Universitat Oberta de Catalunya

uoc.edu

Clasificación de lesiones dermatológicas a partir de imágenes dermoscópicas mediante el aprendizaje automático

Francisco Jesús Montes Mantero

Máster Universitario en Ciencia de Datos

Trabajo Fin de Máster

Jordi de la Torre Gallart

Ferran Prados Carrasco

Enero de 2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Clasificación de lesiones dermatológicas a partir de imágenes dermoscópicas mediante el aprendizaje automático*
Nombre del autor:	*Francisco Jesús Montes Mantero*
Nombre del consultor/a:	*Jordi de la Torre Gallart*
Nombre del PRA:	*Ferran Prados Carrasco*
Fecha de entrega (mm/aaaa):	01/2022
Titulación::	*Máster Universitario en Ciencia de Datos*
Área del Trabajo Final:	*Trabajo Fin de Máster*
Idioma del trabajo:	*Castellano*
Palabras clave	*dermatología, enfermedad, aprendizaje automático*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Las lesiones dermatológicas tienen una gran incidencia en la población humana. No obstante, sus consecuencias pueden mitigarse si se diagnostican a tiempo. En este sentido, resultaría interesante construir herramientas para profesionales del ámbito sanitario que fueran capaces de ayudar en su detección y diagnóstico precoz.

El objetivo de este trabajo es el de crear un modelo que, utilizando técnicas de aprendizaje automático, desarrolle la capacidad de reconocer y clasificar este tipo de lesiones correctamente a partir de imágenes dermoscópicas de pacientes.

Se ha considerado un conjunto de imágenes dermoscópicas correspondientes a 9 categorías de lesiones dermatológicas bien definidas provenientes del archivo ISIC 2019. Esta colección de imágenes, catalogadas y etiquetadas por dermatólogos profesionales, servirá de base para entrenar un modelo de red neuronal convolucional que las clasificará. En este trabajo, se utilizan exclusivamente modelos neuronales basados en la familia de arquitecturas EfficientNet.

El método general que se ha seguido consta de las siguientes etapas:

1.- Recopilación de información sobre el estado del arte de la clasificación de imágenes usando modelos neuronales.

2.- Obtención de modelos candidatos EfficientNet, entrenándolos utilizando técnicas del estado del arte y recurriendo también a técnicas de ensamblado.

3.- Evaluación de los modelos candidatos en base a la puntuación de la medida objetivo definida en la competición ISIC 2019.

Atendiendo únicamente a la medida objetivo, un modelo EfficientNet ensamblado resultó ser más adecuado que el mejor de sus modelos componentes. Sin embargo esta mejoría no resultó ser absoluta cuando se examinaron métricas más específicas.

Abstract (in English, 250 words or less):

Dermatological lesions have a high incidence in the human population. However, its consequences can be mitigated if they are diagnosed early. In this sense, it would be interesting to build tools for healthcare professionals who are capable of helping in its early detection and diagnosis.

The objective of this work is to create a model that, using machine learning techniques, develops the ability to recognize and classify these types of lesions correctly from dermoscopic images of patients.

A set of dermoscopic images corresponding to 9 categories of well-defined dermatological lesions from the ISIC 2019 archive has been considered. This collection of images, cataloged and labeled by professional dermatologists, will serve as the basis for training a convolutional neural network model that will classify them. In this work, exclusively neural models based on the EfficientNet family of architectures are used.

The general method that has been followed consists of the following stages:

1.- Compilation of information on the state of the art of image classification using neural models.

2.- Obtaining candidate EfficientNet models, training them using state-of-the-art techniques and also resorting to assembly techniques.

3.- Evaluation of the candidate models based on the score of the objective measure defined in the ISIC 2019 competition.

According to the target measure, an assembled EfficientNet model turned out to be more suitable than the best of its component models. However, this improvement did not turn out to be absolute when more specific metrics were examined.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

Como parte de su lucha contra el cáncer de piel, la iniciativa International Skin Imaging Collaboration, ISIC, apuesta por la utilización y procesado de la imagen digital como medio fundamental para ayudar, tanto a profesionales como al público en general, en la detección y el diagnóstico precoz de enfermedades dermatológicas en el ser humano. (ISIC, n.d.)

Tal y como se puede leer en la web citada, uno de los objetivos de esta iniciativa consiste en proponer estándares para mejorar la calidad, la privacidad y la interoperabilidad de la imagen dermoscópica digital. De esta manera, ISIC llevó a cabo la creación de un archivo público de imágenes de calidad controlada. El objetivo principal de este archivo es el de fomentar y facilitar el desarrollo de sistemas de diagnóstico y soporte a la decisión médica basados en la Inteligencia Artificial.

El presente trabajo toma un conjunto de datos extraído de este archivo como punto de partida para construir un clasificador de lesiones dermatológicas utilizando el aprendizaje automático. Este clasificador deberá aportar valor como sistema de soporte a la decisión médica en el campo del diagnóstico de este tipo de lesiones.

1.1.1 Justificación personal

He elegido esta línea concreta de TFM por interés propio principalmente. Pienso que una de las aplicaciones del aprendizaje automático más evidentes es la de asistir a la ciencia en cualquiera de sus ámbitos. En el caso de este trabajo, a la medicina y sus profesionales para beneficio de todos.

1.2 Objetivos del Trabajo

1.2.1 Objetivo principal

- Construir un modelo clasificador capaz de clasificar diversos tipos de lesiones a partir de imágenes dermoscópicas. Deberá detectar y reconocer las siguientes patologías:

    - Melanoma
    - Nevus melanocítico
    - Carcinoma de células basales
    - Queratosis actínica
    - Queratosis benigna (lentigo solar / queratosis seborreica / queratosis liquenoide)
    - Dermatofibroma
    - Lesión vascular
    - Carcinoma espinocelular

1.2.2 Objetivos parciales

- Adquirir conocimiento del estado del arte en lo que se refiere a cómo aplicar técnicas de aprendizaje automático para construir modelos capaces de clasificar imágenes.
- Identificar técnicas complementarias más específicas que mejoren los resultados de un modelo clasificador cuando se aplica al campo de la dermatología y, más concretamente, a la imagen dermoscópica.
- Utilizar para el entrenamiento del clasificador, un conjunto compuesto por 25,331 imágenes dermoscópicas en alta resolución previamente etiquetadas por dermatólogos profesionales provenientes del archivo público de ISIC que se utilizó como base para un desafío público celebrado en 2019. (Tschandl et al., 2018) (Codella et al., 2017) (Combalia et al., 2019)
- Evaluar el modelo clasificador final obtenido, utilizando la misma métrica que la utilizada en la competición ISIC 2019 y utilizarla para comprobar que el modelo resultante posee un grado aceptable en términos de efectividad y generalización de lo aprendido.

1.3 Enfoque y método seguido

En primer lugar se examinará cuál es el estado del arte con respecto al diseño y entrenamiento de clasificadores de imágenes, en concreto aquellos que utilizan

modelos basados en redes neuronales convolucionales. Se buscarán fuentes de información científica, en especial, sobre los últimos avances, mejores prácticas y limitaciones en lo referente a la aplicación de este tipo de clasificadores en el campo de la dermatología.

A partir del conocimiento adquirido se elaborará, a continuación, una o varias estrategias que permitirán implementar unos modelos clasificadores candidatos. Todos ellos utilizarán el conjunto de entrenamiento de ISIC para el desafío de 2019.

Por último se efectuará la evaluación y comparación de todos los modelos candidatos en base a las medidas adecuadas para poder seleccionar el mejor de ellos como el producto final de este trabajo.

Python será el lenguaje de programación a utilizar durante la implementación. La librería escogida para diseñar y entrenar redes neuronales convolucionales será Keras de TensorFlow.

1.4 Planificación del Trabajo

1.4.1 Recursos

La creación y entrenamiento de un modelo basado en el aprendizaje automático suele exigir un coste computacional importante. A menudo se recurre a la utilización de procesadores GPU para acelerar el cálculo y ahorrar así en los tiempos de entrenamiento.

Por esta razón y durante la realización de este trabajo se emplearán las siguientes plataformas en la nube para desarrollar y entrenar modelos predictivos. Dichas plataformas ofrecen capacidad de procesamiento tanto de CPU como de GPU, además de permitir el desarrollo utilizando el lenguaje de programación Python y la librería Keras:

- Google Colab: https://colab.research.google.com/
- Kaggle: https://www.kaggle.com/

1.4.2 Planificación temporal

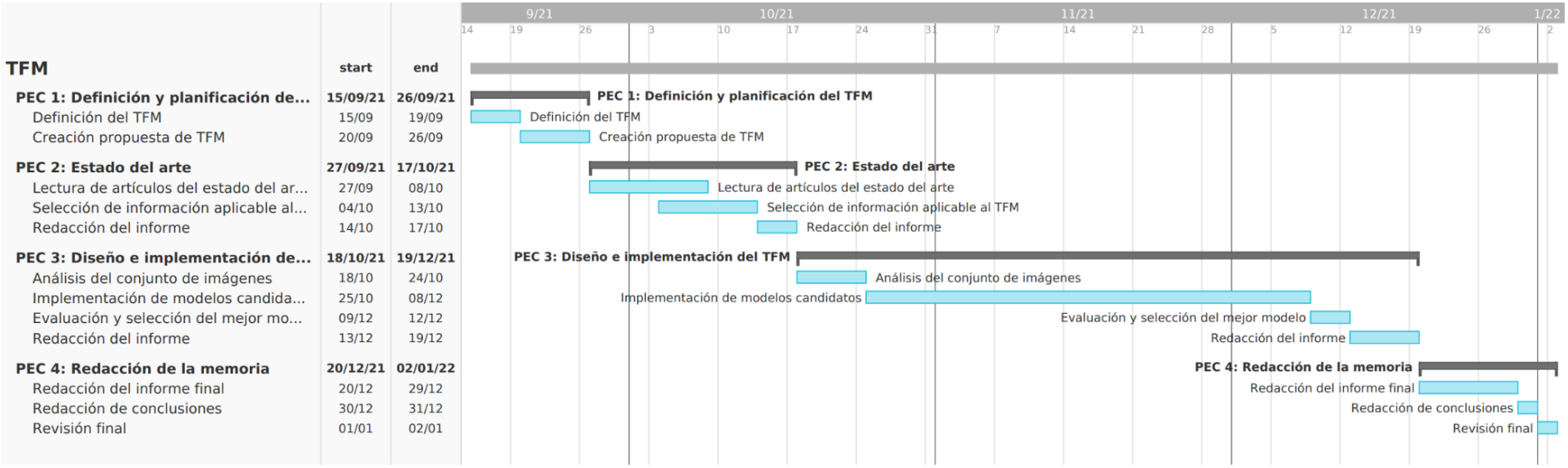

Figura 1.4.2. Planificación temporal

1.5 Breve sumario de productos obtenidos

Un modelo basado en red neuronal capaz de clasificar las imágenes del conjunto de datos ISIC 2019 en las clases que se han definido en el punto sobre los objetivos (1.2).

1.6 Breve descripción de los otros capítulos de la memoria

- Capítulo 2: Estado del arte

  En este capítulo se efectúa una revisión de literatura relacionada con la clasificación automática de imágenes y se intenta explorar sus casos de éxito, problemáticas y diversas técnicas actuales que se han utilizado para resolverlas.

- Capítulo 3: Análisis de los datos de entrada

  La información de entrada se analiza por primera vez en este capítulo y las conclusiones guiarán el diseño y la implementación descrita en los capítulos posteriores.

- Capítulo 4: Análisis y diseño de la solución

  A partir del análisis de la información de entrada y las consideraciones exploradas durante el examen del estado del arte se identifican una serie de especificaciones que la implementación deberá cumplir para conseguir los objetivos.

- Capítulo 5: Implementación de la solución

  Aquí se concretan las especificaciones del diseño y se da solución a otros aspectos importantes que no pudieron preverse durante la fase de diseño.

- Capítulo 6: Experimentos y resultados

  Una vez aplicada la implementación, se describen los resultados obtenidos sobre los modelos candidatos obtenidos que permitirán su evaluación y comparación.

- Capítulo 7: Conclusiones

  En base a los resultados obtenidos se describen una serie de conclusiones finales.

- Capítulo 8: Glosario

  Recoge los acrónimos principales utilizados.

- Capítulo 9: Bibliografía

  Lista de todas las referencias bibliográficas, citadas o consultadas.

2. Estado del arte

2.1 Introducción

La información que se trata en esta sección ha sido obtenida de literatura reciente que explora el problema de la clasificación automática de imágenes.

Dada la naturaleza del presente trabajo, se ha prestado especial atención a aquellas fuentes que tratan más concretamente la clasificación de imágenes en el ámbito de la medicina y particularmente la rama de la dermatología. Los artículos de Adegun y Viriri (2020) y de Bhatt et al. (2020) ofrecen una visión general del estado del arte en este sentido y por tanto han sido un buen punto de partida.

La mayor parte de las fuentes citadas en este trabajo hablan de muy diversas técnicas y estrategias que contribuyen a la obtención de buenos modelos clasificadores de imágenes en medicina y dermatología. Cada una de estas técnicas se encuentra enmarcada dentro de una fase del proceso de construcción del modelo y a menudo este modelo se concibe y presenta dentro del contexto de una competición. Es por ello que se habla de la efectividad de unas o otras técnicas en función de cómo han contribuido a alcanzar un cierto nivel en el ránking de dichas competiciones.

2.2 El aprendizaje profundo y las redes neuronales convolucionales (CNN)

En un primer momento, los sistemas basados en el aprendizaje automático (*Machine Learning* o ML, en inglés) necesitaban de la intervención humana a la hora de extraer un conjunto de características adecuadas de las imágenes que trataban. Estas características servían más tarde como entrada a un modelo clasificador tradicional como, por ejemplo, uno basado en Support Vector Machines (SVM), que se encargaba de predecir la clasificación correcta de la imagen. (Bhatt et al., 2020)

Tal y como se describe en ese mismo artículo, no fue hasta la llegada del aprendizaje profundo (*Deep Learning* o DL, en inglés) que fue posible la completa automatización del proceso de extracción de características con lo que ya no es necesario crear características adecuadas de manera manual durante el proceso de construcción de modelos clasificatorios. Las técnicas de DL proporcionan, por tanto, una solución de principio a fin al problema de la clasificación automática de imágenes. Además, se indica que son las más adecuadas para tratar conjuntos de datos grandes.

Dado su éxito, Bhatt et al. (2020) precisan que en el campo de la clasificación de la imagen médica, los modelos basados en DL se han venido utilizando de manera exponencial desde el año 2014 y en su artículo acaban citando una serie de casos de éxito recientes en la aplicación de modelos DL en imágen médica. Dichos modelos se han aplicado en una variedad de campos como la detección del cáncer de mama o de tumores cerebrales y pulmonares, destacando la utilización de redes convolucionales como base fundamental en ellos.

Por último cabría destacar que son numerosos los estudios que coinciden argumentando que la efectividad de los clasificadores DL de red convolucional supera a la capacidad de los profesionales humanos con lo que se considera el estado del arte actualmente. (Kinyanjui et al., 2019) (Hemavath y Velmuruga, 2020) (Chan et al., 2020) (Bhatt et al., 2020) (Adegun y Viriri, 2020)

2.3 Problemáticas y limitaciones de la clasificación de imágenes con modelos CNN

En esta sección se intenta recopilar los principales problemas encontrados por diversos autores.

2.3.1 Relacionadas con el modelo de clasificación

A pesar de que DL y las redes CNN parecen ser la solución que se utiliza actualmente, cabe destacar que los modelos basados en redes neuronales sufren de ciertas limitaciones (Chan et al., 2020):

- Tienen arquitectura de “caja negra”, es decir, no es posible explicar las reglas internas que utilizan y en las que basan su comportamiento una vez entrenados.
- Para su correcto entrenamiento, necesitan de mucho tiempo e información de entrada.

Debe también tenerse en cuenta el problema del desequilibrio de representatividad entre las clases que se utilizan para entrenar una CNN ya que esto afecta negativamente a su efectividad. (Buda et al., 2018)

Por último, debe tenerse en cuenta que los modelos CNN necesitan que las imágenes de entrada se encuentren en un único tamaño (2D) fijo (Mahbod et al., 2020)

2.3.2 Relacionadas con las imágenes usadas como datos de entrada

Por otro lado, la calidad de las imágenes de entrada afecta a la efectividad de los modelos resultantes. El estudio de Adegun y Viriri (2020) es el que mejor resume las problemáticas principales actuales:

- La gran variabilidad de tamaño y forma de las lesiones.
- La presencia de ruido en las imágenes: pelo, venas, aceites, burbujas y otros elementos.
- Los bordes de las lesiones pueden no estar bien definidos.
- Las imágenes pueden tener bajo contraste dificultando la detección de bordes.
- La iluminación de las imágenes puede tener deficiencias: rayos de luz, reflejos.

Algunos autores como Chan et al. (2020) argumentan, además, que la variedad étnica de los pacientes cuyas imágenes se utilizan para entrenar un modelo afecta negativamente a la efectividad final del modelo debido a que las distintas tonalidades de piel introducen sesgo. Curiosamente, un estudio detallado sobre esta cuestión llevado a cabo por Kinyanjui et al. (2019) no obtuvo resultados concluyentes.

2.4 Soluciones a algunas de las problemáticas actuales

Aquí se citan algunas de las posibles soluciones utilizadas para mitigar o eliminar algunas de las problemáticas expuestas en la sección anterior.

2.4.1 La transferencia de aprendizaje. Redes convolucionales pre-entrenadas.

Para mitigar algunas de las limitaciones de DL y las CNN, es habitual recurrir a la técnica de transferencia de aprendizaje que consigue eficazmente reducir la cantidad de información necesaria para entrenar un modelo y que, además, se aplica de manera creciente a la clasificación de imágenes médicas incluyendo la imagen dermoscópica. (Mahbod et al., 2020)

Siguiendo con su argumentación, Mahbod et al. (2020) indica que los modelos de clasificación que no utilizan transferencia de aprendizaje no se suelen utilizar ya que la cantidad de información de entrenamiento que necesitarían para obtener buenos resultados debería ser alta. Concretamente llegan a citar un ejemplo de modelo

entrenado desde cero con una efectividad aceptable pero que necesitó de al menos 129.000 imágenes de entrenamiento.

