopy

Interactive Atlas of Dermoscopy [3] fue desarrollado como una herramienta de formación para la técnica de dermatoscopia.

Este Atlas provee de una introducción general, así como definición de conceptos y diagnósticos de lesiones en la piel, donde se aplica la histopatología a imágenes dermatoscópicas. Además, contiene un CD-ROM con un programa informático en el que se hace análisis de una gran recopilación de muestras de imágenes dermatoscópicas de melanomas.

Estas muestras están categorizadas según distintos criterios, entre los cuales se incluye profundidad, donde podemos distinguir cuatro categorías:

Profundidad (mm)	Ejemplos
In situ	68
< 0,76	109
0,76 – 1,50	56
> 1,50	33

Tabla 6-1. Ejemplos melanoma Interactive Atlas of Dermoscopy.

6.1.2 The International Skin Imaging Collaboration (ISIC)

The International Skin Imaging Collaboration [41] es una organización que se encarga de recopilar y analizar imágenes de lesiones de la piel para su libre distribución.

Esta organización celebra anualmente, a partir de 2016, retos de diagnóstico de muestras de lesiones de piel por medio de técnicas de análisis de imágenes por computación.

Este *dataset* de lesiones de piel contiene múltiples clasificaciones como: benigno-maligno, tipo de melanoma, profundidad del melanoma…

Dentro de la clasificación de profundidad, hemos podido obtener el siguiente número de ejemplos dentro la categorización elegida:

Profundidad (mm)	Ejemplos
< 0,76	385
0,76 – 1,50	23
> 1,50	12

Tabla 6-2. Ejemplos melanoma ISIC.

6.1.3 Dataset proyecto

El *dataset* para este proyecto es producto de la unión de los *dataset*: Atlas e *ISIC*. Hemos unido todos los ejemplos de ambos conjuntos de datos en la categorización de etapas del melanoma, definida por G. Argenziano et al. [12], que comentamos con anterioridad en este mismo proyecto.

El resultado de esta unión se muestra en la siguiente tabla:

Etapas melanoma	Ejemplos
Etapa I	562
Etapa II	79
Etapa III	45

Tabla 6-3. Dataset proyecto.

Debido a la escasez de ejemplos, hemos dividido el *dataset* de forma que tengamos el mayor número de ejemplos para el entrenamiento, sin prescindir de datos para evaluar el rendimiento del modelo. Por lo que lo hemos dividido en los siguientes tres subconjuntos:

Subconjunto	Porcentaje	Ejemplos
Entrenamiento	80%	550
Validación	10%	68
Test	10%	68

Tabla 6-4. Subconjuntos dataset proyecto.

6.1.3.1 Desequilibrio

Como se puede observar a simple vista nos encontramos frente a un problema de *dataset* desbalanceado.

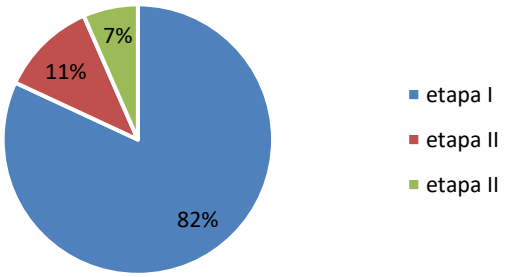

Figura 6–1. Porcentaje ejemplos dataset proyecto.

Para ayudar a combatir esta desventaja hemos empleado la ponderación de la función de cote. La función de coste elegida, *Categorical Cross-Entropy,* estará condicionada por el número de imágenes de cada clase, de forma que la clase con menor número de ejemplos tendrá mayor peso que la clase con más números de ejemplos.

Esto implica un cambio en la ecuación de la función de coste *Categorical Cross-Entropy* (1-13):

$$J(\omega,b)=\frac{1}{m}(\sum_{i=1}^{m}\sum_{k=1}^{k}y_k^i\alpha_k\log(f^k(x^i))+(1-y_k^i)\log(1-f^k(x^i))) \quad (1\text{-}16)$$

Donde:

- α_k : peso de cada clase relativo al número de muestras de esa clase.

6.2 Parámetros de validación

Cada uno de los siguientes parámetros de validación son evaluados para una clase en concreto. Por lo que, cada uno de ellos deberá de ser calculado para cada una de las clases implicadas en la clasificación multiclase.

6.2.1 Verdadero Positivo

Se define verdadero positivo, en inglés *True Positive (TP)* cuando la predicción del modelo coincide con la realidad. Cuando el modelo predice la pertenencia a la misma clase a la que realmente pertenece.

6.2.2 Verdadero Negativo

Se define verdadero negativo, en inglés *True Negative (TN)*, cuando la predicción del modelo coincide con la realidad. Cuando el modelo predice la no pertenencia a una clase a la que realmente no pertenece.

6.2.3 Falso Positivo

Se define como falso positivo, en inglés *False Positive (FP)*, cuando la predicción del modelo no coincide con la realidad. Cuando el modelo predice la pertenencia a una clase distinta a la que realmente pertenece.

6.2.4 Falso Negativo

Se define como falso negativo, en inglés *False Negative (FN)*, cuando la predicción del modelo no coincide con la realidad. Cuando el modelo predice la no pertenencia a una clase a la que realmente si pertenece.

