TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Aplicación móvil para la prognosis y detección del cáncer de piel usando deep learning

Autor

Cristhian Moya Mota (alumno)

Directores

Diego Jesús García Gil

Julián Luengo Martín

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 19 de junio de 2024

Aplicación móvil para la prognosis y detección del cáncer de piel usando deep learning

Autor
Cristhian Moya Mota

Directores
Diego Jesús García Gil
Julián Luengo Martín

Título del Proyecto: Subtítulo del proyecto

Cristhian Moya Mota

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3, ...

Resumen

Poner aquí el resumen.

Project Title: Project Subtitle

First name, Family name (student)

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

Yo, **Cristhian Moya Mota**, alumno de la titulación Grado en Ingeniería informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 20596451C, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Cristhian Moya Mota

Granada a X de junio de 2024 .

D. **Diego Jesús García Gil**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

D. **Julián Luengo Martín**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Aplicación móvil para la prognosis y detección del cáncer de piel usando deep learning***, ha sido realizado bajo su supervisión por **Cristhian Moya Mota**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de junio de 2024 .

Los directores:

Diego Jesús García Gil **Julián Luengo Martín**

Agradecimientos

Poner aquí agradecimientos...

Índice general

1 Introducción **23**
1.1 Motivación . . . 23
1.1.1 El cáncer de piel . . . 25
1.1.2 Dificultades del diagnóstico . . . 26
1.2 Justificación . . . 27
1.3 Objetivos . . . 27
1.4 Planificación . . . 28

2 Tendencias y Estado del arte **29**
2.1 Aprendizaje profundo en dispositivos móviles . . . 29
2.1.1 Squeeze-Net (2016) . . . 30
2.1.2 MobileNet . . . 32
2.1.3 Shuffle Net . . . 37
2.1.4 EfficientNet . . . 38
2.2 Cuantización de modelos . . . 40
2.2.1 Características . . . 41
2.2.2 Modelos cuantizados . . . 42
2.2.3 Conclusión de los modelos cuantizados . . . 44
2.3 Conjuntos de datos disponibles . . . 45
2.3.1 International Skin Imaging Collaboration (ISIC) . . . 46
2.3.2 ASAN Dataset . . . 51
2.3.3 DermNet . . . 53
2.3.4 PH2 . . . 54
2.3.5 PAD-UFES 20 . . . 55
2.3.6 Severance . . . 57
2.3.7 Otros datasets . . . 59

3 Construcción y preprocesado del conjunto de datos **61**
3.1 Estrategia de preprocesado para la fusión . . . 62
3.1.1 ISIC Dataset . . . 63
3.1.2 ASAN . . . 66
3.1.3 PAD UFES 20 . . . 68
3.1.4 PH2 . . . 68
3.1.5 Severance . . . 69
3.1.6 Etiquetado binario . . . 72

3.2 Particionado de los datos 73
3.2.1 Importancia de la separación train-test 73
3.2.2 Particionado de validación 76
3.3 Uso de GANs de super resolución 78
3.3.1 Justificación de uso 78
3.3.2 Proceso de reescalado 80

4 Modelos y entorno de trabajo 83
4.1 Conceptos previos 83
4.2 Modelos preentrenados escogidos 85
4.2.1 ResNet 50 85
4.2.2 MobileNet V2 86
4.2.3 EfficientNet B5 87
4.3 Función de pérdida 88
4.3.1 Cross entropy 88
4.3.2 Focal Loss 88
4.4 Especificaciones y framework de trabajo 89

5 Entrenamiento de modelos completos sobre la base de datos 91
5.1 Clasificación binaria 92
5.1.1 Equilibrio de los datos 92
5.1.2 Representación en Pytorch 95
5.1.3 Resultados 101
5.1.4 Modelo seleccionado 107
5.2 Clasificación multiclase 109
5.2.1 Modelo de enfermedades malignas 109
5.2.2 Modelo de enfermedades benignas 113

6 Cuantización 119
6.1 Optimizaciones de Pytorch mobile 119
6.1.1 Capacidades del framework 119
6.1.2 Optimizaciones disponibles 121
6.2 Creación de los modelos cuantizados 122
6.3 Comparativa de los modelos original y cuantizado 124
6.3.1 Cuantización del modelo binario 124
6.3.2 Cuantización del modelo multiclase maligno 127
6.3.3 Cuantización del modelo multiclase benigno 129
6.4 Conclusión de la cuantización 131

Índice de figuras

2.1 Arquitectura de SqueezeNet 31
2.2 Producto depthwise 33
2.3 Bloque residual invertido 34
2.4 Arquitectura de MobileNetV2 35
2.5 Arquitectura de MobileNetV3 36
2.6 Channel Shuffle de ShuffleNet 37
2.7 Arquitectura de ShuffleNet 38
2.8 Parámetros de EfficientNet 39
2.9 Optimización de EfficientNet Lite 43
2.10 Ganancia de la cuantización en EfficientNet 44
2.11 Proceso de búsqueda seguido 46
2.12 Ejemplo de imágenes de ISIC 2017 50
2.13 Ejemplo de lunares benignos en ASAN (Nevus) 53
2.14 Nevus maligno y benigno en PH2 55
2.15 Batch de ejemplo de PAD-UFES 20 [46] 57
2.16 Imágenes de ejemplo provenientes del dataset Severance 58

3.1 Formato del fichero csv normalizado 63
3.2 Distribución de clases 72
3.3 Arquitectura de ESRGAN. Fuente: [68] 79
3.4 Imagen original (izquierda) vs aumentada (derecha) 81

5.1 Desequilibrio de datos 93
5.2 Demostración gráfica del Early stopping [18] 99
5.3 Curvas del finetuning del modelo binario con Resnet50 103
5.4 Curvas del finetuning del modelo binario con EfficientNet B5 104
5.5 Curvas del finetuning del modelo binario con Mobilenet V2 107
5.6 Espacio (en MB) requerido por cada modelo 108
5.7 Distribución de las clases malignas 110
5.8 Curvas de aprendizaje para clases malignas 111
5.9 Conteo de clases benignas antes de agrupar 114
5.10 Conteo de clases benignas tras la agrupación 116

6.1 Optimización de modelos con Pytorch Mobile [52] 120
6.2 Comparativa de ganancias al emplear modelos cuantizados 132

Índice de tablas

2.1 Número de imágenes duplicadas encontradas [10] 49
2.2 Tabla de imágenes únicas extraída de [10]. En este caso, el autor descarta el uso del dataset de 2016 por su baja aportación 50
2.3 Distribución de clases de ASAN dataset 51
2.4 Tabla de casos diagnosticados en PAD-UFES20 56

4.1 Rendimiento de ResNet en **Imagenet** [7] 86
4.2 Rendimiento de MobileNet en **Imagenet** [45] 87
4.3 Tamaño de las versiones de Efficient Net [63] 87

5.1 Informe de clasificación 94
5.2 Cabecera para trasfer-learning binario 99
5.3 Informe de clasificación Resnet50 102
5.4 Informe de clasificación de EfficientNet B5 104
5.5 Informe de clasificación de EfficientNet B5 106
5.6 Valores de las métricas de cada modelo 107
5.7 Informe de clasificación para validación maligna 111
5.8 Informe de clasificación para validación maligna (Pesos) 112
5.9 Clasificación para validación benigna (Ordenadas de mayor a menor rep., sin pesos) 115
5.10 Informe de clasificación para validación benigna (Ordenadas de mayor a menor rep.) 117

6.1 Resultados de inferencia en test para el modelo original 125
6.2 Resultados de inferencia en test para el modelo cuantizado 125
6.3 Tiempos de inferencia para caso binario en Android 126
6.4 Resultados de inferencia en test para el modelo original 127
6.5 Resultados de inferencia en test para el modelo cuantizado 128
6.6 Tiempos de inferencia para caso multiclase maligno en Android . . . 128
6.7 Resultado obtenido al inferir el conjunto de test (sin cuantizado) . . . 130
6.8 Resultado obtenido al inferir el conjunto de test (cuantizado) 130
6.9 Tiempos de inferencia para caso binario en Android 131

1 Introducción

En este primer punto, describiremos la importancia del correcto diagnóstico del cáncer de piel, y de sus dificultades de detección. Analizaremos la justificación del motivo de realización de este TFG, así como los objetivos a cumplir y la planificación seguida en su desarrollo.

1.1. Motivación

El cáncer es una de las causas de muerte principales en el mundo. Su gran agresividad, así como su dificultad de diagnóstico, debido a su gran variedad de ubicaciones y manifestaciones, provoca que un alto porcentaje de casos no sean diagnosticados a tiempo correctamente. Tan solo en 2023, aproximadamente se registraron 20 millones de nuevos casos de cáncer a nivel mundial, y produciéndose algo menos de 10 millones de defunciones [36].

Estos registros provocan una gran inquietud en la población y entre los expertos de la materia; debido al aumento que se produce cada año, se espera que para el año 2050, el número de nuevos casos sea un 77 % mayor [69]. Por desgracia, no existen formas de prevención claras para este tipo de enfermedad, ni un tratamiento efectivo que permita al paciente recuperarse fácilmente.

La única opción probada es la realización de pruebas rutinarias a colectivos de riesgo, para así acelerar la detección de posibles tumores, y aumentar la esperanza de supervivencia. Esto se ve reflejado en las cifras de los dos tipos de cáncer más frecuentes: el cáncer de mama, y el cáncer colorrectal. Si se detectan en fases iniciales, la correcta recuperación del colon podría aumentarse hasta el 90 % [56], mientras que en el cáncer de mama, podría reducirse su mortalidad entre un 25-31 %. Gracias a la existencia de pruebas rutinarias programadas por servicio de salud, se puede reducir la mortalidad.

El gran problema de estos tipos de cánceres son la escasa visibilidad y síntomas de los mismos; cuando muestran señales, es probable que la tasa de supervivencia sea mucho menor, sobre todo en el colon. Pero existe otro tipo de cáncer que sí se manifiesta de forma más visible y que puede alarmar al paciente de forma más temprana: el cáncer de piel.

Este tipo de tumores puede manifestarse en las diferentes capas de la dermis, y su origen se atribuye a la exposición prolongada a la luz solar sin hacer uso de protección. Debido a los daños que sufre la capa de ozono, y otros factores ambientales, la cantidad de rayos ultravioleta que llegan hasta la superficie ascendió desde que se tienen registros. Si bien la capa de ozono parece recuperarse, debemos ser cautos, y tener cuidado de nuestra piel; los rayos ultravioleta pueden dañar células de la misma, y provocar alteraciones en su material genético. Su principal consecuencia es el crecimiento incontrolado de células, origen de los tumores cancerígenos en la piel.

Se estima que en el mundo, los tumores de la piel representan un tercio de los casos de cáncer diagnosticados. Esta distribución sigue valores parecidos en España, y al igual que las cifras de otros tipos de tumores, los casos diagnosticados aumentan año tras año. Las muertes debidas a esta enfermedad son principalmente, por ser identificadas en fases tardías de su evolución. Debido a que la piel es el órgano más grande del cuerpo humano, y que está en contacto con todos los capilares sanguíneos y el sistema linfático, las células cancerosas se pueden extender por ellos hacia otros lugares del cuerpo.

Aunque este cáncer puede ser identificado de forma más sencilla por su portador, la escasa información acerca del tema, y la confusión con otras lesiones benignas de la piel como verrugas o lunares, provoca una disminución en las posibilidades de supervivencia. Por ello, el objetivo de este trabajo es aportar una nueva forma de diagnóstico que permita a los usuarios obtener una orientación acerca de qué posible lesión están experimentando en la piel, y sirva como complemento del experto. O bien, ayudar a los expertos a tomar la decisión, acortando los tiempos de diagnóstico para aumentar las posibilidades de supervivencia. Esta tarea será realizada gracias al uso de una de las herramientas en auge en la actualidad: la inteligencia artificial, y concretamente, el uso de deep learning y métodos de visión por computador.

Mediante una nueva arquitectura, el propósito es conseguir un modelo capaz de segmentar las manchas de interés en la piel que estén recogidas en una fotografía. Dicha fotografía será capturada con el teléfono móvil del usuario, retirando así la necesidad de disponer de dispositivos especializados. Y posteriormente, clasificar dichas manchas para ofrecer al usuario final una respuesta sólida acerca del posible tipo de lesión de piel que sufre.

1.1.1. El cáncer de piel

Prosiguiendo con el estudio inicial, el diagnóstico del cáncer de piel [9], se dificulta, sobre todo, por su amplia variedad de formas, tamaños, texturas y manifestaciones. Aunque su visibilidad pueda parecer evidente, (ya que es observable a nivel macroscópico) puede ser confundido fácilmente con lesiones benignas. Normalmente, suele dividirse entre dos tipos diferentes:

- **Melanomas de la piel**. Son la variante más peligrosa. Su origen se encuentra en los melanocitos, las células encargadas de dar el color bronceado a la piel. Éstas pueden comenzar a crecer sin control originando tumores, los cuales crecen y se diseminan rápidamente hacia otras regiones del organismo, provocando la metástasis, una extensión a nivel total del organismo. Es el más grave de los diagnósticos. Puede identificarse como una mancha oscura en la piel, formando tumores de color café oscuro. Sin embargo, debido a la gran variedad de reacciones, pueden darse de color rosado si dejan de producir melanina. Este aspecto dificulta su diagnóstico, por lo que el papel de las herramientas de visión por computador pueden ayudar a su identificación.

- **Cánceres no melanomas**. Este tipo de cánceres no se ubican en los melanocitos, y pueden ser tratados mediante otras ténicas menos agresivas debido a su rara probabilidad de expansión. Los más comunes, son los tumores de células basales y los de células escamosas:

  - *Células basales*. Componen la capa inferior de la piel, y son las células encargadas de sustituir aquellas que componen la capa más externa de la piel. Se encuentran , por tanto, en constante reproducción para cubrir aquellas que mueren en la superificie. Si experimentan alguna mutación, producen tumores de color similar a la piel del paciente, con la posibilidad de aparecer en colores como negro brillante en las pieles más oscuras.

  - *Células escamosas*. Son las células externas de la piel, con forma plana. Se regeneran constantemente gracias a las células basales, que producen estas células las cuales se aplanan a medida que ascienden hacia la capa externa. Es frecuente, de nuevo, en zonas expuestas al sol, sobre todo la cara. Normalmente, se encuentran bien localizados, y puede procederse a su extirpación. En casos en los que se haya extendido, se hace uso de radioterapia.

Aunque en base a su descripción parezcan distinguibles, son fácilmente confundidos por su variedad con otros tumores benignos de la piel, como:

- **Lunares**: también conocidos médicamente con nevus, se trata de hiperpigmentación benigna en la piel.

- **Verrugas**: tumores benignos de piel, frecuente debido a virus como el del papiloma humano.
- **Lesiones vasculares**: varices, derrames, y otro tipo de problemas circulatorios.
- **Lipomas**: tumores de tacto blando, debido a su contenido en lípidos (grasa).
- **Queratosis seborreica**: son manchas cerosas, comúnmente desarrolladas en la espalda. De aspecto oscuro y gran relieve, no suponen ninguna amenaza más allá de posible incomodidad al roce o estética.

El uso de deep learning para este fin resulta interesante como forma de mejora del diagnóstico ante casos malignos y benignos de gran similitud, los cuales pueden confundir y dificultar la labor incluso a expertos dermatólogos.

1.1.2. Dificultades del diagnóstico

Existe una gran dificultad de conocer la naturaleza del tumor del paciente. Habitualmente, se suele extraer una muestra del tejido afectado para proceder a su análisis en laboratorio. A esta técnica se le denomina biopsia.

Es un proceso efectivo, que consiste en el estudio bajo microscopio de las células extraídas, y el patrón que constituye el tejido. Su proceso más complejo es la extracción, ya que si ésta no se realiza correctamente, ciertas células cancerígenas pueden no aparecer en la muestra y realizarse un diagnóstico erróneo. Además, debido a la falta de especialistas en la materia que puedan realizar las incisiones, personal médico no cualificado en esta materia suele realizar su extracción. Se estipula que en lesiones inflamatarias, el correcto estudio de la patología se da en el 77 % de los casos si la muestra es recogida por un dermátologo, y un 41 % si la realiza un ayudante [40].

Existen varias técnicas: mediante corte con tijera, mediante rasurado, extracción mediante bisturí, o en forma elíptica, persiguiendo la extracción total de la lesión. En el caso de las dos primeras opciones, solo está indicada si la lesión es superficial y no existe riesgo de que se trata de un melanoma por las complicaciones que esto conlleva de una posible diseminación y metástasis.

Aunque esta técnica suele ofrecer buenos resultados, debido a que la distinción entre un tumor de tipo melanoma y un simple nevus puede ser complicada, sería de interés conocer una evaluación previa. Y es aquí donde podemos ver la utilidad del proyecto propuesto: se busca reforzar el diagnóstico del experto utilizándose un modelo de aprendizaje.

También hay que tener en cuenta otros factores que pueden perturbar el diagnóstico tradicional, como lo es la correcta manipulación de la muestra extraída, y su

coloración adecuada para mejorar el contraste del tejido y poder distinguir el patrón descrito por las células y su núcleo. Además, la herida dejada en la piel puede sufrir complicaciones como hemorragias o infecciones si no se tratan adecuadamente. Reducir por tanto este tipo de operaciones a las estrictamente necesarias gracias a un modelo de aprendizaje profundo es una opción atractiva.

1.2. Justificación

Debido a la necesidad de facilitar el acceso al diagnóstico de forma orientativa, sin necesidad de medios tecnológicos especializados, surge la necesidad de algún medio de diagnóstico gratuito capaz de acelerar el proceso mediante la prognosis, y reducir la tasa de mortalidad de la enfermedad. Encontramos, de esta forma, un área de conocimiento poco explorada: la creación de una aplicación móvil, capaz de emplear modelos detección y prognosis de cáncer de piel, empleando mecanismos específicos para dispositivos de baja potencia.

Los smartphones, al tratarse de un dispositivo altamente extendidos en la sociedad, pueden servir como un medio clave a la hora de extender modelos de salud críticos como el de diagnóstico de enfermedades cancerosas. De esta forma, podríamos acortar los largos tiempos de espera, realizando un diagnóstico previo para declarar cuáles son los casos que requieren mayor prioridad médica.

Sin embargo, no existen modelos de este ámbito lo suficientemente ligeros como para ser portados de forma local dentro del propio terminal, que no necesiten de conexión a Internet, y que velen por la seguridad y privacidad de los datos. Esto convierte el problema en una interesante temática de investigación, la cual será tratada en este documento.

1.3. Objetivos

Teniendo en cuenta los factores estudiados acerca de la enfermedad en los puntos anteriores, y las carencias del estado del arte actual, podemos afirmar que el uso de un modelo de aprendizaje profundo en dispositivos de potencia reducida permite:

1. Facilitar la toma de decisiones de los expertos dermatólogos en casos complicados, a modo de sistema de ayuda a la toma de decisiones.

2. Defender la utilidad de los modelos de aprendizaje profundo en el estudio y evaluación de casos en el ámbito médico, siendo en este caso la identificación de la patología a nivel macroscópico, sin necesidad de realizar una biopsia con el posible riesgo que esto conlleva.

3. Crear una base de datos robusta y completa capaz de representar adecuadamente la población real de la enfermedad, partiendo, para ello, de la construc-

ción de un conjunto de datos mediante una técnica no explotada en el estado del arte.

4. Profundizar en el método de reducción y cuantización de modelos de aprendizaje profundo complejos, para su implementación eficiente en arquitecturas de bajo consumo.

5. Realizar, mediante un enfoque novedoso, la tarea de clasificación de enfermedades con dos niveles de modelos especializados: uso de clasificación binaria para prediagnóstico de la enfermedad, y especificación de la misma mediante modelos especializados separados para enfermedades benignas y malignas.

6. Uso de los principales frameworks de trabajo para la construcción de un modelo eficiente en su conjunto.

7. Estudiar la calidad de un modelo cuantizado frente a los resultados otorgados por un modelo estándar de escritorio.

8. Minimizar los costes, al hacer uso de un dispositivo esencial en nuestra vida diaria como por ejemplo, los teléfonos móviles, aprovechando la posibilidad de que la manifestación de los tumores de piel son visibles a nivel macroscópico.

9. Mostrar el desempeño de los modelos cuantizados y simplificados mediante el uso de una aplicación para dispositivos Android, capaz de ejecutarlos y mostrar su funcionamiento.

En este documento, estudiaremos el estado del arte actual, y analizaremos e implementaremos una aplicación para dispositivos móviles que sea capaz de realizar la segmentación de la mancha cutánea, y su posterior clasificación dentro de una lista de patologías posibles. Se busca informar al usuario del riesgo de su lesión, y de su posible diagnóstico final, el cual debe ser verificado por el experto. De esta forma, el software final puede contribuir a acelerar los diagnósticos de esta enfermedad, y favorecer la esperanza de vida de los pacientes en caso de que su lesión sea cancerosa.

1.4. Planificación

2 Tendencias y Estado del arte

Antes de adentrarnos en el análisis del problema, debemos de tener en cuenta de que este problema es una temática en constante evolución, y por tanto, podemos encontrar diferentes conceptos y procedimientos seguidos en la literatura que pueden servirnos de inspiración para abordar el problema sin cometer los errores ya cometidos en el pasado, y ser capaces de encontrar un nuevo enfoque que nos ofrezca ventajas.

Generalmente, este problema ha sido abordado empleando hardware de computador de escritorio, por lo que la mayoría de modelos se centran en el aprovechamiento de los recursos hardware alojados en un servidor para realizar la clasificación y evaluación de las imágenes potencialmente cancerosas tomadas. Por tanto, nos adentraremos en sus conceptos, pero teniendo en cuenta que el proyecto propuesto hará uso de dispositivos móviles durante el tiempo de inferencia.

2.1. Aprendizaje profundo en dispositivos móviles

Con la creciente tendencia de la potencia de cálculo en los dispositivos actuales, prácticamente todos los aparatos electrónicos que nos rodean han crecido en cuanto a potencia y complejidad de cálculo. Los smartphones son precisamente uno de ellos, y nos acompañan cada día, por lo que es el dispositivo ideal para tareas de uso cotidiano y portabilidad.