Por último, Mahbod et al. (2020) resumen las dos técnicas principales de aplicación de la transferencia de aprendizaje a problemas de clasificación de imágenes dermatológicas:

- Utilización de modelos CNN pre-entrenados como extractores de características fijos cuyas salidas se conectan a las entradas de otros clasificadores (como por ejemplo SVM o perceptrones) para entrenar a estos últimos.
- Utilización de modelos CNN pre-entrenados como extractores de características que se conectan a una serie de capas completamente conectadas (FC layers) concebidas para resolver el problema de clasificación deseado. El modelo completo se ajusta (fine-tune, en inglés) posteriormente al conjunto de entrenamiento deseado.

La transferencia de aprendizaje también ayuda a reducir el tiempo de entrenamiento (Brownlee, 2020). Este autor presenta en su artículo, además, la librería Keras de Tensorflow para Python como una posible API a utilizar para la implementación del clasificador ya que ofrece una colección de distintos modelos muy potentes y que han sido previamente entrenados con el dataset ImageNet. De esta manera, cualquiera de estos modelos podría servir de base para construir un nuevo clasificador más potente ajustado al problema particular que se desee.

2.4.2 La combinación de múltiples modelos. Aprendizaje por conjuntos.

En la literatura consultada, se ha hablado de modelos clasificadores de imágen compuestos por varios sub-modelos CNN entrenados por separado. La unión de todos da como resultado un modelo con mejor efectividad de la que podría darse utilizando cualquiera de sus partes. (Chan et al., 2020)

En el estudio de Adegun y Viriri (2020) puede verse que los modelos clasificatorios que mejor funcionaron utilizando el conjunto de datos de entrenamiento ISIC 2019, el mismo que se utiliza en el presente trabajo, eran modelos compuestos de otros.

Mahbod et al. (2020) también utiliza este tipo de modelos para, no solamente aprovechar la diversidad de funcionamiento de diferentes arquitecturas CNN sino también para poder entrenar el modelo final con varios tamaños de imagen diferentes.

Un caso particular e interesante de este tipo de modelos es el utilizado por Barata et al. (2019) en el que se aprovechan las características jerárquicas de las distintas clases para concebir una estructura de modelos que refleja dicha jerarquía.

2.4.3 La corrección de desequilibrios en el conjunto de entrenamiento

En los artículos de Kinyanjui et al. (2019) y Mahbod et al. (2020) el desequilibrio de clases se utilizan factores de corrección de la tasa de aprendizaje durante el entrenamiento de la CNN que son inversamente proporcionales al grado de representatividad de cada clase. De esta forma el factor permite que se aprenda más de aquellas imágenes correspondientes a las clases menos representadas.

Buda et al. (2018) en su estudio más concreto y exhaustivo sobre los efectos del desequilibrio de clases en el ámbito de las CNN, argumentan que la corrección de la tasa de aprendizaje es sólo uno de los métodos que podrían aplicarse a nivel de clasificador pero también se podría actuar sobre el mismo conjunto de datos utilizando técnicas como oversampling o undersampling.

2.4.4 La utilización de técnicas de mejora de las imágenes

Dada la gran diversidad de operaciones y técnicas de mejora que se pueden aplicar sobre las imágenes de entrada durante la fase de preprocesamiento, Adegun y Viriri (2020) proponen una clasificación a alto nivel:

- Ajuste de contraste para aumentar la visibilidad.
- Ajuste de brillo para aumentar la calidad.
- Ecualización de histograma para incrementar el contraste global de las imágenes.
- Binarización
- Operaciones morfológicas: erosión y dilatación. Para extraer características.
- Aumento de datos
- Estandarización de las imágenes (a través de la reducción de tamaño) para reducir el coste computacional posterior.

Hemavath y Velmuruga (2020) citan algunos métodos de mejora de contraste lineales y no lineales. También, dentro de lo que llaman restauración de la imagen, enumeran algunas técnicas para reducir el ruido.

Resulta interesante mencionar el artículo de Nelson (2020), que habla de ciertas estrategias para tratar el problema de la estandarización del tamaño de las imágenes.

Por último, el estudio comparativo de Hashemi (2019) expone las dos técnicas principales para normalizar el tamaño de las imágenes: recorte o reescalado para acabar justificando la primera opción a partir de resultados experimentales.

2.4.5 La segmentación

Adegun y Viriri (2020) argumentan que la segmentación es una técnica de gran importancia en la clasificación de lesiones dermatológicas y que contribuye a la efectividad de los clasificadores. Se utiliza para separar las zonas enfermas de las sanas en una imagen dermoscópica y citan que los métodos basados en CNN son el estado del arte. Presentan, además, algunos ejemplos de arquitecturas CNN utilizadas para segmentación junto con sus ventajas e inconvenientes: DCNNs, U-Net, Fully Convolutional Network (FCN), Deep Fully Convolutional Residual Network (FCRN) y Convolutional DE-Convolutional Neural Networks (CDCNN).

3. Análisis de los datos de entrada

En una primera etapa y antes de abordar el diseño de una solución, deben estudiarse y analizarse los datos de entrada disponibles.

3.1 Identificación y estructura del conjunto de datos de entrada

Tal y como se estableció en los objetivos de este trabajo el conjunto de datos de entrada disponible se compone de 25,331 imágenes dermoscópicas en alta resolución, etiquetadas por dermatólogos profesionales y provenientes del archivo público de ISIC que se utilizó como base para un desafío público celebrado en 2019.

Este conjunto de datos está igualmente recogido en la plataforma Kaggle y se utiliza como base del presente trabajo. Puede accederse en el siguiente enlace:

https://www.kaggle.com/graf10a/isic-2019

Pueden identificarse los siguientes componentes de información:

- Ficheros de imágen en formato JPG.
- Fichero descriptor y de correspondencia de cada imagen con la clase a la que pertenece. Se trata de un fichero en formato de texto delimitado por comas (CSV).
- Fichero de metadatos que provee información adicional sobre el paciente del que se obtuvo la imagen.

El fichero descriptor representa una tabla en la que cada registro o línea hace referencia a una imagen del conjunto. Consta de los siguientes campos:

- Image: Campo textual. Contiene el nombre de fichero de imagen, sin la extensión.
- MEL: Campo booleano. Determina si el fichero de imagen pertenece a la clase “Melanoma” (valor 1) o no (valor 0).
- NV: Campo booleano. Determina si el fichero de imagen pertenece a la clase “Nevus melanocítico” (valor 1) o no (valor 0).
- BCC: Campo booleano. Determina si el fichero de imagen pertenece a la clase “Carcinoma de células basales” (valor 1) o no (valor 0).
- AK: Campo booleano. Determina si el fichero de imagen pertenece a la clase “Queratosis actínica” (valor 1) o no (valor 0).

- BKL: Campo booleano. Determina si el fichero de imagen pertenece a la clase “Queratosis benigna (lentigo solar / queratosis seborreica / queratosis liquenoide)” (valor 1) o no (valor 0).
- DF: Campo booleano. Determina si el fichero de imagen pertenece a la clase “Dermatofibroma” (valor 1) o no (valor 0).
- VASC: Campo booleano. Determina si el fichero de imagen pertenece a la clase “Lesión vascular” (valor 1) o no (valor 0).
- SCC: Campo booleano. Determina si el fichero de imagen pertenece a la clase “Carcinoma espinocelular” (valor 1) o no (valor 0).
- UNK: Campo booleano. Determina si el fichero de imagen no pertenece a ninguna de las clases anteriores (valor 1) o no (valor 0).

El fichero de metadatos está fuera del ámbito de este trabajo, razón por la que no aparece en este análisis.

3.2 Exploración de las propiedades de las imágenes

Como información básica se extraen una serie de propiedades de las imágenes que serán utilizadas para poder manipular dichas imágenes correctamente en las fases de diseño e implementación. Son las siguientes:

- Rango dinámico. Esto es el rango de valores numéricos puede adoptar cada pixel que componen las imágenes. En todas ellas, este intervalo va desde 0 hasta 255.

- Número de canales que tiene cada imagen. Todas cuentan con tres canales con lo que se tratan de imágenes almacenadas en formato RGB.

- Dimensiones. Categorías de altura y anchura en las que están disponibles, medidas en píxeles. Se informa, además, del número de imágenes que pertenecen a cada categoría y la distribución de clases que existe en cada una. En la figura 3.2 se puede apreciar que, aunque las imágenes están disponibles en muchos formatos de altura y anchura, los dos mayoritarios son:
    - 1024 x 1024
    - 450 x 600

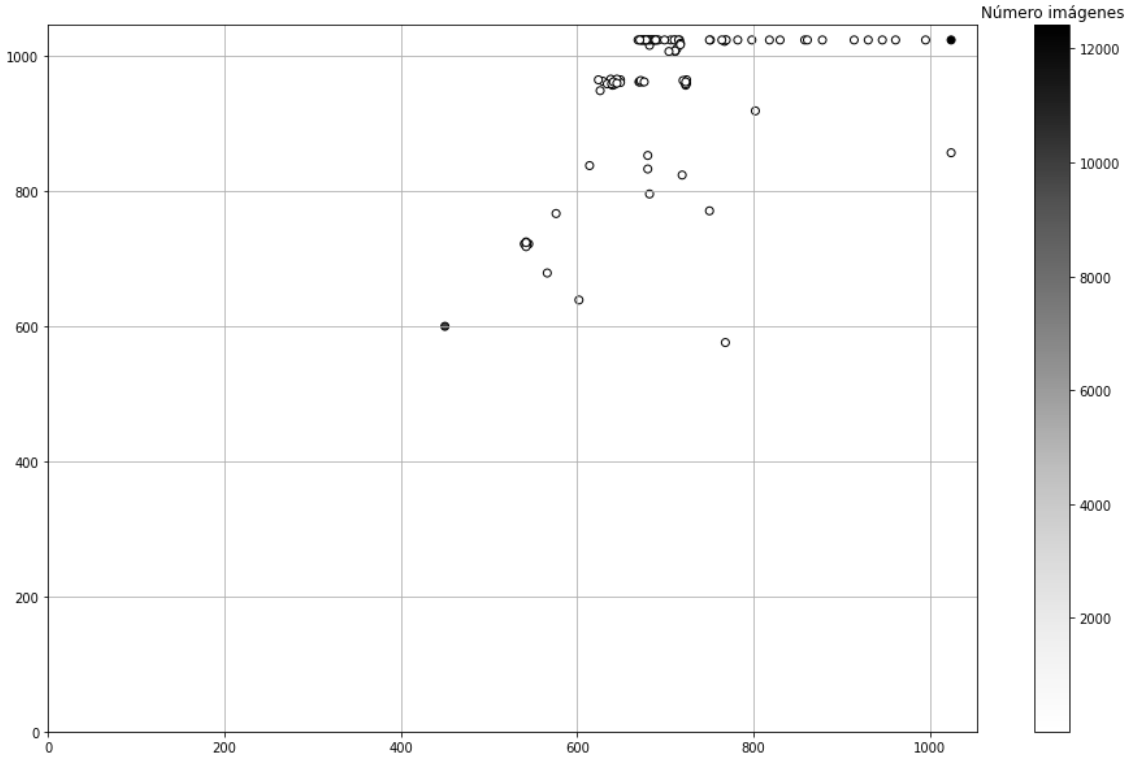

Figura 3.2. Distribución de dimensiones de las imágenes.

3.3 Análisis de la distribución de los datos de entrada

El siguiente gráfico muestra el grado de representatividad de cada clase en el conjunto de imágenes.

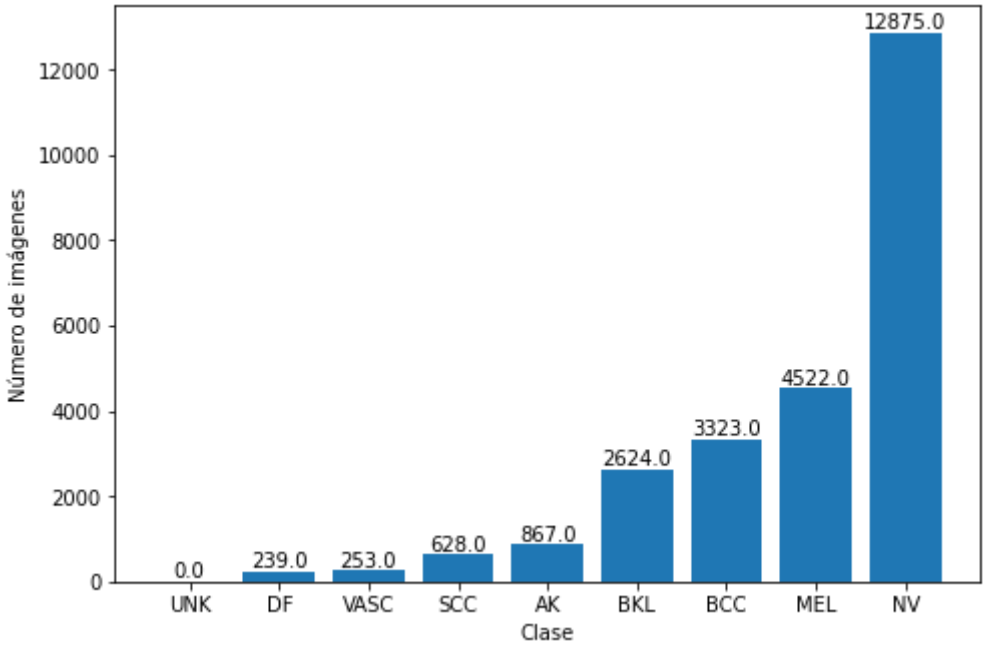

Figura 3.3. Distribución de clases en el conjunto de datos

La distribución obtenida demuestra claramente la existencia de un fuerte desequilibrio en la representatividad de las clases del conjunto de datos. Este hecho deberá tratarse posteriormente tal y como se determinó en el punto 2.4.3 del estado del arte.

Queda patente, además, la ausencia de la clase UNK. Tal y como se explica en la web de ISIC, esta clase representa la ausencia de pertenencia a cualquiera de las otras. Sin embargo, en el ámbito de este trabajo, esta clase no se tendrá en cuenta.

4. Análisis y diseño de la solución

4.1 Introducción

En este capítulo se abordará la especificación de una solución a la clasificación automática de las imágenes analizadas en el capítulo 3 mediante la aplicación de algunas de las técnicas del estado del arte ya exploradas y recogidas en el capítulo 2. En este sentido, ya se ha mencionado que una CNN representa el pilar fundamental sobre el que se basan muchas de las soluciones, consideradas estado del arte, como respuesta al problema descrito.

También se ha comentado ya que son muchas las arquitecturas de red neuronal que se han concebido para el desarrollo de modelos destinados al análisis y clasificación de imágenes bidimensionales. En este trabajo se ha considerado una familia de arquitecturas concreta conocida como EfficientNet, consideradas como uno de los modelos estado del arte en el tratamiento de imágenes en general, y también para el tratamiento de aquellas relacionadas con el ámbito médico en particular. (Mahbod et al., 2020). Por otro lado, consigue resultados similares a las de otras arquitecturas utilizando un número bastante menor de parámetros en comparación. Este hecho facilita su entrenamiento. (Agarwal, 2020)

A continuación se van a examinar algunos aspectos clave que se han tenido en cuenta en el diseño de la solución. Los objetivos principales de este capítulo son los siguientes:

- Redactar una serie de especificaciones que servirán para dirigir la implementación de la solución.

- Identificar una serie de estrategias de aplicación de dicha implementación que permitirá encontrar uno o más modelos clasificadores candidatos. Dichos modelos podrían constituir la solución buscada.

En el capítulo 5, se abordarán los detalles de dicha implementación.

4.2 Aspectos considerados en el diseño

4.2.1 Preprocesamiento de los datos de entrada. Estrategia de aumento de datos.

Para ayudar a mejorar el nivel de generalización obtenido tras sucesivos entrenamientos es muy corriente aplicar algunas transformaciones a las imágenes de entrada de un CNN. Estas transformaciones intentan desensibilizar lo más posible a la red neuronal de factores como la posición u orientación exacta de los distintos elementos o estructuras presentes en las imágenes de entrada. (Brownlee, 2019)

Dentro de los objetivos del presente trabajo y teniendo en cuenta la naturaleza de una imagen dermoscópica, se ha juzgado pertinente la aplicación de las siguientes transformaciones básicas sobre las imágenes:

- Volteos horizontales y verticales
- Cambios de brillo o nivel de luminosidad/oscuridad de la imagen

La clasificación de las distintas lesiones no debería verse alterada si se aplica cualquiera de las transformaciones descritas anteriormente. Por ello, es deseable que la red neuronal aprenda no sólo de las imágenes originales, sino de cualquiera de estas variantes.

Esta técnica es conocida como aumento de datos y es habitual aplicarla a la hora de entrenar un modelo CNN. (Brownlee, 2019)

Por simplicidad, se ha preferido no considerar operaciones que dieran como resultado imágenes con zonas indefinidas, que hubiera que rellenar con un valor neutro artificial, como sería el caso si se realizaran giros aleatorios utilizando ángulos que no fueran múltiplo de 90 grados.

4.2.2 Aplicación de transferencia de aprendizaje

De acuerdo con las razones que se barajaban en el punto 2.4.1 del presente trabajo podemos decir que:

- El conjunto de datos de 25,331 imágenes dermoscópicas que se utilizará para entrenar un modelo clasificador puede no ser suficiente por sí mismo para asegurar que el modelo resultante llegue a alcanzar un grado de generalización

adecuado. En concreto y siguiendo criterios que se desprenden de Mahbod et al. (2020) y Marcelino (2018) se puede considerar que es un conjunto relativamente pequeño de imágenes.

- El tiempo de entrenamiento que sería necesario si solo se utilizara dicho conjunto para entrenar la red neuronal desde cero podría ser muy elevado hasta conseguir buenos resultados.

Por estas razones y, tal y como se describe en Marcelino (2018), el modelo clasificador objeto de este trabajo debería estar basado en un modelo previamente entrenado que sea capaz de extraer las características adecuadas de las imágenes de entrada. Una vez obtenido, se añadirán una serie de capas clasificadoras adicionales que tendrán como objetivo hacer corresponder dichas características extraídas por el modelo de base con cada una de las clases de lesión dermatológica que se está considerando en el presente trabajo y que fueron descritas en el punto 1.2.1. La figura 4.2.2 muestra un esquema de cómo quedaría el modelo descrito.