6.2.5 Exactitud

La exactitud, en inglés *Accuracy*, se define como el número de predicciones correctas o acertadas entre el número total de predicciones.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad (1\text{-}17)$$

6.2.6 Precisión

La precisión, en inglés *Precision*, identifica la proporción identificaciones como positivas que realmente fueron correctas. Se define como el número total de verdaderos positivos predichos entre verdaderos positivos y falsos positivos.

$$Precision = \frac{TP}{TP + FP} \qquad (1.18)$$

6.2.7 Sensibilidad

La sensibilidad, en inglés más conocido como *Recall*, o Tasa de Verdaderos Positivos (*True Positive Rate, TPR*), es una métrica que evalúa la capacidad del modelo para predecir correctamente los casos positivos o pertenecientes a una clase específica.

Se define como el número de verdaderos positivos predichos entre el número total de positivos, de manera independiente para cada una de las clases.

$$Recall = TPR = \frac{TP}{TP + FN} \qquad (1\text{-}19)$$

6.2.8 Especificidad

La especificidad, en inglés *Specificity*, o Tasa de Verdaderos Negativos (*True Negative Rate, TNR*), es una métrica que evalúa la capacidad del modelo para predecir correctamente los casos negativos o no pertenecientes a una clase específica.

Se define como el número de verdaderos negativos predicho entre el número total de negativos, de manera independiente para cada una de las clases.

$$TNR = \frac{TN}{TN + FP} \tag{1-20}$$

6.2.9 Tasa de falsos positivos

La Tasa de Falsos Positivos, en inglés *False Positive Rate (FPR)*, o Tasa de Falsa Alarma, hace referencia a los casos en los que el modelo predice una clase a la que realmente no pertenece. Se define como la probabilidad de que el modelo clasifique erróneamente la clase a la que pertenece.

$$1 - Recall = FPR = \frac{FP}{FP + TN} \tag{1-21}$$

6.2.10 F1-score

El valor F1, en inglés *F1-score*, es una métrica que relaciona la precisión con la sensibilidad, de manera que podemos evaluar ambas métricas de manera conjunta.

$$F1 = 2\frac{precision \cdot recall}{precision + recall} \tag{1.22}$$

6.2.11 Matriz de confusión

La matriz de confusión es una métrica que ayuda a evaluar la eficiencia de un modelo de clasificación. Muestra la cantidad de aciertos y fallos que realiza el modelo frente a los datos reales del conjunto de datos utilizados para su evaluación.

La matriz de confusión de un problema de clasificación de n clases tendrá una dimensión de nxn, en la que las columnas se nombran con las predicciones y las filas con la clasificación reales.

La diagonal de dicha matriz representa el número de verdaderos positivos (*TP*) de cada clase, mientras que los demás valores representan el número de falsos positivos (*FP*).

Realidad/Predicción	Clase A	Clase B	Clase C
Clase A	TP	FP	FP
Clase B	FP	TP	FP
Clase C	FP	FP	TP

Tabla 6-5. Matriz de confusión.

6.2.12 Curva ROC

La curva ROC (*Receiver Operating Characteristic Curve*) es una métrica que evalúa la capacidad del modelo para distinguir entre clases que pertenecen a una clase. Muestra la relación que existe entre la tasa de verdaderos positivos (*TPR*), o sensibilidad, y la Tasa de Falsos Positivos (*FPR*), para diferentes umbrales de decisión.

En el eje de ordenadas tendremos la Tasa de Verdaderos Positivo (*TPR*), mientras que en el eje de abscisas tendremos la tasa de falsos positivos (*FPR*). La curva ROC se traza dentro de este gráfico para diferentes umbrales de decisión del modelo. Ésta muestra como a medida que se ajusta el umbral de decisión, cambia la relación entre *TPR* y *FPR*.

Un buen rendimiento de nuestro modelo haría que la curva se sitúe cerca de la esquina superior izquierda del gráfico, lo que significaría una alta *TPR* frente a una baja *FPR*.

En clasificación multiclase se utiliza el método *One-vs-Rest* que se implementa de la manera que:

- Una curva para cada clase.
- Clasificación binaria en la que se evalúa la clase en cuestión frente al resto.

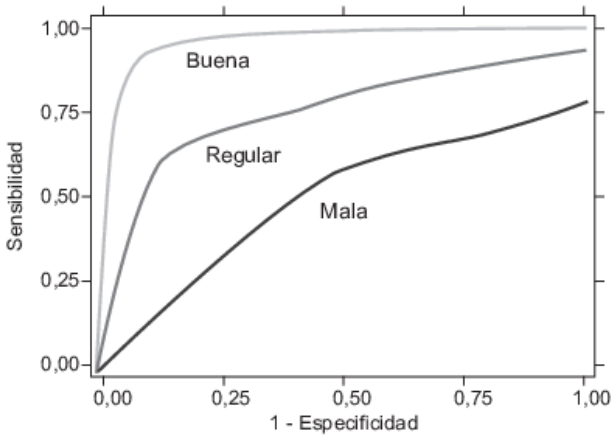

Figura 6–2. Ejemplo curvas ROC [42].

6.2.13 AUC

El AUC (*Area Under the Curve*) es una métrica que se usa para medir la capacidad del modelo basado en la curva ROC.

Representa el área bajo la curva ROC, de manera que un valor de uno indicaría que el modelo clasifica fielmente con forme la realidad.

Muestra una medida de rendimiento que es útil en la comparación entre modelos sobre una misma clasificación.

6.3 Resultados obtenidos

A continuación expondremos la evolución del entrenamiento, como los resultados obtenidos, sobre el subconjunto de test, por los modelos elegidos después de éste: VGG16, VGG19, Resnet50V2 e InceptionV3.

6.3.1 VGG16

El modelo VGG16 con respecto a la exactitud ha conseguido estabilizarse sobre la época 50, aunque en lo relativo a la función de coste no consigue estabilizarse. El valor de los pesos obtenidos da lugar en la época 73, donde la función de coste toma su valor más bajo sobre el subconjunto de validación en el entrenamiento.