Estos dispositivos, a diferencia de los computadores tradicionales, normalmente basado en la arquitectura x86 o AMD64, se basan en ARM, siguiendo como concepto de diseño ofrecer el máximo rendimiento posible dentro de unos consumos contenidos,mejorando el ahorro de energía y la pérdida de la misma mediante calor. El entrenamiento de modelos que encontramos habitualmente en la literatura, como ResNet, Inception o similares, es prácticamente inviable de forma nativa.

Sin embargo, en lo que respecta a la inferencia, éstos son capaces de ofrecer muy buenos resultados, gracias a la incorporación de hardware dedicado capaz de ofrecer estas características. Como prueba, podemos observar infinidad de aplicaciones que hace uso de ello, como Google Lens, que si bien se ayuda del uso de servidores de búsqueda especializados, es capaz de realizar procedimientos locales en los dispositivos de gama alta.

El objetivo del proyecto es aprovechar dicho vacío en la existencia de apliaciones de inferencia local para ofrecer una app que no necesite de conexión de red

permanente para ofrecer resultados acerca de las manchas de piel identificadas.

Aprovechando el auge de los smartphones, grandes empresas, como Google y Meta, centran sus esfuerzos en la creación de arquitecturas basadas en redes convolucionales capaces en realizar detección de imágenes en tiempo real, para la clasificación de distintos objetos que podemos encontrar en nuestra vida cotidiana, y servir así como una herramienta de apoyo para diferentes necesidades. Sin embargo, esto es una tarea completa, ya que suelen carecer de complejas operaciones, o sacrificar en profundidad para lograr un rendimiento aceptable de unas decenas de milisegundos por inferencia.

A continuación, evaluaremos algunos de los modelos más conocidos y efectivos de propósito general, como SqueezeNet[35], MobileNet [32, 57, 31], ShuffleNet [70] y EfficientNet[63, 21].

2.1.1. Squeeze-Net (2016)

Squeeze-Net[35] se centra en la reducción de complejidad de la arquitectura, sin pérdida de capacidad predictiva y evitando aplicar técnicas de compresión y cuantización[38] de modelos. Se autodefine como "una red al nivel de AlexNet[37] pero con una quincuagésima parte de los parámetros", haciendo alusión a disponer de una capacidad de cálculo similar a AlexNet, pero recortando en cuanto a número de parámetros necesarios.

Aunque el motivo de su creación no es de forma directa el uso de la arquitectura en dispositivos móviles, ha sido ampliamente utilizada en ellos al formar parte de la tendencia actual de reducción de coste computacional para reducir las necesidades de potencia de cálculo. De esta forma, se puede facilitar la implementación de redes convolucionales en sistemas empotrados con escasa capacidad de memoria y cómputo, haciendo uso de FPGAs [42].

Siguiendo este punto de vista, fue capaz de igualar e incluso superar levemente el rendimiento de AlexNet[37], empleando las siguientes técnicas:

- Uso de módulos **Fire**. Se trata de un nuevo tipo de estructura convolucional modular que puede ser apilado en capas al estilo de los módulos Inception [62] de Google. Consiste en una unidad modular ajustable en función de 3 parámetros: el número de convoluciones 1x1, y el número de filtros 1x1 y 3x3 de "expasión" a aplicar. El objetivo de añadir las convoluciones 1x1 es, por un lado, la reducción de dimensionalidad del volumen a convolucionar, y por otro lado, la simplificación en número de parámetros. En arquitecturas de gran profundidad, como VGGNet o ResNet, quedó demostrado que este tipo de convoluciones permitían llegar más allá sin perder información relevante para el aprendizaje. 2.1.1

- Desplazamiento de los métodos de **reducción** de dimensionalidad hacia las capas más **profundas** de la arquitectura. En lugar de realizar pooling o aplicar

stride a la hora de aplicar el filtro para reducir el volumen de salida en las primeras capas de la red, este tipo de transformaciones se reparten en capas más profundas para evitar que las capas cercanas al Head, de forma que se reduce la pérdida de características si retrasamos el subsampling del filtro.

- Eliminación de capas totalmente conectadas. Estas capas son, normalmente, las que mayor complejidad añaden al modelo por su gran cantidad de parámetros. Gracias al uso de Average Pooling en su última capa, podemos tener una red completamente independiente del tamaño de la entrada sin gran cantidad de parámetros ni necesidad de capas adicionales.

Figura 2.1: Arquitectura de SqueezeNet

Esta red ha sido empleada en multitud de aplicaciones: detección de objetos en tiempo real, clasificación semántica, y modelos preliminares de conducción autónoma. Entre todas sus aplicaciones, podríamos destacar su utilización en imagen médica, concretamente MRI (Resonancias magnéticas). Ha permitido facilitar el diagnóstico de ciertas enfermedades y lesiones cerebrales en un espacio de memoria y recursos contenido.

Sin embargo, a pesar de las mejoras recibidas en sus versiones sucesivas, como los módulos Fire de doble nivel para reducir dimensionalidad, o la introducción de más reducciones mediante pooling, sacrifica resultados a nivel de accuracy respecto a la competencia, y no ha sido aplicada de forma firme y exitosa sobre imágenes de enfermedades cutáneas.

2.1.2. MobileNet

MobileNet es el fruto del proyecto de investigación de Google Research para la implementación de redes convolucionales en dispositivos móviles. El objetivo era encontrar un modelo eficiente que pueda ser incluso utilizado en tareas de segmentación en tiempo real, pero reduciendo el número de parámetros del red así como el número de operaciones de producto necesarias, para poder ejecutarlas de forma nativa en dispositivos móviles como smartphones y tablets.

Esta arquitectura consta de 3 versiones diferentes, siendo cada una más sofisticada que la anterior. Disponemos de MobileNet V1, MobileNet V2 y MobileNetV3.

MobileNet V1 (2017)

La versión original de la arquitectura convolucional MobileNet [32] fue publicada en 2017. En esta publicación, se busca reducir el número de operaciones realizadas para conseguir un menor impacto de las operaciones en punto flotante sobre el rendimiento.
El punto clave de esta arquitectura reside en las llamadas "pointwise convolutions", haciendo uso del concepto de separabilidad, ampliamente estudiado desde el año 2012 por la literatura.

Las nuevas convoluciones descomponibles se pueden separar en dos pasos bien delimitados: la convolución en profundidad y la convolución puntual.

- Las convoluciones en profundidad realizan el producto del filtro con el volumen de entrada capa a capa. Es decir, no se tiene en cuenta la dimensionalidad total de la imagen, sino que se realiza por cada nivel de profundidad el mismo producto. Esto reduce considerablemente el número de parámetros, ya que la dimensionalidad del problema es mucho menor.

- La segunda fase es la convolución puntual, cuyo objetivo no es más que acumular el producto de todas las capas calculadas independientemente mediante una simple combinación lineal, la cual es de coste computacional muy bajo.

$$G_{k,l,m} = \sum_{i,j} K^{i,j,m} * F_{k+i-1,l+j-1,m}$$

Este producto es calculable eficientemente por las técnica de álgebra lineal GEMM, que permite aplicar propiedades de la suma y la multiplicación para el producto matricial de forma eficiente mediante Tensor cores.

En resumen, gracias a la separabilidad convolucional, se adquieren varias ventajas:

- El número de productos se reduce considerablemente. Como se puede verificar en [32] se traduce en una reducción de entre 8 y 9 veces el número de operaciones con respecto a las arquitectura tradicional de convolución
- Se reduce el espacio necesario en memoria.
- Se puede aprovechar el hardware específico.
- No se pierde precisión de cálculo gracias a que la separabilidad de convoluciones no afecta al resultado.

(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figura 2.2: Producto depthwise

Adicionalmente, también incorporaron dos hiperparámetros: α y ρ. Estos parámetros controlan la anchura y la resolución de entrada, respectivamente.
El hiperparámetro α hace referencia a la anchura de cada capa convolucional que compone la red, adquiriendo un valor de 1 cuando la arquitectura no se ve reducida, y gradualmente podrá ser reducida en el intervalo (0,1]. Así se conseguirán modelos más simples en anchura para dispositivos con menores recursos.

En cuanto a ρ, este se encuentra implícito en la resolución de la imagen de entrada. Por defecto, la red acepta imágenes de hasta 224x224, pero en función de dicho valor ρ, podremos reducir su resolución también dentro del intervalo (0,1].

Ambos parámetros sacrificarán bondad y ajuste en los resultados a favor de una mayor eficiencia.

MobileNet V2 (2018)

Dos años más tarde de la publicación de MobileNets, da a luz su versión V2 [57]. Esta conserva los hiperparámetros de la versión anterior, así como el producto punto a punto. Sin embargo, añade tres nuevas características, algunas de ellas no triviales y que requieren experimentación:

- Se introduce el concepto de "residuo invertido". Ésta mejora reside en la utilización de los bloques residuales, propuesto por la arquitectura de ResNet. Su objetivo es evitar la degradación del gradiente, y que se frene el aprendizaje en modelos de gran profundidad. Normalmente, estas conexiones se realizan entre capas de gran profundidad, siendo las capas intermedias bloques estrechos. Sin embargo, en MobileNet V2, se propone la composición inversa, de forma que sean los bloques intermedios entre los residuales aquellos que poseen una mayor anchura, y así reducir el número de parámetros sin perder expresividad en el modelo [51].

Figura 2.3: Bloque residual invertido

- En las capas donde el volumen de entrada es estrecho, al hacer uso de bloques residuales invertidos, la eliminación de la no linealidad aportada por ReLu favorece a la conservación de características y permite obtener mejores resultados de accuracy en tareas generales como clasificación en imagenet. Esto se debe a que al realizar los saltos entre bloques "estrechos" perdemos rendimiento de la red, y simplemente con eliminar la última transformación no lineal del bloque, contrarrestamos este problema.

- ReLu6. Se mantiene una versión modificada de la original función de activación. En lugar de utilizar la tradicional función ReLu entre 0 y 1, se extiende este intervalo hasta 6, permitiendo mantener la precisión en caso de utilizar coma fija, ya que se aseguran 3 dígitos de parte entera, y el resto queda destinado a la mantisa, que se almacena de forma precisa.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figura 2.4: Arquitectura de MobileNetV2

MobileNet V3 (2019)

En su tercera versión[31], MobileNet incorpora métodos avanzados de diseños de redes basados en NetAdapt. Este algoritmo se basa en la transformación de modelos preentrenados para escritorio, y, en base a una serie de requisitos de potencia especificados, adapartar la arquitectura a una plataforma móvil perdiendo las mínimas capacidades posibles de la red original. El modelo de partida empleado fue ajustado con precisión para mejorar latencias y uso de memoria, aplicando los siguientes conceptos:

- Se añade la capa Squeeze-and-Excite, dentro de las conexiones residuales. Se trata de un mecanismo surgido en 2018 [34]. Este estudio afirma que existen filtros de imagen con mayor importancia para el cómputo global que otros, como, por ejemplo, los bordes. Por tanto, les aporta un mayor "peso" durante el entrenamiento a dichos filtros haciendo uso de una serie de parámetros adicionales. Éstos añaden una carga computacional muy pequeña, por lo que se trata de una técnica eficaz. Para obtener los parámetros de relevancia, se dispone de dos módulos: squeeze, y excite. El módulo squeeze se encarga de representar cada filtro mediante un valor numérico, obtenido por average pooling de la imagen. Y por otro lado, el módulo excite se encarga de aprender los pesos que dar a cada uno de estos filtros o canales, haciendo uso de un MLP. El resultado final serán los pesos de cada canal en cuanto a su importancia, normalizados entre 0 y 1 por una función sigmoide.

- Se incluyeron nuevas capas al inicio y al final de la red de tipo residual invertidas.

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Table 1. Specification for MobileNetV3-Large. SE denotes whether there is a Squeeze-And-Excite in that block. NL denotes the type of nonlinearity used. Here, HS denotes h-swish and RE denotes ReLU. NBN denotes no batch normalization. s denotes stride.

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

Table 2. Specification for MobileNetV3-Small. See table 1 for notation.

Figura 2.5: Arquitectura de MobileNetV3

Debido a la gran complejidad adquirida por el modelo, los problemas de latencia y rendimiento en dispositivos de menor potencia, se opta por dividir la arquitectura en dos modelos parametrizables: MobileNet Small y Large. Mientras que la versión Large mejora los resultados de la versión 2 aumentando las prestaciones, el modelo Small otorga importancia sobre todo a la eficiencia y el uso de memoria, enfocado al hardware embebido o dispositivos de poca potencia.

Aplicaciones en dermatología y cáncer de piel

MobileNet, concretamente en su segunda versión, ha sido utilizado en la literatura para el diagnóstico de enfermedades de la piel. En [11], es utilizado para realizar la clasificación de 7 enfermedades cutáneas extraídas del Humans against Machine, HAM10000 [66], que podemos encontrar en ISIC archive [4], un repositorio web de acceso libre con enfermedades de la piel tanto beningnas como cancerosas.

También se utilizó más recientemente para su implementación en dispositivos de IOT, [59], donde se logra alcanzar el 99 % de accuracy en un pequeño conjunto extraido de ISIC, haciendo uso de la versión V3 junto a un algoritmo de Squeeze. Dicho algortimo se encarga de localizar la ubicación de los pelos y otros posibles artefactos para preprocesar la imagen y lograr una fotografía resultante libre de interferencias.

Para ello, hace uso de un filtro black hat, que binariza la imagen y obtiene los píxeles objetivo de eliminar, que son sustituidos por los colores de los píxeles adyacentes, junto al uso del aumento de datos. Sin embargo, no queda especialmente claro la tasa de precisión del modelo para cada una de las enfermedades que se intentan diagnosticar.

2.1.3. Shuffle Net

Shuffle Net [70, 65] surge con el objetivo ofrecer un modelo capaz de ofrecer un buen modelo con la mínima pérdida de rendimiento frente a modelos profundos. Es capaz de superar en resultados a la primera versión de MobileNet, logrando un error de aproximadamente tres puntos menos que MobileNet V1. Su mejor rendimiento se debe sobre todo al uso de Channel Shuffle para las convoluciones grupales, y creando una arquitectura basada en módulos shuffle.

La convolución en grupo mediante mezcla de canales (Channel Shuffle) surge tras el estudio del funcionamiento de las convoluciones grupales en AlexNet[37] y ResNext. En ambos modelos, se uitilizan convoluciones grupales, donde cada canal de salida sólo se relaciona con los canales de entrada del que proviene. Esto podría debilitar la relación entre cada canal, y debilitar los resultados; para evitarlo, y no poner en riesgo el rendimiento con demasiadas convoluciones 1x1 para relacionarlos, se hace uso del mezclado de canales; es decir, es como si permitiésemos que cada grupo convolucional pudiera obtener información de otros grupos adyacentes, para así mejorar la relación entre ellos.

Para evitar un sistema complejo a la hora de representar dichas interconexiones, es como surge el channel shuffle 2.6: se mezclan los canales de forma que los grupos ya no quedan aislados con sus respectivas entradas y salidas.

Figura 2.6: Channel Shuffle de ShuffleNet

Esta técnica se aplicará sobre la primera y última convolución 1x1 realizada sobre los bloques residuales de la red, que siguen una estructura parecida a la que adoptaría MobileNetV2 posteriormente. Mediante esta propuesta, podemos además aplicar una mayor capacidad de procesamiento en anchura añadiendo stride, y aplicando average pooling. El resultado, es conseguir modelos más anchos en procesamiento que no impacten negativamente en el rendimiento de los dispositivos con menor capacidad. En la figura 2.7 , podemos apreciar la arquitectura de la red, cuyos filtros

pueden ser escalados mediante el parámetro s, aunque teniendo en cuenta una penalización en la complejidad, equivalente a s^2 sobre Shuffle Net base, equivalente a $s = 1$

Layer	Output size	KSize	Stride	Repeat	Output channels (g groups)				
					$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 8$
Image	224×224				3	3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24	24
MaxPool	56×56	3×3	2						
Stage2	28×28		2	1	144	200	240	272	384
	28×28		1	3	144	200	240	272	384
Stage3	14×14		2	1	288	400	480	544	768
	14×14		1	7	288	400	480	544	768
Stage4	7×7		2	1	576	800	960	1088	1536
	7×7		1	3	576	800	960	1088	1536
GlobalPool	1×1	7×7							
FC					1000	1000	1000	1000	1000
Complexity					143M	140M	137M	133M	137M

Figura 2.7: Arquitectura de ShuffleNet

Sus aplicaciones han sido variadas, pero en lo que respecta a la detección de lesiones cutáneas, la cercana salida de MobileNet V2 y su mejor rendimiento provocó que ShuffleNet quedase relegada a un segundo plano, y no fuese muy utilizada para este fin. Podemos encontrar algunos trabajos [17] donde podemos observar una comparativa de este modelo frente a la completitud de los modelos del estado del arte de 2022, y podemos confirmar que MobileNetV2 es capaz de superar su rendimiento en la mayoría de pruebas, siendo estas comparaciones en cuanto a tiempo de entrenamiento, precisión, accuracy y tamaño del conjunto de entrenamiento. Sólo consigue superar a MobileNetV2 en tiempo de entrenamiento, donde es aproximadamente 900 más rápida, pero ofrece peores resultados en promedio.

2.1.4. EfficientNet

EfficientNet [11] es un conjunto de arquitecturas de redes creadas por el departamento de investigación de Google con el fin de conseguir una familia de modelos variada que fuese capaz de adaptarse fácilmente mediante parámetros a diferentes conjuntos de imágenes, y a requisitos de hardware más o menos limitados.

Parte de que una red convolucional sigue el siguiente esquema:

$$\mathcal{N} = \odot_{i=1\ldots s} F_i^{L_i}(X_{(H_i,W_i,L_i)})$$

Donde se denota que la capa F_i es repetida L_i veces la etapa i de la red, y la dimensionalidad de la capa queda representada con (W_i, L_i). Fijando F_i, efficient net intenta dar versatilidad a sus modelos variando las dimensiones restantes, Li, Ci, Hi, Wi mediante el uso de 3 constantes de escalabilidad 2.8:

- Profundidad, Depth (d): Aumentar la profundidad es la tendencia habitual presente en las redes convolucionales. Pero llegar a un equilibrio es crítico, ya que aumentar demasiado la profundidad sin modificar otros parámetros puede ocasionar pérdidades rendimiento por el desvanecimiento del gradiente a menor profundidad.

- Anchura, Width (w): Aumentar la anchura suele ser beneficioso para modelos de pocos recursos donde el aumento de profundidad supone un gran aumento de la carga computacional. Permite conseguir mayor cantidad de características de grado fino, pero si la red es demasiado poco profunda, el modelo careceré de características de alto grado que permitan aprender patrones generales.

- Resolución, (r): al emplear tamaños de entrada mayores, damos opciones a obtener una mayor cantidad de características de grado fino, pero un exceso de resolución puede provocar grandes tiempos de ejecución y puede ser contraproducente, al reducirse la ganancia con tamaños demasiados grandes.

Figura 2.8: Parámetros de EfficientNet

Experimentalmente, estos parámetros pueden ser ajustados, y dan lugar a una serie de modelos distinto: los conocidos EfficientNetB0 - B7, denotando el valor numérico la profundidad y complejidad del modelo, siendo esta mayor a mayor valor del índice. Cada una de ellas fue ajustada utilizando una profundidad concreta, y puede ser posteriormente ajustada con el resto de parámetros libres no fijados a las características del conjunto de entrada.

En smartphones de alta gama, los modelos B0 a B4 pueden ser ejecutados con un rendimiento aceptable para aquellas aplicaciones que no requieran un alto tiempo de respuesta, pero si necesitan dar al usuario una respuesta aceptable. Para modelos

de mayor complejidad computacional, se usan las variantes lite, que estudiaremos más adelante.

Aplicaciones en dermatología y cáncer de piel

En el problema que nos concierne, esta arquitectura ha conseguido grandes resultados en el dataset del ISIC abierto al público como competición en la plataforma Kaggle, habiendo sido utilizado como parte de un ensemble de modelos, o bien como modelo único entrenado en el top 3 de ganadores de la competición. En el caso de la segunda mejor solución clasificada [47], se menciona la utilización de EfficientNet-B6, con tamaño de entrada de 512x512, y un tamaño de batch de 64, obteniendo 0.9485 de accuracy como resultado final a la hora de emplear los datasets ISIC 2019 y 2020
En la primera solución, es usada en conjunto a Resnet50, y una red especializada en los metadatos de la imagen, y todos los modelos juntos someten su resultado a votación [15].

Ambos resultados han sido evaluados con computadores de alta gama, haciendo uso de múltiples tarjetas gráficas para el entreanmiento y la inferencia. Este proceso es demasiado pesado para un dispositivo móvil, por lo que de cara a este trabajo, se buscarán alternativas capaces de ahorrar en espacio y potencia como concepto de cuantización.

2.2. Cuantización de modelos

Las arquitecturas y soluciones propuestas por la literatura que han sido analizadas en los apartados anteriores proponen métodos que ofrecen buenos resultados. Sin embargo, existe un problema: todas ellas han sido entrenadas con un computador cuya capacidad de cálculo supera incluso las características de un ordenador doméstico promedio, como en [47], donde se emplean 4 GPU Nvidia Quadro RTX 6000 24GB. Aunque su utilización engloba sobre todo el proceso de entrenamiento, la inferencia de estos modelos también sigue siendo extremadamente costosa para un dispositivo móvil, y este proceso también es realizado a través del computador.

Esto resume a que el teléfono simplemente adquiere el papel de cliente dentro de una arquitectura cliente-servidor, donde el dispositivo host de todo el procesamiento de la imagen y de su clasificación es un ordenador de gran potencia de cálculo, y el teléfono movil únicamente debe compartir con este la imagen que desea examinar. Pero esto supone una gran desventaja: en ausencia de conexión de red, o interrupción del servicio por parte del servidor, el usuario no sería capaz de emplear la aplicación para el diagnóstico. La dificultad e interés de este proyecto se ve reforzado por este argumento, y resulta ahora de interés la posibilidad de realizar la inferencia en el teléfono, a pesar de que el entrenamiento sea realizado en un

ordenador.