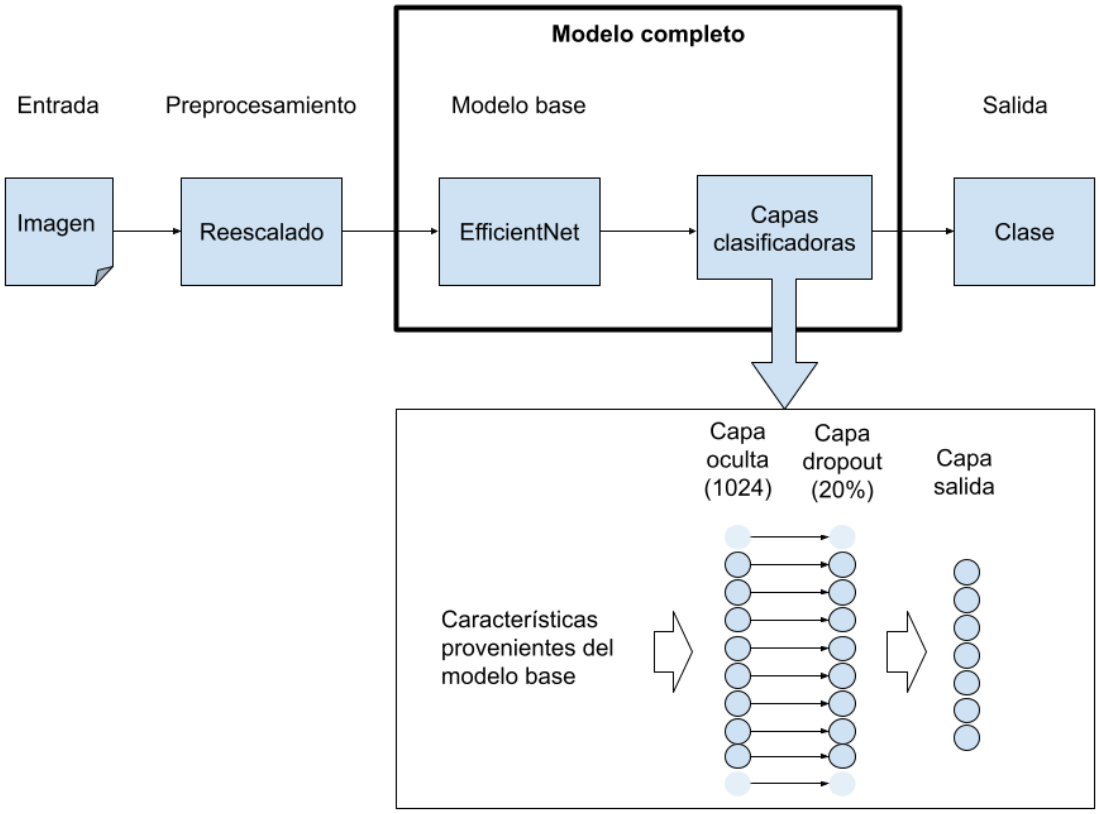

Figura 4.2.2. Modelo con transferencia de aprendizaje

El diseño de las capas clasificadoras será siempre el mismo para todos los modelos que se desarrollen en este trabajo. Este conjunto estará compuesto por una primera capa completamente conectada, seguida de una capa de dropout para controlar el sobreajuste del modelo. Por último, una capa con activación softmax proporcionará un

vector de salida que expresará la probabilidad de que la imagen de entrada pertenezca a una de las 7 clases posibles.

A continuación, siguiendo el planteamiento de Marcelino (2018) debe concretarse una estrategia de entrenamiento que se aplicará sobre el modelo anterior. A este respecto y teniendo en cuenta que el conjunto de datos de ISIC es pequeño, pueden considerarse las dos estrategias posibles siguientes dependiendo del grado de similitud entre las imágenes que fueron utilizadas para entrenar el modelo base y las que componen el conjunto de ISIC:

- **Estrategia 1:** Si la similitud entre ambos conjuntos es alta entonces debería entrenarse solo la parte correspondiente a las capas clasificadoras que se añadieron al modelo base dejando intacto este último.

- **Estrategia 2:** Si el grado de similitud no es tan alto entonces no solamente habría que entrenar las capas clasificadoras sino también una parte del modelo base. Si se confirma este escenario, sería necesario determinar qué capas del modelo base deberían entrenarse también y cuáles no. Las capas que no se entrenan se dice que están congeladas (“frozen”, en inglés).

Llegado a este punto, es evidente la necesidad de evaluar cuál de las dos estrategias debería aplicarse para permitir encontrar una buena solución. Existen una serie de modelos EfficientNet que han sido previamente entrenados con el conjunto de datos ImageNet (Fu, 2020) accesible en https://image-net.org. Estos modelos se utilizarán más tarde en la implementación. Teniendo en cuenta que la naturaleza de las imágenes existentes en Imagenet es bastante diferente a la del conjunto de imágenes dermoscópicas de ISIC 2019 resulta evidente que el grado de similitud entre ambas no es muy alto y, por tanto, sería más adecuada la aplicación de la estrategia 2.

Queda, sin embargo, determinar qué capas el modelo base debería ajustarse o reentrenarse. En este sentido, hay que saber que la arquitectura de cualquier modelo de la familia EfficientNet (desde B0 hasta B7) se compone de 7 bloques (Agarwal, 2020). Cada bloque lo forman una serie de capas cuya composición específica puede diferir según la arquitectura concreta considerada. Sabiendo que existen conexiones entre la primera y la última capa de cada bloque se recomienda que, a la hora de congelar pesos de un modelo base EfficientNet, se haga congelando los pesos de todas las capas que componen un bloque completo. Hacerlo de otra forma afectaría negativamente la efectividad del modelo resultante. (Fu, 2020)

Teniendo en cuenta lo anterior, cabe pensar que existen 7 formas distintas de entrenar un modelo EfficientNet según la estrategia 2 (independientemente de la arquitectura que se utilice). Se ilustra a continuación:

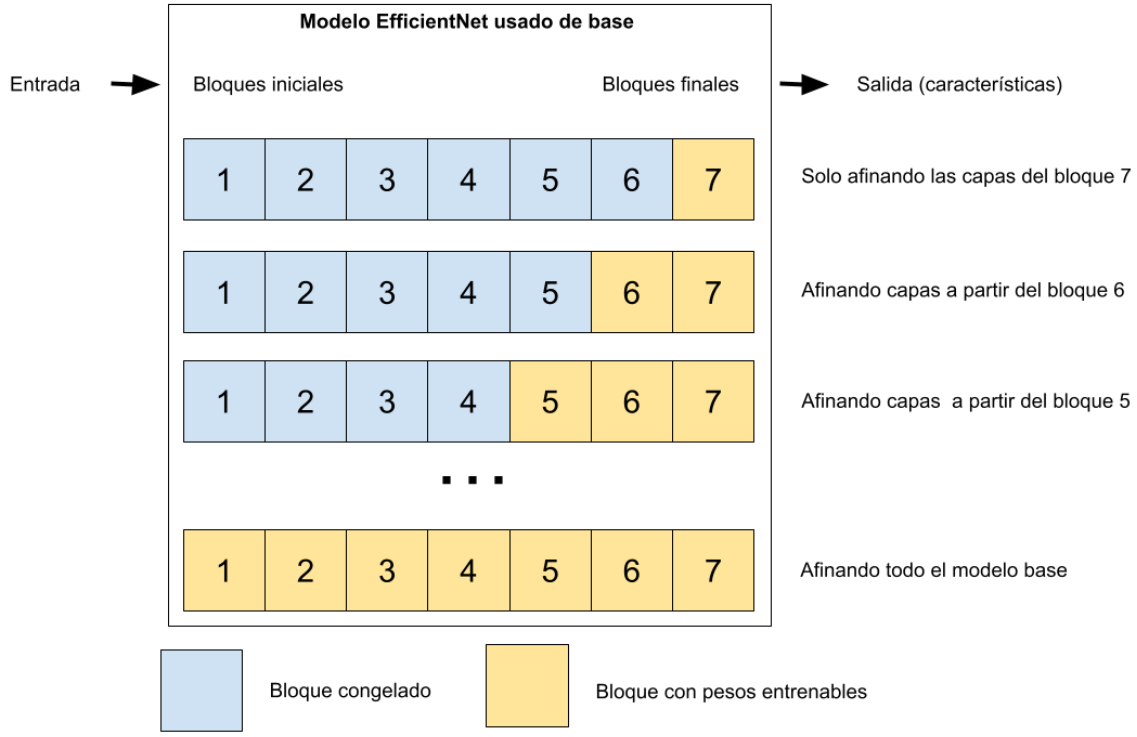

Figura 4.2.2.2 Posibilidades de afinamiento del modelo base

Es por ello que hay que determinar a partir de qué bloque debería entrenarse el modelo base EfficientNet. En este trabajo, se considerará que este factor depende más del grado de similitud (o diferencia) existente entre las imágenes del conjunto ImageNet utilizado para entrenar previamente el modelo base y las imágenes del conjunto de ISIC. Por esto, se probarán todas estas posibilidades de afinación únicamente con la arquitectura B0 y se extrapolará el resultado obtenido como el más adecuado a utilizar con cualquier otra arquitectura de la familia EfficientNet.

4.2.3 Estrategia de entrenamiento

Tal y como se comentó sección anterior, deberá aplicarse un entrenamiento sobre un modelo que constará de dos partes bien diferenciadas:

- Un modelo base previamente entrenado con el conjunto de imágenes ImageNet y que se utilizará como extractor de características de las imágenes de entrada.

- Unas capas clasificadoras que tomarán como entrada las características extraídas por el modelo base para hacerlas corresponder con una de las clases correspondientes a lesiones dermatológicas que se han descrito previamente en el punto 1.2.1.

Para efectuar un correcto entrenamiento de este tipo de modelos se ha decidido seguir la metodología de dos etapas defendida en el artículo de Chollet (2020). Estas dos etapas se describen a continuación:

- **Entrenamiento de las capas clasificadoras:** durante toda esta etapa, los pesos de las capas del modelo base se mantienen inalterados (congelados) y solo se entrenan aquellos de la parte clasificadora.

- **Reentrenamiento (afinamiento o "fine tuning") parcial del modelo base:** en esta segunda etapa se liberan de nuevo los pesos del modelo base y se vuelve a entrenar el modelo completo (es decir, el modelo base conjuntamente con la parte clasificadora). Sin embargo, la tasa de aprendizaje deberá ser muy baja con respecto a la que se utilizó en la primera fase. Esto se hace por dos razones:
    - Adaptar algo más, aunque de manera muy suave, la extracción de características al conjunto de imágenes dermatoscópicas realizada por el modelo base.
    - Preservar al máximo lo ya aprendido por el modelo base.

Según los artículos de Chollet (2020) y Fu (2020), un aspecto especialmente importante a tener en cuenta durante el entrenamiento en la segunda etapa es el de preservar inalterados los pesos utilizados en las capas de normalización del modelo base (batch normalization layers, en inglés). Estas capas tienen pesos internos que ya se ajustaron a los valores de media y varianza de las imágenes utilizadas durante el entrenamiento inicial del modelo base con ImageNet y deberían permanecer congelados durante todo el entrenamiento que se va a efectuar para no destruir lo ya aprendido.

4.2.4 Gestión del grado de desequilibrio en la representatividad de cada clase

Tal y como se detectó durante el análisis de los datos en el punto 3.3, la representatividad de cada clase en el conjunto de imágenes de entrada presenta un fuerte grado de desequilibrio entre la clase más representada (12.875 imágenes) y la menos representada (239 imágenes).

Durante la implementación se deberá tener en cuenta este aspecto para evitar los efectos negativos que este desequilibrio puede suscitar sobre el entrenamiento, la convergencia y el grado de generalización que puede llegar a desarrollar el modelo resultante. (Buda et al., 2018)

En este sentido, Chollet (2019), describe una solución, también citada por Buda et al. (2018) consistente en ajustar el entrenamiento de manera que cada vez que se presenta al modelo un ejemplo correspondiente a una de las clases poco representadas en el conjunto de datos se provoque un cambio mayor en los pesos del modelo que cuando se le presentan ejemplos de clases con una representación mayor. Esto se hace encontrando un factor por cada clase proporcional a su grado de representación en el conjunto de datos. Este factor se utiliza posteriormente durante el cálculo del gradiente para que cada ejemplo provoque un mayor o menor cambio en el modelo.

4.2.5 Gestión del conjunto de datos y evaluación de modelos

La evaluación de los modelos que se entrenan es fundamental para poder llegar a encontrar una solución de clasificador aceptable. Para ello debe tomarse una decisión sobre dos aspectos:

- Diseñar una política de división de los datos de entrada en tres grupos. Uno destinado para entrenar los modelos, otro que servirá durante los entrenamientos para validar si el proceso se está realizando correctamente y uno de test que servirá para obtener una evaluación final de cualquier modelo una vez entrenado.

- Efectuar una selección de métricas que no solamente resulten útiles para evaluar cualquier modelo una vez se haya entrenado, sino que también permita supervisar la evolución del proceso de entrenamiento en sí.

En referencia al primer punto, se van a tomar las siguientes decisiones:

- Del conjunto inicial de imágenes dermoscópicas se seleccionará un conjunto de test, constituído por el 10% del total, es decir, 2533 imágenes. Este conjunto permanecerá fijo y, para servir bien a su propósito, no se utilizará de ninguna manera durante el entrenamiento de ningún modelo. Esto evitará el problema de la filtración de datos (Vashisht, 2020) reservándose así para efectuar la evaluación final de cualquier modelo candidato.

- Durante el entrenamiento de un modelo, se tomará otro 10% del conjunto inicial, 2.533 imágenes, para conformar el conjunto de validación y el 80% restante se utilizará para entrenamiento. Un aspecto importante a destacar es que, a diferencia del conjunto de test que permanece fijo e inalterado, los conjuntos de validación y entrenamiento pueden variar su composición exacta durante cada entrenamiento. Solo su tamaño permanecerá fijo: 2.533 imágenes para validación y 20.265 para entrenamiento.

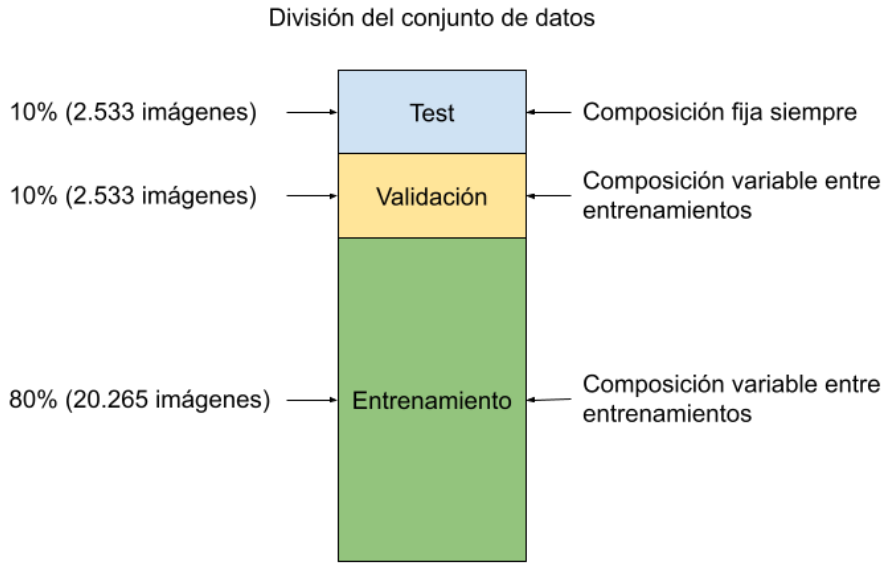

Figura 4.2.3. División de los datos en conjuntos de entrenamiento, validación y test

Para poder medir el grado de adecuación de un modelo candidato en el ámbito del presente trabajo se utilizará la medida de exactitud equilibrada entre múltiples clases (balanced multiclass accuracy, en inglés) definida en las bases de la competición ISIC 2019 y disponible en la siguiente url:

https://challenge2019.isic-archive.com/evaluation.html

Se considerará dicha medida como la más importante a la hora de decidir el grado de efectividad de un modelo. No obstante, se han considerado también las siguientes medidas adicionales para seguir la evolución de un entrenamiento:

- Exactitud (accuracy, en inglés). Medida de uso común aunque no muy útil en una situación de desequilibrio de clases como la que se ha constatado en este trabajo.
- Pérdida (loss, en inglés). Expresa la diferencia entre el resultado obtenido con el clasificador y el resultado verdadero. Da información sobre la evolución de los entrenamientos. Su valor debería descender conforme el modelo aprende.

- Macro precisión (macro precision, en inglés). Esta medida es el resultado de combinar, mediante la media aritmética, la medida de precisión obtenida para cada clase individualmente. (Shmueli, 2019)
- Macro exhaustividad (macro recall, en inglés). Similar a la macro precisión solo que utilizando la medida de la exhaustividad. (Shmueli, 2019)
- Macro F1. El valor F1 es una media armónica entre la precisión y la exhaustividad para cada clase. Es una forma de combinar ambas medidas en una sola de manera equilibrada. Por ello es útil en escenarios donde se le da la misma importancia a ambas. La Macro F1 resulta de la combinación, utilizando la media aritmética, de los valores F1 de cada clase. (Shmueli, 2019)

De igual manera se deberá poder obtener al final del entrenamiento de cualquier modelo:

- Un informe detallando la precisión y exhaustividad (recall) obtenidos por cada una de las posibles clases.
- Una matriz de confusión que permita ver cómo se han clasificado cada imagen de un conjunto y qué errores se han cometido.

4.3 Identificación de estrategias para encontrar modelos candidatos

La implementación una vez finalizada deberá permitir la realización de una serie de acciones encaminadas a encontrar un modelo capaz de dar solución al problema de clasificación de imágenes dermatoscópicas. Estas acciones formarán parte de las estrategias que se detallan a continuación.