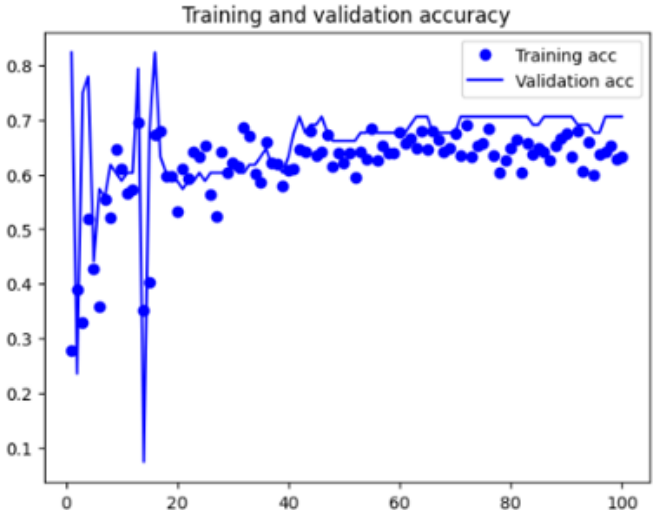

Figura 6–3. Valor exactitud VGG16.

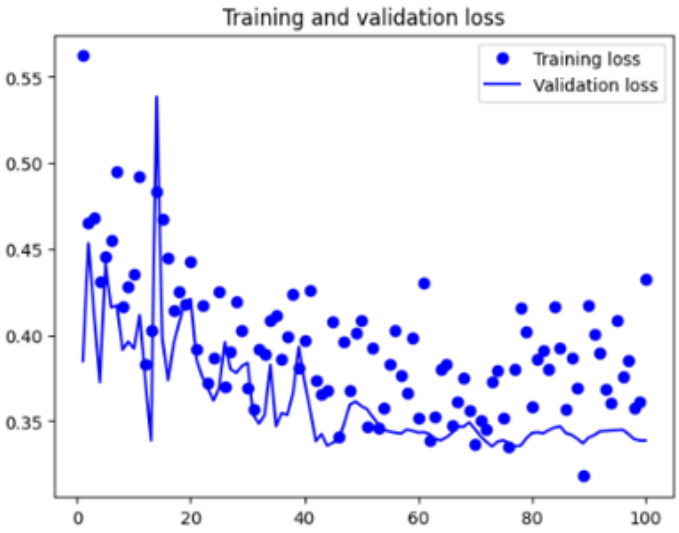

Figura 6–4. Valor función de coste VGG16.

Los resultados obtenidos para las métricas *precision, recall* y *f1-score* son los siguientes:

	Precision	*Recall*	*F1-score*	*Samples*
Etapa I	0.9167	0.7857	0.8462	56
Etapa II	0.1250	0.1250	0.1250	8
Etapa III	0.2500	0.7500	0.3750	4
Accuracy		0.7059		

Tabla 6-6. Resultados métricas VGG16.

La matriz de confusión resultado de las predicciones del modelo sobre el subconjunto de test es la siguiente:

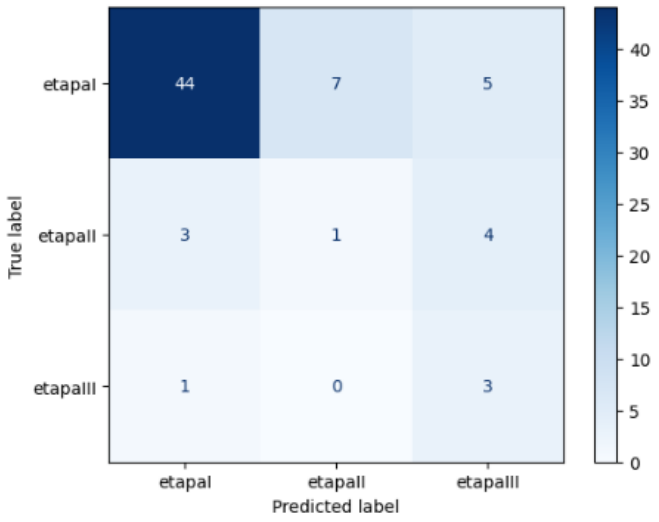

Figura 6–5. Matriz de confusión VGG16.

La curvas ROC se han obtenido mediante el método *One-vs-Rest* para cada una de las etapas. Además, se ha obtenido también el AUC correspondiente para cada una de ellas.

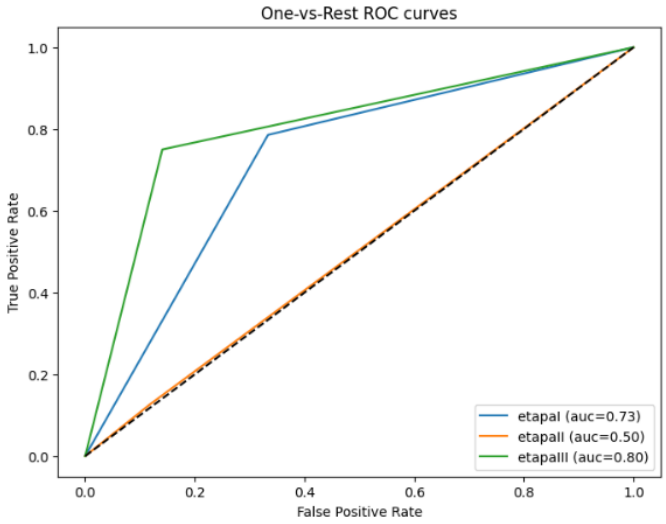

Figura 6–6. Curva ROC y AUC VGG16.