Existe una técnica que nos permitirá obtener un modelo optimizado a partir de uno entrenado de forma tradicional: la cuantización.

2.2.1. Características

La cuantización de modelos se centra en la simplificación y optimización de un modelo preentrenado por un computador, reduciendo algunas de sus características buscando un impacto mínimo sobre los resultados obtenidos, pero reduciendo de forma considerable el tiempo de inferencia para dispositivos de baja potencia. Aunque existen mecanismos de poda, donde la arquitectura del modelo se ve simplificada de forma directa, el método que evaluaremos será la cuantización de modelos basada en la reducción de la precisión numérica, es decir, una disminución en la profundidad en bits de la variable flotante.

Este concepto ya fue brevemente mencionado durante el desglose de características de MobileNet V3 [31]. En la arquitectura AMD64, el almacenado de variables en coma flotante emplea una precisión de 32 bits. Por tanto, al realizar operaciones de cualquier tipo, las 32 posiciones del nuevo número han de ser actualizadas al valor resultado. El tiempo empleado en realizar dicha operación en cada dígito de este número puede parecer despreciable, pero, cuando el conteo de operaciones alcanza los miles de millones, supone una diferencia significativa en el rendimiento.

En dispositivos de baja potencia, esta precisión suele verse reducida a 8 bits, de forma que reducimos la longitud del máximo número almacenable en 4 veces menos, y reducimos también así el tiempo por operación.

Este mecanismo es empleado tanto por los frameworks de trabajo habituales para aprendizaje mediante redes convolucionales, como Pytorch y TensorFlow, y por los propios fabricantes de teléfonos móviles de forma nativa. Es el caso de Qualcomm, conocido por su gama de procesadores Snapdragon. Esta empresa realizó un estudio del impacto de la cuantización en los modelos [38] usando flotantes de 8 bits del precisión. Demuestran que el uso de números flotantes de 8bits en lugar de enteros con exponenciación para desplazar el punto ofrece un mayor rendimiento. Las consideraciones a la hora de defender la utilidad de la cuantización son las siguientes:

- La operación más costosa realizada durante la inferencia es el producto de matrices. Para simplificar la complejidad de la operación, normalmente se suelen emplear matrices de números enteros reescalados a un nuevo intervalo de valores. Esto es, se aplica una transformación de rangos, donde una matriz $\mathbb{R}^{mxn}$ se cuantiza a una matriz $X^{(int)}$ asociado a un valor de reescalado s: $X^{(int)} = clip([\frac{X}{s}], x_{min}, x_{max})$, donde el producto pasa a ser el valor redondeado más cercano al número tras aplicar la transformación de rango mediante escala, y la operación de clip asegura que el número es representable dentro de los valores establecidos para el extremo. Esto proporciona una simplificación

contra la habitual notación IEEE-754 de 32-bit, FP32, donde se usa un bit de signo, 23 de mantisa, y 8 bits de exponente.

- La utilización de FP8 puede ser beneficiosa al ser capaz de encontrar un término medio entre la precisión y variabilidad de rango numérica del resultado, como se puede ver en [38]. Los resultados empíricos muestran que tras un ajuste entre el número de bits de mantisa y exponente, se logra un equilibrio adecuado usando valores cercanos a 3 bits de mantisa y 4 de exponente, y en caso de post training quantization, 5 de mantisa y 2 de exponente, en caso de que el aprendizaje se realice de forma lenta.

- Para mejorar estos resultados, se puede usar la técnica Quantization Aware Training (QAT), o entrenamiento preparado para cuantización, donde el objetivo es facilitar el paso a FP8 desde el proceso de entrenamiento teniendo en cuenta los rangos de valores adoptados durante el entrenamiento, como por ejemplo, controlando el diferencial mínimo que puede tomar el algoritmo de optimización. Esta operación muestra mejor resultado sobre valores enteros, pero también es capaz de mejorar los resultados obtenidos por PF8.

En los frameworks Pytorch y TensorFlow, podemos encontar estas optimizaciones en sus funcionalidades relacionadas con la optimización de modelos en post entrenamiento, con las dependencias de Pytorch Mobile y TensorFlow lite, respectivamente. Ambas aplican cuantización numérica de los tensores, y en caso de tratarse de modelos tradicionales de la literatura, existen arquitecturas simplificadas de forma profunda, con redes como EfficientNet o ResNet, que cuentan con variantes lite.

2.2.2. Modelos cuantizados

El framwork de TensorFlow lite contiene dos de los modelos más utilizados en la literatura en sus variantes lite: EfficientNet Lite y ResNetV2. Ambos han recibido un tratamiento de simplificación, que pasa por el uso de INT8 como modelo de representación numérica y la simplificación de la arquitectura, retirando algunas de las capas más resentidas en el proceso de optimización. En el caso de EfficientNet Lite, podemos encontrar un descripción detallada sobre aquellos cambios realizados para la mejora de rendimiento [54]:

- Se hace uso de cuantización post entrenamiento mediante el algoritmo de cuantización empleado por Tensorflow lite. Esta cuantización se hace de forma dinámica, dependiendo del hardware en el que se ejecuta el modelo. Puede realizarse una cuantización de rango dinámico, donde el factor de escalado puede aplicarse de forma distinta dependiendo de la capa en la que se realice la operación; cuantización de enteros, para los dispositivos menos potentes; y

por último, cuantización de FP16, de forma que se logra un término medio entre ambas soluciones, y se puede emplear en dispositivos con GPU de potencia considerable. Por defecto, es la opción de rango variable la utilizada.

- Se eliminan algunos módulos Squeeze and Excite presentes en capas intermedias, debido a su coste e imprecisión para dispositivos que no soportan gran precisión numérica.

- Las funciones de activación se reemplazan con RELU6, al igual que se realizó en MobileNetV3 [31]

Figura 2.9: Optimización de EfficientNet Lite

Este procedimiento variable, con múltiples posibles combinaciones, se realiza debido a la gran variedad del mercado actual. Existe una gran diversidad de dispositivos: conviven dispositivos de bajo consumo con escasa potencia gráfica, así como terminales especializados con unidades de TPU para acelerar el cálculo neuronal. Y, de esta forma, con optimizaciones modulares, podemos priorizar la eficiencia en dispositivos con poca capacidad de cálculo , y aumentar la precisión en aquellos que pueden ejecutarlos. En el dispositivo de ejemplo, un Google Pixel 4, se consigue un tiempo medio de inferencia de 30 ms, una mejora de casi el 60 % sobre ResNet50.

2.2.3. Conclusión de los modelos cuantizados

A la vista de los resultados, los modelos cuantizados pueden suponer una ventaja; obtenemos resultados cercanos a los modelos complejos del estado del arte, con una pequeña penalización ganada en forma de rendimiento, y que la ejecución ofrezca tiempos de respuesta razonables. Además, este tipo de transformaciones no se han empleado para el reconocimiento y detección de enfermedades de la piel, y abren una nueva linea de investigación en la que obtener resultados.

Como puntos en contra, debemos tener en cuenta que la penalización obtenida al entrenar este tipo de modelos no es siempre la misma; dependiendo del problema a clasificar, y del modelo seleccionado, la penalización de la cuantización puede ser o no más marcada; para ello, es necesario obtener empiricamente resultados que nos permitan conocer el tipo adecuado para nuestro problema. Este conflicto para encontrar el equilibrio ya fue detectado por los desarrolladores de Google en EfficientNet Lite [21, 54], donde inicialmente, sus pruebas en ImageNet arrojaron resultados pésimos al obtener un 46 % de accuracy en clasificación TOP1 a diferencia del 75.1 % obtenido con FP32. Pero, tras ajustar el rango de amplitud de los enteros utilizados para cuantización, se consiguió una ganancia de 1.85x sobre el modelo original con tan solo un 0.7 % de pérdida en los resultados.

	float32	int8	Improvement
model size	17.7MB	5.17MB	3.42x
accuracy (top1)	75.1%	74.4%	-0.7 percent point
latency (CPU)	12ms	6.5ms	1.85x

* Benchmarked on Pixel 4 CPU with 4 threads

Figura 2.10: Ganancia de la cuantización en EfficientNet

Como aspecto negativo, aunque la varianza de los resultados pueda verse paliada mediante la selección de la cuantización más adecuada, aún existen riesgos de resultados inesperados. Con este mismo modelo, por ejemplo, se han alcanzado resultados inesperados, como que los modelos de menor tamaño (B0,B1,B2) poseen un mejor desempeño que B3 y B4 en datasets de propósito general, como podemos observar en el artículo de Agarwal [6], pero esto puede deberse por la necesidad de una mayor cantidad de datos para realizar el entrenamiento.

Dado a los buenos resultados en general, la cuantización será la senda seguida por este TFG a la hora de optimizar los modelos de escritorio en post entrenamiento.

2.3. Conjuntos de datos disponibles

La obtención de datos para el proceso de aprendizaje automático es una fase crucial para la obtención de un modelo correctamente entrenado y que proporcione resultados de calidad. Esta fase es un factor común en todos los problemas de aprendizaje, ya que a mayor diversidad y cantidad de datos, más cercana será la muestra de la que disponemos y, por ende, más cercanos serán los resultados al diagnóstico real. Múltiples puntos de vista, tonos de iluminación, y otro tiempo de variantes pueden ser beneficiosas.

Para el diagnóstico del cáncer de piel, este proceso es igualmente importante. La piel es el órgano más extenso del cuerpo humano, y debido a que es el área de mayor esposición diaria a los elementos, presenta una gran variedad en cuanto pigmentación, porosidad, y otros elementos influyentes como el envejecimiento o el daño provocado sobre ella. Esto provoca que el diagnóstico de las lesione sea complejo, pues existen tanto lesiones benignas como malignas de gran similaridad. Además, cada localización geográfica del mundo posee diferentes tonos de piel., claros y más oscuros. La inexistencia de un tipo de piel o de lesión en el conjunto de entrenamiento podrían provocar resultados sesgados indeseados durante la predicción de la imagen tomada.

En la red, podemos encontrar varios reposotorios de imágenes cutáneas de acceso público, que permiten su utilización de forma abierta con fines académicos para el desarollo de modelos de inteligencia artificial. Sin embargo, no todos ellos son indicados para su uso, ya que el control realizado sobre el diagnóstico y la certeza de los mismos puede ser verificada en mayor o menor medida, y necesitamos que estos sean los más fiables posibles. Es crítico desarrollar un plan de búsqueda que nos facilite la decisión de tomar o desechar un conjunto de datos que encontremos.

Para facilitar este proceso, podemos apoyarnos en diferentes artículos recientes donde se analizan los conjuntos de datos públicamente accesibles en la actualidad, y conocer la calidad de la imagen, el porcentaje de fiabilidad del etiquetado y el balance de clases. Destacamos el estudio realizado por M. Goyal et Al. [23], o el reciente artículo de Sana Nazari y Rafael García [44], donde se analiza la disponibilidad de los datasets a fecha de 2023.

Basado en sus estudios y recomendaciones acerca de los conjuntos de datos disponibles, he elaborado un diagrama de flujo a seguir para evaluar qué conjuntos decartar y cuales anotar para su uso en el entrenamiento:

Figura 2.11: Proceso de búsqueda seguido

Siguiendo dicho procedimiento, se han encontrado 6 datasets diferentes, cuyo origen son instituciones públicas que han cedido datos con fines académicos y de investigación.

Un factor determinante para la elección de estos conjuntos es la diversidad. Es indispensable, para este proyecto, encontrar datos lo suficientemente variados como para distinguir lesiones cancerígenas y no cancerígenas, y disponer de diferentes tonos de piel para entrenar. Aunque los tonos de piel más oscuras sufren lesiones de tipo cancerígeno en menor proporción gracias a su protección natural, también pueden sufrir este tipo de patologías, y es clave ser capaces de detectarlas en cualquier posible paciente.

2.3.1. International Skin Imaging Collaboration (ISIC)

El repositorio ISIC (International Skin Imaging Collaboration [4]), es un repositorio en línea sin ánimo de lucro que recoge datos procedentes de diferentes instituciones

públicas para elaborar un atlas online donde obtener datos para mejorar el diagnóstico de enfermedades cutánea y ofrecer un conjunto de imágenes accesible para la Inteligencia aritificial.

Dispone de un buscador de imágenes en el que se pueden filtrar las enfermedades de las que se deseen obtener imágenes o metadatos para su estudio. En su mayoría, las imágenes que podemos encontrar son de enfermedades benignas, pero la web hace especial énfasis en disponer de imágenes demoscópicas de lesiones principalmente cancerosas. En la literatura, se trata de uno de los conjuntos de datos más citados, debido a su utilización anual durante los años 2016-2020 para la realización de un reto virtual de Machine Learning descrito.

El objetivo era modificada cada año, pero este oscilaba entre la identificación de melanomas frente a lesiones no cancerosas, como se propone en el ISIC Challenge-2020 [55], o bien, identificar diferentes subtipos de lesiones cancerosas frente a lesiones benignas en la competición ISIC Challenge 2019, [66, 12, 13], cuyo propósitos se acerca más al objetivo de este trabajo.

En el estado del arte, destacan las soluciones que hacen uso de métodos de Deep Learning, como el planteado por Ian Pan [47]] o el ya analizado anteriormente ganador del año 2020 [15], que realiza un enfoque híbrido por votación entre modelos que hacen uso de redes convolucionales para la clasificación de los datos de tipo imagen, y clasificación mediante Machine Learning para los metadatos.

Debido a que cada uno de estos subconjuntos de datos anuales eran escasos para modelos profundos, normalmente se ha recurrido a la reutilización de los conjuntos anteriores para enriquecer el conjunto de entrenamiento. Se trata de una técnica que permite obtener buenos resultados ya que, a mayor conjunto de entrenamiento, más cercana será la muestra a la población real. Hay que prestar especial atención a la repetición de datos, ya que la aparición duplicada de imágenes podría provocar un sobreaprendizaje sobre dicho ejemplar, además de posibles fugas de información entre entrenamiento, validación y test, al poder existir imágenes exactamante iguales entre ellos.

Fusión y deduplicado

Podemos encontrar una análisis del procedimiento de unión de los subconjuntos en [10]. En este artículo, se realiza una análisis exhaustivo de los datos disponibles, cubriendo aspectos como métodos de deduplicado de imágenes, y una clasificación de posibles modelos a aplicar.

Si desglosamos el conjunto de datasets disponibles, nos encontramos con 5 conjuntos:

Challenge Dataset Year	Train	Test	Total	Tipo de problema
ISIC 2016	900	379	1279	Clas. binaria
ISIC 2017	2000	600	2600	Clas. Multiclase
ISIC 2018	10015	1512	11527	Clas. Multiclase
ISIC 2019	25331	8238	33569	Clas. Multiclase
ISIC 2020	33126	10982	44108	Clas. Binaria

- ISIC 2016 [24]: Es el dataset de menor tamaño de todos los propuestos. Hace distinción únicamente de los casos malignos y benignos. Contiene imágenes dermoscópicas anotadas con información acerca de la localización de la mancha, y la edad del paciente. Contiene información adicional para la segmentación de la mancha pigmentada de interés (máscaras).

- ISIC 2017 [12] Es un conjunto de mayor tamaño al anterior, y hace alusión a 4 clases diferentes: melanomas, nevus, y seborheic keratosis. Contiene también información acerca de la edad del paciente, y otros metadatos de interés. La escasa cantidad de datos provoca que normalmente se recurra a aislar distintas clases del conjunto entre sí, y realizar comparaciones de tipo binario entre cada par de clases.

- ISIC 2018 [66, 12, 13]. Este dataset contiene un número de imágenes considerable, siendo un total de 10015 imágenes para entrenamiento, y 1512 para test. En este caso, se realiza subclasificación de tipos, a través de las clases melanocytic nevus, basal cell carcinoma, actinic keratosis, benign keratosis, dermatofibroma y lesiones vasculares. Es de especial interés destacar que este dataset proviene, a su vez, de HAM10000 (Human against machine, [66]) y MSK Dataset. El challenge original comprendía, de nuevo, la clasificación de los diferentes tipos realizando previamente una discriminación de la mancha en cuestión mediante segmentación. Existen gran cantidad de publicaciones que tratan este conjunto de datos, como [8], donde se emplea este dataset para demostrar mejores resultados al emplear transformaciones polares de la imagen y aumentar la invarianza.

- ISIC 2019 [66, 12, 13]. Se trata del mayor conjunto de datos para clasificación multiclase propuesto por ISIC. Este conjunto es la unión del empleado para el año 2018, con la adición de BCN_20000 Dataset [13], cuyos datos provienen del Hospital Clínic de Barcelona. Las clases a clasificar se amplían hasta 9, encontrando dos subtipos de melanomas en el conjuto: aquellos que muestran sangrado, frente a los que no.

- ISIC 2020 [55]. El último dataset propuesto públicamente, contiene únicamente datos binarios acordes a melanomas y lesiones no malignas. Esta construido principalmente de los datasets mencionados anteriormente.

Todos estos datos pueden ser acoplados entre sí para dar un dataset global de ISIC[10], donde obtendríamos las siguientes clases:

Clase	2017	2018	2019	2020
Melanoma	374	1113	4522	584
Atypical melanocytic proliferation	-	-	-	1
Cafe-au-lait macule	-	-	-	1
Lentigo NOS	-	-	-	44
Lichenoid keratosis	-	-	-	37
Nevus	-	-	-	5193
Seborrheic keratosis	254	-	-	135
Solar lentigo	-	-	-	7
Melanocytic nevus	-	6705	12.875	-
Basal cell carcinoma	-	514	3323	-
Actinic keratosis	-	327	867	-
Benign keratosis	-	1099	2624	-
Dermatofibroma	-	115	239	-
Vascular lesion	-	142	253	-
Squamous cell carcinoma	-	-	628	-
Other / Unknown	1372	-	-	27.124
Total	2000	10.015	25.331	33.126

Sin embargo, sería necesario tener en cuenta la eliminación de imágenes repetidas, debido a que durante cada edición de ISIC, un número considerable de imágenes han sido incluidos en varios años. Este procedimiento engloba:

1. Eliminar las imágenes idénticas por hash. Todas las imágenes de ISIC están numeradas de forma única para facilitar la identificación de cada una de ellas. Si unimos todos los datatesets, y tomamos las repeticiones, podemos remover:

	2016	2017	2018	2019	2020
Train	291	1283	0	0	0
Test	95	594	0	0	0

Tabla 2.1: Número de imágenes duplicadas encontradas [10]

2. Eliminación del ISIC 2018. Como éste se encuentra contenido en la composición para el año 2019, puede prescindirse totalmente de él a favor de la versión de 2019.

3. Eliminación de imágenes "downsampled" del conjunto. En los años 2019 y 2020, se añadieron imágenes de challenges anteriores con una reducción en

resolución. Para ahorar en espacio y tiempo de cómputo, pueden eliminarse las imágenes reducidas para quedarnos con una única copia de mayor calidad de la lesión, y luego realizarles manualmente un reescalado en caso de que sea necesario.

Atendiendo de nuevo a los resultados propuestos por [10], obtenemos la tabla 2.2, donde se recogen las imágenes removidas y restantes.

Year	Task No.	Images Removed	Images Remaining
2016	3	826	74
2017	3	801	1199
2018	3	10,015	0
2019	1	2235	23,096
2020	-	433	32,693
Total	-	14,310	57,0621

Tabla 2.2: Tabla de imágenes únicas extraída de [10]. En este caso, el autor descarta el uso del dataset de 2016 por su baja aportación

Obtendríamos un total de 57000 imágenes, los cuales podrían clasificarse, con sus respectivas clases extraídas de los metadatos. Componen, en resumen, un conjunto de datos robusto que puede formar parte del dataset de entrenamiento de este trabajo.

Figura 2.12: Ejemplo de imágenes de ISIC 2017

Uso de la galería de ISIC

Como ya se mencionó anteriormente, el repositorio de ISIC contiene una galería donde encontrar imágenes y metadatos de diferentes enfermedades diagnosticadas por especialistas dermatólogos, y verificadas en gran cantidad de casos por biopsia. Este conjunto de datos, en la actualidad, puede ser más atractivo que la fusión de los 5 datasets descritos anteriormente, ya que desde el año 2020, no les ofrece ninguna mejora o procesado.

En total, podemos encontrar casi 80000 imágenes disponibles, siendo un total de 53781 imágenes correctamente etiquetadas y revisadas. Aunque se trata de unas 3500 imágenes menos que las competiciones en conjunto, la variedad de enfermedades e imágenes es mayor, ya que no se trata de las mismas imágenes empleadas en las competiciones, y no han sufrido ningún tipo de preprocesado o selección manual que pudiese alterar los resultados finales de forma positiva.

Por esta serie de argumentos, este conjuntos de imágenes será el empleado para realizar el diagnóstico de las enfermedades mediante la app móvil.

2.3.2. ASAN Dataset

ASAN [27, 28, 26] es un conjunto de datos de origen surcoreano compuesto por lesiones malignas y benignas de la piel. Nos permiten obtener un mayor grado de variedad de las imágenes, ya que el repositorio ISIC se centra sobre todo en lesiones de piel de población europea y norteamericana.

Tipo de lesión	Número de ejemplares
Actinic keratoses and intraepithelial carcinoma (AKIEC)	651
Basal Cell Carcinoma (BCC)	1082
Dermatofibroma (DF)	1247
Hereditary angioedema (HAO)	2715
Intraepithelial Carcinoma (IC)	918
Lentigo (LEN)	1193
Melanoma (ML)	599
Nevus (NV)	2706
Pyogenic Granuloma (PG)	375
Squamous Cell Carcinoma (SCC)	1231
Seborrhoeic Keratosis (SK)	1423
Wart	2985
Total	**17125**

Tabla 2.3: Distribución de clases de ASAN dataset

Tal y como se describe en la publicación de Goyal [23] , este dataset tiene 12 tipos de enfermedades, sumando un total de 17125 imágenes clínicas. Se dispone de escasa información acerca de este conjunto de datos, ya que los archivos de descarga no están acompañados por documentación acerca de su uso.