Es importante resaltar que la aplicación de estas estrategias deberá hacerse en el orden expuesto ya que cada una depende de los resultados obtenidos con las anteriores.

4.3.1 Establecimiento de afinamiento óptimo de un modelo base EfficientNet

En el punto 4.2.2 se estableció que, antes de intentar construir un modelo de solución definitivo, se deberá evaluar qué partes o bloques de un modelo base EfficientNet deberá afinarse para obtener buenos resultados. Es por esto que, en primer lugar, deberá encontrarse este valor tras la realización de las pruebas necesarias.

4.3.2 Búsqueda de modelos aislados EfficientNet candidatos

De acuerdo con Agarwal (2020), la familia EfficientNet consta de 8 arquitecturas distintas nombradas desde B0 hasta B7 y que van en orden progresivo de complejidad. Las capas de cualquiera de estas arquitecturas están organizadas en 7 bloques distintos.

Por otro lado, el tamaño de imagen de entrada recomendado para cada una de estas arquitecturas varía. En el artículo de Fu (2020) puede apreciarse que este tamaño comienza siendo de 240 x 240 píxeles para una EfficientNet B0 y va creciendo progresivamente hasta llegar a 600 x 600 píxeles en una B7.

Como se puede comprobar, a medida que la complejidad de la red aumenta, también lo hace la cantidad de información que admite en la entrada. Este hecho lleva a pensar que sería interesante probar las distintas arquitecturas en orden creciente hasta llegar a encontrar un modelo clasificador candidato adecuado.

4.3.3 Ensamblando modelos EfficientNet

En el punto 2.4.2 se introdujo el concepto de aprendizaje por conjuntos que consiste en intentar unir distintos modelos con la intención de crear otro de mejor rendimiento. En efecto, la hipótesis principal de este tipo de modelos es que cada uno de los modelos componentes puede cometer errores de manera individual según las particularidades del entrenamiento al que estuvo sometido pero, si se agregan los resultados de varios de ellos, la tendencia debería ser a mejorar la clasificación de manera global al cancelarse o atenuarse los errores entre sí. Esta técnica da lugar a la consideración de una nueva estrategia de búsqueda de modelos que tomará como punto de partida varios de los modelos resultantes de la estrategia anterior.

Se proponen dos tipos de modelos ensamblados que se detallan a continuación:

- **Ensamblado de modelos de arquitectura mixta.** Se trata de combinar varios modelos EfficientNet de distintas arquitecturas una vez entrenados con el conjunto de datos ISIC. La hipótesis principal de esta idea es que los modelos de arquitectura más compleja podrían contribuir a mejorar el resultado final al poder encontrar relaciones más sutiles entre los distintos elementos de las imágenes de entrada mientras que los modelos de arquitecturas más sencillas podrían mejorar la generalización ya que tenderían a sobreajustarse menos a los datos.

- **Ensamblado de modelos de arquitectura única.** Este tipo de modelo ensamblado estará compuesto de submodelos que utilizarán la misma arquitectura EfficientNet. Dicha arquitectura se corresponderá con la del mejor modelo resultante de la aplicación de la estrategia definida en el punto 4.4.2. En este caso, se entrenarán más modelos con esa misma arquitectura para que sirvan como modelos-componente adicionales. La única diferencia entre ellos será que fueron entrenados con conjuntos de validación y entrenamiento obtenidos mediante distintos muestreos.

4.4 Especificaciones para la implementación

Teniendo en cuenta todos los aspectos de diseño abordados en los puntos anteriores, se pueden extraer la siguiente lista de especificaciones con las que deberá cumplir la implementación. En este sentido la solución deberá:

- Poder manejar el conjunto de datos de imágenes inicial de acuerdo a las decisiones detalladas en el punto 4.2.5. En concreto se deberá:
    - Extraer un conjunto de test fijo.
    - Extraer conjuntos de composición variable pero respetando el tamaño que se fijó para los conjuntos de entrenamiento y validación.

- Poder efectuar un procesamiento básico a las imágenes de entrada con el objetivo de insensibilizar al modelo sobre ciertas características de las imágenes como, por ejemplo, la posición exacta de elementos significativos en la imagen o su orientación según los detalles descritos en el punto 4.2.1.

- Posibilitar la construcción de modelos que permitan aprovechar lo ya aprendido por otro que se utilizará como base. La arquitectura de estos modelos deberá estar dividida en dos partes:
    - Un modelo base EfficientNet previamente entrenado con el conjunto de datos de ImageNet y que se utilizará como extractor de características.
    - Una serie de capas que compondrán la parte clasificadora de acuerdo con las características descritas en la figura 4.2.2.

- Permitir realizar un entrenamiento particular de dos etapas sobre un modelo que tenga la arquitectura descrita en el punto anterior y que responde a las características descritas en el punto 4.2.3.

- Permitir elegir las partes del modelo base EfficientNet que deberán ajustarse y cuáles deberán permanecer congeladas durante la segunda etapa de entrenamiento (“fine tuning”). Concretamente podrá elegirse a partir de qué

bloque de la arquitectura EfficientNet que se esté utilizando se podrá efectuar este ajuste, dejando el resto de bloques congelados (ver figura 4.2.2.2).

- Permitir ajustar el entrenamiento del modelo para que las imágenes de clases menos representadas en el conjunto de datos de entrada provoquen un cambio mayor en el modelo y aprenda más de ellas que cuando se le presentan imágenes de clases con mayor representación.

- Permitir el cálculo de las medidas identificadas en el punto 4.2.5 tanto para evaluar un modelo ya entrenado como durante su entrenamiento para posibilitar el seguimiento y efectividad del mismo.

- Permitir guardar un modelo ya entrenado para poder reutilizarlo más adelante. Este paso es crucial para poder permitir el ensamblado de varios modelos para crear uno nuevo más adelante tal y como se describe en el punto 4.3.3.

- Generar los informes identificados también en el punto 4.2.5 de cara a evaluar cualquier modelo una vez entrenado.

5. Implementación de la solución

5.1 Elementos del entorno de trabajo

En este punto se describen los elementos básicos que definen el entorno utilizado para la implementación. En primer lugar, se establece el lenguaje de programación elegido. En segundo lugar, se describe por un lado el entorno de desarrollo que se ha utilizado y por otro, el entorno de explotación donde se han ejecutado realmente los procesos. Se citan, además, las limitaciones que se han encontrado en este último.

5.1.1 Lenguaje de programación y librerías principales

Para poder implementar el modelo clasificador se ha elegido el lenguaje de programación Python. Por otro lado, la librería Keras de Tensorflow v.2.0 ofrece un entorno de desarrollo concebido para facilitar la realización de proyectos relacionados con el aprendizaje profundo.

En particular, Keras permite dar solución a los siguientes puntos clave del diseño que se abordó en el capítulo anterior:

- Acceso a modelos previamente entrenados de la familia EfficientNet para poder utilizarlos como extractores de características.

- Posibilidad de ajustar el proceso de entrenamiento para poder:
    - Flexibilidad para poder implementar a medida el cálculo de cualquier métrica que se necesite.
    - Grabar el estado completo de un modelo durante el entrenamiento para poder reutilizarlo más tarde.

- Facilitar el manejo de cantidades importantes de información visual de entrada para entrenar un modelo neuronal de manera eficiente y a la vez mantener un consumo de memoria adecuado.

Como librería secundaria importante, Scikit Learn (https://scikit-learn.org/) se utiliza para aspectos como el cálculo de las métricas, manejo del conjunto de datos de entrada y la generación de distintos informes de evaluación de modelos una vez entrenados.

5.1.1 Entornos de ejecución y recursos disponibles

Para el desarrollo de la solución se ha utilizado un ordenador personal trabajando con solo una pequeña parte del conjunto de datos original. El objetivo era únicamente asegurar el buen funcionamiento del código ya que esta máquina, al carecer de GPU, no sería adecuada para efectuar entrenamientos de modelos debido a la cantidad considerable de tiempo que sería necesario para completarlos.

Para poder acceder a máquinas con procesadores GPU y acelerar así los entrenamientos de los distintos modelos, se utilizó la plataforma pública gratuita Kaggle. Es en esta plataforma donde se sube el código una vez desarrollado y depurado para su explotación.

Durante la realización de este trabajo Kaggle tenía las siguientes limitaciones principales:

- La ejecución de cualquier proceso en segundo plano estaba limitada a 9 horas. Si el proceso necesitaba de más tiempo, se interrumpía.
- El tiempo máximo de utilización de una GPU estaba limitado entre 37 y 42 horas semanales.

Durante la fase de implementación de este trabajo se descartó el uso de Google Colaboratory ya que esta plataforma no facilitaba la ejecución de procesos en segundo plano (sin necesidad de tener un navegador abierto) y, desde un punto de vista de usuario, no era posible saber cuál era la disponibilidad de procesadores GPU en un momento dado. Esto último imposibilitaba la planificación de las sesiones de entrenamiento necesarias.

5.2 Modularización de la implementación

Para la implementación de la solución se ha adoptado la orientación a objetos como paradigma principal a seguir. Esto permite potenciar la reutilización de las distintas partes del código y facilitar su mantenimiento.

Desde un punto de vista general, el código de la solución puede dividirse en dos grandes categorías de acuerdo con el papel que cumple en la misma:

- **Clases o tareas principales:** Comprende el código principal desarrollado para efectuar una tarea dentro de la solución. Una vez subidos a Kaggle, este código toma la forma de un Jupyter Notebook. Cada uno de ellos contribuye a aportar una parte de la solución. Se tienen los siguientes:
  - Notebook: **"TFM: Clasificador de dermatoscopias (análisis)"**. Efectúa un análisis de los datos de entrada.
  - Notebook: **"TFM: Clasificador de dermatoscopias (modelado)"**. En el diagrama de clases Construcción y entrenamiento de un modelo de red neuronal EfficientNet
  - Notebook: **"TFM: Clasificador de dermatoscopias (ensamblado)"**. Ensamblado de varios modelos de red neuronal EfficientNet previamente entrenados

- **Objetos de apoyo:** El resto del código necesario se encapsula en diversas clases de apoyo para que las clases principales que realizan las tareas se sirvan de ellas a la hora de llevar a cabo su cometido.

En la figura Figura 5.2.1 se muestra el diagrama de clases de la solución completa. Pueden distinguirse las dos categorías de código que se han expuesto y la relación de dependencia entre todas las clases. En este tipo de diagramas, las flechas muestran la dirección de la dependencia, surgen de los objetos que necesita una dependencia y apuntan hacia aquellos de los que dependen.

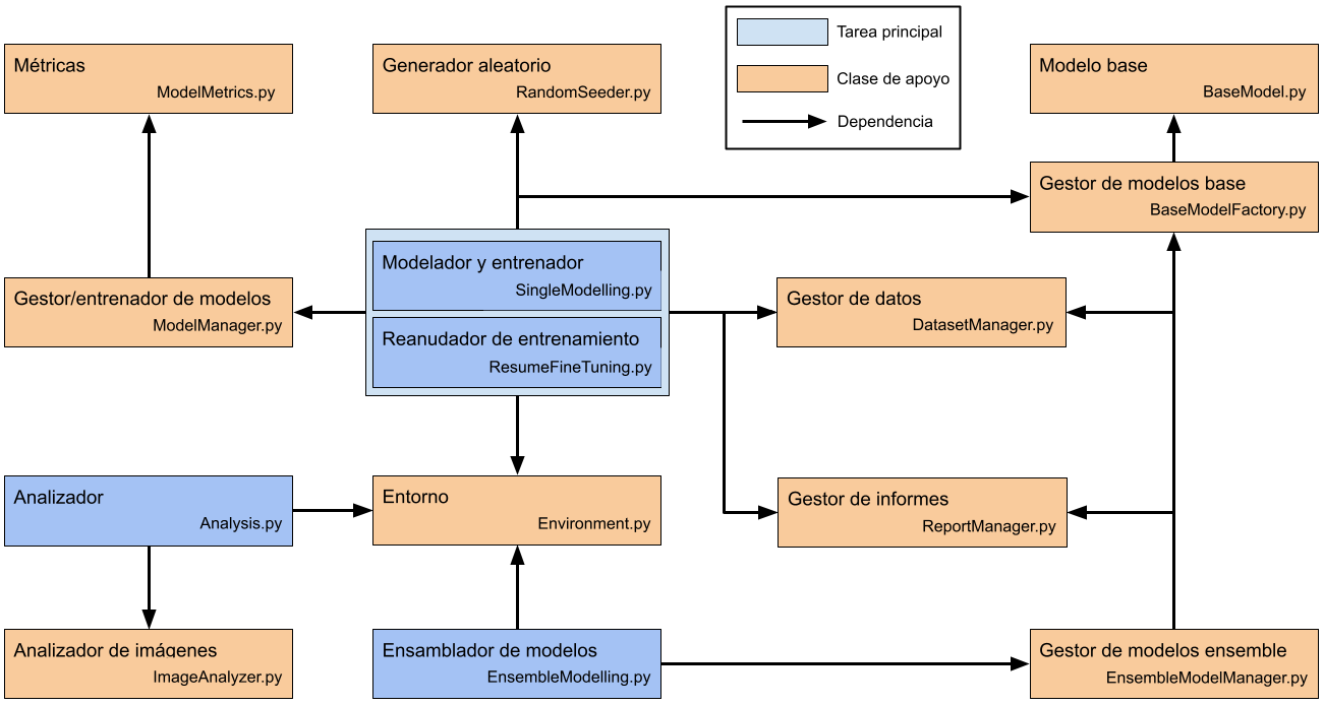

Figura 5.2.1. Diagrama de clases de la implementación completa.

A continuación se describe el papel que cumplen el código de cada una de las tareas principales. En el punto 5.6 se describe más detalladamente la función y responsabilidad de cada uno de los objetos de apoyo.

5.3 Análisis de los datos de entrada

Durante la ejecución de esta tarea se efectúa una exploración de las imágenes que componen el conjunto de datos de ISIC 2019. El objetivo principal es el de generar un informe con la información necesaria para poder efectuar un análisis y diseño de la solución tal y como aparece en el capítulo 4.

Los resultados de este informe se recogen en el capítulo 3.

5.4 Construcción y entrenamiento de un modelo EfficientNet

El objetivo de esta tarea es el de posibilitar la construcción y entrenamiento de un modelo que utiliza transferencia de aprendizaje siguiendo las especificaciones de la figura 4.2.2. Se puede utilizar cualquier arquitectura de la familia EfficientNet como modelo base.

5.4.1 Aspectos de arquitectura

La implementación interna de un modelo EfficientNet facilitada por Keras hace que no sea necesario normalizar el rango dinámico de las imágenes de entrada siendo el propio modelo el que realiza esta normalización internamente. (Fu, 2020)

La arquitectura de la capa clasificadora es siempre la misma y se corresponde con la que se muestra en la figura 4.2.2. De manera más detallada, se compone de:

- Una capa oculta de 1024 neuronas con función de activación RELU.
- Una capa dropout actuando al 20% para contrarrestar los posibles efectos de un sobreajuste.
- Una capa final de salida con 7 neuronas, cada una se corresponde con una de las clases posibles. Para conseguir ésto se utiliza una función de activación Softmax.

5.4.2 Entrenamiento de los modelos

La implementación del proceso de entrenamiento de un modelo sigue las especificaciones expuestas en el punto 4.2.3 y por tanto se ha tenido en cuenta el proceso de dos etapas allí descrito. Durante la segunda etapa de entrenamiento, llamada igualmente etapa de ajuste (o “fine tuning” en inglés), puede especificarse a partir de qué bloque se debe reentrenar el modelo base. Este hecho se ilustra en la figura 4.2.2.2.

Keras hace posible un cierto grado de adaptabilidad del proceso de entrenamiento a través de la utilización de puntos de anclaje de funcionalidad conocidos como “callbacks”. A continuación se detalla cómo se han utilizado estos puntos para cubrir las especificaciones del punto 4.4 en lo referente al proceso de entrenamiento.

En primer lugar, la duración del entrenamiento de cualquier modelo se ha fijado a un máximo de 20 épocas por cada etapa. Esto significa que un modelo podría llegar a entrenarse durante un máximo de 40 épocas. Para ayudar a contrarrestar el posible sobreajuste que esto podría causar, se aplica la técnica de parada prematura (o “early stopping”, en inglés). Esta técnica consiste en detener el entrenamiento de la etapa en curso si no se obtienen mejoras en la métrica objetivo principal (exactitud equilibrada entre múltiples clases) durante un máximo de 5 épocas. De esta manera, si esto ocurre durante la primera etapa de entrenamiento, ésta se detendrá y se pasará a la segunda. Si ocurre durante la segunda etapa, el entrenamiento del modelo llegará a su fin. La especificación de la técnica de parada prematura se declara en Keras mediante un callback específico: **`EarlyStopping`**.

Para el entrenamiento se ha optado por utilizar el optimizador Adam para las dos etapas. La diferencia fundamental entre ambas radica en la tasa de aprendizaje, tal y como se comentaba en el punto 4.2.3. En la primera etapa se utiliza 0.001 y en la segunda 0.0001, es decir, un valor diez veces más bajo.

Las imágenes de entrada se presentan a la red neuronal durante el entrenamiento en bloques de 32. Esto conforma el número de imágenes por lote (o “batch” en inglés). Hay que destacar que el valor de este hiperparámetro tuvo que reducirse a 16 debido a problemas de consumo excesivo de memoria en la plataforma Kaggle solamente durante el entrenamiento de modelos EfficientNet B4.

La función de pérdida de Keras más apropiada ha sido la de “categorical cross entropy” ya que es la que se utiliza cuando la salida del clasificador es un vector codificado utilizando la técnica “one hot encoding” que indica el grado de pertenencia de cada imagen de entrada a una serie de clases predefinidas.

Por otro lado, si durante el proceso de entrenamiento se alcanza un máximo en la métrica objetivo principal, entonces el estado interno del modelo se graba en un fichero. Esto posibilita la recuperación del mejor modelo conseguido al final del entrenamiento lo que permite utilizarlo para componer un modelo ensamblado más tarde. Esto se consigue mediante la especificación de otro callback: **`ModelCheckpoint`**. La extensión/formato de los ficheros de modelo Keras guardados es “hdf5”.