6.3.2 VGG19

El modelo VGG19 con respecto al valor de la exactitud y de la función de coste ha conseguido estabilizarse sobre la época 70. Precisamente en esta época es donde la función de coste toma su valor más bajo sobre el subconjunto de validación en el entrenamiento, por lo que hemos obtenido el valor de los pesos en esta época.

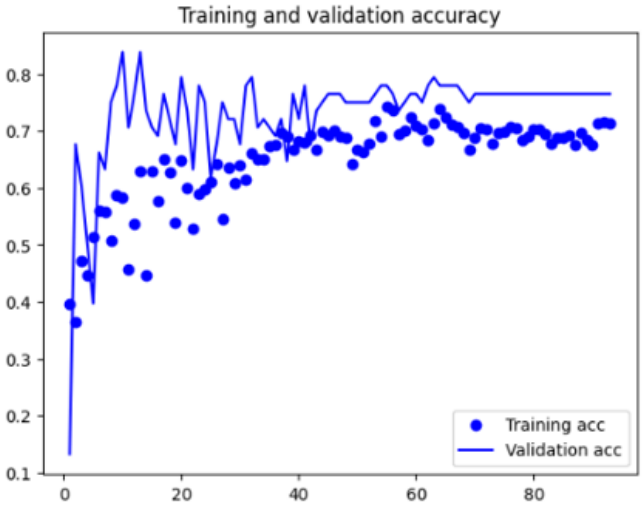

Figura 6–7. Valor exactitud VGG19.

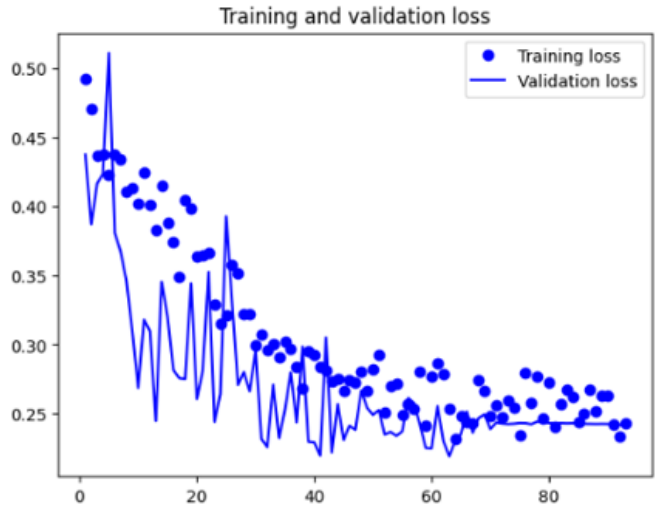

Figura 6–8. Valor función de coste VGG19.

Los resultados obtenidos para las métricas *precision, recall* y *f1-score* son los siguientes:

	Precision	*Recall*	*F1-score*	*Samples*
Etapa I	0.8846	0.8214	0.8519	56
Etapa II	0.2857	0.5000	0.3636	8
Etapa III	0.5000	0.2500	0.3333	4
Accuracy		0.7500		

Tabla 6-7. Resultados métricas VGG19.

La matriz de confusión resultado de las predicciones del modelo sobre el subconjunto de test es la siguiente:

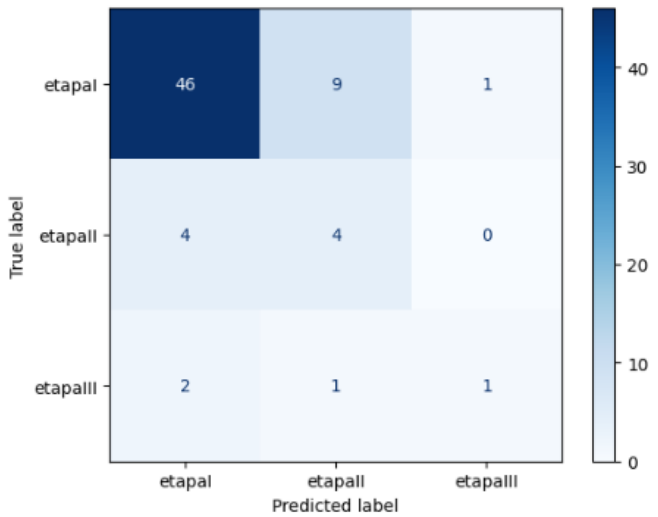

Figura 6–9. Matriz de confusión VGG19.

La curvas ROC se han obtenido mediante el método *One-vs-Rest* para cada una de las etapas. Además, se ha obtenido también el AUC correspondiente para cada una de ellas.

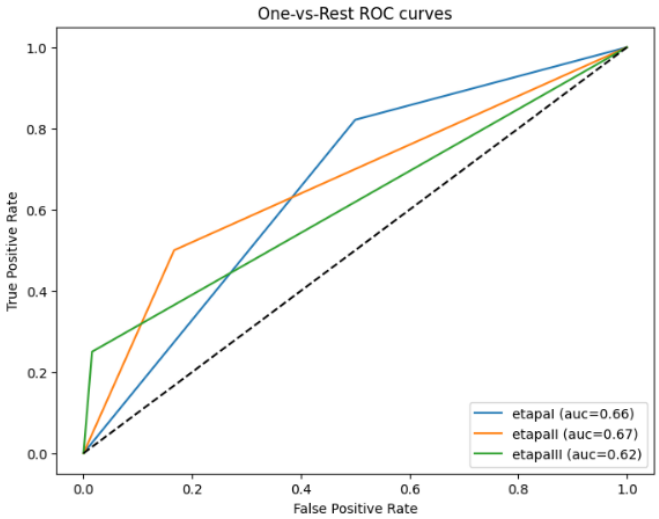

Figura 6–10. Curva ROC y AUC VGG19.