En las publicaciones asociadas a este conjunto de datos [27, 28, 26], se hace alusión a la existencia de varios subconjuntos de datos, designados de la A a la D, pero siendo el conjunto A el conjunto intersección de todos ellos. En la breve descripción de su publicación original [27], podemos encontrar que existe un conjunto de imágenes de alta calidad sólo accesible bajo petición, y sólo se dispone de acceso público a sus

miniaturas, las cuales se encuentran organizadas en grandes páginas de alta resolución en configuración de matriz. En caso de necesitar las imágenes por separado para entrenamiento, sería necesario separarlas manualmente.

Tampoco encontramos un fichero de metadatos donde se especifiquen atributos como la edad, la procedencia o el sexo del paciente, pero sí existe constancia de la verificación mediante biopsia de las imágenes gracias al nombre de las mismas, ya que cada matriz de imágenes tiene como nombre la clase a la que pertenecen, y si fue verificada o no por biopsia. Se requiere, por tanto, una fase de preprocesado inicial. El tamaño de cada miniatura es de 98 x 98, siendo un tamaño bastante reducido, en comparación con los 1024 x 1024 de las imágenes contenidas en ISIC.

Adicionalmente, podemos encontrar también 125 imágenes proporcionadas por el dataset Hallym, cuya procedencia es desconocida, pero que solo contiene imágenes cancerosas de tipo melanoma.

Teniendo en cuenta este segundo subconjunto, el dataset contiene un total de 12 clases:

- Benignas: Dermatofibromas (acumulaciones de colágeno), Agiodemas hereditarios, lentigos (manchas debidas a exposición solar o por envejecimiento), nevus (lunares comunes de color marrón o azulado), granuloma piógeno (proliferación de capilares sanguíneos), queratósis seborreica (en forma de descamación de la piel) y verrugas comunes.

- Malignas: Carcinomas intraepitelial, carcinoma de célula basal, melanomas (el más mortal e invasivo) y cáncer de células escamosas.

Encontramos, por tanto, una mayoría de imágenes benignas sobre malignas, lo cual puede introducir ciertos desequilibrios que provoquen sesgo a la hora del aprendizaje. Los resultados de la literatura con este dataset ronda el 90 % en el caso de que se empleen las imágenes de resolución completa. En [26], los resultados ofrecidos se acercan al 91 % de media, tras haber utilizado ResNet153 como arquitectura de red.

Diagnosis	AUC	Sensitivity	Specificity	Threshold[1]
		Test= Asan Test Dataset		
Basal cell carcinoma	0.96 ± 0.01	88.8 ± 3.8	91.7 ± 3.5	0.0429 ± 0.0539
Squamous cell carcinoma	0.83 ± 0.01	82.0 ± 3.6	74.3 ± 3.7	0.0018 ± 0.0014
Intraepithelial carcinoma	0.82 ± 0.02	77.7 ± 6.1	74.9 ± 3.1	0.0030 ± 0.0024
Actinic keratosis	0.92 ± 0.01	92.5 ± 2.5	84.3 ± 2.5	0.0009 ± 0.0002
Seborrheic keratosis	0.90 ± 0.01	82.5 ± 2.1	85.6 ± 3.6	0.0172 ± 0.0140
Malignant melanoma	0.96 ± 0.00	91.0 ± 4.3	90.4 ± 4.5	0.0305 ± 0.0426
Melanocytic nevus	0.95 ± 0.01	91.5 ± 1.9	86.9 ± 1.4	0.0166 ± 0.0134
Lentigo	0.95 ± 0.01	93.9 ± 4.1	86.1 ± 2.8	0.0039 ± 0.0031
Pyogenic granuloma	0.89 ± 0.02	81.1 ± 4.7	89.6 ± 4.0	0.0014 ± 0.0019
Hemangioma	0.89 ± 0.00	81.5 ± 3.7	83.6 ± 5.2	0.1107 ± 0.0721
Dermatofibroma	0.95 ± 0.01	87.6 ± 2.8	92.6 ± 1.9	0.0227 ± 0.0191
Wart	0.94 ± 0.01	86.9 ± 2.2	86.5 ± 2.6	0.0726 ± 0.0280
Average	0.91 ± 0.01	86.4 ± 3.5	85.5 ± 3.2	0.0270 ± 0.0210

Figura 2.13: Ejemplo de lunares benignos en ASAN (Nevus)

Los resultados han sido confirmados por expertos dermatólogos y verificados en su mayoría mediante biopsia, por lo que las etiquetas asociadas a cada lesión están completamente verificadas. Como aspecto negativo, de cara al proyecto que se desarrolla en este documento, estos resultados distan bastante de lo que se puede conseguir con un smartphone, ya que, ResNet153 es uno de los modelos más profundos existentes en la literatura, y requieren grandes cantidades de memoria de gráficos. Además, sólo se dispone de las miniaturas, por lo que no será posible obtener características de grado fino aunque se aumente en profundidad o anchura el modelo. En preprocesado de datos, veremos algunas formas de evitar que se produzca esta penalización mediante la aplicación de mejora de imágenes con GANs.

2.3.3. DermNet

DermNet [3] es un portal online que ofrece imágenes en alta resolución sobre lesiones cutáneas, tanto cancerígenas como benignas, para el uso como material docente para la formación de dermatólogos, y tiene a disposición un conjunto de imágenes complementado por datos en formato tabular, especialmente preparado para su uso en Inteligencia artificial.

Podemos encontrar enfermedades cutáneas recogidas de pacientes alrededor de todo el mundo, sin limitarse únicamente manchas y lesiones parecidas a tumores cancerosos, y ofreciendo también heridas vinculadas a enfermedades infecciosas y

hongos. E incluso lesiones de tipo alérgico, así como acné, dermatitis severa o celulitis.

Estos datos se encuentran alojados de en una web que podemos visitar sin restricciones. Para facilitar la extracción, existen gran cantidad de herramientas que podemos encontrar en GitHub [64]. En total, se pueden obtener hasta 23000 imágenes, compuestas por un total de 23 clases no balanceadas. Carecen de metadatos asociados, y además. poseen una marca de agua en la imagen que podría crear sesgos en el aprendizaje. Si se desea acceder a dicha información, es necesario solicitar permiso en su web oficial, y abonar una cantidad económica no especificada, determinada en función del número de imágenes solicitadas y su procedencia.

Debido a la escasez de presupuesto, se descarta la opción de emplear este conjunto de datos, pero en la literatura podemos encontrar algunas menciones que datan del período anterior a 2020, momento en el cual parte de estos datos se ofrecía de forma abierta bajo otro portal de nombre similar, DermNet.

2.3.4. PH2

El conjunto de datos PH2 [43] es un conjunto de 200 imágenes obtenidas gracias al hospital Pedro Hispano de Portugal. Está compuesto por 200 imágenes de alta resolución que contienen 3 posibles casos de lesiones:

- Lunar común (Common Nevus), 80 ejemplares
- Lunar atípico (Atypical Nevus), 80 ejemplares
- Melanomas, 40 ejemplares.

Dichas imágenes fueron captadas bajo las mismas condiciones, haciendo uso de un sistema de análisis "Tuebinger Mole Analyzer", capaz de realkizar una magnificación de 20 veces mayor al tamaño original. Su resolución es de 768x560 pixeles, haciendo uso de los 3 habituales canales RGB a 8 bit de profundidad cada uno. Se trata de unas imágenes de buena calidad, que llevan su correspondiente información asociada en forma de metadatos. Estos describen el color, la extensión de las manchas, la textura, la forma del borde, la localización, y otro tipo de características, como el tipo de piel.

De acuerdo a los requisitos especificados en su web, [2], podemos acceder a los datos y sus metadatos asociados de forma totalmente abierta y sin restricciones siempre que su uso se limite fines académicos y personales, como es este caso.

Figura 2.14: Nevus maligno y benigno en PH2

En la figura 2.3.4 podemos encontrar una muestra de estas imágenes, concretamente un lunar común (nevus) y un lunar de tipo cancerígeno.

2.3.5. PAD-UFES 20

A patient may have one or more skin lesions and a skin lesion may have one or more images. In total, there are 1373 patients, 1641 skin lesions, and 2298 images present in the dataset. PAD-UFES-20 [46] se trata de un conjunto de imágenes disponible públicamente acerca de lesiones cutáneas benignas y malignas. Concretamente, dispone de imágenesde 1.641 lesiones cutáneas únicas recopiladas, procedenetes de 1373 pacientes. Tal y como se especifica en su publicación [46], cada paciente perteneciente al dataset puede ofrecer imágenes de piel sana, una lesión, o varias de ellas. Por tanto, el conteo final de imágenes asciende a un total de 2.298 ejemplares.

Las imágenes a analizar han sido extraídas en sus totalidad de dispositivos móviles, los cuales se conectaban a un servidor al cual enviaban la imagen para su posterior diagnóstico y validación mediante biopsia en caso de sospecha de caso positivo de cáncer. Todas ellas fueron almacenadas de forma directa, sin aplicar ningún tipo de ecualización o filtro, ya que el formato de guardado escogido es PNG, caracterizado por ser un formato sin pérdidas.

Entre sus clases, podemos encontrar, de forma equilibrada tres enfermedades benignas de la piel y tres tipos de cánceres de piel:

- Queratosis actínica: manchas escamosas y oscurecidas en la piel debido a la exposición solar. Benignas de naturaleza.
- Cáncer de células basales: tumores generalmente aislados que surgen en las capas profundas de la piel
- Melanoma: tumores y manchas cancerosas de color oscuro, en algunos casos acompañadas de necrosis de tejidos, y altamente mortal
- Nevus: lunares comunes
- Queratosis seborreica: enfermedad benigna, que forma manchas protuberantes de color oscuro no contagiosas.

- Cáncel de células escamosas: cancer con baja tasa de mortalidad, que surge en la capa intermedia de la piel, y cuyo crecimiento puede tener graves riesgos para la salid.

Todos estos datos fueron revisados por dermatólogos expertos, quie aportaron metadatos muy detallados acerca del historial del paciente, gracias a la asociación con la Unidad Patológica de Anatomía del Hospital universitario Cassiano Antonio Moraes (HUCAM) , perteneciente a la Universidad Federal de Espírito Santo, en Brasil. EDe entre los metadatos ofrecidos, podemos destacar:

- ID de paciente
- ID de lesión,
- ID de imagen
- Si la lesión benigna fue o no probada por biopsia.
- Enfermedad diagnosticada
- Información del paciente: fumador o no, localización de la lesión, edad, exposición a químicos, historial cancerígeno, etc.

En su publicación original[46], podemos encontrar un resumen de su contenido de forma más específica:

Diagnóstico	**Ejemplares**	**% biopsied**
Actinic Keratosis (ACK)	730	24.4 %
Basal Cell Carcinoma of skin (BCC)	845	100 %
Malignant Melanoma (MEL)	52	100 %
Melanocytic Nevus of Skin (NEV)	244	24.6 %
Squamous Cell Carcinoma (SCC)	192	100 %
Seborrheic Keratosis (SEK)	235	6.4 %
Total	**2298**	**58.4 %**

Tabla 2.4: Tabla de casos diagnosticados en PAD-UFES20

Donde podemos apreciar que todos los casos de enfermedades cancerígenas han sido probados mediante biopsia, y el cáncer de célula basal se trata del tipo de enfermedad más frecuente.

Figura 2.15: Batch de ejemplo de PAD-UFES 20 [46]

La gran ventaja de este dataset es que sus imágenes han sido recopiladas mediante smartphones, por lo que las condiciones de recopilación han sido prácticamente idénticas a las que se recibirán por el modelo entrenado para este trabajo. Esto nos aporta como beneficio un conjunto de entrenamiento mucho más cercano a la población real a la que será destinada el modelo.

2.3.6. Severance

Severance es otro conjunto de lesiones cutáneas que podemos encontrar para su libre acceso de forma digital [25]. Se trata de un conjunto de imágenes de lesiones cutáneas recopiladas de pacientes de Corea del Sur por el hospital Severance. Los datos recopilados pertenecen a los casos diagnosticados por dermatólogos entre enero de 2008 y marzo de 2019, teniendo en cuenta el historial medido de cada uno de los pacientes y asegurando un grado alto de confianza en el diagnóstico.

La población de este muestreo se centra sobre pacientes mayores a 19 años, que fueron diagnosticados con lesiones provenientes de tumores. Se excluyó de este estudio a todas aquellas lesiones de difícil acceso, como las lesiones intraoculares, postoperatorias, o causas por medios externos, como láseres o costuras operatorias. Además, algunos pacientes tenían diferentes lesiones en zonas próximas, por lo que se tomaron las imágenes de forma que el diagnóstico para cada una de ellas fuera individual, y no se pudieran producir incongruencias entre los expertos en el diagnóstico.

En total, se dispone de 40331 imágenes clínicas, que provienen de 10426 pacientes distintos que participaron en la cesión de datos. Al igual que ASAN, este dataset posee distintas versiones publicadas en la red, siendo accesible únicamente su partición A, y podemos encontrar 38 tipos de diagnósticos distintos de los 43 disponibles

en total, los cuales no siguen una distribución equilibrada.

Realizando un test estadístico previo, es posible verificar que las 6 clases más comunes contenidas en este dataset componen el 75 % del conjunto total. Está compuesto por queratosis actínica (22.5 %), angiofibromas (14.4 %), angiokeratomas(13.8 %), cáncer células basales (8.1 %), lunares de Becker (7.5 %), bluenevus (6.2 %), y la enfermedad de Bowen (carcinomas)(6.1 %).

El interés en este dataset se debe a que algunas de estas clases mayoritarias, como los nevus azules y los lunares de Becker, son condiciones benignas que suelen ser retirados únicamente con fines estéticos, permitiendo complementar con el resto de los diagnósticos negativos. Este tipo de lunares son los más complejos de diagnosticar, debido a sus colores similares a un melanoma, y suelen requerir una biopsia, por lo que su diagnóstico suele alargarse.

Las imágenes publicadas son miniaturas recogidas en forma de matriz, presentando un formato similar a las encontradas en ASAN. La diferencia, en este caso, es que disponemos de los metadatos asociados a cada imagen, y el proceso de separación de miniaturas se ve simplificado. En la figura 2.16 podemos apreciar el formato matricial, y la toma de imágenes desde diferentes puntos de vista para portar una mayor variabilidad al modelo. Es posible separar cada una de estas imágenes logrando un conjunto de aproximadamente 10400 fotografías a 98 x 98 píxeles en formato jpeg, debido a la compresión con pérdidas aplicada para almacenarlas.

Figura 2.16: Imágenes de ejemplo provenientes del dataset Severance

En cuanto a los resultados obtenidos en tareas de aprendizaje, de nuevo encontramos que se han utilizado grandes modelos para realizar el aprendizaje, siendo en este caso utilizado los modelos VGG16 y ResNext50, y siendo el resultado de accuracy top1 un 53.0 %, dada a la complejidad del problema al utilizar clases infrarepresentadas.

2.3.7. Otros datasets

En los puntos anteriores, se han descritos los datasets disponibles con mayores referencias y mayor peso dentro del espacio de investigación cubierto por el campo de investigación acerca del cáncer de piel. Pero, existe una serie de bases de datos con grandes cantidades de datos que fueron retiradas de su acceso público, o simplemente, fueron eliminadas debido a limitaciones legales por licencias. A continuación, se destacan algunas de ellas que tuvieron un gran impacto en la literatura de los años 2015 y 2020:

- **DermQuest**. Se trataba de un atlas virtual que contenía información detallada acerca de lesiones cutáneas, con el fin de servir como atlas de estudio para dermatólogos en formación. Esta información, a pesar de ser principamente académica, podía ser recopilada por técnicas de minería de datos. Posteriormente, este conjunto de datos fue dado de baja, pero reapareció como un subconjunto del repositorio Derm101. Sin embargo, desapareció de forma definitiva en el año 2019, por lo que no es posible encontrar estos datos para uso.

- **SD-198 y SD-260** [61] son dos datasets de origen chino que recopilan fotografías y metadatos acerca de 98 y 260 imágenes respectivamente, asegurándo al menos un total de 50 imágenes como mínimo por patología. Ambos datasets se pueden ver referenciados en mutitud de papers como [23]. Habitualmente, el más usado de ambas variantes era SD-198. Existen datasets con mayor y menor cantidad de clases, pero son subconjuntos de estos, ya que los datasets se organizaban en niveles de carpetas, donde cada nivel descendido aportaba un mayor nivel de granularidad en el diagnóstico. En la actualidad, sólo el subconjunto 198 permanece en línea, pero bajo solicitud. Contiene 6,584 imágenes, que fueron capturadas para historias clínicas mediante smartphones. Sin embargo, debido a las dificultades del idioma, y la dificultad de contacto con el autor original, se prescindirá de este dataset para el estudio del problema.

- **DermIS** [1] es un atlas online creado con fines académicos. Aunque no dispone de metainformación asociada, podría emplearse para el problema que se trata en este trabajo. Pero, debido a la incorporación de otras enfermedades que no vinculan con el dominio del problema, se descarta su utilización.

- **Mednode** [19] contiene 70 melanomas and 100 nevus provenientes del Departamento de dermatologia de la universidad University Medical Center Groningen (UMCG) , cuyo objetivo es la creación de una aplicación de diagnóstico de enfermedades cutáneas no desmocópicas que faciliten la decisión a los dermatólogos. Debido a su pequeño tamaño, y a que este aporta datos acerca de clases ya mayoritarias en otros datasets, se propone su no utilización para el experimento.

Además de estos datasets no disponibles en la actualidad, existen publicaciones recientes sobre nuevos conjuntos de datos utilizados de uso interno para dicha investigación, fortaleciendo así la visión de que la inteligencia artificial aplicada a la dermatología es un campo de interés y competitividad. No es posible acceder a este tipo de conjuntos debido a su gran valor competitivo dentro de la industria de las aplicaciones de asistencia al diagnóstico dermatológico.

3 Construcción y preprocesado del conjunto de datos

Una vez examinados todos los conjuntos mencionados anteriormente, podemos llevar a cabo la unión de todos los datos en un único subconjunto. Esto nos permitirá conseguir un dataset completo y variado con diferentes tipos de piel y diferentes lesiones que nos permitirán identificar multitud de tipos de patologías, siendo posible ajustar el grado de granularidad en función de la agrupación o no de posibles subclases.

Inicialmente, el conjunto de datos construido contendrá todos los subtipos de lesiones cutáneas vistos, pero dispondrán de una segunda etiqueta que indicará si se trata de un caso canceroso o no, atendiendo a la enfermedad que lo etiqueta. En total, tenemos como subclases 52 posibles etiquetas, las cuales iremos examinando a medida que se preprocese cada uno de los subconjuntos.

En el caso de las lesiones potencialmente cancerosas, como se trata de condiciones de la piel no cancerosas con posible evolución a cancerosas, se tendrán en cuenta como imágenes benignas, ya que la condición de malignidad sólo podría aparecer en el futuro, el cual sigue siendo desconocido.

Este proceso es una fase muy delicada del entrenamiento, ya que se ha demostrado empíricamente que una correcta preparación y normalización de los datos permiten hallar soluciones más cercanas a la optima que con datos no procesados. Es importante tener en cuenta que no existe una metodología de preprocesado única, y que es necesario adaptarse al tipo de dato que estamos tratando. Para este proyecto, además, existe una dificultad adicional, y es la existencia de diferentes procedencias para los datos, pues en total se dispone de 5 datasets distintos, cada uno recopilado con diferentes metodologías e instrumentación. Por tanto, será clave adaptarse a cada uno de los destinos, y realizar la partición final de forma estratificada para evitar sesgos que perturben el resutado.

A continuación, se describe la estrategia seguida para el procesado global de los datos, y los ajustes necesarios para cada uno de los conjuntos empleados.

3.1. Estrategia de preprocesado para la fusión

En el punto de partida, antes del preprocesado, contamos con 5 datasets muy diferentes entre sí. Cada uno ha sido documentado y organizado siguiendo unos criterios no estándares que nos afectan en gran medida a la hora de emplear estos datos para el aprendizaje. Antes de proceder con el desarrollo de los modelos, debemos de estadarizarlos a un formato común para evitar que existan clases con el mismo diagnóstico que, por cuestiones de formato, se consideren etiquetadas como clases distintas, por usar criterios distintos de escritura, como ausencia de espacio, mayúsculas o one hot encoding.

Concretamente, los datos recopilados poseen el siguiente formato de etiquetado:

- ISIC: etiquetas escritas a formato completo, como nombre de carpeta, con la primera letra de la enfermedad en mayúscula.
- ASAN: nombres escritos en el nombre de la fotografía, haciendo uso de caracteres especiales, y de abreviaturas.
- Severance: etiquetas escritas en la propia imagen, la cuales habrá que etiquetar y organizar manualmente, debido a que su csv está incompleto.
- PH2: fichero csv, con diagnósticos en formato one hot encoding. Es decir, una fila de ceros y unos, siendo uno la clase a la que pertenece, y 0, el resto.
- PAD UFES 20: fichero csv, con los nombres de diagnóstico escritos en minúsculas, sin espacios.

Podemos observar la gran variedad de formatos de registro empleados, y que por tanto, es completamente obligatorio y necesario realizar una transformación para hacerlo homogéneo. En este caso, por decisión propia, he considerado adecuado realizar una transformación de las etiquetas al siguiente formato: Uso únicamente de minúsculas, con ausencia de caracteres especiales, nombres sin abreviaturas, y evitando el uso de espacios, con el uso de la barra baja como carácter sustitutivo. Para la clasificación binaria, se utilizará one hot encoding, denotando como 0 los casos negativos y como unos, los positivos.