Dado que Kaggle impone una limitación de 9 horas como máximo para el tiempo de ejecución de un proceso cualquiera, es posible que un entrenamiento que utilice uno de los modelos EfficientNet más complejos pueda no acabar antes de ese límite y se interrumpa. Para poder llevar a cabo un entrenamiento que sobrepasa esta limitación es necesario guardar tanto el estado del modelo como las métricas que se han ido registrando al final de cada época en un fichero CSV. Para ello se utilizan dos callbacks distintos: **`BackupAndRestore`** y **`CSVLogger`**. Tal y como se cita en la documentación online de Keras, el callback `BackupAndRestore` efectúa una salvaguarda del modelo, en un directorio especificado, especialmente adecuada para completar un proceso de entrenamiento tras una interrupción.

Teniendo en cuenta los tiempos de entrenamiento que se han observado durante la realización de este trabajo, se puede asumir que, en caso de producirse una interrupción, ocurrirá sin lugar a dudas durante la segunda etapa (o de “fine tuning”). Conociendo este hecho, se ha implementado una tarea o notebook de Kaggle específico para recuperar entrenamientos de fine tuning interrumpidos denominado **“TFM: Continuar entrenamiento interrumpido”**. En estos casos, el propio callback `BackupAndRestore` detecta los ficheros escritos durante un entrenamiento anterior y los carga para restaurar el estado del modelo antes de que se produjera la interrupción.

5.4.3 Tratamiento del desequilibrio

La solución adoptada para el tratamiento del desequilibrio de clases se describió en el punto 4.2.4.

Su implementación es sencilla. En primer lugar, se calcula el grado de representatividad de cada clase utilizando la función `compute_class_weight` de Scikit-learn Esto devolverá un peso distinto para cada clase. Estos pesos deben pasarse posteriormente al método de entrenamiento (`fit`) para tenerlos en cuenta durante el entrenamiento.

5.4.4 Métricas del modelo

El código que se ha utilizado para calcular la métrica de exactitud equilibrada entre múltiples clases es el mismo utilizado durante la fase de evaluación de la competición ISIC 2019. Se ha tomado del repositorio GIT siguiente:

https://github.com/ImageMarkup/isic-challenge-scoring

El cálculo de las métricas se realiza a través de un callback de Keras dedicado y se llama al final de cada época de entrenamiento.

5.5 Ensamblado de modelos EfficientNet

Tal y como se introdujo ya en el punto 4.3.3, la implementación debe permitir la construcción de modelos ensamblados compuestos de una serie de submodelos ya sean de arquitectura distinta entre sí (figura 5.5.1) o compartiendo una misma (figura 5.5.2).

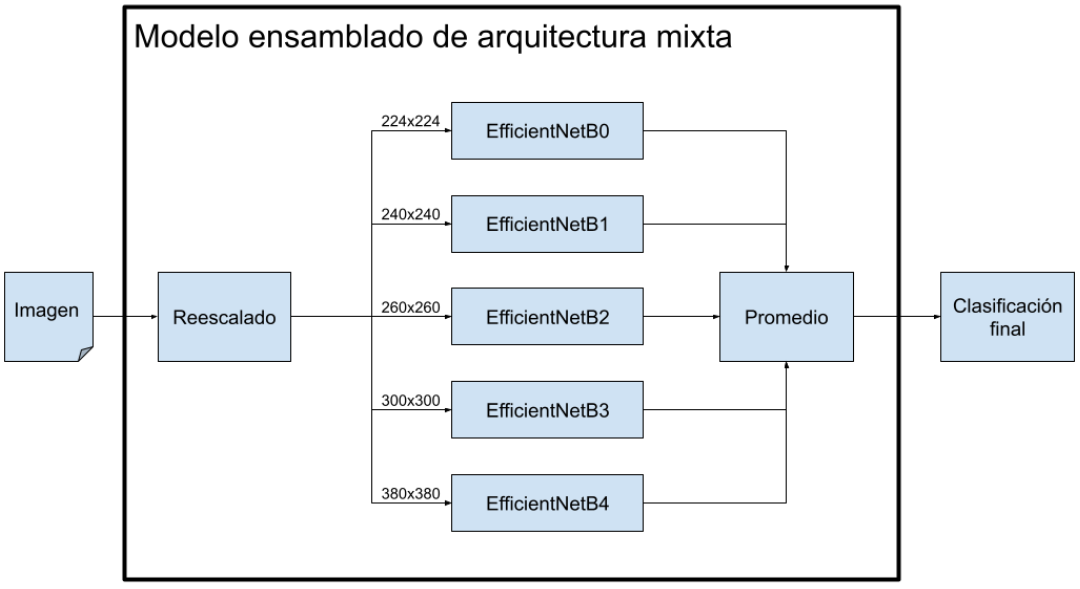

Figura 5.5.1. Diagrama de un modelo ensamblado con mezcla de arquitecturas

Como puede verse, el ensamblado de submodelos tanto de arquitectura mixta como de arquitectura única se realiza en paralelo, es decir, todos ellos toman los mismos datos de entrada y generan un vector de probabilidad de pertenencia a cada una de las clases definidas en el punto 1.2.1. Para unificar la salida de los submodelos, se realiza un promedio de todos estos vectores. Este vector promedio se obtiene

calculando la media de los valores de cada una de las posiciones de los vectores de los submodelos por separado. Este vector resultado se corresponde con la respuesta del modelo ensamblado en su conjunto e informará del grado de pertenencia de la imagen a cada una de las clases. La clase final será aquella que tenga el valor más alto asignado en dicho vector.

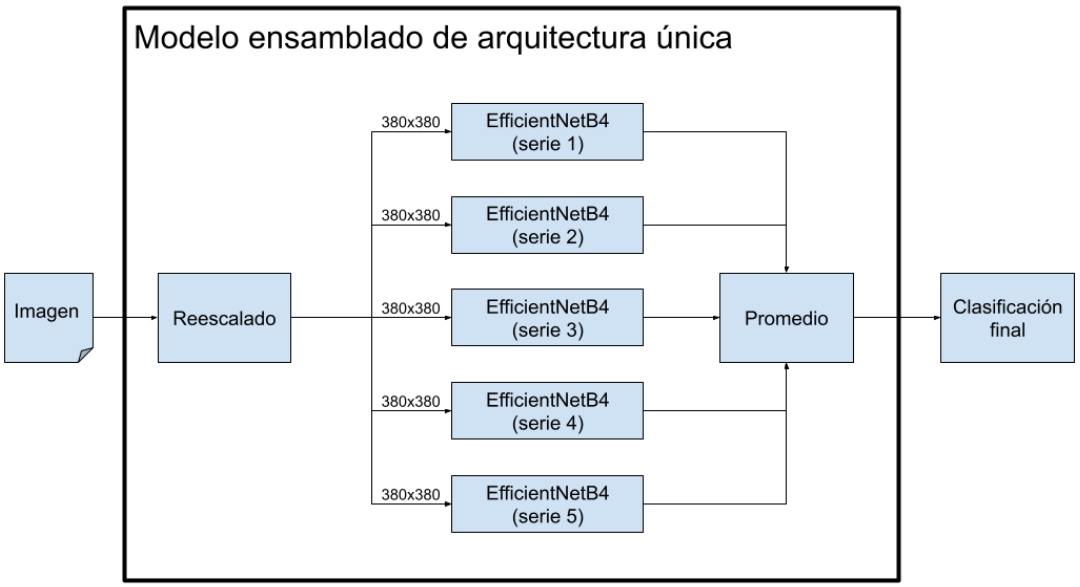

Figura 5.5.2. Diagrama de modelo ensamblado de arquitectura única

La implementación es capaz de crear un modelo ensamblado como los presentados. Tan solo es necesario especificar un directorio donde se encuentren los ficheros correspondientes a los modelos EfficientNet componentes en formato hdf5. Naturalmente, cada uno de estos modelos estará ya entrenado con las imágenes de ISIC 2019.

El nombre de cada uno de estos ficheros debe seguir una nomenclatura concreta. Primeramente el nombre de la arquitectura a la que se corresponde “EfficientNetB0”, “EfficientNetB1”, etc. Seguidamente se añadirá un número natural separado por guión bajo indicando así valor de serie que permite tener varios ficheros correspondientes a modelos de una misma arquitectura en el mismo directorio. Si este número no se encuentra se asumirá un número de serie “1” por defecto. Finalmente deberá tener la extensión “hdf5”. Algunos ejemplos:

- “EfficientNetB3_2.hdf5” - Modelo EfficientNet de arquitectura B3 con un número de serie igual a 2.
- “EfficientNetB0.hdf5” - Modelo EfficientNet de arquitectura B0 con un número de serie (asumido) igual a 1.

Cuando la implementación encuentra ficheros de modelo nombrados de esta forma podrá montarlos en paralelo. Tendrá, además, la información necesaria para crear los iteradores adecuados que prepararán las imágenes de entrada al tamaño recomendado para cada modelo componente de acuerdo con su arquitectura.

5.6 Descripción de los objetos apoyo

Para finalizar el capítulo, se describirán un poco más en detalle cada una de las clases de apoyo. Todas estas clases aparecen en la figura 5.2.1 y como su nombre indica, dan el soporte necesario para que las tareas principales (notebooks en Kaggle) puedan llevarse a cabo.

5.6.1 Métricas

La responsabilidad de esta clase es la de implementar las funciones necesarias para el cálculo de las métricas que se supervisan durante el entrenamiento de un modelo. Siendo las siguientes:

- Exactitud equilibrada entre múltiples clases. Utilizando el código del repositorio GIT que ya se comentó en el punto 5.4.4.
- Macro-precisión
- Macro-exhaustividad
- Macro-F1

5.6.2 Generador aleatorio

Esta clase genera una semilla aleatoria que se utiliza principalmente para preparar el conjunto de datos de entrenamiento y validación durante el proceso de entrenamiento de un modelo cualquiera. Dicha semilla depende de dos factores: de la arquitectura del modelo utilizado y de un número natural considerado un número “de serie”.

De esta forma, se tiene la posibilidad de organizar entrenamientos. Todos ellos utilizarán conjuntos de validación y entrenamiento de distinta composición (aunque siempre respetando el tamaño de cada conjunto).

La figura siguiente muestra este hecho gráficamente:

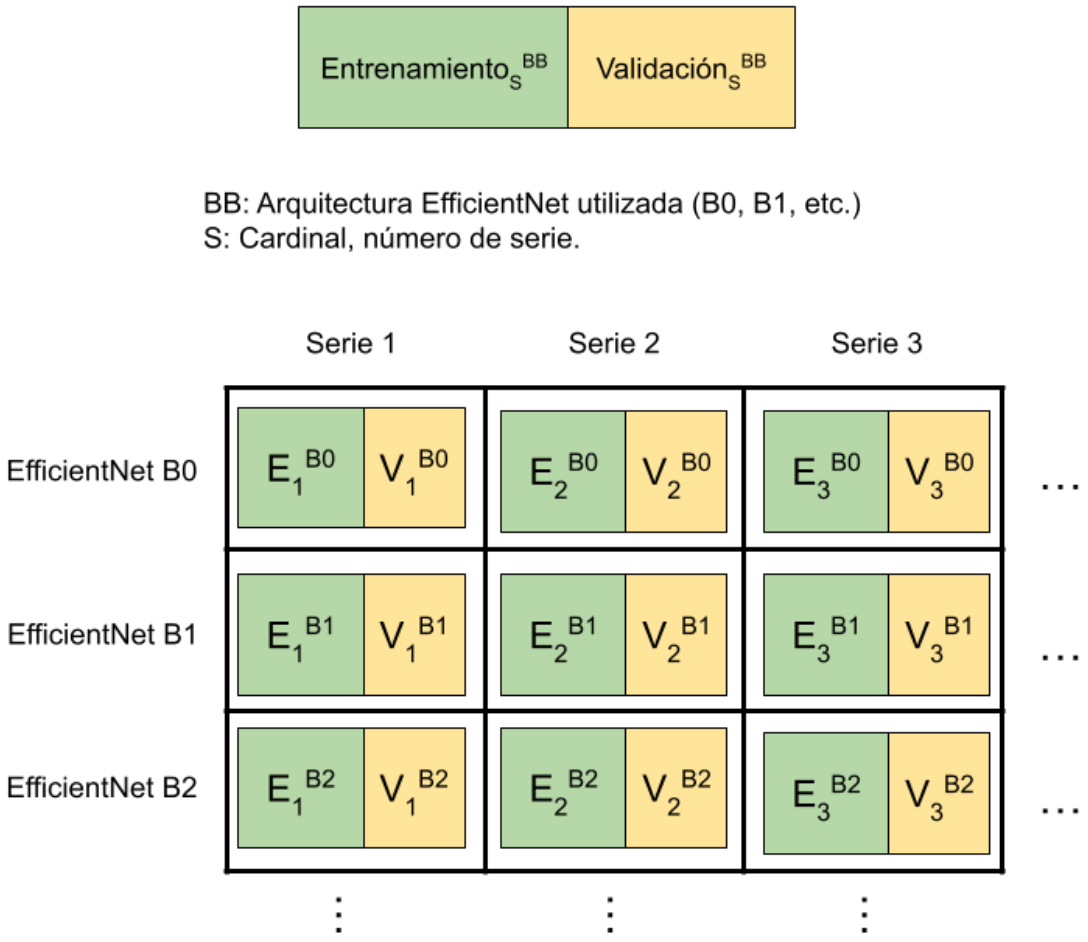

Figura 5.6.2. Organización de entrenamiento de modelos aislados.

5.6.3 Modelo base

Esta clase representa un modelo concreto de la familia EfficientNet que ha sido previamente entrenado con el conjunto de datos ImageNet. Su responsabilidad es la de describir uno de estos modelos en relación a los siguientes puntos clave:

- **Nombre del modelo EfficientNet**: Puede ser EfficientNetB0, EfficientNetB1, etc. hasta EfficientNetB7.
- **Tamaño de imagen y número de canales recomendado:** filas y columnas de la imagen en píxeles y su número de canales. Por ejemplo, para una EfficientNetB0, estos valores son 224 x 224 x 3.
- **Capa inicial de cada bloque**: aquí se especifica en qué capa de la arquitectura del modelo comienza cada uno de los sietes bloques con los que cuenta toda arquitectura EfficientNet. Se utiliza durante la segunda etapa de

entrenamiento ya que es necesario saber a partir de qué capa se pueden afinar los pesos del modelo base.

5.6.4 Gestor de modelos base

La creación de una instancia de Modelo Base es responsabilidad de esta clase ya que encapsula toda la información necesaria sobre cualquier modelo base de la familia EfficientNet.

Tan solo será necesario especificar la arquitectura del modelo base EfficientNet a instanciar (B0, B1, etc.) para que este gestor proporcione un objeto de tipo Modelo Base con la información correspondiente.

5.6.5 Gestor/entrenador de modelos

Como su nombre indica, esta clase es la responsable de dos tareas:

- Instanciar un modelo cuyo diseño se corresponde con el ya comentado en la figura 4.2.2 con un parte correspondiente al modelo base y otra parte formada por capas clasificadoras.

- Efectuar un entrenamiento de dos etapas definido en el punto 4.2.3 sobre un modelo como el mencionado anteriormente.

5.6.6 Gestor de datos

Esta clase encapsula las operaciones necesarias que se pueden realizar con el conjunto de datos de ISIC 2019. Las más importantes son:

- Dividir el dataset en tres conjuntos. Uno para entrenamiento, otro para validación y un último para test. La división se hace de forma que los conjuntos creados tengan la misma proporción de clases que el conjunto completo original. De esta manera se reduce el sesgo durante el entrenamiento y la validación final. Es importante hacer notar que el conjunto de test es siempre el mismo cada vez que se solicita su extracción del conjunto original. Sin embargo los conjuntos de validación y entrenamiento se extraen del conjunto original restante de acuerdo a una semilla aleatoria que debe especificarse.

- Obtener objetos iteradores de imágenes para entrenamiento y validación. Un objeto iterador es responsable de proporcionar las imágenes del conjunto de entrada a un modelo EfficientNet de una arquitectura concreta (B0, B1, etc.) durante su entrenamiento. La presentación de estas imágenes a la red neuronal se realiza por lotes de un tamaño fijo. Por otro lado, el tamaño de las imágenes se adapta al tamaño recomendado para la arquitectura del modelo EfficientNet al que se alimenta. Además, las operaciones definidas en el punto 4.2.1 en relación con la estrategia de aumento de datos a aplicar, también son realizadas por el iterador al vuelo de manera eficiente. Esto es así porque no es necesario tener que escribir las imágenes en disco una vez tratadas y solo se leen en memoria aquellas que forman parte del lote que se esté procesando.

5.6.7 Entorno

Representa un entorno de trabajo. Se diseñó para dar soporte a tres entornos posibles:

- **Local**: representa la máquina de desarrollo
- **Kaggle**: representa la plataforma de explotación Kaggle
- **Google**: representa la plataforma de explotación Google Colab

La responsabilidad principal de esta clase es la de especificar rutas a ficheros y directorios necesarios en cada uno de estos entornos. Por ejemplo, dónde se encuentran las imágenes ISIC o dónde deben guardarse los ficheros de modelo una vez entrenados. Todas las tareas principales dependen de esta clase ya que necesitan tener conocimiento de estas rutas para desempeñar su trabajo.

5.6.8 Gestor de informes

La responsabilidad del gestor de informes es la de generar las gráficas e informes necesarios a partir de los datos que se le proporcionen. En concreto se pueden obtener los siguientes:

- Gráfico de evolución de exactitud (accuracy en inglés) y pérdida a través de las distintas épocas que resulta tras el entrenamiento de un modelo.

- Gráfico de evolución de las métricas definidas durante el entrenamiento:
    - Exactitud equilibrada entre múltiples clases
    - Macro-precisión
    - Macro-exhaustividad
    - Macro-F1

- Matriz de confusión resultante de evaluar el modelo contra un conjunto definido de datos. Por ejemplo, de validación o de test.

- Informe de clasificación que informa de la precisión, exhaustividad y valor F1 por cada una de las clases.