6.3.3 Resnet50V2

El modelo Resnet50V2 no consigue estabilizarse con respecto a los valores de exactitud y función de coste.

Vemos como, a medida que pasan las épocas, la exactitud y la función de coste aumentan y disminuyen su valor respectivamente para el subconjunto de entrenamiento. Mientras que para el subconjunto de validación la exactitud oscila entre 0.70 y 0.77, mientras que el valor de la función de coste oscila entre 0.25 y 0.35.

La función de coste toma su valor más bajo en la época 33, por lo que tomamos el valor de los pesos de la red en este punto.

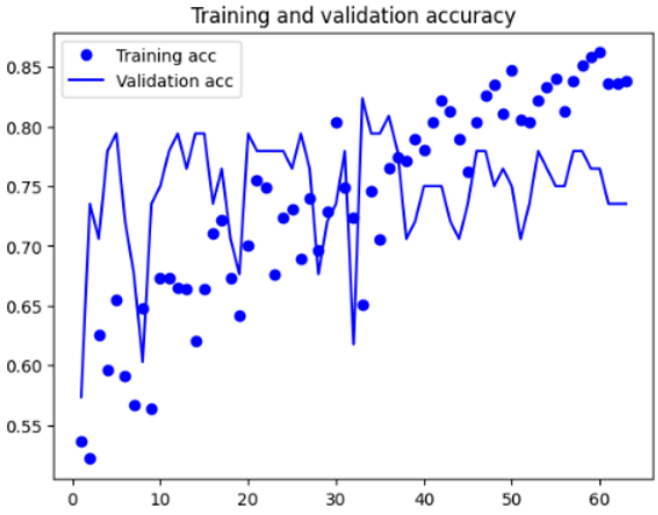

Figura 6–11. Valor exactitud Resnet50V2.

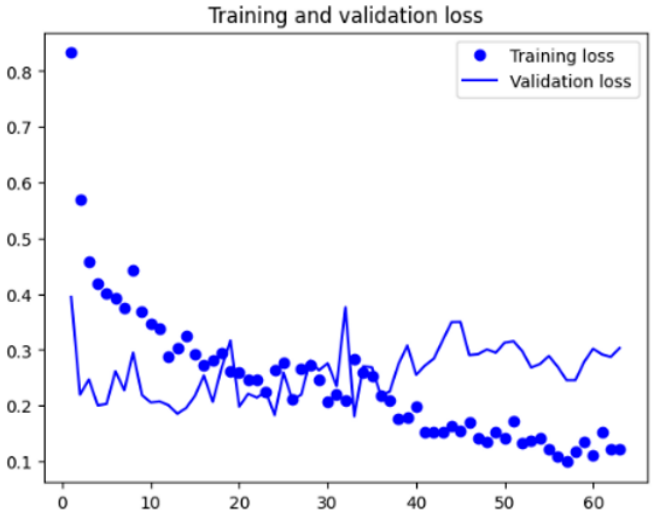

Figura 6–12. Valor función de coste Resnet50V2.

Los resultados obtenidos para las métricas *precision, recall* y *f1-score* son los siguientes:

	Precision	*Recall*	*F1-score*	*Samples*
Etapa I	0.9231	0.8571	0.8889	56
Etapa II	0.2500	0.3750	0.3000	8
Etapa III	0.5000	0.5000	0.4444	4
Accuracy		0.7794		

Tabla 6-8. Resultados métricas Resnet50V2.

La matriz de confusión resultado de las predicciones del modelo sobre el subconjunto de test es la siguiente:

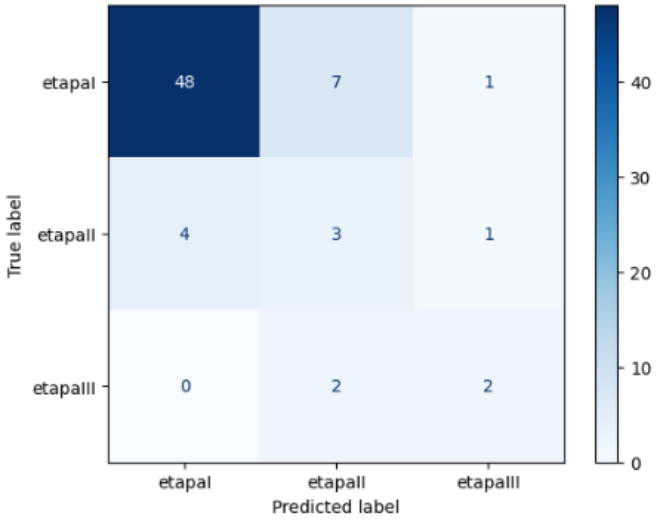

Figura 6–13. Matriz de confusión Resnet50V2.

La curvas ROC se han obtenido mediante el método *One-vs-Rest* para cada una de las etapas. Además, se ha obtenido también el AUC correspondiente para cada una de ellas.