De esta forma, se obtiene un fichero .csv donde encontrar los valores necesarios para entrenar los modelos. Para poder localizar cada imagen, se mantendrá el arbol de directorios por defecto de cada dataset, y se anotará su directorio en un nuevo fichero .csv, que contendrá las etiquetas estandarizadas y los nombres de los ficheros de imagen con y sin el directorio. En resumen, contará con los campos:

- image: nombre la imagen, sin el path en su nombre, y con la extensión de formato

- dir: directorio donde se aloja la imagen, respetando la estructura de carpetas original seguida por el dataset de origen
- label: etiqueta con el diagnóstico de la lesión, siguiendo las pautas indicadas anteriormente
- dataset: columna que indica el dataset de procedencia de la imagen, por si fuese necesario utilizar solo un subconjunto de todos los datos.
- bin: columna para la etiqueta que indica si se trata de clase Benigna (0) o Maligna (1)

	image	dir	label	dataset	bin
0	ASAN_8196.png	datasets/Asan/dataset/hem…	hemangioma	ASAN	0
1	ASAN_10059.png	datasets/Asan/dataset/der…	dermatofibroma	ASAN	0
2	SEV_248_74.png	datasets/Severance/thumbn…	pyogenic_granuloma	SEVERANCE	0
3	ISIC_0068761.JPG	datasets/ISIC/Melanoma/IS…	melanoma	ISIC	1
4	SEV_062_17.png	datasets/Severance/thumbn…	squamous_cell_carcinoma	SEVERANCE	1
5	SEV_218_91.png	datasets/Severance/thumbn…	melanocytic_nevus	SEVERANCE	0
6	ISIC_0009272.JPG	datasets/ISIC/Nevus/ISIC_…	nevus	ISIC	0
7	ASAN_1867.png	datasets/Asan/dataset/nev…	nevus	ASAN	0
8	ISIC_0011887.JPG	datasets/ISIC/Angioma/ISI…	angioma	ISIC	0
9	ASAN_14675.png	datasets/Asan/dataset/bas…	basal_cell_carcinoma	ASAN	1

Figura 3.1: Formato del fichero csv normalizado

Una vez establecido el formato común, podemos pasar al análisis y adaptación propia de cada conjunto.

3.1.1. ISIC Dataset

El dataset ISIC es el mayoritario de la lista, ya que posee casi 60.000 imágenes de alta resolución, del total de casi 108.000 imágenes de las que disponemos. Para su descarga, se han empleado la galería de la web oficial [4], donde podemos filtrar cómodamente las enfermedades que queremos descargar. Como criterio de descarga, se han tenido en cuenta únicamente aquellas fotografías correctamente diagnosticadas, ya que existe un total de 27896 imágenesno etiquetadas dentro del repositiorio, las cuales descartaremos. El problema se centrará en resolver un problema de aprendizaje supervisado, por lo que las imágenes no etiquetadas suponen una complejidad adicional y un ruido para el modelo.

Cada clase descargada, además, se ha sometido a un proceso de filtro, sobre todo por cuestiones numéricas; existen nuevas clases, con escasas cantidades de datos, las cuales poseen menos de 10 imágenes, cantidad insuficiente a la hora de clasificar frente a clases como lunares comunes, que tienen en total 32697 ejemplares. De esta forma, nos queda el siguiente conjunto de clases:

- Nevus
- Seborreic keratosis
- Actinic keratosis
- Pigmented benign keratosis
- Solar lentigo
- Dermatofibroma
- Vascular lesion
- Lichenoid keratosis
- Acrochordon
- Lentigo NOS
- Atypical melanocytic proliferation
- Aimp
- Wart
- Angioma
- Lentigo simplex
- Neurofibroma
- Scar

Estas clases se almacenan en ficheros zip cada una, así que tras ser descargadas, deben ser extraídas y añadidas al fichero csv que definimos anteriormente. Al tratarse del primer subconjunto que se añadirá, será la parte del código encargada de crear el fichero y establecer las columnas mencionadas. Además, se realizará la transformación de las etiquetas, dispuestas en el formato de la enumeración anterior, a notación snake case. Numerando el proceso, se ha creado un script de python que realiza las siguientes tareas:

Algorithm 3.1 Algoritmo de descompresión de fotografías por carpetas

```
1: procedure EXTRACTISIC(path)        ▷ Obtener una lista de todos los archivos ZIP
2:     archivosZip : list of strings
3:     for all file in path where file.name endswith('zip') do
4:         archivosZip.add(file)
5:     end for
                                                    ▷ Iterar sobre cada archivo ZIP
6:     for all archivo in archivosZip do
7:         var rutaArchivoZip ← join(path, archivo)
8:         var carpetaSalida = path.splitext(rutaArchivoZip).get(0)     ▷ Eliminar la
   extensión
9:         if not exists(carpetaSalida) then
10:            makedir(carpetaSalida)
                        ▷ Extraer el contenido del archivo ZIP en la carpeta de salida
11:            OPENZIPFILE(rutaArchivoZip) as zipRef
12:            zipRef.extractall(carpetaSalida)
13:        end if
14:    end for
15: end procedure
```

1. Extraer las imágenes mediante uzip en una carpeta con el mismo nombre de la clase a la que pertenecen 3.1.1
2. Crear un fichero .csv, denominado preprocessedData.csv, donde se alojarán las 5 columnas: images, dir, label, dataset, bin.
3. Recorrer cada carpeta creada, y añadir los 4 primeros campos
4. Una vez añadidas todas las imágenes, se renombran las etiquetas a camel case mediante las funciones upper(), lower() y replace() de la clase string de python.

Algorithm 3.2 Algoritmo de creación del csv

```
procedure CREAR_CSV(path, clases, nombre_dataset)   ▷ Inicializa una lista vacía para almacenar la información de los archivos
    var info_archivos : list of strings
    i ← 0
    for nombre_carpeta in dir(path) do
        ruta_carpeta ←join(path, nombre_carpeta)
        if path.isdir(ruta_carpeta) then
                                                    ▷ Itera sobre cada archivo en la carpeta
            for nombre_archivo in listdir(ruta_carpeta) do
                ruta_archivo ← join(ruta_carpeta, nombre_archivo)
                if isfile(ruta_archivo) and nombre_archivo.ends() != "jpg" then
                    var info_archivos.add(nombre_archivo, ruta_archivo, clases[i], nombre_dataset)
                end if
                i = i + 1
            end for
        end if
    end for
    ruta_archivo_csv ← 'preprocessedData.csv'          ▷ Define la ruta del CSV
    open(ruta_archivo_csv) as archivo_csv
    archivo_csv.writerow([ïmage", "dir", çlass", "dataset"])   ▷ Escribe la cabecera
end procedure
```

Para facilitar el procesado, el rellenado de los datos se realiza sobre una estructura tabular de pandas, para así transformar la columna label fácilmente. En cuanto a la quinta columna, la clase binaria, dicha tarea se realizará cuando todos los datasets estén añadidos al .csv, de forma que el recorrido de los datos sólo se realice una vez, cuando tengamos disponible todas las clases.

En cuanto al estudio estadístico de los datos, este se realizará una vez dividido los datos en los conjuntos de entrenamiento y test, definido en entradas posteriores.

3.1.2. ASAN

ASAN es uno de los dos datasets cuyo formato de entrega de los datos consistía en una matriz de imágenes en un canvas de gran resolución. En el caso de este dataset, tenemos un total de 32 imágenes de este tipo, cuya etiqueta se encuentra escrita en el nombre del fichero. El procesado de este dataset será más complejo que el anterior, ya que debemos recortar cada una de las imágenes, evitando que queden bordes blancos que puedan perturbar la predicción, y sesgar el aprendizaje.

Podríamos idear una solución codificada de forma estricta en la cual la imagen se subdivida en n filas y m columnas para extraer las fotografías; sin embargo, cada uno de los canvas del datasets tiene un número filas y columnas concreto que dificultaría esta tarea de forma automática. En su lugar, se ha medido mediante una herramienta de recorte fotográfico el tamaño de una de las miniaturas, siendo este de 98 píxeles, y será el valor que utilizaremos a la hora de realizar el recortado.

No se debe pasar por alto que las imágenes se encuentran separadas vertical y horizontalmente por espacios en blanco, cuya distancia es variable. Es el factor causante de la imposibilidad de partición regular, por lo que se emplearán técnicas de visión mediante la librería OpenCV para la detección de bordes:

1. Eliminar 4 píxeles en blanco de los extremos para que todas las bandas blancas queden del mismo grosor

2. Hallar el numero de imágenes por fila y columana de forma aproximada, teniendo en cuenta el tamaño de miniatura y el borde.

3. Recortar la imagen usando el método findContours() de openCV. Este método binaria la imagen transformándola a blanco y negro, y trazado con técnicas de detección de puntos de interés en imágenes los bordes de cada una de las miniaturas, y devolviendo las coordenadas de sus equinas en un vector multidimensional.

4. Para cada imagen, obtenemos la esquina superior izquierda de la imagen, y mediante el ancho y alto de la imagen, recortamos dicha sección de la imagen y se almacena en una nueva variable.

5. Se recortan los bordes de dicha imagen y se almacena el resultado en disco, en una carpeta que posee el mismo nombre que la imagen de la que fue extraída.

6. Se repite el paso 3-6 para cada imagen de la matriz, pasando a abrir la siguiente matriz hasta que no quede ninguna por recortar.

Algorithm 3.3 Recorte de las imágenes de ASAN mediante OpenCV

```
procedure recortarImagenesASAN(path, i, title='ASAN')
    var files : list of strings
    for f in pathlib.Path().iterdir() do
        if f is_file()] then
            files.add(f)
        end if
    end for
    var names : list of strings
    var diss_class : list of strings
    var i ← 0
    for each f in files do
        name ← str(f)[str(f).rfind('#') + 1:-4]
        if f.ends = 'png' and not exists(path) then
            makedir(name)
            var image = cv2.imread(str(f), cv2.IMREAD_UNCHANGED): image
            var gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) : image
            var kernel = cv2.getStructuringElement(cv2.MORP H_RECT, (5, 5))
            var gradient = cv2.morphologyEx(gray, cv2.MORPH_GRADIENT, ker-
nel)
            var contours ← cv2.findContours(gradient, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
            for each cnt in contours do
                x, y, w, h ← cv2.boundingRect(cnt)
                var box_image = image[y: y + h, x: x + w] : image
                                        ▷ Recorta la imagen para eliminar el borde
                var img_sin_borde = box_image[grosor_borde:-grosor_borde,
grosor_borde:-grosor_borde]
                cv2.imwrite(f"name/title_i.png", img_sin_borde)
                names.append(f"title_i.png")
                diss_class.add(name)
                i = i + 1
            end for
        end if
    end for
end procedure
```

Una vez finalizado el proceso, el proceso a aplicar es similar a ISIC; pero, en este caso, en lugar de simplemente convertir a camelcase, debemos de cambiar los nombres por completo para no usar el formato por abreviaturas original, y poder hacer merge de las clases de este dataset con ISIC que sean de la misma enfermedad.

Para ello, simplemente se crea un diccionario clave-valor, donde la clave es el nombre que deseamos cambiar, y el valor, el nuevo nombre. Mediante pandas [48], el proceso de sustitución se puede hacer de forma inmediata mediante la función replace.

Es importante destacar que el dataset Hallym, también será incluido en el conjunto final, siendo el procedimiento de preprocesado a aplicar exactamente el mismo al descrito en este punto.

3.1.3. PAD UFES 20

PAD UFES 20, como ya describimos en el apartado de Estado del arte, se trata de un dataset diseñado para el entramiento de sistemas de asistencia en diagnóstico computado, donde el experto dermatólogo puede utilizarlo como un medio de apoyo. Contiene 6 enfermedades distintas, siendo 3 cancerosas (células basales, células escamosas o melanoma maligno) y 3 benignas (actinic keratosis, nevus, keratosis seborreica).

La estructura de presentación de los datos es más sencilla que ASAN, pues las imágenes son individuales, y cuentan con un fichero .csv donde se describen las etiquetas y otros metadatos asociados a las imágenes. La única modificación necesaria es actualizar el path de cada imagen y la nomenclatura del diagnóstico de la enfermedad, teniendo en cuenta que debemos de tranformar de abreveviatura a camel case:

- NEV → nevus
- SEK → seborreic_keratosis
- ACK → actinic_keratosis
- BCC → basal_cell_carcinoma
- SCC → squamous_cell_carcinoma
- MEL → melanoma

De esta forma, podemos simplemente hacer fusión de las nuevas filas con el fichero anterior en modo de apertura “append”.

3.1.4. PH2

Este dataset recoge imágenes provenientes del Servicio de Dermatología del Hospital Pedro Hispano (Matosinhos, Portugal), que recoge pruebas cutáneas realizadas con el sistema Tuebinger Mole Analyzer, y un aumento de 20x. Recordando lo analizado en el capítulo anterior, consta de 200 imágenes dermatoscópicas de lesiones melanocíticas, incluidos 80 lunares comunes, 80 nevus atípicos y 40 melanomas, en

Algorithm 3.4 Formato de las imágenes de PAD UFES

```
var PAD_UFES_PATH : directory of PAD UFES 20 dataset
dict ← {'BCC': 'basal_cell_carcinoma', 'SEK': 'seborreic_keratosis', 'SCC': 'squa-
mous_cell_carcinoma', 'NEV': 'nevus', 'ACK': 'actinic_keratosis', 'MEL': 'melano-
ma'}
procedure PREPARAR_PADUFES
    df_pad_ufes ← file.read()
    df_pad_ufes['diagnostic'] ← df_pad_ufes['diagnostic'].replace(dict)
    for each img in df_pad_ufes do
        df_pad_ufes['im_dir'] ← PAD_UFES_PATH + '/images/ + img
        df_pad_ufes['dataset'] = 'PAD_UFES'
    end for
end procedure
```

una resolución de 768x560 píxeles. La base de datos PH² incluye anotaciones médicas de todas las imágenes: segmentación médica de la lesión, diagnóstico clínico e histológico, y evaluación de varios criterios dermatoscópicos. Entre ellos, encontramos distinción por colores; formación del tejido; puntos/glóbulos; rayas; áreas de regresión; velo azul blanquecino (pgimentación difusa).

Los datos se organizan en directorios de carpetas de varios niveles, pudiendo encontrar en su interior la imagen dermoscópica, la plantilla de segmentación y regiones de interés de la imagen destacadas mediante una plantilla binaria. Para este trabajo, sólo se utilizarán las imágenes correspondientes al directorio de datos dermoscópicos, ya que con debido a la cercanía de la lensión, la imagen contiene en su mayoría solo información útil.

Como preprocesado de las imágenes, al disponer de ellas correctamente clasificadas en carpetas y con un fichero de metadatos bastante completo, el único procedimiento necesario sería la adición al fichero preprocessedData.csv, realizando las conversión a camel case de las labels. El método a emplear es equivalante al utilizado en PAD UFES 3.4.

3.1.5. Severance

Este dataset, como ya se comentó, provinene del hospital Severance, y el dataset con mayor cantidad de patologías identificadas de la lista. El formato de este repositorio sigue una estructura similar a la de ASAN: las imágenes están organizadas en varias páginas de gran tamaño, donde, a diferencia de ASAN, podemos encontrar varias imágenes por lesión separadas entre cuadros blancos, que contienen escritos en él, la etiqueta de las siguientes imágenes leídas de izquierda a derecha, y de arriba a abajo, hasta encontrar el siguiente cuadro en blanco con texto.

La dificultad de este preprocesado radica precisamente en la existencia de los cua-

drados blancos, ya que la lectura del texto escrito en ellos es prácticamente imposible. La baja resolución, unido a la existencia de etiquetas cuyo nombre se encuentra dividido en varias líneas, provoca que algoritmos de reconocimiento de texto como Pytesseract [30] no fuesen capaces de detectar las etiquetas adecuadamente.

Por este motivo, fue necesario identificar manualmente cuántos casos de cada enfermedad existían por matriz de imágenes, y realizar un recuento para saber cuándo aplicar una etiqueta u otra. Conociendo que las etiquetas asignadas aparecían por orden alfabético, el proceso a seguir se resumió en aumentar un contador cada vez que aparecía una imagen en blanco durante el recorrido de las imágenes, y aumentar el contador hasta llegar al valor total de ejemplares de esa clase; en ese momento, se procede a contar los valores de la siguiente. Con este procedimiento, nos queda el algoritmo 3.5(3.1.5)

La ventaja respecto a ASAN reside en que los espacios en blanco entre imágenes tienen el mismo grosor, y fue posible de separar en submatrices sin necesidad de utilizar ténicas de visión, que ralentizan el proceso extracción. Cada imagen es etiquetada siguiendo el formato establecido y siendo añadida a una nueva carpeta, desde la cual se referenciarán mediante el fichero preprocessedData.csv.

Como dificultades a destacar durante el desarrollo de este procedimiento, comentar que la detección de la imagen en blanco que separa las etiquetas entre clases no es trivial. El método elegido para distinguirla es emplear un umbral de número de píxeles con valor de escala de grises blanco, es decir, 255. El valor de dicho umbral se calculó de forma experimental , ya que si el valor era demasiado bajo, pieles sanas o de color claro también cumplían la restricción. El cambio de etiquetado no se produce si el número de píxeles totales de la imagen encontrada no supera un valor x de miles de píxeles. De forma experimental, el valor 20.000 fue el que mejor resultados aportó, funcionando en todos los casos.

Aumentar de forma desmesurada este valor también puede tener consecuencias negativas, ya que el nombre de la etiqueta se encuentra escrita en la parte inferior de estas imágenes y aportan píxeles de color negro que no cumplirían la condición.

Algorithm 3.5 Recorte de las imágenes de Severance mediante OpenCV

```
procedure RECORTARIMAGENESSEVERANCE(path)
    df_sev ← read('SeveranceA.xls')
    if not os.path.exists('thumbnails') then
        makedir('thumbnails')
    end if
    var files : list of strings
    for f in pathlib.Path().iterdir() do
        if f is_file()] then
            files.add(f)
        end if
    end for
    var names : list of strings
    diss_class : list of strings
    j ← 0
    for f in files do
        if f.ends = 'jpg' then
            name ← f.name[0:-4]
            makedir(name)
            var image = cv2.imread(name, cv2.IMREAD_UNCHANGED): image
            var image_split = [image_1 image_2, ..., image_8]^T        ▷ División por filas
            var i ← 0
            for each im in image_split do        ▷ Recorta la imagen en columnas
                image_split_col = np.split(im[:, 2:-3], 15, axis=1)
                for each imcol in image_split_col do
                    img_sin_borde←        imcol[grosor_borde:-grosor_borde,
grosor_borde:-grosor_borde]
                    if np.count_nonzero(imcol == 255) >20000) then   ▷ Si es imagen
de separador con nombre
                        j ← j + 1        ▷ Cambiamos de etiqueta al siguiente paciente
                        continue
                    else
                        actual ← df_sev.get(j)
                        actual.add('name_i.png')
                        trainProcessed.add(actual)        ▷ Guardamos la imagen y
añadimos su nombre
                    end if
                    cv2.imwrite(f'thumbnails/name_i.png', img_sin_borde)
                    names.append(f"name_i.png")
                    i ← i + 1
                    diss_class.add(name)
                end for
            end for
        end if
    end for
    trainProcessedDF.write("severanceTrainingSet_thumbnails.csv")   ▷ Guardado
en disco
end procedure
```

3.1.6. Etiquetado binario

Una vez disponemos de todas las imágenes correctamente etiquetadas y redireccionadas desde el fichero preprocessedData.csv, podemos continuar con el etiquetado binario de las enfermedades. Para disponer también de los datos de malignidad de las enfermedades, se ha realizado manualmente una distinción de las distintas enfermedades entre benignas y malignas. Debido a la minoría de imágenes por defecto, para reducir el tamaño del diccionario de enfermedades, por defecto se establecieron todas las etiquetas por defecto a 0, y aquellas pertenecientes al subconjunto de malignas, a 1, siguiendo el siguiente procedimiento:

1. Crear una nueva columna, de nombre "bin", que contendrá 0 como valor de inicialización.

2. Construir un diccionario o lista con aquellas enfermedades que son malignas.

3. Se recorre la columna target para comparar la enfermedad del ejemplar con la lista de enfermedades malignas. En caso de ubicarse en ella, se cambia el valor de "bin" por un 1.i

De esta forma, obtenemos la quinta columna, expresada con ceros y unos. Dependiendo de la función de pérdida y el framework posteriormente utilizado, estas etiquetas tendrán que ser categóricas, pero dicho paso puede realizarse de forma inmediante con un casting de tipos, o bien, un el uso de la función replace de python ya mostrada en la fase de homogeneización de los datos.

Figura 3.2: Distribución de clases

Si agrupamos por lesiones benignas, cancerosas, y potencialmente cancerosas, podemos obtener el siguiente diagrama de sectores de la figura 3.2, donde se puede

observar cómo la mayoría de imágenes disponibles engloban problemas de piel no cancerosos, mientras que el segundo tipo más común de lesión si es la cancerosa. Las lesiones potenciales, como ya se comentó, pasan a formar parte de las lesiones benignas debido a que en el momento de la captura, continúan siendo benignas.

Con este paso, finalizamos la uniformización de los datos en lo que corresponde al formato del etiquetado y gestión de directorios. A continuación, se realizarán las fases de partición de los datos, y de preprocesado de la imagen en sí, empleando como referencia el estudio realizado en el estado del arte.

3.2. Particionado de los datos

En este punto, estudiaremos la importancia de un correcto particionado de los datos para el entrenamiento del modelo y su posterior verificación de rendimiento. Se estudiará la importancia de disponer de un estimador insesgado del modelo entrenado, y las decisiones tomadas para el problema que nos concierne en lo que respecta a la calidad de imagen. Para su mejora en calidad, emplearemos GANs con las imágenes de inferior resolución.

3.2.1. Importancia de la separación train-test

En las tareas de aprendizaje de Machine learning y Deep learning, el objetivo es disponer del mayor número de datos posible, por lo que la opción de privar al proceso de entrenamiento de un subconjunto de ellos, o incluso proceder a su eliminación, puede ser una decisión demasiado drástica que ha de tomarse con fundamento.

Sin embargo, cuando realizamos un modelo de aprendizaje, realmente estamos creando una hipótesis, un modelo que construimos y queremos hacer que se acerque lo máximo posible a la expresión que define la población real del problema. Pero, para poder estimar la bondad de nuestro ajuste de forma fiable, necesitamos algún estimador no sesgado que nos indique la calidad del modelo, que nos permita saber el error que comete nuestra hipótesis con respecto a la distribución de la población real, al que denominaremos como error fuera de la muestra, abreviado como E_{out}. Esto se debe a que, para construir este modelo aproximado, estamos empleando una muestra de la población, la cual es continuamente iterada para ir ajustando los filtros aplicados sobre el conjunto y que permite inferior de sus características extraídas las etiqueta de cada una de las imágenes.