5.6.9 Analizador de imágenes

El analizador efectúa un recorrido exhaustivo de todas las imágenes que conforman el conjunto de datos ISIC 2019, cargándolas una a una y examinando sus propiedades. Durante este recorrido se recoge información sobre:

- El rango dinámico de las imágenes y su número de canales.
- Los distintos grupos de tamaños de imagen encontrados y la distribución de clases dentro de cada uno de estos grupos.
- Explora la estructura del fichero de clases que relaciona cada imagen con su clase.
- La distribución de imágenes de acuerdo con su clase.

También extrae una muestra de 5 imágenes por cada clase y las presenta en pantalla.

5.6.10 Gestor de modelos ensemble

La misión de este gestor es la de crear un modelo ensamblado. Para ello basta con especificar un directorio donde se encuentren una serie de modelos EfficientNet ya entrenados con las imágenes de ISIC y guardados en ficheros hdf5 tal y como se describió en el punto 5.5.

5.7 Código de la implementación

5.7.1 Repositorio principal en GitHub

El código completo de la implementación, debidamente comentado, puede encontrarse en el siguiente repositorio público GitHub: https://github.com/uranio255/tfm

5.7.2 Código desplegado en Kaggle

De manera alternativa, puede también accederse a la versión desplegada del código específica para Kaggle donde las clases principales se han convertido en Jupyter notebooks:

- **"TFM: Clasificador de dermatoscopias (análisis)".** Notebook de análisis de los datos de entrada: https://www.kaggle.com/uranio255/tfm-clasificador-de-dermatoscopias-an-lisis
- **"TFM: Clasificador de dermatoscopias (modelado)"**. Notebook de construcción de modelos EfficientNet individuales: https://www.kaggle.com/uranio255/tfm-clasificador-de-dermatoscopias-modelado
- **"TFM: Continuar entrenamiento interrumpido".** Notebook de reanudación de un entrenamiento EfficientNet interrumpido: https://www.kaggle.com/uranio255/tfm-continuar-entrenamiento-interrumpido
- **"TFM: Clasificador de dermatoscopias (ensamblado)".** Notebook de construcción de modelos EfficientNet ensamblados: https://www.kaggle.com/uranio255/tfm-clasificador-de-dermatoscopias-ensamblado

Los notebooks necesitan código y datos para funcionar. Estos elementos deben hacerse disponibles en Kaggle en forma de datasets. De esta manera, se han creado los siguientes:

- Las clases de apoyo utilizadas por cualquiera de los notebooks se ha desplegado en un dataset disponible en: https://www.kaggle.com/uranio255/code-tfm-classes

- Los datos de entrada para el notebook que construye modelos de EfficientNet ensamblados ha sido:
  https://www.kaggle.com/uranio255/tfm-input-modelo-ensamblado

- Los datos de entrada para el notebook que reanuda entrenamientos interrumpidos ha sido:
  https://www.kaggle.com/uranio255/interrupted-efficientnetb4-training-data-files

6. Experimentos y resultados

En este capítulo se presentan los resultados que surgieron de los distintos modelos candidatos que se construyeron y entrenaron para dar respuesta a la estrategias de búsqueda de modelos candidatos consideradas en este trabajo y que ya se comentaron en el punto 4.3 dentro del capítulo de análisis y diseño de la solución.

6.1 Métrica objetivo

Un hecho importante que se comprobó empíricamente fue que la métrica objetivo de la competición ISIC 2019 resultó ser la misma que la exhaustividad media de todas las clases (o macro recall, en inglés). Hay que recordar que esta medida es igual a la media aritmética de los valores de exhaustividad calculados para cada clase.

Es por esto por lo que en este capítulo, dedicado a la presentación de los resultados, se hace referencia a macro recall como la medida más importante a la hora de comparar modelos, decir cuál es el mejor y tomar decisiones.

6.2 Afinamiento del modelo base

Antes de comenzar la búsqueda de modelos candidatos es importante establecer a partir de qué bloque debe reentrenarse un modelo EfficientNet con las imágenes de ISIC 2019 para conseguir buenos resultados. Este hecho ya se abordó en el punto 4.3.1. Este dato es necesario a la hora de aplicar el afinamiento del modelo base en la segunda etapa de cualquier entrenamiento.

Para ello se decidió entrenar una arquitectura EfficientNet B0 con todas las imágenes de ISIC 2019, reservando un 20% del conjunto original para validación y el 80% restante para entrenamiento. Dicho entrenamiento se efectuó de manera idéntica en la primera etapa (capas de clasificación) pero se utilizó un bloque de comienzo distinto para efectuar el afinamiento del modelo base en la segunda etapa. La tabla 6.2 muestra los resultados de los entrenamientos que se efectuaron. Los valores de bloque de fine tuning van desde 1 hasta 7. Un valor de 1 realiza un afinamiento de todos los bloques convolucionales del modelo base mientras que un valor de 7 reentrena el último bloque convolucional únicamente (aquel más cercano a la salida).

Arquitectura	Bloque fine tuning	Macro Recall validación (etapa 1)	Macro Recall validación (etapa 2)
B0	7	0,508560	0,656570
B0	6	0,523080	0,701530
B0	5	0,550580	0,729730
B0	4	0,529290	0,749640
B0	3	0,506980	0,721220
B0	2	0,518730	**0,765880**
B0	1	0,525280	0,717950

Tabla 6.2. Resultados de afinación de modelo base con EfficientNet B0

En todos los entrenamientos que se realizaron puede apreciarse la mejora cualitativa del clasificador desde la primera etapa hasta que el modelo base se afina en la segunda etapa tras su evaluación contra el conjunto de validación. Sin embargo, solo se obtuvo el mejor resultado tras la afinación cuando el modelo base se reentrenó con las imágenes de ISIC 2019 a partir del bloque 2.

Este experimento solo se efectuó con la arquitectura EfficientNet B0. En principio, hubiera debido repetirse con cada una de las otras arquitecturas más complejas para así haber encontrado el bloque más adecuado para cada una, sin embargo, se decidió que estos resultados podrían extrapolarse a cualquier otra arquitectura más allá de la B0. Para justificarlo, se asumió que el bloque a partir del cual se debe reentrenar depende más del grado de similitud entre las imágenes ISIC 2019 y el conjunto de datos ImageNet (con el que se entrenó el modelo base originalmente) (Marcelino, 2018) que de la arquitectura concreta EfficientNet que se esté utilizando.

De esta manera resultó que los modelos base EfficientNet debería afinarse a partir del bloque 2 independientemente de la arquitectura concreta que se utilice.

6.3 Modelos EfficientNet aislados

En la tabla 6.3 se resumen los resultados de los 25 modelos aislados EfficientNet que se construyeron y entrenaron. Las arquitecturas probadas van desde la B0 hasta la B4. Por cada una de ellas se entrenaron 5 modelos. Cada uno de ellos se entrenó con conjuntos de entrenamiento y validación de distinta composición gracias a una semilla aleatoria que depende de la arquitectura y el número de serie del modelo. Esta

organización de los modelos responde al método que del que se habló en la figura 5.6.2.

La arquitectura B4 representó el límite de complejidad de modelos que se consideró en este trabajo debido a la dificultad de entrenar otras arquitecturas más complejas en la plataforma Kaggle con las limitaciones que imponía.

Arquitectura	Serie	Macro Recall validación (etapa 1)	Macro Recall validación (etapa 2)	Macro F1 test	Macro Precision test	Macro Recall test
B0	1	0,503770	0,761110	0,5044382903	0,5010478831	0,6016581314
B0	2	0,535770	0,778830	0,4975017924	0,5086172907	0,5500478203
B0	3	0,533280	0,746720	0,4018258738	0,4464304368	0,5195806378
B0	4	0,482450	0,779320	0,5417741334	0,5420802003	0,5929864223
B0	5	0,486770	0,777490	0,5388920621	0,5346088944	0,5832654798
B1	1	0,498460	0,765330	0,5391821527	0,5361541238	0,6345475635
B1	2	0,516410	0,759900	0,5546085422	0,5442274866	0,6348684489
B1	3	0,490910	0,775540	0,4927737879	0,5052682055	0,5849556167
B1	4	0,483730	0,766310	0,4914659659	0,4973318173	0,5973160892
B1	5	0,485590	0,766180	0,549676205	0,5506063397	0,6193082455
B2	1	0,457840	0,782360	0,5866895648	0,6031716381	0,6272950856
B2	2	0,434440	0,809360	0,5692207737	0,5696506229	0,6434302789
B2	3	0,452850	0,746170	0,4632592351	0,4649547453	0,6185315594
B2	4	0,449510	0,778580	0,594616178	0,5695191851	0,6621288311
B2	5	0,386570	0,789990	0,5178822174	0,526668482	0,6172662625
B3	1	0,336370	0,776130	0,5885097052	0,5617257434	0,6840857352
B3	2	0,364750	0,808900	0,6291917822	0,6046554586	0,6887448715
B3	3	0,336920	0,767450	0,5366809161	0,5202508627	0,6421986091
B3	4	0,375780	0,805290	0,6300205025	0,615303583	0,6898759652
B3	5	0,280370	0,791730	0,5880684915	0,5539993671	0,685100426
B4	1	0,379623	0,819563	0,6529677804	0,6472894536	0,7089477849
B4	2	0,372337	0,805850	0,6763661438	0,6430048384	**0,7394543836**
B4	3	0,401550	0,802550	0,6470958245	0,6260177574	0,699376369
B4	4	0,399220	0,823170	0,5255105865	0,5175448873	0,6559095762
B4	5	0,439370	0,763390	0,6096823184	0,5995784224	0,6820507484

Tabla 6.3. Resultados obtenidos con modelos EfficientNet aislados

La figura 6.3 muestra el valor promedio de la medida macro recall obtenido contra el conjunto de test por cada uno de los 5 modelos entrenados de cada arquitectura EfficientNet utilizada. Puede observarse una tendencia al alza de esta medida, y por tanto de la efectividad global del modelo, conforme aumenta la complejidad de la misma.

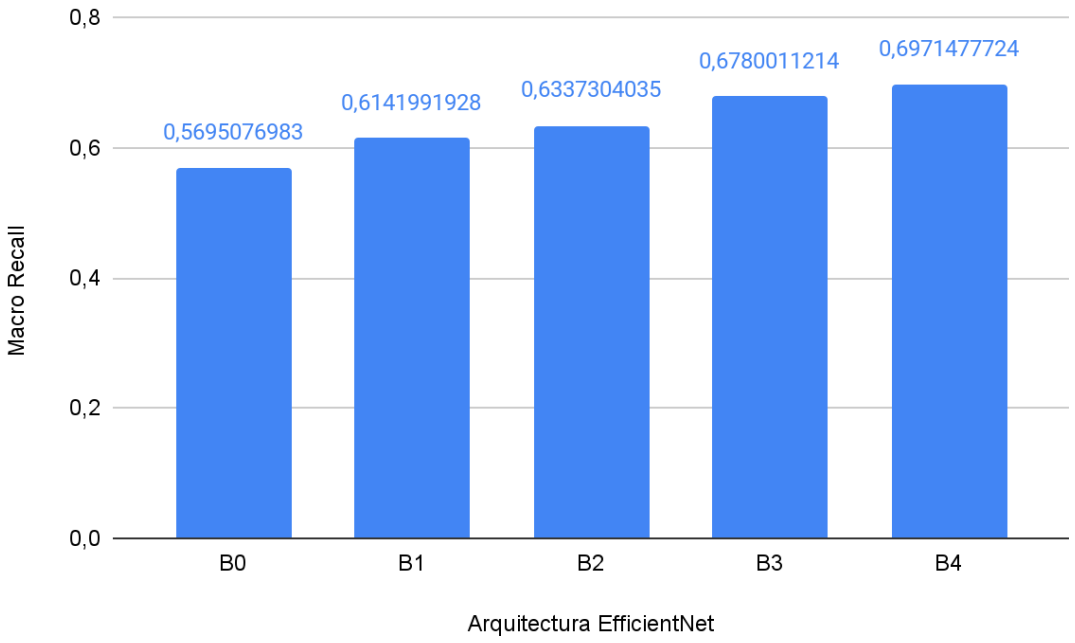

Figura 6.3. Promedio de la medida objetivo por cada arquitectura EfficientNet

6.4 Modelos EfficientNet ensamblados

El mejor modelo aislado que se encontró pertenece a la arquitectura B4, la más compleja de todas las que se probaron y que, por tanto, ha resultado ser la mejor de todas.

Aplicando la idea de ensamblar modelos que se introdujo en el punto 4.3.3, se consideró la siguiente lista de modelos ensamblados candidatos:

- **Todos excepto los de la mejor arquitectura:** Modelo compuesto por 20 modelos aislados de los 25 que se entrenaron. La idea detrás de esta propuesta fue la de comprobar si el ensamblado, por sí mismo, podría ser suficiente para dar buenos resultados aunque no estuviera compuesto necesariamente por los mejores sub-modelos. Para ello se tomaron todos aquellos que no pertenecieran a la mejor arquitectura.

- **Todos los modelos:** Modelo compuesto por los 25 modelos aislados que se entrenaron. La hipótesis central de este modelo fue considerar que la arquitectura de los modelos componentes no es un factor determinante en el resultado del modelo ensamblado sino que lo que cuenta es, simplemente, utilizar el mayor número posible de sub-modelos.

- **Solo el mejor de cada arquitectura:** Para este modelo ensamblado se utilizaron los mejores modelos de cada arquitectura entre la B0 y la B4. Por tanto, está compuesto de 5 sub-modelos. En este caso, la idea fue utilizar únicamente aquellos sub-modelos que hubieran dado mejor resultado con la métrica objetivo.

- **Solo los modelos de la mejor arquitectura:** Modelo compuesto por 5 de todos los modelos aislados que se entrenaron. En este caso la hipótesis era considerar que solo los modelos pertenecientes a la mejor arquitectura podrían conseguir el mejor resultado una vez ensamblados. Incluso si algunos de ellos no hubieran conseguido una métrica mejor que algunos de arquitecturas menos complejas.

Los resultados que se obtuvieron para cada uno de los modelos descritos se presentan en la tabla 6.4. Se encuentran ordenados comenzando por el que presenta mejor macro recall.

Modelo ensamblado	Número modelos	Macro F1	Macro Precision	Macro recall
B4 únicamente	5	0,702862371	0,6791327465	**0,7659438638**
Mejores de cada arquitectura	5	0,6744159385	0,6652495974	0,7215734261
Todos	25	0,6530964353	0,6453543056	0,7145027572
Todos excepto B4	20	0,6241781239	0,6219162077	0,6889449319

Tabla 6.4. Modelos ensamblados y sus métricas resultantes

6.5 Mejores modelos encontrados

Se presentan aquí, de forma más detallada, los resultados que se obtuvieron para el mejor modelo aislado y el mejor modelo ensamblado.

6.5.1 Mejor modelo EfficientNet aislado

Las gráficas de la figura 6.5.1.1 resumen el valor de accuracy y pérdida que tuvo este modelo durante el entrenamiento. Puede verse el descenso del valor de pérdida característico de la convergencia del modelo hacia una solución.

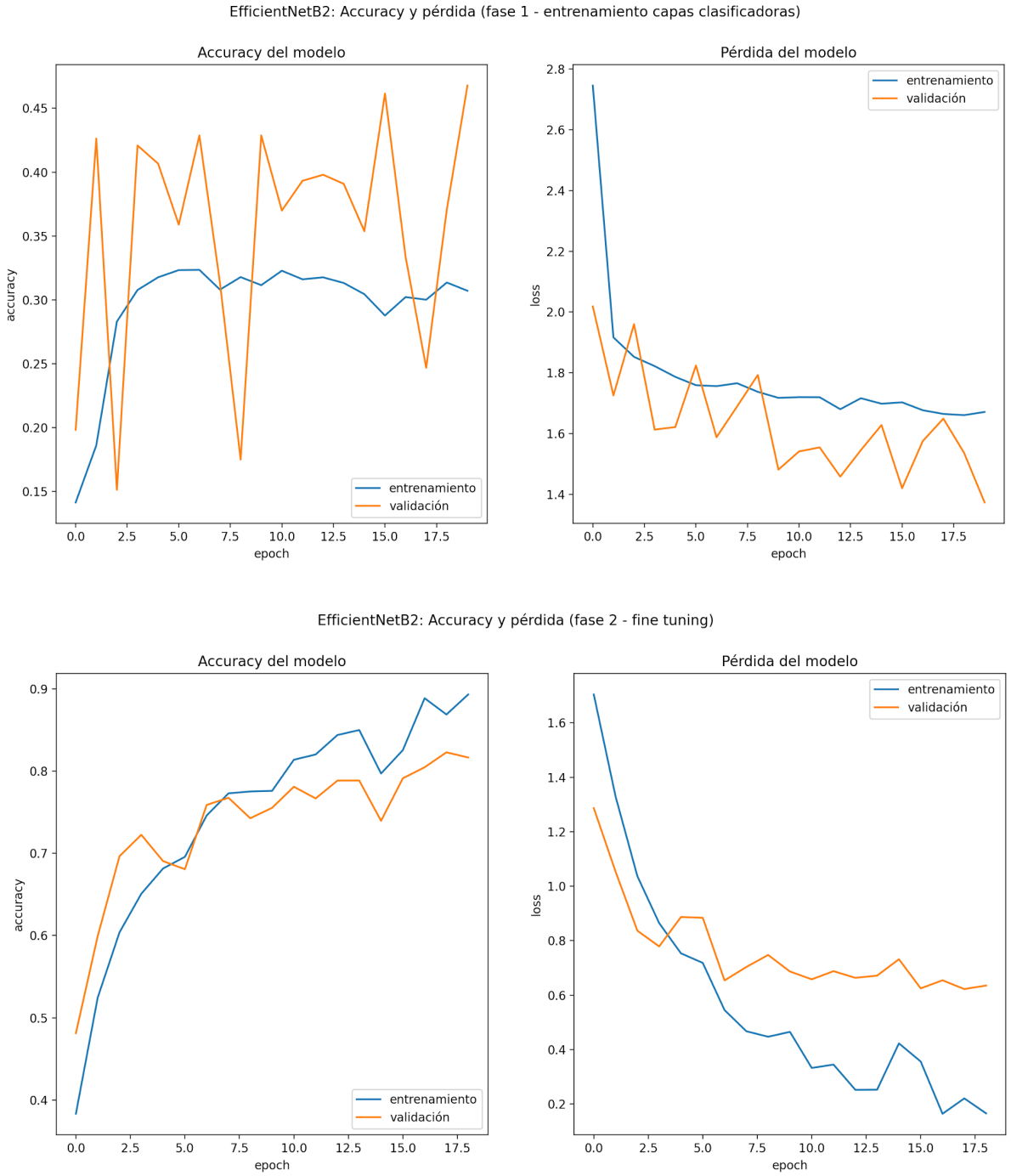

Figura 6.5.1.1. Accuracy y pérdida del modelo durante el entrenamiento

La figura 6.5.1.2 muestra la evolución de los valores de la métrica objetivo (la accuracy equilibrada) y también de la macro-precision y la macro-recall. Puede observarse la equivalencia de la métrica objetivo y la macro-recall tal y como se adelantó en el punto 6.1 además de la tendencia general a la mejoría de cada una de ellas conforme progresa el entrenamiento.