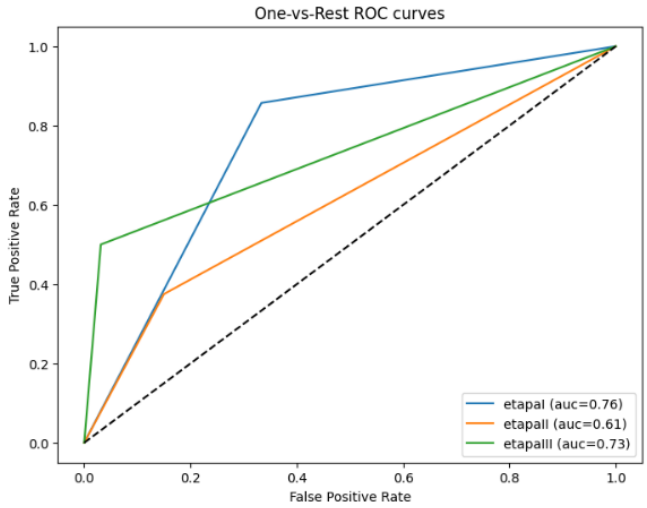

Figura 6–14. Curva ROC y AUC Resnet50V2.

6.3.4 InceptionV3

El modelo InceptionV3 con respecto al valor de la exactitud y de la función de coste en la época 20, por lo que es este el momento de los pesos de la red que hemos usado para realizar nuestras validaciones sobre el modelo.

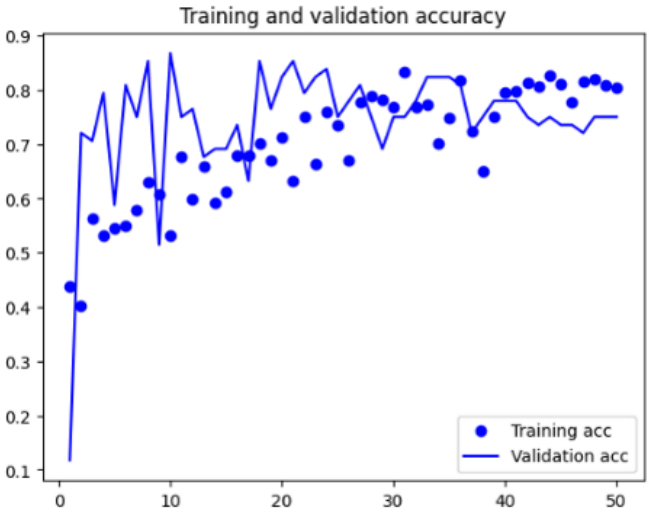

Figura 6–15. Valor exactitud InceptionV3.

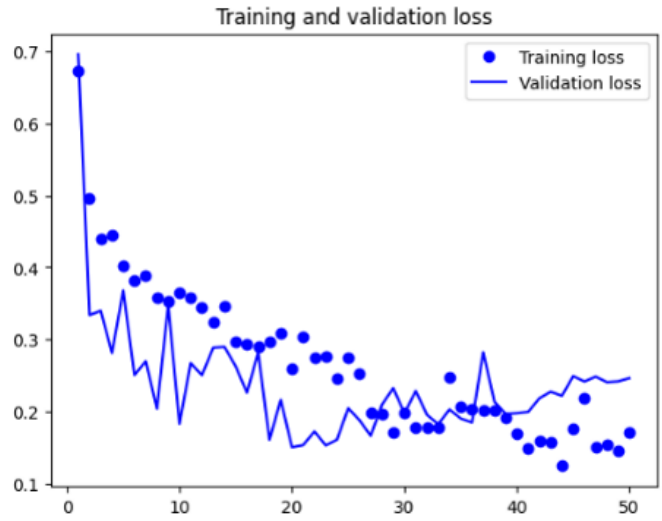

Figura 6–16. Valor función de coste InceptionV3.

Los resultados obtenidos para las métricas *precision, recall* y *f1-score* son los siguientes:

	Precision	*Recall*	*F1-score*	*Samples*
Etapa I	0.9216	0.8393	0.8785	56
Etapa II	0.3333	0.5000	0.4000	8
Etapa III	0.8000	1.0000	0.8889	4
Accuracy		0.8088		

Tabla 6-9. Resultados métricas InceptionV3.

La matriz de confusión resultado de las predicciones del modelo sobre el subconjunto de test es la siguiente:

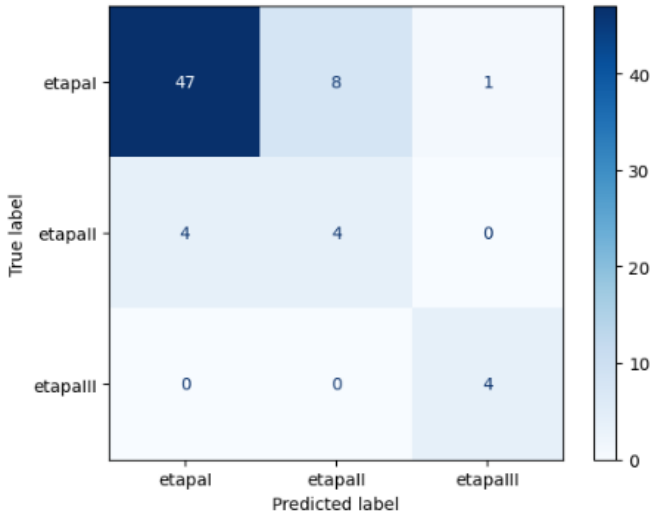

Figura 6–17. Matriz de confusión InceptionV3.

La curvas ROC se han obtenido mediante el método *One-vs-Rest* para cada una de las etapas. Además, se ha obtenido también el AUC correspondiente para cada una de ellas.