Podemos conseguir un estimador insesgado separando un conjunto de datos del total, y obteniendo un conjunto de test, el cual no se ha visto involucrado en ninguna fase del proceso de aprendizaje. La hipótesis final, a la que llamaremos g, es evaluada en este subdataset, y el resultado será un buen estimador del E_{out}. A este error

podemos llamarlo E_{test}. Al seleccionarlo como estimador de E_{out}, estamos afirmando de alguna manera de que éste se tratará de un estimador que generaliza muy bien el error out-of-sample.

Este conjunto recibirá un número efectivo de hipótesis $|\mathcal{H}| = 1$, es decir, únicamente es evaluado con la hipótesis final de nuestro modelo (su ajuste paramétrico final). Esta hipótesis se elige sin conocer el comportamiento de los datos que el conjunto de test contiene, ya que si la estimación de g se ve afectada de alguna forma por los ejemplares ya utilizados en el entrenamiento, no conseguiremos un estimador insesgado, y la ecuación de Hoeffding, no será aplicable. Dicha ecuación enuncia, que para muestras idénticamente distribuidas, e independientes entre sí, se cumple la siguiente desigualdad [5]:

$$P(\mathcal{D} : |E_{out}(h)E_{in}(h)| > \epsilon \leq 2e^{-2\epsilon^2 N}$$

Donde, para cualquier valor de ϵ:

- $\mathcal{D}$ es el nombre que recibe el conjunto de entrenamiento.
- E_{out} es el error out-of-sample, error teórico con respecto a distribución real de los datos.
- E_{in} es el error in-sample, es decir, el error cometido durante el proceso de entrenamiento entre las etiquetas inferidas, y su valor real, también conocido como y verdadero.
- N es el tamaño de la muestra de datos disponible.

Es decir, se cumple que E_{out} se convierte en un estimador muy cercano al valor real del modelo. Dichas restricciones de independencia y pertenencia a la misma distribución son claves, motivo por el cual la eliminación de imágenes duplicadas fue clave (ya que cualquier contaminación entre el conjunto de entrenamiento y aprendizaje puede crear sesgos al alza en los resultados).

El valor del entrenamiento E_{in} sí que sería un estimador sesgado de forma optimista, pues el modelo ha adaptado los filtros para ser capaz de inferior el mayor número posible de etiquetas correctas del conjunto de entrenamiento.. Pero esto no ha ocurrido con test, por lo que podemos dar como resultado un estimador fiable y robusto de cómo se comportará nuestro modelo con otros elementos de la población real.

Cuanto mayor es el tamaño del conjunto, más cercano será el valor del error de nuestra partición de test, E_{test}, con respecto a E_{out}. Pero, cuantos menos datos para entrenar dispongamos, menos se parecerá la estimación al comportamiento real

de la población, y a f. Tenemos que conseguir un equilibrio entre ambos tamaños para aprovechar la mayor cantidad de datos posibles, pero disponiendo de un E_{test} fiable. Normalmente, los valores recomendados suelen ser entre un 10 %-30 % de porcentaje de los datos dedicados al conjunto de test, quedando entre un 70-90 % de datos de entrenamiento. Para este problema, la separación entrenamiento-test elegida ha sido 60-40, respectivamente, que a priori, puede ser muy elevada y salir de la recomendación empírica, pero debido a la existencia de grandes desiquilibrios y una cantidad de datos más que suficiente, de aproximadamente 110.000 ejemplares, podemos obtener de esta forma un estimador muy fiable.

A continuación, se realizará todo el proceso de separación de datos, entre entrenamiento y test en proporción 60-40. Aunque, idealmente, este proceso de hace de forma completamente aleatoria y sin intervención, he optado por realizar la separación de forma estratificada. Esto hace referencia al mecanismo de separación de los datos teniendo en cuenta el conteo de cada una de las clases. Lo que he realizado ha sido, en primer lugar organizar los datos por sus respectivas clases, dentro de cada dataset, de forma que se tienen dos vectores cuyo contenido son las tuplas de cada clase por separado.

Pero este aspecto no solo se da a nivel de subdataset, sino también a nivel del dataset global; es importante tomar una cantidad proporcional de cada clase para evitar que el algoritmo sesgue sus resultados al encontrar patrones comunes propios de cada subconjunto, y este no sea capaz de aprender la visión general de los elementos que definen a un tipo de lesión concreto. Por tanto, definiré un "doble nivel de estratificación", donde:

1. A nivel de cada subdataset, extraer mediante partición estratificada un porcentaje de entrenamiento, y otro de test en proporción 60-40 a la hora entrenar.

2. Volver a unir las clases de entrenamiento de cada dataset, y test de cada dataset, en los dos ficheros que comprenderán las imágenes asociadas al conjunto de entrenamiento y al conjunto de test.

De esta forma, podemos conseguir proporcionalidad entre dataset y entre clases entre train y test, y hacer que la distribución de las imágenes sea prácticamente la misma. Aunque normalmente se suele realizar la separación de forma completamente aleatoria, en clasificación podría darse el caso extremo en el que una clase completa no aparezca en el conjunto de entrenamiento, y nuestro modelo obtenga pésimos resultados de E_{test}. Si bien la probabilidad es muy baja, matemáticamente puede suceder estadísticamente hablando, y consideramos que no afecta negativamente al análisis haber realizado la partición de forma proporcional a cada clase y dataset.

Comentando en detalle la implementación realizada, es de interés destacar el empleo de la función train-test split de SKlearn [50]. Se trata de una función que realiza

de forma bastante simplificada el proceso de partición estratificada. En este caso, además, se verificará la existencia de clase con menos de 16 ejemplares de imagen presentes para confirmar que no se han filtrado clases minoritarias las cuales son imposibles de clasificar con el modelo actual. El resultado es el código de la figura 3.6

Algorithm 3.6 Separación de las filas según el valor de la columna "column"

```
MIN_COUNT = 16
procedure SEPARAR_SUBDATA(dataframe, column)
    var separated ← dataframe [c_1, c_2, ···, c_n] / c_{i_j} = column y separated[k] ∩
separated[l] = 0 para k ≠ l
    var trainData, testData: list of tuples
    lista_elementos: array of tuples
    for each data_part, contenido in separated do
        lista_elementos = lista_elementos ∪ contenido
        if lista_elementos[çlass"].map(counts) >MIN_COUNT then
            var train, test = TRAIN_TEST_SPLIT(subdata, test_size=0.4,seed=19, shuf-
fle=True, stratify=subdata['class'])
            trainData = trainData ∪ train
            testData = testData ∪ test
        end if
    end for
return lista_elementos
end procedure
```

De esta forma, seguimos manteniendo la proporción a nivel de conjunto global, y de subdataset. En caso de necesitar entrenar únicamente con uno de los subconjuntos añadidos, podemos filtrar el cojunto de tuplas por aquellos cuya clase sea igual a la buscada.

3.2.2. Particionado de validación

Una vez separado nuestro conjunto de test, únicamente trabajaremos con el resto de valores, que componen el conjunto de entrenamiento. Con ellos, realizaremos el estudio estadístico, ya descrito parcialmente en el punto anterior, y todo el procedimiento de preprocesado de imágenes y ajuste del modelo.

Durante las fases de ajuste del modelo elegido en nuestras hipótesis, necesitaremos estudiar qué parámetros favorecen mejores resultados, así como el tipo de regularización a emplear y el coeficiente λ de participación de este.

Una forma de realizar esta práctica consiste en el uso del subconjunto de validación. Este concepto se asemeja al concepto del conjunto de test, con la diferencia de que se utiliza como estimador insesgado para evaluar los parámetros relevantes de

cada algoritmo, o los modelos a comparar. Estimamos el error out-sample de forma directa mediante el error obtenido E_{val}. Lo conseguimos al evaluar el modelo sin tener en cuenta los K valores extraídos para la validación, utilizándolo como estimador, E_{val}.

$$E_{out}(g) \approx E_{out}(g^{-}) \approx E_{val}(g^{-})$$

Siendo $E_{out}(g)$ el error final cuando se utiliza todo el conjunto de datos $\mathcal{D}$. La parte izquierda de la ecuación funciona mejor cuanto más pequeña sea k, mientras que por la derecha, un mayor valor de k permitirá una mejor estimación a traves de E_{val}.

El caso ideal sería, para acercanos lo máximo posible entre $E_{out}(g)$ y $E_{out}(g^{-})$ es elegir k lo más pequeño posible, cuyo caso extremo es k=1. De esta forma estaríamos haciendo **leave one out**, método el cual evalua todo el conjunto como entrenamiento, reservando solo un único valor para validación. La combinación de este proceso aplicado a todos los elementos nos permitirá conseguir el error de valicación cruzada, dado por:

$$E_{cv} = \frac{1}{N} \sum_{n=1}^{N} e_n$$

donde $e_n = E_{val}(g_n^{-}) = e(g_n^{-}(x_n), y_n)$, tal y como se ha explicado. Pero es muy costoso computacionalmente, sobre todo ante casos como este, donde el conjunto de entrenamiento contiene alrededor de 60000 valores. Es por ello que simplificaremos este enfoque al conjunto de validación único.

Un caso menos costoso computacionalmente es V Cross-Validation, donde dividimos todo el conjunto de datos en V subconjuntos, 5 o 10 habitualmente, llamados **folds**. En este proyecto, donde disponemos de un conjunto considerable de valores, no he considerado conveniente realizar el proceso de Validación cruzada (CV), ya que este consistiría en disponer de varios conjuntos de validación distintos, y entrenar el modelo para cada uno de los conjuntos de datos restantes no pertenecientes a validación. Debido a que los tiempos de entrenamientos de redes convolucionales pueden superar los días y semanas de duración, no permitiría tener en cuenta otros aspectos claves del proyecto más allá del ajuste de parámetros.

Los valores escogidos para esta partición será un **70 % de entrenamiento, y 30 % de validación**.

3.3. Uso de GANs de super resolución

Las imágenes procedentes de los datasets seleccionados poseen una buena resolución, siendo esta, en media, prácticamente equivalente a lo que se conoce como alta resolución en aprendizaje profundo (1024 x 1024). Sin embargo, aquellas imágenes pertenecientes a los datasets Severance y ASAN, que fueron almacenadas en forma de matriz de imágenes, poseen una calidad muy pobre, de apenas 98x98 píxeles, como ya destacamos durante su preprocesado. En caso de que estos datasets aporten clases completamente nuevas respecto a los 3 conjuntos restantes, la red podría clasificarlas erróneamente basándose únicamente en la resolución.

Un simple reescalado solo provocaría altas frecuencias en la imagen que colaborarían en la aportación de ruido al modelo. Hacen falta técnicas más avanzadas, que permitan hacer crecer el tamaño de la imagen sin interferir en su estructura conjunta. Una de estas alternativas, es el uso de redes generativas adversarias (GANs).

3.3.1. Justificación de uso

Las redes generativas adversarias (GANs) [20] son una arquitectura de aprendizaje profundo de tipo generativo, cuyo objetivo suele ser la generación de imágenes sintéticas capaces de imitar con un alto grado de confianza un conjunto de imágenes concreto dado como entrenamiento, estudiando su distribución y siendo capaces de generar imágenes capaces de hacerse pasar por las originales.

Se trata de una arquitectura surgida en 2014 del grupo de investigación de Ian Goodfellow [20], cuya arquitectura consiste en el uso de dos redes profundas: la red generadora, y de discriminatoria.

- La de red generadora genera imñagenes que intentan asemejarse a las proporcionadas como entrenamiento, de forma que estas se hagan pasar por las originales.
- La red discriminativa, infiere si la imagen recibida como entrada es o no verídica, y le transmite dicho valor de confianza a la red generadora, para que ésta sea capaz de ajustarse para mejorar el resultado.

Este tipo de arquitectura, basada en la colaboración entre las dos redes, tiene cierta similitud con el aprendizaje con refuerzo, ya que los ajustes a realizar sobre el modelo generador, depende con fuerza del feedback recibido por la función de pérdida de la red discriminativa.

Se trata de un proceso comúnmente vinculado al aprendizaje no supervisado, en el que el interés reside en la generación de imágenes sintéticas para fines como el entrenamiento de otros modelos profundos con clases infrarrepresentadas. La

GAN puede encargarse de aumentar el tamaño sintéticamente de aquellas clases infrarrepresentadas para mejorar la calidad del aprendizaje y que este se realice de forma equilibrada. Por tanto, puede considerarse como una técnica de aumento de datos.

Sin embargo, esta no se trata de la única aplicación. El problema al que nos enferentamos, donde requerimos un aumento de la resolución para la mejora de calidad de la imagen, también es una tarea realizable y demostrada por la literatura en la que las GANs logran buenos resultados. Al expandir una imagen de tamaño, requerimos "rellenar" aquellos píxeles intermedios faltantes al redimensionar la imagen. Ésta tarea es bastante similar a la realizada pora una GAN tradicional, con la diferencia de que sólo debemos generar los píxeles intermedios, y de forma que éstos se adapten al entorno del resto de la imagen. ESRGAN[68] logra este efecto.

Esta red ha demostrado cuantiosos éxistos en la literatura al ser utilizada como forma de reescalado de datos, debido a sus mejoras propuestas frente a las GANs tradicionales. Una de las claves reside en la función de pérdida; en lugar de discriminar si la imagen es adecuada o no, se ha empleado un "grado de percepción"(Perceptual Loss) de la imagen, donde se discrimina la calidad de la imagen generada en una escala.

Figura 3.3: Arquitectura de ESRGAN. Fuente: [68]

Además, incorpora a la arquitectura las conexiones residuales al estilo de ResNet, con la diferencia de que todos los niveles de un bloque están conectados a todas las siguientes una conexión residual, arquitectura la cual han denominado Residual-in-Residual. Esto les ha permitido evitar la decaída de gradiente, y además, prescindir de la capa de Batch Normalization, que introducía artefactos en el resultado final.

Concluimos que este modelo es el adecuado para el problema, debido a sus buenos resultados a nivel de mejora de imagen, y de sus resultados ya mostrados en modelos de la literatura [22], donde la aplicación de ESRGAN ofrece mejoras en el resultado del modelo evaluado, siendo este resultado evaluado con distintas arquitecturas y modelos de redes convolucionales del estado del arte.

Gracias a este modelo, el aumento de resolución podrá darse para equiparar la calidad de las imágenes recopiladas.

3.3.2. Proceso de reescalado

La aplicación del proceso de mejora de las imágenes se llevará a cabo únicamente sobre el conjunto de entrenamiento, para así evitar la contaminación del conjunto de test. Podemos encontrar el modelo de ESRGAN ya preentrenado en su repositorio de Github [67].

Dentro del software incorporado para su uso, podemos encontrar el modelo preentrenado para una multiplicación de la resolución a 4 veces mayor. Este será el modelo con aquellos subconjuntos de menor resolución, para así aumentar la resolución de 98x98 a 392x392 píxeles. El procedimiento a seguir ha sido el siguiente:

1. Listar las imágenes de entrenamiento pertenecientes a los subconjuntos ASAN y Severance

2. Aplicar el aumento de resolución, generando las nuevas imágenes en una carpeta nueva

3. Sustituir las imagenes originales por las aumentadas, conservando las imágenes fuente en un directorio de respaldo.

De forma organizada, nos queda resumido en el siguiente pseudocódigo:

Algorithm 3.7 Aumento de resolución para ASAN y Severance

```
procedure REESCALAR_IMG(path_orig, nombre_carp)
    if not exists(nombre_carp) then          ▷ Renombrar la carpeta a "nombre_lr"
        RENAME(path_orig, nombre_carp)
        MKDIR(path_orig)                     ▷ Creamos carpeta con el nombre original
        READ_IMAGES(path_orig + "_lr", path_orig)
    end if
end procedure
```

Donde la función 'read_images' 3.8 es la encargada de recorrer la carpeta, tomar las imágenes y generar su imagen aumentada por super resolución:

Algorithm 3.8 Aumento de resolución para ASAN y Severance

```
procedure READ_IMAGES(path_orig, nombre_carp)
    var im_path += "/*"
    var model_path = 'ESRGAN/models/RRDB_ESRGAN_x4.pth':path
    LOAD_MODEL(model_pat)
    var idx ← 0
    for each path in im_path do
        idx ← idx + 1
        var base_dir= osp.splitext(osp.basename(path))[0] :path
        var img = IMREAD(path, format = IMREAD_COLOR)
        img = img ∗ 1.0/255                    ▷ Normalización antes de aumentar
        img = img^T.squeeze()                  ▷ Formato de entrada del modelo
        img_LR = img.unsqueeze(0).to(GPU)
        output ← model.predict(img_LR)
        output ← output,squeeze()
        output = model(img_LR).data.squeeze().float().clamp_(0, 1)
        img = img^T ∗ 255                      ▷ Deshacemos cambios
        Call imwritef'im_out/' + 'png'.format(base), output)   ▷ Guardo en disco
    end for
end procedure
```

De esta forma, conseguimos reescalar cada una de las imágenes sin necesidad de modificar el archivo .csv previamente guardado para entrenamiento y test. Los resultados son bastante notables, ya que al aumentar la resolución a una 4 veces mayor, pueden apreciarse detalles con mayor claridad 3.4.

l

(a) Original (b) Reescalada

Figura 3.4: Imagen original (izquierda) vs aumentada (derecha)

Como inconvenientes, destacar la gran necesidad de espacio en disco, ya que las

imágenes al ser reescaladas y almacenadas en formato sin pérdidas (PNG), han multiplicado el tamaño del dataset completo de entrenamiento en casi 6 veces. Adicionalmente, se han requierido una hora y doce minutos para el reescalado de las imágenes, haciendo uso de una GPU Nvidia Geforce RTX 3080.

4 Modelos y entorno de trabajo

Una vez disponemos de los datos correctamente preprocesados, podemos pasar a la fase de construcción del modelo. En este proyecto, se usará la cuantización de modelos tras el entrenamiento (Post Training Quantization), por lo que el diseño del modelo , su arquitectura y su forma de entrenamiento se realizan de la forma habitual, con la particularidad del proceso posterior. Sin embargo, existe una serie de factores a tener en cuenta durante la creación del modelo:

- La memoria y potencia del dispositivo móvil son limitadas. Aunque la cuantización consigue reducir el tiempo de inferencia y el espacio ocupado por el modelo, la complejidad del mismo sigue siendo proporcional, por lo que un modelo excesivamente profundo o pesado a nivel espacial puede suponer un problema.
- La resolución de las imágenes de entrada es variable, ya que depende del terminal en el que se ejecute y su cámara. Es posible que se requiera un reescalado de la entrada (resize) o la posibilidad de disponer de un tamaño de entrada sin fijar. Esto es posible mediante la no utilización de capas totalmente conectadas, o de pooling.

En los siguientes apartados, entraremos en detalle en cada uno de estos aspectos, y describiremos los modelos a probar: MobileNet, y las familias de redes EfficientNet y ResNet.

4.1. Conceptos previos

Habitualmente, para el estudio y creación de modelos cuya información de entrada son imágenes, se hace uso de redes convolucionales. Estas redes, a diferencia de las redes neuronales habituales, donde tenemos una serie de neuronas conectadas entre sí formando distintas capas, tenemos un conjunto de capas de procesado local, que permiten aplicar transformaciones sobre las imágenes para simplificar su estructura y destacar elementos característicos que permitan obtener características útiles y distintivas para realizar clasificación de la información que muestran.

Los pesos que se aprenden son, precisamente, los coeficientes y parámetros de transformaciones (convoluciones) a realizar sobre la imagen , con la peculiaridad de que estos se aprenden automáticamente dentro de la red, y no es necesario de especificarlos manualmente. La propia red, en base a su función de pérdida, intentará

maximizar los resultados entre el valor predicho y real, y encontrará los parámetros más adecuados. Únicamente, debemos indificar las capas que compondrán la red, y la dimensionalidad de entrada y salida para que sean correctamente interconectables.

Por tanto, el nombre convolucional proviene de la capacidad de aprender los valores para los filtros que queremos aplicar para destacar propiedades. Para una convolución, exiten las siguientes modificaciones y variantes:

- Padding. Consiste en el relleno de bordes auxiliares en la imagen para obtener una imagen resultante con la misma dimensionalidad que la entrada de una capa. Habitualmente, dicho relleno se realiza con ceros (zero-padding).
- Stride. Indica el desplazamiento que realiza el filtro sobre la imagen. por defecto, dicho valor es 1, por lo que la imagen se recorre píxel a píxel. Pero puede ser aumentado para reducir la dimensionalidad del problema.
- Dilation: espacio entre los valores con los que opera el kernel. Por defecto, su valor es 1, es decir, los kernels operan con los valores adyacentes al píxel sobre el que se trabaja, pero puede ser una distancia mayor. Por ejemplo, con padding de 2, el área de influencia de los píxeles del entorno alejados dos píxeles de su centro adquieren mayor presencia en el resultado de la convolución.

Habitualmente, encontramos en ellas dos tipos de capas para el aprendizaje de pesos. Se encuentran en la mayoría de modelos del estado del arte:

- Capa totalmente conectada. Son las capas hasta ahora vistas en Aprendizaje automático, donde cada neurona de un nivel está conectada a todas las del siguiente. Son costosas computacionalmente, y requieren un formato de entrada concreto, especificando de antemano su tamaño de entrada. En este problema, las imágenes tomadas con el teléfono movil pueden variar en tamaño, debido al recorte de las imágenes o a la calidad variable de la cámara que la capta. Supone un problema, el cual que podemos solucionar mediante una capa previa de Average Pooling: es una operación convolucional capaz de reducir la dimensionalidad de la imagen de entrada, aprendiendo de la información resumida para extrear de características. Es una de las capas que habitualmente se usa cuando tomamos modelos ya entrenados sobre un tamaño de entrada distinto al que requerimos usar.
- Capa convolucional. Son el resultado de un dot-product sobre una región concreta del volumen de entrada, operando a nivel local. Gracias a la localidad, no es necesario que cada neurona está conectada a todas del siguiente nivel, por lo que se produce un ahorro computacional considerable. En resumen: se trata de una matriz filtro que se desplaza a lo largo y alto de la imagen, realizando una operación de producto.