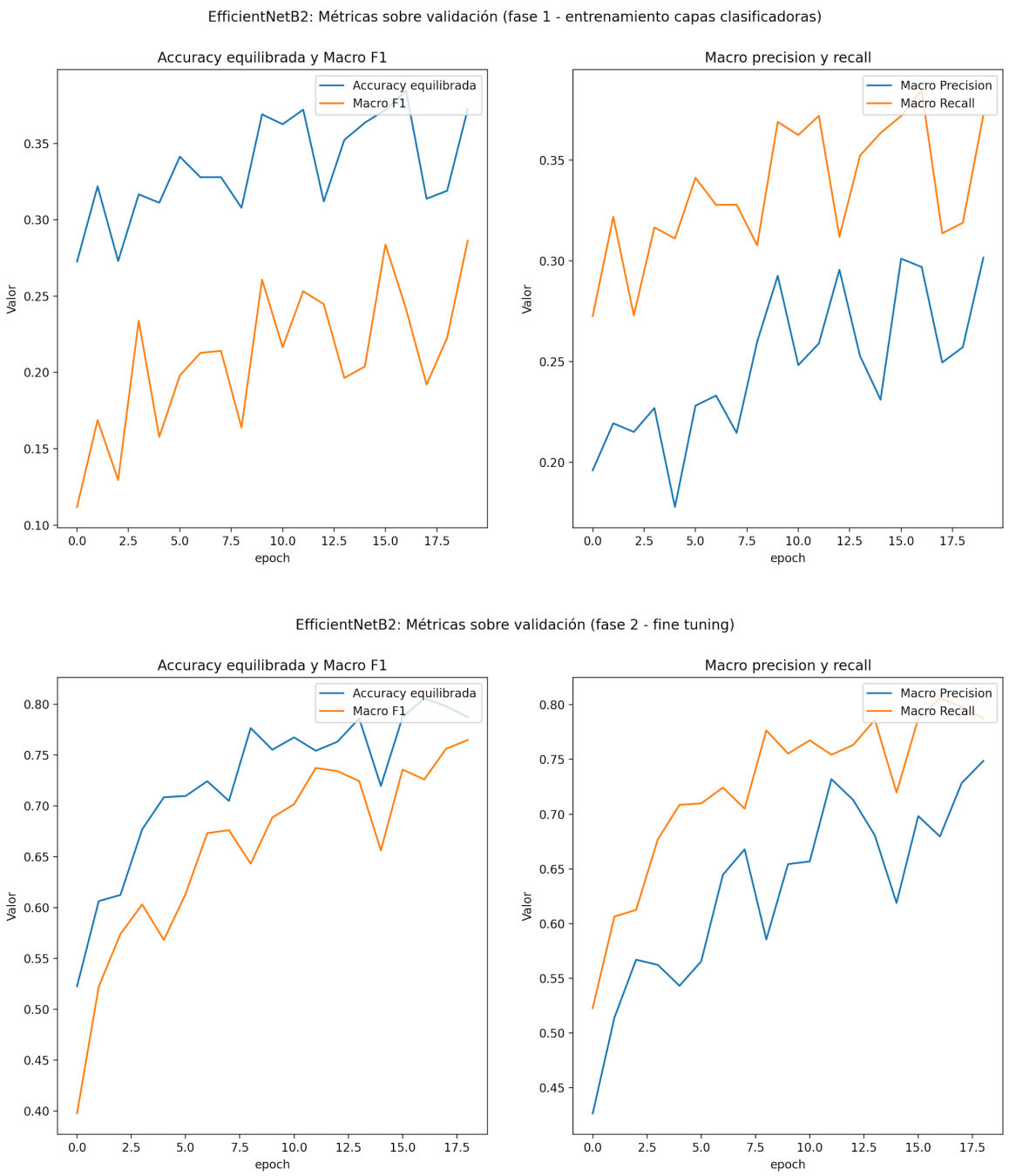

Figura 6.5.1.2. Evolución de las métricas durante el entrenamiento del modelo

La figura 6.5.1.3 muestra el informe de clasificación donde puede verse un cálculo de los valores de precisión, exhaustividad y medida F1 por cada una de las clases consideradas.

	precision	recall	f1-score	support
C0 (MEL)	0.76	0.52	0.62	452
C1 (NV)	0.87	0.86	0.86	1288
C2 (BCC)	0.80	0.86	0.83	332
C3 (AK)	0.56	0.56	0.56	87
C4 (BKL)	0.58	0.75	0.66	262
C5 (DF)	0.59	0.83	0.69	24
C6 (VASC)	0.52	0.88	0.66	25
C7 (SCC)	0.46	0.65	0.54	63
accuracy			0.77	2533
macro avg	0.64	0.74	0.68	2533
weighted avg	0.78	0.77	0.77	2533

Figura 6.5.1.3. Informe de clasificación del mejor modelo EfficientNet aislado encontrado

Los valores más importantes son los de exhaustividad (recall) de cada clase. A continuación se listan las clases que tienen los mejores valores (considerados a partir de 0.75). Esto significa que el clasificador es relativamente eficaz detectando estas lesiones:

- C6 (VASC)
- C1 (NV)
- C2 (BCC)
- C5 (DF)
- C4 (BKL)

El resto de clases con valor de recall menor de 0.75 representan aquellas que el clasificador tiene más dificultad en reconocer:

- C7 (SCC)
- C3 (AK)
- C0 (MEL)

Como complemento que ayuda a ilustrar la información anterior, la figura 6.5.1.4 muestra la matriz de confusión que resume la manera en que el modelo acabó clasificando cada una de las imágenes del conjunto de test.

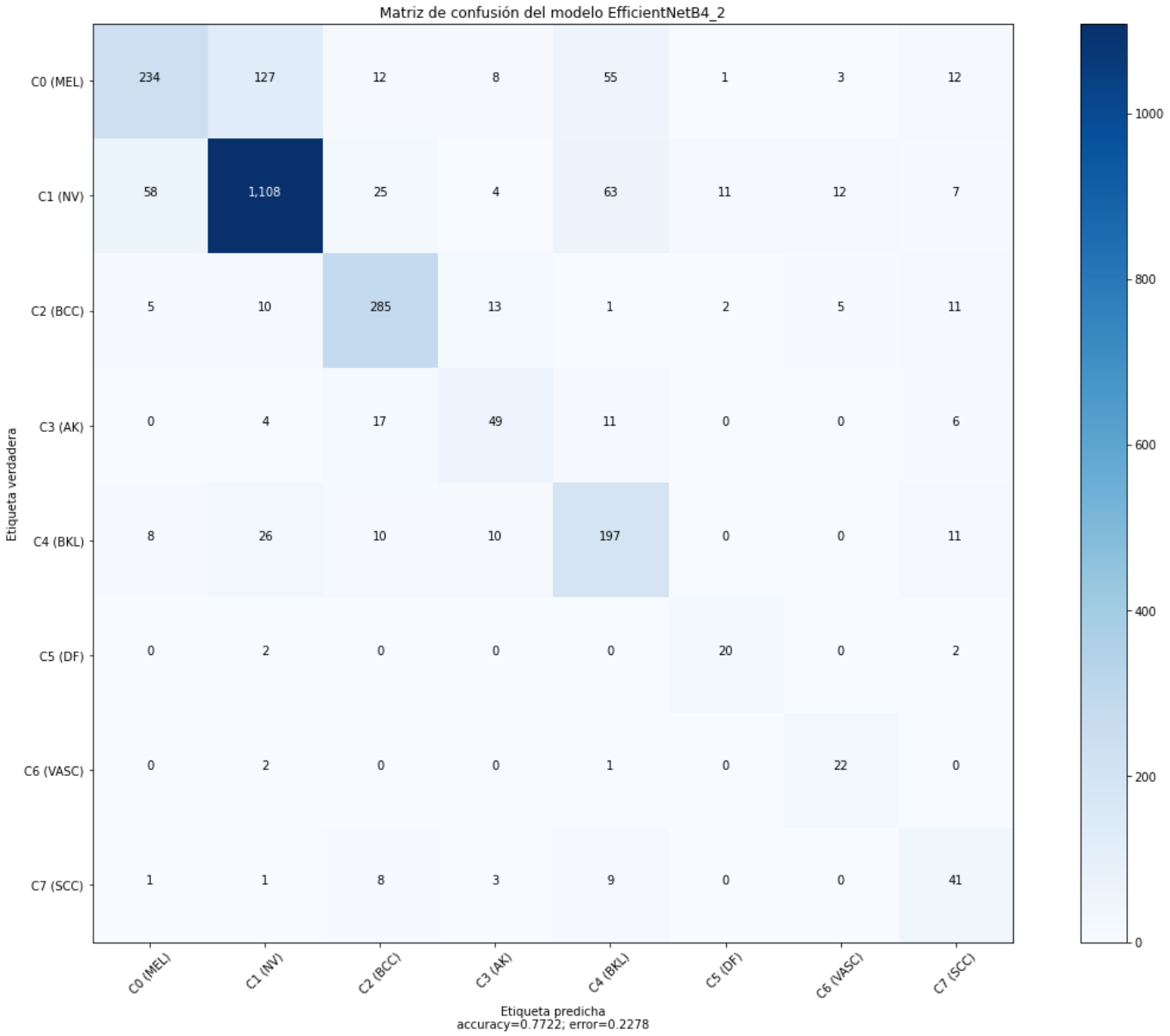

Figura 6.5.1.4. Matriz de confusión del mejor modelo EfficientNet aislado encontrado

6.5.2 Mejor modelo ensamblado

Naturalmente, el mejor modelo ensamblado no tuvo entrenamiento como tal. El "modelo" sencillamente interroga a cada uno de sus modelos componentes para obtener un vector de pertenencia de cada uno por cada imagen. Posteriormente, se calcula el vector de pertenencia final como promedio de todos los vectores de pertenencia obtenidos de cada modelo componente.

Al igual que para el mejor modelo aislado, se muestra en la figura 6.5.2.1 el informe de clasificación obtenido para el mejor modelo ensamblado.

	precision	recall	f1-score	support
C0 (MEL)	0.85	0.46	0.59	452
C1 (NV)	0.86	0.90	0.88	1288
C2 (BCC)	0.82	0.84	0.83	332
C3 (AK)	0.54	0.70	0.61	87
C4 (BKL)	0.59	0.73	0.65	262
C5 (DF)	0.51	0.79	0.62	24
C6 (VASC)	0.79	0.92	0.85	25
C7 (SCC)	0.46	0.79	0.58	63
accuracy			0.78	2533
macro avg	0.68	0.77	0.70	2533
weighted avg	0.80	0.78	0.78	2533

Figura 6.5.2.1. Informe de clasificación del mejor modelo ensamblado

Los valores más importantes son los de exhaustividad (recall) de cada clase. A continuación se listan las clases que tienen los mejores valores (considerados a partir de 0.75). Esto significa que el clasificador es relativamente eficaz detectando estas lesiones:

- C6 (VASC)
- C1 (NV)
- C2 (BCC)
- C5 (DF)
- C7 (SCC)

El resto de clases con valor de recall menor de 0.75 representan aquellas que el clasificador tiene más dificultad en reconocer:

- C4 (BKL)
- C3 (AK)
- C0 (MEL)

Y finalmente, la figura 6.5.2.2 muestra la matriz de confusión que resume la manera en que el modelo acabó clasificando cada una de las imágenes del conjunto de test.

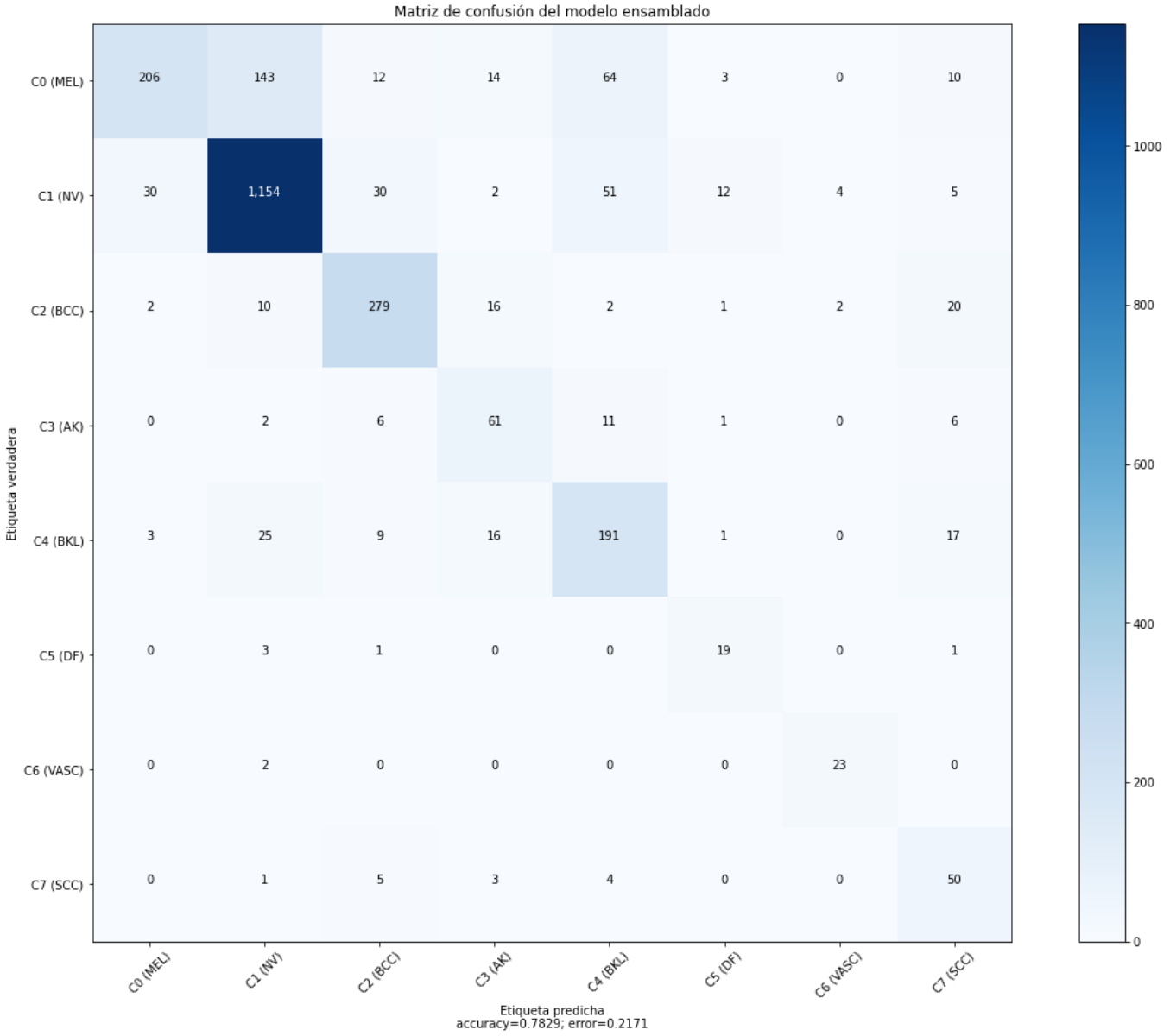

Figura 6.5.2.2. Matriz de confusión del mejor modelo ensamblado encontrado

6.5.3 Comparativa de los dos mejores modelos

Una vez obtenidos los resultados de los dos mejores modelos, se pueden obtener algunas observaciones más precisas a partir de su comparación. La figura 6.5.3.1 muestra cómo variaron los valores de macro recall por cada clase.

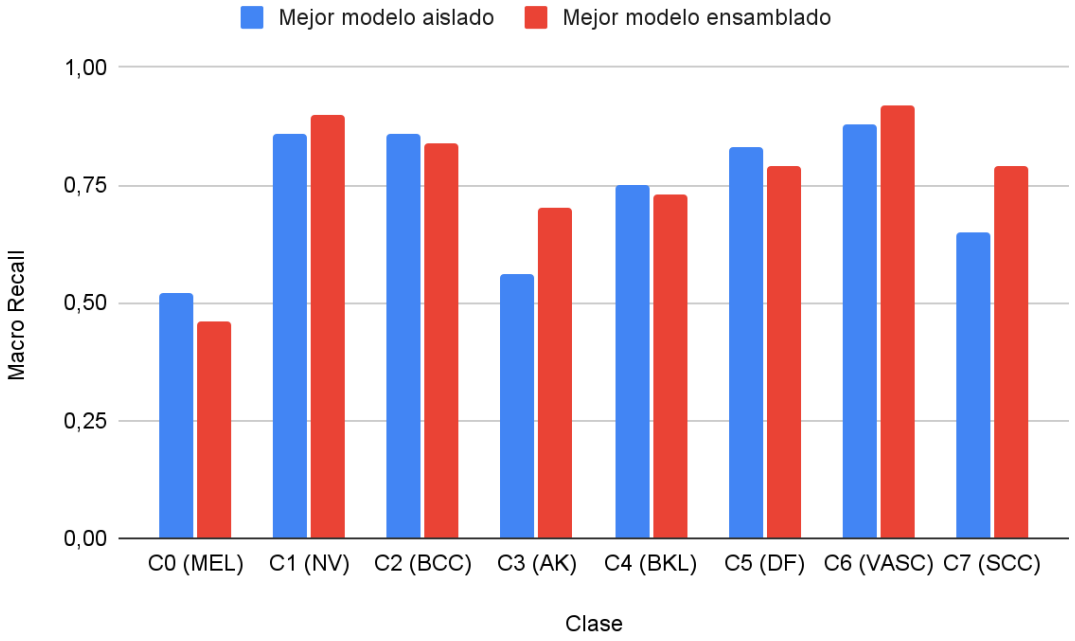

Figura 6.5.3.1. Comparación del macro-recall entre los dos mejores modelos

Se puede comprobar que con el modelo ensamblado algunas clases pudieron incrementar su valor de exhaustividad. En concreto, la clasificación de la lesión C7 (SCC) experimentó una mejora sustancial:

- C1 (NV)
- C3 (AK)
- C6 (VASC)
- C7 (SCC)

Puede verse que la exhaustividad del resto de las clases empeoró. La clase C0 (MEL), que tenía el peor resultado con el mejor modelo aislado, sufrió aún más con el modelo ensamblado. Esta clase representa la peor reconocida y clasificada independientemente del modelo utilizado.