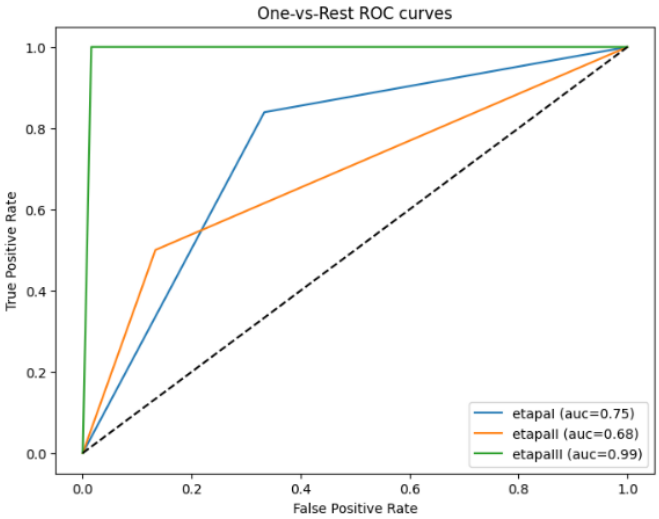

Figura 6–18. Curva ROC y AUC InceptionV3.

7 CONCLUSIONES

La mayor dificultad que afrontar por este proyecto ha sido la escasez de muestras de las distintas etapas del melanoma, que hemos podido minimizar mediante el aumentado de los datos.

Afortunadamente, el número de muestras de lesiones perteneciente a la primera etapa del melanoma es mucho mayor que las demás. Este factor conlleva que la diferenciación entre clases por parte de la red aumente en su dificultad. Hemos conseguido minimizar este problema mediante la adecuación de la función de coste, aunque aun así ha sido el gran limitante para la mejora de resultados.

A través de los resultados, hemos podido llegar a las siguientes conclusiones:

- Las redes coinciden en una mayor capacidad de distinción entre las etapas más distantes, teniendo dificultades y peores resultados en la predicción de la etapa intermedia.
- El tiempo de entrenamiento de las redes más complejas es mucho menor que el de las redes menos profundas.
- A medida que la red aumenta su complejidad, su capacidad de distinción entre las etapas también aumenta, clasificando con mayor precisión las muestras según su etapa.

El problema de *Overfitting*, que hemos comentado anteriormente, se hace ver en el modelo *Resnet50V2*. En los resultados obtenidos, vemos como los resultados de entrenamiento mejoran con las épocas, mientras que los resultados de validación no mejoran. La red se sobre ajusta y extrae características del entrenamiento que no son generales o relevantes para la distinción entre etapas.

Como hemos comentado con anterioridad, el diagnóstico del estado del melanoma es crucial para su temprano tratamiento para así impedir que desarrolle metástasis.

Dado este importante factor, una red que pueda predecir con precisión la etapa más avanzada del melanoma es determinante para designar la mayor urgencia a los pacientes con lesiones pertenecientes a esta etapa.

En nuestros resultados se puede apreciar como *InceptionV3,* además de presentar los mejores resultados, es la red que más se ajusta a este objetivo, identificando con una sensibilidad igual al 100% y una precisión del 80% el melanoma perteneciente a la etapa III, sin ningún caso de Falso Negativo (*FN*).

REFERENCIAS

[1] "Difference between AI, ML and DL | Towards Data Science." https://towardsdatascience.com/understanding-the-difference-between-ai-ml-and-dl-cceb63252a6c (accessed Aug. 31, 2023).

[2] "¿Qué son los cánceres de piel de células basales y de células escamosas?," *American Cancer Society*, Jul. 26, 2019.

[3] G. Argenziano *et al.*, *Interactive atlas of dermoscopy*. Milan, Italy: Edra Medical Publishing & New Media, 2000.

[4] J. S. Zager, V. K. Sondak, and R. Kudchadkar, *Melanoma*. New York, UNITED STATES: Oxford University Press, Incorporated, 2016. [Online]. Available: http://ebookcentral.proquest.com/lib/uses/detail.action?docID=4310816

[5] Jonathan. Bowling, *Diagnostic Dermoscopy The Illustrated Guide*. Hoboken: Wiley, 2011.

[6] W. Stolz *et al.*, "ABCD rule of dermatoscopy: A new practical method for early recognition of malignant melanoma," *European Journal of Dermatology*, vol. 4, no. 7, pp. 521–527, 1994, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0027987427&partnerID=40&md5=5adb30dd1e3f702a9901130e50da13ed

[7] G. Argenziano, G. Fabbrocini, P. Carli, V. De Giorgi, E. Sammarco, and M. Delfino, "Epiluminescence microscopy for the diagnosis of doubtful melanocytic skin lesions: Comparison of the ABCD rule of dermatoscopy and a new 7-point checklist based on pattern analysis," *Arch Dermatol*, vol. 134, no. 12, pp. 1563–1570, 1998, doi: 10.1001/archderm.134.12.1563.

[8] I. Zalaudek *et al.*, "Three-point checklist of dermoscopy: An open internet study," *British Journal of Dermatology*, vol. 154, no. 3, pp. 431–437, 2006, doi: 10.1111/j.1365-2133.2005.06983.x.

[9] S. W. Menzies, C. Ingvar, and W. H. McCarthy, "A sensitivity and specificity analysis of the surface microscopy features of invasive melanoma," *Melanoma Res*, vol. 6, no. 1, pp. 55–62, 1996, doi: 10.1097/00008390-199602000-00008.

[10] J. S. Henning *et al.*, "The CASH (color, architecture, symmetry, and homogeneity) algorithm for dermoscopy," *J Am Acad Dermatol*, vol. 56, no. 1, pp. 45–52, 2007, doi: 10.1016/j.jaad.2006.09.003.