En las redes neuronales, nos interesa procesar los datos con capas convolucionales que estimen los parámetros para las traformaciones adecuadas de la imagen, mientras que las capas fully-connected serán más útiles en capas finales, ya que al estar conectadas totalmente, tienen acceso a todos los valores de entrada, y permitirán obtener de forma condensada los resultados para la clasificación de las imágenes. Es recomendable el uso esta capa cuando la imagen es suficientemente reducida por las convoluciones.

El proceso de entrenamiento de estas redes es generalmente costoso, debido al entrenamiento de una gran cantidad de parámetros, que habitualmente superan los varios millones. Normalmente, los modelos convolucionales empleados para la resolución de problemas suelen ser preentrenados, modelos ya configurados entrenados con millones de datos durante períodos de tiempo prolongados, de forma que adquieren capacidades generales de filtro para un espectro muy amplio de imágenes.

Cuando se requiere su aplicación para una tarea específica, que en este caso se trata de la piel, basta con realizar un ajuste de los parámetros mediante el entrenamiento del modelo durante una serie de epocas reducidas: a esto se le conoce como transfer learning, y será la técnica que aplicaremos para la creación de los modelos mediante MobileNet, EfficientNet y Resnet.

4.2. Modelos preentrenados escogidos

A cotinuación, se detallan lar arquitecturas seleccionadas para el entrenamiento del modelo, teniendo en cuenta aquellos modelos del estado del arte que no fueron existosos. Los 3 modelos serán sometidos a procesos de entrenamiento similares, de forma que el de mejor desempeño de los 3 muestre será el elegido para ser parte de la aplicación móvil.

Al tratarse de modelos ya entrenados con un dataset de gran tamaño, es necesario adaptar su salida para que ésta se adecúe a nuestras necesidades; dicha información la veremos posteriormente, ya que la capa de adaptación será muy similar para los 3 modelos seleccionados.

4.2.1. ResNet 50

La familia de redes preentrenadas Resnet [29] es bastante amplia. Dispone de sus versiones 18, 34, 50, 101 y 153. Cada una de estas redes hace uso de la misma configuración de capas, pero replicando esta con mayor o menor profundidad. Para nuestro problema, las versiones de 18 y 153 quedan descartadas, ya que 18 unidades de profundidad son insuficientes para la complejidad y variedad de nuestro proble-

ma. No serían capas suficientes para extraer todas las características necesarias del entrenamiento.

En cuanto a Resnet153, su excesiva profunidad requiere grandes cantidades de datos para ajustar las capas más profunda de la red, ya que tras cada capa supera, el gradiente que regula la optimización del modelo hacia el óptimo decae, y provoca que los ajuste de las últimas capas sean mínimos. A este fenómemo se le llama desfallecimiento de gradientes, y es un suceso bastante común en redes profundas como esta.

A pesar de que Resnet se caracteriza por la aplicación de conexiones residuales, que permiten interconectar distintas capas distanciadas entre sí sin necesidad de transcurrir sobre las unidades intermedias, la cantidad de datos requerida, y el tiempo de inferencia necesario para el modelo en un dispositivo móvil son excesivos.

En la literatura, se encuentran algunos casos de utilización de esta arquitectura, pero no se tienen datos de la aplicación de este modelo sobre teléfonos móviles mediante cuantización. Teniendo en cuenta este factor, podemos considerar este modelo como parte de los candidatos.

Model	Input	Size (model)	Size (feat.)	Est. FLOPS
resnet18	224 x 224	45 MB	23 MB	2 GFLOPs
resnet34	224 x 224	83 MB	35 MB	4 GFLOPs
resnet-50	224 x 224	98 MB	103 MB	4 GFLOPs
resnet-101	224 x 224	170 MB	155 MB	8 GFLOPs
resnet-152	224 x 224	230 MB	219 MB	11 GFLOPs

Tabla 4.1: Rendimiento de ResNet en **Imagenet** [7]

Concretamente, se procede a escoger la versión ResNet50 del modelo, cuya característica clave es ser el punto de equilibrio entre modelos de pequeño y gran tamaño por sus requisitos de memoria y tiempo de inferencia. Podemos apreciar estos valores en la tabla 4.2.1

4.2.2. MobileNet V2

Mobile Net es una red convolucional especializada en su uso directo en dispositivos móviles. Este modelo ya fue descrito en su totalidad en el capítulo 2.1.2. Entre sus modelos preentrenados que podemos encontrar, debemos destacar la existencia de los modelos V1 y V2:

Ambos modelos son bastante ligeros, por lo que no requieren cuantización para obtener buen rendimiento. En este caso, se elige la versión V2, debido a su mejor eficiencia computacional sin pérdida de resultados. En cuanto ala profundidad y ancho de la red, parámetros clave, se utilizan profunidad = 1 y anchura = 1, es decir,

Model	Input	Size (model)	Est. FLOPS
MobileNet-V1	224 x 224	16.9 MB	0.569 GFLOPs
MobileNet-V2	224 x 224	14.0 MB	0.3 GFLOPs

Tabla 4.2: Rendimiento de MobileNet en **Imagenet** [45]

no se realizará ninguna reducción sobre el modelo, de forma que pueda tener profundidad suficiente para aprender todas las clases del conjunto de entrenamiento. Se descarta el uso del modelo V3 por su especialización en segmentación, ya que el problema que estamos tratando es clasificación.

Sin embargo, tal y como veremos en el estudio de resultados posteriomente, los resultados ofrecidos no estarán finalmente a la altura de la aplicación.

4.2.3. EfficientNet B5

EfficientNet, modelo descrito en el capítulo 2.1.4, se trata de otro conjunto de arquitectura de red cuyo funcionamiento en problemas de clasificación de enfermedades cutáneas es positivo, tal y como demuestran los resultados ganadores de la competición de ISIC Challenge [15, 47].

Como ya detallamos anteriormente, estos modelos son demasiado costosos para entrenar y ejecutar en sus versiones de gran tamaño, como las utilizadas en dichas soluciones. En este caso, emplearé de nuevo el modelo intermedio, EfficientNet B5, el cual logra un equilibrio suficiente entre tamaño y calidad del resultado. Además, por limitación del hardware disponible, no es posible utilizar sus variantes superiores, ya que ni el entrenamiento e inferencia de los mismos es posible.

Model	# of Parameters	Est. FLOPs
EfficientNet-B0	5.3M	0.39B
EfficientNet-B1	7.8M	0.70B
EfficientNet-B2	9.2M	1.0B
EfficientNet-B3	12M	1.8B
EfficientNet-B4	19M	4.2B
EfficientNet-B5	30M	9.9B
EfficientNet-B6	43M	19B
EfficientNet-B7	66M	37B

Tabla 4.3: Tamaño de las versiones de Efficient Net [63]

El tamaño del model final no es calculable debido a la amplia variedad de parámetros a configurar en cuanto a su arquitectura. En el caso de la competición antes mencionada, este rondaba entre 300 y 400MB para la versión B7, motivo por

el cual se ha decidido seleccionar la versión B5, y obtener un tamaño estimado de 150-200MB.

4.3. Función de pérdida

El modelo a entrenar requiere el uso de una función que nos permita transformar la salida paramétrica del modelo, en un valor numérico legible, que nos permita conocer su progreso en el entrenamiento. Es decir, nos interesaría saber tanto el error E_{in} como E_{val}, como un valor de pérdida, que queremos minimizar para ajustarnos lo máximo posible a la distribución real. Mientras que la probabilidad de pertenencia a cada clase nos la ofrece la función Softamax, la función de pérdida puede ser elegida entre varias alternativas. Comúnmente se utiliza CrossEntropy loss, pero para este problema, utilizaré también FocalLoss, especializada en el ajuste de modelos con clases desbalanceadas, como es nuestro caso.

4.3.1. Cross entropy

La entropía cruzada es una función de pérdida utilizada comúnmente en problemas de clasificación. Su imagen se utiliza como valor a minimizar, y representa la calidad de ajuste del modelo.

Se apoya en las salidas de la función softmax. Conocemos que Softmax proporciona como salida, en un problema de clasificación, la probabilidad de que el ejemplar que estamos clasificando pertenezca a una clase concreta. La entropía cruzada se basa en calcular la distancia existente entre las salidas probabilísticas de la función softmax, y el valor real de su etiqueta. Se calcula cuánto divergen los valores entre sí de los valores predichos contra los valores reales.

Como su resultado es nuestra representación del error, queremos que sea lo más pequeño posible. Por eso, se intenta minimizar esta función para obtener mejores resultados.

Su expresión analítica es la siguiente:

$$H(P,Q) = -\sum_{x \in X} p(x) log(q(x))$$

Donde p es el valor real, y q es el valor estimado por nuestro modelo para cada ejemplo del conjunto X de entrenamiento.

Esta función es adecuada para la mayoría de casos, aunque puede ver perjudicado su rendimiento cuando las clases son dispares en número entre sí.

4.3.2. Focal Loss

Focal loss [39] es una función de perdida que modula el aprendizaje en conjuntos de datos desbalanceados, para fomentar el aprendizaje de ejemplos infrarepresentados,

o más complejos, sobre aquellos más sencillos.

Para ello, parte de las expresiones de CrossEntropy loss, aplicando el efecto de una valor regularizador, al que llamaremos α. Este peso aportará más valor a los casos complejos, mediante la siguiente expresión:

$$FL = \begin{cases} -\alpha(1-p)^{\gamma} log(p) & y = 1 \\ \\ -(1-\alpha)p^{\gamma} log(1-p) & otherwise \end{cases}$$

Donde $log(p)$ hace alusión a CrossEntropy evaluado a un único elemento, y el valor αes aquel que modula el efecto regularizador.

Los casos díficiles hacen referencia a los Falsos negativos' mientras que los sencillos son los correctamente clasificados, en especial los Verdaderos negativos. Cuanto más valor se aporte a γ, mayor será la importancia de los objetos no clasificados correctamente, y con α, cuanto mayor, mayor es la importancia del conjunto no clasificado en completo.

Esta función de pérdida es muy útil para el problema, ya que el dataset sufre de un problema de desbalanceo importante en sus clases, tanto a nivel de clasificación benigno-maligno, como a nivel de enfermedades de cada tipo. Por tanto, esta función de pérdida se hace ideal para dar más importancia a las clases menos representadas, y que por ende, tenderán a ser marginales para el modelo si no se ajusta adecuadamente. Su Su configuración por defecto, con $\gamma = 2$ es la adecuada para la mayoría de problemas, y será la configuración empleada para este dataset de entrenamiento.

4.4. Especificaciones y framework de trabajo

La reaización de este proyecto usará el lenguaje Python, debido a la gran extesión de las librerías y frameworks de Deep learning desarrollados en la última decada. Concretamente, se basará en el uso de la librería de aprendizaje Pytorch[49], con el apoyo del framework de alto nivel FastAI[33].

Se ha elegido Pytorch por su tendencia creciente en las publicaciones científicas, así como su interfaz simplificada frente a Tensorflow. Aunque ambas soluciones son equivalentes, se ha elegido la primera por su reciente incorporación de capacidades para modelos Android, capaz de superar levemente en rendimiento promedio a Tensorflow lite.

FastAI, por su parte, esta basado en Pytorch, y permite la implementación de modelos y la organización de los datasets de forma simplificada mediante un mayor grado de abstracción. Permitirá simplificar la implementación en aquellas tareas que

funcionen adecuadamente con las configuraciones ofrecidas.

Para las ténicas de aumento de datos, se emplearán dos librerias:

- Imgaug: librería de aumento de datos, específicamente diseñada para tareas de aprendizaje profundo, y agnóstica en cuanto al framework de desarrollo empleado. Esto permite su utilización de forma independiente, de forma previa al aprendizaje, para realizar otras tareas, como el oversampling de clases minoritarias, procedimiento el cual aplicaremos para equilibrar el entrenamiento en clasificación binaria (benigno vs maligno).
- Albumentations: se trata de otra librería de aumento de datos, que se integra de forma nativa con el formato datablock de Fastai. Nos permitirá realizar aumento de datos de la forma habitual, cuando no se necesite aplicar oversampling a los datos de entrenamiento.

En cuanto a las características del sistema, se usarán las dos configuraciones siguientes:

- Intel Core i7 12700K, Nvidia RTX 3080, 32GB de RAM
- Intel Xeon Silver 4216, Nvidia Quadro RTX 8000 32GB, 20GB RAM

Una vez definido el framework y las especificaciones a utilizar, podemos pasar a detallar el proceso de aprendizaje completo y sus resultados.

5 Entrenamiento de modelos completos sobre la base de datos

Tras las descripción de los modelos a utilizar, y las funciones de pérdida a emplear, comienza el proceso de entrenamiento de los modelos. Este proceso es uno de los más lentos de todo el desarrollo del proyecto, debido a la cantidad de horas necesarias a esperar para recopilar los resultados del modelo. Se estudiará la eficiacia de cada modelo a la hora de evaluar el problema, y las decisiones tomadas para su mejora y elección.

Para resolver este problema, se ha tomado un enfoque de dos niveles; en primer lugar, distinguiremos si la enfermedad que se recibe como entrada se trata de un ejemplar de enfermedad maligna o benigna, y posteriormente, su salida indicará un segundo modelo a aplicar sobre la enfermedad, especializado únicamente en uno de los dos tipos de enfermedades. Así, conseguimos un conjunto de modelos especialziados que permiten otorgar el diagnóstico con mayor precisión y seguridad.

Un aspecto clave de este proceso será la eficiencia, pero sobre todo el correcto diagnóstico de las enfermedades, por lo que usaremos como métricas de selección precision, el recall y balanced accuracy:

- Precision. Proporción de valores bien clasificados dentro de una clase teniendo en cuenta verdaderos y falsos positivos.

$$Precision = TP/(TP + FP).$$

- Recall. Valores correctamente identificados como positivos respecto al total de elementos positivos.

$$Recall = TP/P$$

- Balanced accuracy: combina sensitivity y specificity. La primera, devuelve el valor real de proporción de valores correctamente clasificados entre el total de casos positivos que tenemos (contando predicciones positivas y falsos negativos), mientras que la especifidad, nide el caso dual: la proporción de casos negativos bien clasificados respecto al total de datos negativos tanto bien clasificados como mal clasificados, y que son identificados como falsos positivos. Esto nos permite calcular el accuracy de forma proporcional al porcentaje de

presencia de las clases, e intentamos así paliar el efecto del desequilibrio de los datos:

$$Sensitivity = TP/(TP + FN)$$

$$Specificity = (TN/(TN + FP)$$

$$Balanced\ accuracy : \frac{Sensitivity + Specificity}{2}$$

En base a esta expresión, podemos saber que si el valor es aproximadamente 1/númeroClases, es posible que gran parte de las clases no estén siendo correctamente clasificadas en caso de ser minoritarias. Valores mayores nos harán conocer que el resultado de clasificar todas las clases es satisfactorio.

5.1. Clasificación binaria

El problema de la clasificación binaria consiste en la distinción de las enfermedades malignas de aquellas que son benignas. Se trata del primer problema a resolver antes de especificar el tipo de enfermedad que posee el paciente, y así poder especializar los modelos segundo nivel.

5.1.1. Equilibrio de los datos

El primer inconveniente que nos encontramos para este problema es el inmenso desbalaceo entre la clase benigna y maligna. Si redibujamos el gráfico mostrado durante la fase de preprocesamiento de datos, únicamente con los datos de entrenamiento, obtenemos lo siguiente:

Figura 5.1: Desequilibrio de datos

Se puede observar la alta disparidad existente entre ambas, siendo 35664 casos benignos y 9752 malignos. Es una diferencia de 3.65 veces más casos benignos que malignos, lo que se traduce en un equilibrio de 79 % y 21 % respectivamente. Para paliarlo, podemos efectuar dos estrategias: aplicar técnicas de oversampling (también conocido como sobremuestreo sintético), donwsampling de la clase mayoritaria, o bien, la aplicación de penalziaciones sobre la clase minoritaria para que su incorrecta clasificación tenga mayores consecuencias (haciendo uso de FocalLoss).

Técnicas descartadas

La ténica de downsampling de la clase mayoritaria queda descartada, debido a que dentro de esa clase, podemos encontrar etiquetas de segundo nivel asociadas a las enfermedades que se pueden diagnosticar. Como algunas de estas clases son muy escasas en número, si realizamos downsampling de forma aleatoria sin ningún tipo de restricciói n, podría darse el caso límite en la que una clase completa desapareciese del cojunto. Y esto conllevaría a la obtención de errores al evaluación de test, pues tendríamos una clase presente en test no estudiada en entrenamiento, que sólo provocaría un aumento de las métricas de error.

El uso de la función de pérdida FocalLoss podría ayudar a paliar este efecto. Sin embargo, debido al desajuste 79-21 del que disponemos, y la gran variedad de imágenes, se trata de una solución demasiado arriesgada. Para comprobarlo, se sometió a evaluación con los valores por defecto recomendados de Resnet50. A priori, los resultados podrían parecer correctos en validación 5.1; sin embargo,

observamos que la clase minoritaria, la maligna, obtiene resultados que no están a la altura de la clase mayoritaria. Observamos un valor 0.41 en recall, valor preocupante frente al 0.95 de la clase benigna. Al existir un recall bajo, quiere decir que esta clase no se está detectando adecuadamente como maligna, y casi el 60 % de sus casos son calificados como benignos. Esto es un grave problema, ya que los casos malignos son los que más riesgo conllevan para la salud. Por tanto, se descarta también esta estrategia.

	Precision	Recall	F1-score	Support
Benign	0.86	0.95	0.90	15338
Malignant	0.68	0.41	0.51	4126
Accuracy			0.83	19464
Macro avg	0.77	0.68	0.71	19464
Weighted avg	0.82	0.83	0.82	19464

Tabla 5.1: Informe de clasificación

Propuesta: oversampling

Sólo nos queda disponible la estategia de oversampling: la ampliación sintética de la clase minoritaria. Este proceso es muy delicado, ya que se basa en realizar alteraciones sobre las imágenes disponibles para dotarlas al conjunto de una mayor variabilidad, hasta equiparar el número de ejemplares con el de la clase mayoritaria. Existen distintas técnicas a aplicar, pero en este caso, utilizaré la estrategia de realizar alteraciones sobre las imágenes mediante una biblioteca de transformaciones: imgaug.

Se aplicarán una serie de transformaciones la imagen que alteren su estructura, pero con cuidado de no generar imágenes demasiado modificadas que se alejen de la distribución original de la clase maligna. Para realizar las transformaciones, he contruido un pipeline, una secuencia de transformaciones que aplica, en orden aleatorio, y con cierta probabilidad, las siguientes variaciones:

- Fliplr(p=0.5). Se trata de un efecto de espejo de la imagen, respecto a su eje horizontal. Es equivalente a realizar una rotación de la imagen de 180º sobre su eje teórico horizontal que transcurre por la mitad de la imagen. Esta alteración se realziará con un 50 % de probabilidad.

- Flipud(p=0.5). Se trata de un efecto de espejo de la imagen, respecto a su eje vertical. Es un caso de efectos similares al anterior, aplicado con un 50 % de probabilidad.

- Gaussian blur (p=0.5). Se aplicará filtro gaussiano en el 50 % de los casos, especificándose un valor de sigma muy pequeño de 0.5.

- Modificación de contraste (p=0.25). Se aplicará un aumento o decremento del contraste del 20 % en el 25 % de los casos. Esto emula posibles cambios de luminosidad ambiente y de profundidad de color de la cámara

- Ruido Gaussiano aditivo (p=0.5). Se aplica en caso de que el ruido gaussiano estándar no sea aplicado. Aplica un filtro gaussiano a nivel de canal, de forma que solo parte de la imagen se encuentra difuminada. Se aplica también en un 50 % de los casos

- Transformaciones afines (p=0.2): se trata de una alteración compuesta de escala y zoom, y una posterior traslación o rotación de la imagen. Se trata de una opcionalidad ofrecida por imgaug de forma no configurable directamente.

Estos valores han sido escogidos tomando como inspiración las transformaciones realizadas por el equipo ganador de la competición de ISIC [15], debido a sus buenos resultados.

Es importante destacar que dichas transformaciones únicamente se han realizado sobre los datos de entrenamiento; los datos de validación y test permanecen inalterados de forma que éstos sigan siendo un estimador no sesgado del rendimiento del modelo. Poir tanto, la extracción del 30 % de validación se realiza antes del sobremuestreom y los dos subconjuntos se guardan en carpetas distintas, a las que llamaremos train y test.

5.1.2. Representación en Pytorch

Con los datos preparados, configurar mediante Fastai y Pytorch los modelos que deseamos probar. Los 3 modelos siguen la misma configuración, a excepción de la importación del modelo, por lo que será explicado de forma paramétrica. Tenemos 3 tareas diferenciables: la creación del datablock para la gestión de los datos, el entrenamiento del modelo, y la transformación del mismo al modelo cuantizado empleado en el dispositivo Android.

Creación del datablock

FastAI ofrece el tipo de objeto Datablock para la creación de una estructura que organiza los datos de entrenamiento. Permite construir un bloque con todos los datos, de forma que cualquier consulta o verificación pueda realizarse sobre un mismo objeto. Consta de varios parámetros que podemos ajustar para normalizar las imágenes, adaptar el tamaño de entrada, o realizar la separación train-validación. Desglosadando los parámetros más importantes, tenemos:

- blocks. Nos permite establecer el formato de entrada de datos, y la salida que queremos obtener de los mismos. En problemas de clasificación, habitualmen-

te, tendremos como entrada un bloque de imágenes, y como salida, queremos obtener un conjunto de categorías asociados al bloque.

- get_items. Establece la fuente de la cual obtener los datos para la construcción del DataBlock. Se trata de una función preconfigurada para la lectura de datos dado un repositorio o dirección de disco. En nuestro caso, como las imágenes no contienen información compleja, como plantillas de segmentación, se importan directamente de esta forma.