De manera análoga a como se ha hecho con la exhaustividad también puede analizarse los resultados desde el punto de vista de la precisión. Hay que decir que este análisis es menos importante de cara a la métrica objetivo ya que, para su cálculo, solo se tiene la exhaustividad en cuenta y no la precisión. En la figura 6.5.3.2 se muestra cómo varió esta métrica entre ambos modelos por cada clase.

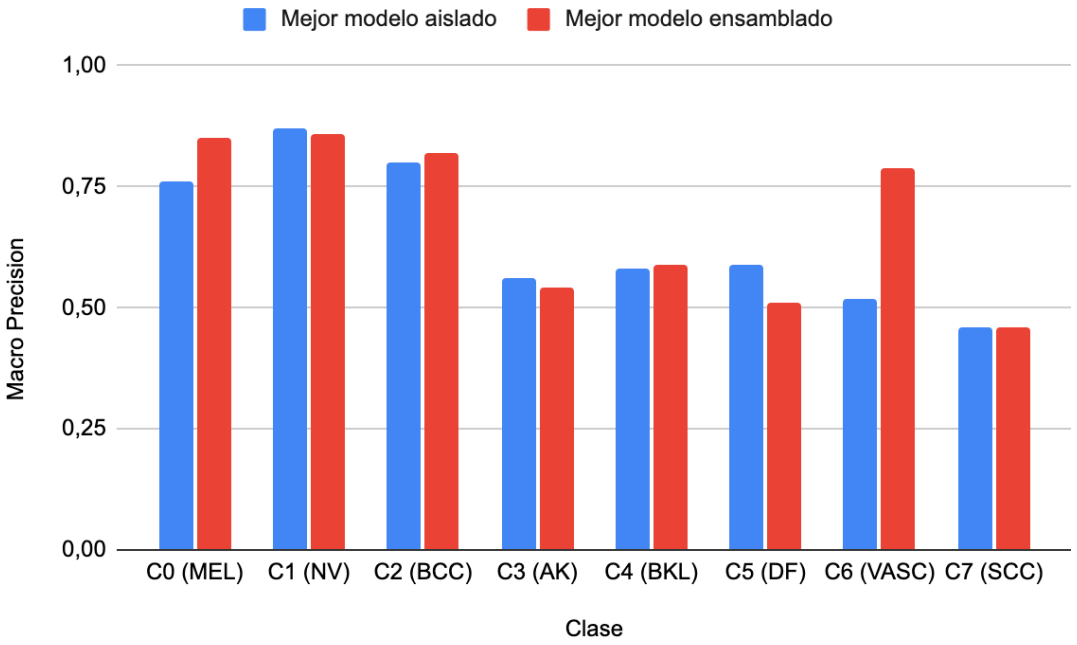

Figura 6.5.3.2. Comparación del macro-precision entre los dos mejores modelos

Se puede comprobar que con el modelo ensamblado algunas clases pudieron incrementar su valor de precisión. En concreto, la clasificación de la lesión C6 (VASC) aumentó en gran medida su nivel de precisión:

- C6 (VASC)
- C0 (MEL)
- C2 (BCC)
- C4 (BKL)

El resto de clases mantuvo su nivel de precisión o empeoró en el modelo ensamblado.

7. Conclusiones

En este capítulo se recogen las conclusiones principales del presente trabajo, reflexiones finales y algunas propuestas de trabajo futuro que podrían utilizarlo como punto de partida.

7.1 Conclusiones del trabajo

- Un clasificador que utiliza un modelo neuronal basado en la arquitectura EfficientNet resultó ser adecuado para dar una solución al problema de clasificación automática de imágenes dermatológicas descrito en los objetivos del presente trabajo. Para obtener buenos resultados es importante decir que no fue necesario aplicar ningún preprocesamiento de las imágenes con la intención de mejorar la capacidad de reconocimiento de ciertos patrones o matices clave por parte del clasificador.
- Los modelos basados en la arquitectura EfficientNet B4, la más compleja tratada en este trabajo, han resultado dar los mejores resultados. Sin embargo, este hecho también podría deberse a que los modelos más complejos tratados en este trabajo también accedían a mayor cantidad de información por imagen. En las líneas de trabajo futuro se propone ahondar más sobre este aspecto.
- En términos globales, el mejor modelo conseguido ha resultado de ensamblar una serie de modelos de la arquitectura más compleja (EfficientNet B4).
- Cualquier otro modelo ensamblado formado por sub-modelos de menor complejidad que la EfficientNet B4 resultó en un clasificador que presentaba un valor de la métrica objetivo por debajo del mejor modelo aislado que se consiguió entrenar.
- Si bien el ensamblaje de modelos de arquitectura más compleja consiguió mejorar ligeramente la medida objetivo del mejor modelo aislado, esto no significó que el modelo resultante fuera mejor que el aislado en todos los aspectos. El factor de exhaustividad mejoró para algunas clases pero también empeoró para otras.
- El entrenamiento de todos los modelos basados en red neuronal necesarios para la realización de este trabajo ha requerido una cantidad de tiempo de cómputo considerable. En este sentido la plataforma Kaggle resultó ser más adecuada que Google Colaboratory para construir y entrenar modelos con GPU hasta la complejidad de una EfficientNet B4. A pesar de ello y a partir de ese punto, las restricciones aplicadas en Kaggle hacían muy difícil continuar explorando arquitecturas más complejas.

7.2 Reflexión crítica

Se ha conseguido cumplir con todos los objetivos definidos al principio de este trabajo y descritos en el punto 1.2. Finalmente fue posible obtener un clasificador que, aunque ciertamente mejorable, consigue demostrar una cierta capacidad de generalización sobre lo aprendido a partir del conjunto de imágenes dermoscópicas ISIC 2019. Más adelante, se proponen una serie de líneas de trabajo futuro que podrían mejorar los resultados que se han obtenido.

Por otro lado, hay que decir que los conceptos y principios finalmente aplicados en este trabajo (transferencia de aprendizaje, entrenamiento en dos fases, tratamiento del desequilibrio, ensamblado de modelos, etc.) son generales del modelado de clasificadores automáticos utilizando el aprendizaje profundo. Es decir, no han sido específicos del campo de la dermatología ni del tratamiento de la imagen dermoscópica en particular sino que podrían aplicarse igualmente a otros tipos de problemas de clasificación automática de imágenes que nada tuvieran que ver con esta rama de la medicina.

7.3 Planificación y metodología

La planificación inicial se realizó teniendo en cuenta las distintas entregas (PEC) y se mantuvieron durante todo el desarrollo del trabajo.

La metodología resultó también adecuada para llegar a conseguir modelos candidatos cuya efectividad fue mejorando conforme se aumentaba el tamaño de la imagen de entrada y la complejidad del modelo hasta llegar a aplicar el concepto de ensamblado para conseguir el mejor modelo.

7.4 Líneas de trabajo futuro

A continuación se proponen una serie de líneas de acción complementarias que podrían seguirse en un futuro con la intención de mejorar los resultados del presente trabajo.

- **Utilizar un método alternativo para adaptar el tamaño de las imágenes a cada arquitectura EfficientNet.** Los modelos con arquitectura más compleja han dado los mejores resultados en este trabajo. Sin embargo, hay que recordar que dichas arquitecturas también admiten imágenes de mayor tamaño. Sería interesante comprobar si estos resultados se deben a la complejidad de la arquitectura únicamente o si el aumento del tamaño de la

imagen (y por tanto de la información entrante) también contribuye a ello. Una posible propuesta consiste en utilizar recortes de la imagen (cropping, en inglés) en lugar de reducir sus dimensiones. Es decir, evitar la disminución de la cantidad de información que se introduce en la red neuronal. Utilizar recortes aleatorios de las imágenes con el tamaño objetivo en lugar de reducirlas es una idea que ya pareció dar muy buenos resultados en otros trabajos (Mahbod et al., 2020). Sin embargo, habría que desarrollar un método para hacerlo de manera que no se generen imágenes de zonas sin lesión que podrían confundir al modelo o, si esto no es posible, intentar mitigar su posible efecto negativo sobre la calidad del clasificador final de alguna forma. La aplicación de técnicas de segmentación podrían ayudar en el recorte de imágenes para garantizar que se centrasen en zonas con lesión.

- **Mejorar algunas características de las imágenes de entrada:** ciertas transformaciones aplicadas en los datos de entrada podrían facilitar la tarea de un modelo neuronal posteriormente. Por ejemplo aplicar normalizaciones de histograma, filtrados para reducir el ruido, etc.
- **Intentar mejorar algún punto del diseño:** Los resultados extraídos en este trabajo constituyen ya un punto de partida comparativo para poder evaluar futuros cambios que se introduzcan en el diseño con la intención de mejorar aún más la efectividad de los clasificadores obtenidos. Algunos de estos puntos podrían ser:
    - Repetir el experimento que intenta encontrar a partir de qué bloque se necesita reentrenar el modelo base para obtener resultados óptimos con cada arquitectura. Esto serviría, de paso, para corroborar o desmentir la presunción que se ha adoptado en este trabajo de que solo es necesario hacerlo para una arquitectura.
    - Modificar el diseño de las capas de clasificación. Introduciendo más capas o alterando el número de neuronas o probabilidades de la(s) capa(s) dropout.
    - Probar con otros parámetros de entrenamiento: optimizador, tamaño del batch, valor de paciencia definido en la aplicación de parada prematura (Early Stopping), etc.
    - Utilizar otros métodos alternativos para el tratamiento del desequilibrio. El método de sobremuestreo (oversampling, en inglés), por ejemplo, parece haber dado buenos resultados en otros trabajos (Buda et al., 2018).
- **Introducir modificaciones para poder utilizar procesadores más avanzados (TPU).** Mediante la introducción de algunos cambios en el código de la implementación y aplicando algunas modificaciones a la manera en que están almacenados los datos de imágenes ISIC 2019 posibilitaría la utilización optimizada de procesadores TPU de alto rendimiento en la plataforma Kaggle. Este hecho no solamente debería acelerar el proceso de entrenamiento de modelos sino que haría posible hacerlo con aquellos pertenecientes a arquitecturas EfficientNet más complejas de los que se han tratado en este trabajo. Lo que abre la posibilidad de encontrar modelos más eficientes aunque también más complejos.

8. Glosario

CNN: Convolutional Neural Network. Red neuronal convolucional.

DL: Deep Learning. Aprendizaje profundo.

FC: Fully connected layer. Capa completamente conectada.

GPU: Graphics Processing Unit. Unidad de procesamiento gráfico.

TPU: Tensor Processing Unit. Unidad de procesamiento de tensores.

ML: Machine Learning. Aprendizaje automático

CSV: Comma separated values. Formato de fichero de texto donde los valores se separan por comas.

9. Bibliografía

Adegun, A., y Viriri, S. (2020, Junio 27). *Deep learning techniques for skin lesion analysis and melanoma cancer detection: a survey of state-of-the-art.* Artificial Intelligence Review, (54), 811–841. 10.1007/s10462-020-09865-y

Agarwal, V. (2020, Mayo 24). Complete Architectural Details of all EfficientNet Models | by Vardan Agarwal. Towards Data Science. Consultado el 4 de Diciembre 4 de 2021, url: https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142

Barata, C., Marques, J. S., & Celebi, M. E. (2019). *Deep Attention Model for the Hierarchical Diagnosis of Skin Lesions.* EEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2757-2765. 10.1109/CVPRW.2019.00334

Bhatt, C., Kumar, I., Vijayakumar, V., Singh, K. U., & Kumar, A. (2020, Septiembre 25). *The state of the art of deep learning models in medical science and their challenges.* Multimedia Systems, (27), 599–613. 10.1007/s00530-020-00694-1

Brownlee, J. (2019, Abril 12). How to Configure Image Data Augmentation in Keras. Machine Learning Mastery. Retrieved Diciembre 3, 2021, from https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

Brownlee, J. (2020, Julio 24). Train-Test Split for Evaluating Machine Learning Algorithms. Machine Learning Mastery. Consultado el 27 de Noviembre 27 de 2021, url: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

Brownlee, J. (2020, Agosto 18). *Transfer Learning in Keras with Computer Vision Models. Machine Learning Mastery.* Consultado el 10 de Octubre de 2021, url: https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/

Buda, M., Maki, A., & Mazurowski, M. A. (2018, Julio 29). *A systematic study of the class imbalance problem in convolutional neural networks.* Elsevier, Science Direct, 106, 249-259. https://doi.org/10.1016/j.neunet.2018.07.011

Chan, S., Reddy, V., Myers, B., Thibodeaux, Q., Brownstone, N., & Liao, W. (2020). *Machine Learning in Dermatology: Current Applications, Opportunities, and Limitations.* Dermatol Ther (Heidelb). 10.1007/s13555-020-00372-0

Chollet, F. (2019, Mayo 28). *Imbalanced classification: credit card fraud detection.* Keras. Consultado el 10 de Diciembre de 2021, url: https://keras.io/examples/structured_data/imbalanced_classification/

Chollet, F. (2020, Abril 15). *Transfer learning & fine-tuning.* keras.io. Consultado el 1 de Noviembre de 2021, url: https://keras.io/guides/transfer_learning/

Codella, N. C.F., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., & Halpern, A. (2017). *Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI)*, Hosted by the International Skin Imaging Collaboration (ISIC). arXiv. 1710.05006

Combalia, M., Codella, N. C.F., Rotemberg, V., Helba, B., Vilaplana, V., Reiter, O., Halpern, A. C., Puig, S., & Malvehy, J. (2019). *BCN20000: Dermoscopic Lesions in the Wild*. arXiv:1908.02288

Fu, Y. (2020, Junio 30). *Image classification via fine-tuning with EfficientNet.* keras.io. Consultado el 1 de Noviembre de 2021, url: https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/#tips-for-fine-tuning-efficientnet

Gupta, K. (2021, Junio 19). *How to Load Kaggle Datasets Directly into Google Colab?* Analytics Vidhya. Consultado el 16 de Noviembre 16 de 2021, url: https://www.analyticsvidhya.com/blog/2021/06/how-to-load-kaggle-datasets-directly-into-google-colab/

Hashemi, M. (2019). *Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation.* Journal of Big Data. 10.1186/s40537-019-0263-7

Hemavath, S., y Velmuruga, K. J. (2020). *Skin disease prediction and provision of medical advice using deep learning.* Journal of Physics: Conference Series, 1724. 10.1088/1742-6596/1724/1/012048

ImageMarkup/isic-challenge-scoring: Automated scoring code for the ISIC Challenge. (n.d.). GitHub. Consultado el 4 de Diciembre 4 de 2021, url: https://github.com/ImageMarkup/isic-challenge-scoring

ISIC. (n.d.). About ISIC. *The International Skin Imaging Collaboration (ISIC).* Consultado el 2 de Octubre de 2021, url: https://www.isic-archive.com/#!/topWithHeader/tightContentTop/about/aboutIsicOverview

Kinyanjui, N. M., Odonga, T., Cintas, C., Codella, N. C. F., Panda, R., Sattigeri, P., & Varshney, K. R. (2019). *Estimating Skin Tone and Effects on Classification Performance in Dermatology Datasets.* arxiv.org. arXiv:1910.13268v1

Li, Y. (2020, Febrero 27). *Are You Making This Mistake when Implementing the Macro F1 Score in Keras?* Towards Data Science. Consultado el 18 de Diciembre 18 de 2021, url: https://towardsdatascience.com/implementing-macro-f1-score-in-keras-what-not-to-do-e9f1aa04029d

Mahbod, A., Schaefer, G., Wang, C., Dorffner, G., Ecker, R., & Ellinger, I. (2020). *Transfer learning using a multi-scale and multi-network ensemble for skin lesion classification.* Computer Methods and Programs in Biomedicine. 10.1016/j.cmpb.2020.105475

Marcelino, P. (2018, Octubre 23). *Transfer learning from pre-trained models.* Towards Data Science. Consultado el 4 de Diciembre 4 de 2021, url: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

Nelson, J. (2020, Enero 31). *You Might Be Resizing Your Images Incorrectly.* Roboflow. Consultado el 8 de Octubre de 2021, url: https://blog.roboflow.com/you-might-be-resizing-your-images-incorrectly/

Shmueli, B. (2019, Julio 3). *Multi-Class Metrics Made Simple, Part II: the F1-score*. Towards Data Science. Consultado el 18 de Diciembre de 2021, url: https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1

Tschandl, P., Rosendahl, C., & Kittler, H. (2018). *The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions.* Sci. Data 5, 180161 doi.10.1038/sdata.2018.161.

Vashisht, R. (2020, Octubre 22). *What is data leakage? And how to mitigate it?* atoti. Consultado el 10 de diciembre de 2021, url: https://www.atoti.io/what-is-data-leakage-and-how-to-mitigate-it/

Vijayabhaskar, J. (2018, Septiembre 21). *Tutorial on Keras flow_from_dataframe*. Medium.com. Consultado el 31 de Octubre de 2021, url: https://vijayabhaskar96.medium.com/tutorial-on-keras-flow-from-dataframe-1fd4493d237c

Vijayabhaskar, J. (2019, Enero 26). *Multi-label image classification Tutorial with Keras ImageDataGenerator.* Medium.com. Consultado el 31 de Octubre de 2021, url: https://vijayabhaskar96.medium.com/multi-label-image-classification-tutorial-with-keras-imagedatagenerator-cd541f8eaf24