[11] C. Ferrándiz, *Dermatología clínica*, 5ª edición. Barcelona: Elsevier España, 2019.

[12] G. Argenziano, G. Fabbrocini, P. Carli, V. De Giorgi, and M. Delfino, "Clinical and dermatoscopic criteria for the preoperative evaluation of cutaneous melanoma thickness," *J Am Acad Dermatol*, vol. 40, no. 1, 1999, doi: 10.1016/S0190-9622(99)70528-1.

[13] S. Polesie, M. Gillstedt, H. Kittler, C. Rinner, P. Tschandl, and J. Paoli, "Assessment of melanoma thickness based on dermoscopy images: an open, web-based, international, diagnostic study," *Journal of the European Academy of Dermatology and Venereology*, vol. 36, no. 11, pp. 2002 – 2007, 2022, doi: 10.1111/jdv.18436.

[14] P. Rubegni *et al.*, "Evaluation of cutaneous melanoma thickness by digital dermoscopy analysis: A retrospective study," *Melanoma Res*, vol. 20, no. 3, 2010, doi: 10.1097/CMR.0b013e328335a8ff.

[15] A. Sáez, J. Sánchez-Monedero, P. A. Gutiérrez, and C. Hervás-Martínez, "Machine Learning Methods for Binary and Multiclass Classification of Melanoma Thickness From Dermoscopic Images," *IEEE Trans Med Imaging*, vol. 35, no. 4, 2016, doi: 10.1109/TMI.2015.2506270.

[16] C. Hervás-Martínez and F. Martínez-Estudillo, "Logistic regression using covariates obtained by product-unit neural network models," *Pattern Recognit*, vol. 40, no. 1, pp. 52–64, 2007, doi: 10.1016/j.patcog.2006.06.003.

[17] J. Jaworek-Korjakowska, P. Kleczek, and M. Gorgon, "Melanoma Thickness Prediction Based on Convolutional Neural Network With VGG-19 Model Transfer Learning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 2748–2756. doi: 10.1109/CVPRW.2019.00333.

[18] K. Simonyan and A. Zisserman, "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION," 2015, Accessed: Aug. 27, 2023. [Online]. Available: http://www.robots.ox.ac.uk/

[19] J. Bapu Ahire, "The Artificial Neuronal Networks Handbook: Part 4." https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e (accessed Apr. 14, 2023).

[20] J. Casas Roma, *Deep learning*. Editorial UOC, 2020.

[21] E. A. Galindo, J. A. Perdomo, and J. C. Figueroa-García, "Estudio comparativo entre máquinas de soporte vectorial multiclase, redes neuronales artificiales y sistema de inferencia neuro-difuso auto organizado para problemas de clasificación," *Información tecnológica*, vol. 31, pp. 273–286, 2020.

[22] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, vol. 133. Springer, 2004.

[23] F. Sancho Caparrini, "Entrenamiento de Redes Neuronales: mejorando el Gradiente Descendente." http://www.cs.us.es/~fsancho/?e=165 (accessed Aug. 26, 2023).

[24] S. Marsland, *Machine Learning: An Algorithmic Perspective*. in Chapman & Hall/CRC The R Series. CRC Press, 2009. [Online]. Available: https://books.google.es/books?id=x1d0mAEACAAJ

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014, [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[27] D. Calvo, "Red Neuronal Convolucional CNN." https://www.diegocalvo.es/red-neuronal-convolucional/ (accessed Aug. 26, 2023).

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. in Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press, 2016. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2565107&lang=es&site=ehost-live&scope=site

[29] N. Chakrabarty, "A Novel Strategy for Gender Identification from Hand Dorsal Images using Computer Vision," Aug. 2019. doi: 10.1109/ICCMC.2019.8819804.

[30] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/S11263-015-0816-Y.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks", Accessed: Aug. 27, 2023. [Online]. Available: https://github.com/KaimingHe/

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", Accessed: Aug. 27, 2023. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/

[33] E. Qazi, T. Zia, and A. Almorjan, "Deep Learning-Based Digital Image Forgery Detection System," *Applied Sciences*, vol. 12, p. 2851, Aug. 2022, doi: 10.3390/app12062851.

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, "Rethinking the Inception Architecture for Computer Vision".

[35] "Advanced Guide to Inception v3 | Cloud TPU | Google Cloud." https://cloud.google.com/tpu/docs/inception-v3-advanced (accessed Aug. 28, 2023).

[36] "Keras Applications." https://keras.io/api/applications/ (accessed Aug. 27, 2023).

[37] "Colaboratory." https://colab.research.google.com/?hl=es (accessed Aug. 29, 2023).

[38] F. Chollet, *Deep learning with Python*. Shelter Island, N.Y: Manning Publications, 2018.

[39] "API reference - Keras." https://keras.io/api/ (accessed Sep. 01, 2023).

[40] "API Reference — scikit-learn 1.3.0 documentation." https://scikit-learn.org/stable/modules/classes.html (accessed Sep. 01, 2023).

[41] "ISIC | International Skin Imaging Collaboration." https://www.isic-archive.com/ (accessed Aug. 28, 2023).

[42] B. Me and C. Manterola, "Cómo interpretar un artículo sobre pruebas diagnósticas," *Revista Chilena de Cirugia*, vol. 62, pp. 301–308, Aug. 2010, doi: 10.4067/S0718-40262010000300018.

ÍNDICE DE CONCEPTOS

No se encuentran entradas de índice.

GLOSARIO

No se encontraron elementos de tabla de autoridades.