- splitter. Permite especificar divisiones de los datos. Habitualmente, se utiliza la función RandomSplitter, que nos permite separar los datos en entrenamiento y validación. Como parámetros de entrada, recibe el porcentaje de datos para validación que queremos obtener, y opcionalmente una semilla, por si deseamos repetibilidad de los experimentos. Como en este caso, queremos que el conjunto de entrenamiento sea el conjunto sobremuestreado, y validación, sea la carpeta valid, se ha creado una función que indica 1 si la imagen pertenece a la carpeta valid, y 0, si se encuentra en train. 5.1

- get_y. Parámetro al cual se asocian las etiquetas de los elementos de entrada. Puede recibir una función o bien una lista con las etiquetas asociadas a los elementos del dataset. FastAI toma las etiquetas por defecto del nombre de carpeta, por lo que tendremos que especificarle una función que tome la etiqueta del dataset 5.2.

- item_tfms. Establece modificaciones sobre las imágenes de forma previa a la ejecución del modelo. En este caso, como ya hemos realizado oversampling, no realizaremos ninguna otra modificación sobre las imágenes para no transformar en exceso los datos.

- batch_tfms. Permite realizar transformaciones a nivel de batch. Permite aplicar operaciones como la normalización de las imágenes. Este procedimiento favorece la convergencia del modelo, aunque el cálculo de la media y desviación típica es costoso. En su lugar, utilizaré la normalización que provee ImageNet, donde ya se incluyen los valores estadísticos de media y desviación para los datos en la variable *image_stats provenientes de decenas de millones de imágenes.

Algorithm 5.1 Función para la distinción de entrenamiento y validación

```
1: procedure VAL_SPLITTER(fname)        ▷ Comprueba si la carpeta contenedora es validiación
2:     var pertenece: boolean = False
3:     if Path(fname).parent.name = valid then
4:         pertenece = True
5:     end if
6: return pertenece
7: end procedure
```

Algorithm 5.2 Función para la consulta de etiquetas

```
1: procedure BINARY_LABEL(fname, df_data: Dataframe)
2:     coincidence = df_data[i] where df_data[i] = fname, i ∈ Integer
3:     var label = coincidence.label
4: return label
5: end procedure
```

Una vez definido, podremos instanciar el datablock, configurando uno de los parámetros más críticos del entrenamiento: el tamaño de batch.

Cuando entrenamos una red, estamos ajustando una serie de pesos según recorremos los datos. Éstos, por tanto, pueden actualizarse en diferentes momentos: tras procesar cada imagen, tras procesarlas todas, y antes de volver a empezar, o cada n imágenes. A dicho tamaño n, se le conoce como tamaño de batch: es el tamaño del conjunto de imágenes que han de procesarse para realizar una actualización de pesos del modelo.

Este número es un hiperparámetro clave, ya que tiene relación directa con la convergencia: a mayor tamaño, mayor velocidad de convergencia, pero menor precisión; y el caso inverso ocurre con el caso extremo, si, por ejemplo, actualizamos con cada ejemplar procesado. En la literatura, existen gran cantidad de estudios acerca de este valor, y establecen el valor adecuado en un intervalo $[2, 32]$, compuesto únicamente por las potencias de 2 [41]. Debido a limitaciones de memoria, y la demostración de su optimalidad, emplearé tamaño 32 para los 3 modelos, aunque otros tamaños, como 64 o 128, también son viables.

A partir de ahora, denominaremos época al proceso de recorrer cada uno de los batches del datablock.

Definición del modelo. Transfer learning

Con los datos correctamente definidos, podemos construir el modelo propiamente dicho para el entrenamiento. En Fastai, la construcción de modelos puede hacerse de forma modular, mediante la especificación de las capas que componen la red, ya

que la librería dispone de los módulos ya configurados a excepción de los canales de entrada y salida. A pesar de ello, la creación de un modelo desde cero es muy costoso computacionalmente, ya que se parte el entrenamiento de valores aleatorios. Por este motivo, normalmente se tiende a emplear modelos preentrenados, y adaptar su salida mediante el uso de transfer learning.

El transfer-learning es una métodología que permite utilizar complejas redes convolucionadas preeentrenadas con datasets de gran tamaño que son adaptadas para utilizarse con otros datasets diferentes y obtener buenos resultados, sin necesidad de realizar un entrenamiento desde cero del modelo.

De forma intuitiva, el proceso a seguir es el siguiente: partimos de las capas ocultas ya entrenadas del modelo, y sustituimos su cabecera por una nueva que se adapte a nuestro problema. Por cabecera(head), entendemos la parte final de la red, punto en el cual se elabora la salida final que la red pasa a la función SoftMax para obtener las probabilidades de pertenencia a cada clase de la imagen de entrada. Como la red fue entrenada para otro dataset, es probable que el número de clases de salidas sea diferente, y por eso, debemos de sustituirla por una nueva que se adapte a las necesidades de nuestro problema.

Una vez sustituida el head de la red, debemos de entrenar la nueva cabecera para que sus pesos se adapten a los datos de entrada y ofrezcan buenas métricas. Pero no nos interesa modificar el resto de capas ocultas, ya que fueron entrenadas previamente y tienen valores de W adecuados. Realizaremos un çongelado"de estas capas, para únicamente entrenar la nueva adición.

Posteriormente, para afinar el comportamiento de la red completa, “descongelaremos” la red para hacer un entrenamiento breve a la red completa, durante un número pequeño de épocas. Así, obtemos un modelo final adaptado a nuestro problema.

En esta primera fase, debemos ofrecer una salida binaria, que nos ofrezca una respuesta de 0 (benigno) y 1 (maligno). Para conseguirlo, recurriremos a retirar la capa predefinida con la que viene nuestro modelo, y para todos ellos, emplearemos la misma cabecera, la cual es creada de forma automática por fastai tras especificar el número de salidas.

Capa	Tam. Entrada	Tam. Salida
Average pooling	n x n	32 x 2048
Flatten	32 x 2048	4096
Fully connected	4096	512
Fully connected	512	32 x 2048
Fully connected	32	2
Softmax	-	-

Tabla 5.2: Cabecera para trasfer-learning binario

Dicha cabecera, observable en la figura 5.2, recibe como tamaño de entrada n x n, siendo n el tamaño de la salida de la última capa de la red preentrenada que estemos utilizado. Mediante una oporación de average pooling, podemos convertir dicha entrada a un tamaño de salida concreto, el cual nos permitirá tener a partir de este punto una red totalmente conectada la cual no requiere ser modificada aunque modifiquemos el tamaño de entrada de la red por imágenes de mayor o menor resolución.

La configuración escogida tras el flattening de la imagen, que consiste en vectorizar su contenido, es una tradicional triple capa de redes totalmente conectadas: la estructura tradicional de una red neuronal donde cada neurona se encuentra conectada a las siguientes. Dicha estructura de entradas y salida sigue forma de "embudo", en el que el tamaño de la capa final coincide con el número de salidas, en este caso, dos clases.

Para instanciar el modelo ya configurado, se hace uso de la clase vision_learner, la cual recibe como entreada el modelo, expresado como cadena, un vector con los nombres de las métricas, el datablock de entrenamiento y validación creado anteriormente, y las funciones de callback. Estas últimas son funciones que serán ejecutadas cada vez que se complete una época de nuestro modelo. Como entrada, he decidido añadir un callback de salvado, que guarde el modelo tras cada época finalizada, y un monitor de Early stoping.

Figura 5.2: Demostración gráfica del Early stopping [18]

Resumiendo, early stopping es un mecanismo de parada automática del entrenamiento, el cual, especificando una paciencia, en época, si no se cumple una condición, se detiene el modelo; en este problema, he decidido que el monitor de parada supervise el error de pérdida en validación, para así evitar el sobreaprendizaje de la red: la memorización completa del conjunto de entrenamiento, conllevando el riesgo de pérdida de generalidad. Controlaremos este fenómeno gráficamente mediante la muestra de las pérdidas de entrenamiento y validación.

Entrenamiento

El entrenamiento es la otra parte clave de la creación de modelos de deep learning. Esto se debe a la existencia de usa serie de hiperparámetros que debemos regular para asegurar el correcto entrenamiento del modelo. En este apartado, trataramos los dos parámetros clave: el número de épocas de entrenamiento, y el learning rate.

El número de épocas es equivalente al número de iteraciones realizadas sobre la totalidad de los datos del modelo, Éste debe ser fijado de forma estratégica para evitar que se produzca sobreentrenamiento, donde en lugar de aprender las características generales de los datos y su distribución, pasamos a su memorización completa. Este fenómeno, como ya se ha explicado, provoca pérdida de generalidad del modelo fuera del conjunto de entrenamiento, y no es lo deseable. En lugar de establecer dicho número de forma fija, usaremos, como ya enunciamos, Early stoping. De esta forma, será el propio modelo quien se detenga sin necesidad de que establezcamos un valor manual. Concretamente, emplearemos una paciencia de 3 épocas sobre la pérdida de validación.

En cuanto al learning rate, éste es un valor crítico para definir el tamaño de salto que se produce a la hora de optimizar el modelo. Define el tamaño del salto (step) que se produce a través del hiperplano definido por la función que modela los datos, en búsqueda de su mínimo local (o global). AL igual que el tamaño de batch, define de manera implícita la precisión de la búsqueda del óptimo; a mayor valor, mayor velocidad de convergencia, pero menor certeza de encontrar el óptimo de forma precisa; y a menor valor, mayor tiempo de entrenamiento, pero mayores garantías de encontrar un valor cercano al óptimo de la función.

La decisión de este parámetro normalmente se realiza mediante pruebas experimentales. Sin embargo, al disponer de recursos hardware limitados, no es posible realizar experimentos en paralelo en búsqueda de este valor, cuyo entrenamiento podría conllevar semanas. Por tanto, en lugar de usar un learning rate fijo, se ha tomado la decisión de usar un learning rate adaptativos, mediante la política de entrenamiento de un solo ciclo (Fit one cycle en FastAI) propuesta por Leslie Smith [60].

Con esta técnica, el valor del learning rate se ajusta de forma iterativa, partiendo de un valor pequeño, llegando a un valor máximo, y finalizando en un valor más

pequeño que el inicial.

Este proceso se realiza para conseguir una buena detección de óptimos, sin necesidad de conocer el valor exacto de learning rate, que es una tarea compleja. Este se basa en el compartamiento de la teoría de learning rate cíclicos, donde el learning rate parte de un valor pequeño, llega a un máximo, y luego se decrementa de nuevo hasta un valor mínimo, de forma periódica. Pero la diferencia radica en que fit one cycle realiza un único ciclo de crecimiento y decrecimiento del valor del learning rate.

Otra gran ventaja de este método es que no se requieren gran cantidad de parámetros. Con la implementación realizada en Fastai, solo es necesario especificar el número de épocas a realizar, y un valor máximo de learning rate, clave para especificar el valor pico de la fase ascendente del lr.

Fastai también implementa un método para ayudar en la búsqueda del learning rate pico, llamado lr_find. Este método realiza un estudio de diferentes valores de learning rate y su capacidad minimizadora de la función de pérdida, y devuelve puntos de partida razonables para el inicio del entrenamiento: puntos pico, valle , y pendiente. De todos ellos, nos quedaremos con el punto valle, ya que es el que mejores propiedades de generalización posee.

La decicisión, por tanto, de elegir este método, es la considerable reducción de costo computacional que se produce, así como la mejora de los resultados, tal y como se demuestra en el paper original publicado por L. Smith [60].

5.1.3. Resultados

Una vez realizado el proceso de preparación de los modelos, podemos comenzar el entrenamiento de los mismos. Para comparar los resultados de cada uno de los modelos, se ha recopilado información acerca de las métricas ya mencionadas, las pérdidas, y, sobre todo, el accuracy balanceado, para observar la tasa de acierto de cada uno de los modelos.

Destacar también que, en esta parte del proyecto, la gran cantidad de imágenes y el impacto de la calidad de las mismas provocaban problemas desbordamientos de memoria, que interrumpían el entrenamiento por falta de recursos hardware. Debido a este motivo, únicamente se ha utilizado el subconjunto de imágenes que pertenecen al dataset ISIC, el cual, engloba un aproximado 60 % de los datos totales recopilados, a un tamaño máximo de entrada de 512x512. Reservaremos las imágenes para posible evaluación y equilibrado de los datos.

En los siguientes puntos, desglosaremos el resultado y comportamiento obtenido por cada modelo.

ResNet 50

Comenzamos por los resutados obtenidos con ResNet50. Anteriormente, ya evaluamos un modelo para justificar la necesidad de aplicar oversampling sobre el conjunto de entrenamiento. Podemos apreciar, con respecto a dicho modelo entrenado de manera preliminar, que los nuevos resultados con ResNet 50 son razonablemente mejores 5.3 en lo que a métricas respecta , ya que consiguen mejorar los ya vistos anteriormente cuando se probó el entrenamiento mediante Focal Loss.

	Precision	**Recall**	**F1-score**	**Support**
Benign	0.89	0.90	0.90	7205
Malignant	0.70	0.69	0.69	2460
Accuracy			0.84	9665
Macro avg	0.79	0.79	0.79	9665
Weighted avg	0.84	0.84	0.84	9665

Tabla 5.3: Informe de clasificación Resnet50

Podemos apreciar que el aprendizaje de la clase minoritaria, la maligna, ha mejorado considerablemente respecto a la alternativa de ajustar la función de pérdida unicamente. La mejora de los resultados es apreciable, sobre todo, en el aumento de las métricas precision y recall. Mientras que sin hacer oversampling, obteníamos 68 %(0.68) y 41 % (0.41), respectivamente, ahora obtenemos 70 % (0.70) y 69 % (0.69), siendo por tanto el aumento en 0.28 en el caso del recall. Esto significa que estamos consiguiendo mejores resultados a la hora de distinguir las lesiones malignas, y estas se diagnostican correctamente como tal.

En cuanto a la clase benigna, hemos sacrificado parte de los resultados al realizar el equilibrado, pero sus valores siguen siendo muy buenos: entorno al 89 % de precision, y 90 % de recall.

Durante el entrenamiento del modelo, ha sido almacenada la curva de aprendizaje del modelo, de forma que podemos evaluar de forma bastante visual el desempeó de la red con el progreso de las épocas. Tal y como podemos apreciar en el gráfico 5.3, la curva de aprendizaje muestra una tendencia descendente hasta la época 15, donde el conjunto de validación siempre contiene un valor de la función de pérdida superior a entrenamiento, que se trata del comportamiento habitual.

1

(a) Fase de congelado

(b) Descongelado

Figura 5.3: Curvas del finetuning del modelo binario con Resnet50

A partir de dicha época, podemos apreciar un punto claro de inflexión de la curva, donde los valores de validación se disparan, alejándose de la curva de entrenamiento. Esto es detectado por el callback de Early Stopping, el cual cancela el entrenamiento al detectar un empeoramiento en 3 épocas continuas. En resumen, podemos concluir que el entrenamiento durante la fase de congelado se ha realizado correctamente.

En la segunda fase del proceso de finetuning, la fase de descongelado, los resultados no han sido tan positivos; el modelo sólo ha conseguido mejorar durante las dos primeras épocas, y las tres últimas han supuesto un empeoramiento que ha provocado la activación del early stopping en la quinta época. Si bien los resultados no han supuesto grandes mejoras, este comportamiento denota la complejidad que posee el modelo completo descongelado, y lo adecuadamente afinado que ya se encontraba gracias al preentrenamiento de la red con ImageNet.

A modo de referencia para comparación de los dos siguientes modelos a probar, este modelo ha requerido 4 horas y media para el proceso de entrenamiento, y 1.65 horas para el proceso de oversampling de la clase minoritaria, que ha llegado a ocupar 240GB de disco. Sin embargo, este último proceso sólo ha sido realizado para esta primera prueba, y el resto de modelos utilizarán este conjunto ya configurado.

En conclusión, la técnica de oversampling ha dado sus frutos, y hemos conseguido una mejora significativa en las métricas de la clase minoritaria sin afectar negativamente a la mayoritaria. A continuación, probaremos otros dos modelos para seleccionar el que exhiba mejor comportamiento.

Efficient Net

EfficientNet B5 es la segunda arquitectura elegida para su comparación. Esta arquitectura es el modelo más profundo de los 3 evaluados, por lo que los resultados han de ser considerablemente mejores para que se justifique la mayor complejidad. Aplicando el mismo conjunto de datos con oversampling creado para el entrenamiento de ResNet50, el resultado obtenido puede apreciarse en la tabla 5.4.

	Precision	**Recall**	**F1-score**	**Support**
Benign	0.89	0.93	0.91	7205
Malignant	0.76	0.65	0.70	2460
Accuracy			0.86	9665
Macro avg	0.83	0.79	0.8	9665
Weighted avg	0.85	0.83	0.86	9665

Tabla 5.4: Informe de clasificación de EfficientNet B5

A la vista de los resultados, podemos apreciar que se produce una mejora en las métricas obtenidas en lo que respecta a accuracy, pero las métricas de precision y recall tienden a ofrecer mejores resultados para la clase mayoritaria. La tasa de aciertos para ambas clases (accuracy) ronda el valor de 0.85, que se traduce en iun 85 % de los casos correctamente clasificados. Sin embargo, si analizamos las curvas de aprendizaje 5.4, podemos observar un fenómeno llamativo: la curva no sigue la tendencia habitual a minimizarse, y existen crecidas repentinas de la función, dando lugar a picos de error:

1

(a) Fase de congelado (b) Descongelado

Figura 5.4: Curvas del finetuning del modelo binario con EfficientNet B5

Este fenómeno puede deberse al reajuste de pesos fruto del learning rate varia-

ble, donde un subconjunto de las imágenes correctamente etiquetadas es cambiada de clase a la hora de modificar los valores para adaptarse a otro conjunto distinto no correctamente clasificado hasta el momento. También puede tratarse de un mal ajuste del learning rate, pero esto no es posible al utilzar la política de un ciclo con learning rate adaptativo.

Este fenómeno, de forma aislada, no supone ningún problema destacable sobre el entrenamiento, pero en este caso, observamos cierta periodicidad en el fenómeno que lo convierte en un comportamiento preocupante.

Curiosamente, este fenómeno no se ve reflejado de la misma forma en la fase de descongelado del modelo; a diferencia de Resnet, el modelo si ha mejorado considerablemente en esta fase, aunque la tendencia descendente de la curva se ve eclipsada por el pico obtenido en la época número 3. En este caso, sí que podríamos considerar el pico como un fenómeno aislado sin mayor importancia.

En este modelo, el tiempo de aprendizaje ha sido de aproximademante 7.7 horas, pero ha requerido el reinicio del entrenamiento en 5 ocasiones. Dichos problemas, asociados principalmente al desbordamiento de la memoria de GPU, y la precisión de los tensores, denota que el entrenamiento de este modelo no se ha realizado en condiciones óptimas, siendo uno de los factores posibles la necesidad de una mayor memoria de GPU. Por tanto, los resultados de este modelo han de considerarse con cautela, ya que la inestabilidad del entrenamiento y el aumento considerable de coste computacional lo convierten en una alternativa menos atractiva.

Mobile Net

MobileNet es el tercer modelo a evaluar con el conjunto de entrenamiento con sobremuestreo. Se trata del modelo con menor número de parámetros y profundidad, por lo que no se esperan mejores resultados en lo que respecta a la métrica. Sin embargo, su simplicidad, especialmente dedicada para dispositivos de baja potencia, puede convertirse en una gran ventaja si los resultados no se ven considerablemente penalizados.

Tras realizar el entrenamiento, los resultados obtenidos por el conjunto de validación, nuestro estimador, son los mostrados en la tabla 5.5, donde, como era de esperar, no apreciamos mejores resultados en cuanto a métricas:

	Precision	Recall	F1-score	Support
Benign	0.87	0.86	0.86	7205
Malignant	0.60	0.61	0.60	2460
Accuracy			0.80	9665
Macro avg	0.73	0.74	0.73	9665
Weighted avg	0.80	0.80	0.80	9665

Tabla 5.5: Informe de clasificación de EfficientNet B5

A la vista de los resultados, podemos apreciar como la métrica especialmente penalizada es la columna de precision, donde la diferencia con los modelos anteriores es superior a una décima. Tratándose de la clase minoritaria, es un factor clave que este valor sea lo más alto posible para evitar casos malignos incorrectamente clasificados como benignos. El coste de una enfermedad maligna mal clasificada es superior al de una benigna, por lo que en cuanto a resultados, no se trata de un modelo que destaque. Sin embargo, su reducido tamaño, aproximadamente 4 veces menor que ResNet, lo haría una opción interesante para dispositivos de escaso potencia y con requisitos de bajo consumo.

Si estudiamos las curvas de aprendizaje (figura 5.5), se puede apreciar un comportamiento prácticamente ideal de la función de pérdida de validación con respecto a entrenamiento; en la fase de congelado tanto el estimador de la pérdida en la función de entrenamiento como el de validación progresan en un intervalo de valores muy similares, sin picos bruscos, y con tendencia descendente. Como aspecto destacable, podemos observar cierto aplanamiento del modelo, y sin tendencia clara de sobreaprendizaje. Esto significa que el modelo ha alcanzado su límite de optimización, y debido a su arquitectura poco profunda, no es capaz de obtener valores más cercanos a la función real al disponer de menor dimensionalidad.

En la fase de descongelado, el resultado obtenido no es para nada similar a la tendencia de entrenamiento: obtenemos una aprendizaje muy ruidoso, con gran cantidad de picos y oscilaciones. Además, existe una tendencia clara de aumento de la función de pérdida en entrenamiento, lo cual refleja, en conjunto, que se está produciendo sobreaprendizaje. Por este motivo, en la cuarta época finaliza el entrenamiento, y se conservan los valores de la fase de congelado al haber empeorado los resultados en todas las épocas.

1

(a) Fase de congelado

(b) Descongelado

Figura 5.5: Curvas del finetuning del modelo binario con Mobilenet V2

En resumen, se trata de un modelo fácilmente entrenable, con apenas 2.18 horas de entrenamiento, pero con capacidades de cálculo bastante limitadas.

5.1.4. Modelo seleccionado

Tras el entrenamiento de los 3 modelos, y su correspondiente evaluación mediante el conjunto de validación, es posible tomar la decisión de qué modelo emplear para realizar la cuantización del modelo. Para ello, de la forma tradicional, bastaría con mirar la tabla de resultados de forma conjunta, y elegir aquel que demuestre un mejor resultado promedio en las métricas obtenidas. Pero, en este caso, tendré en cuenta otros factores, como la calidad de las curvas de aprendizaje, el número parámetros como medida de complejidad del modelo (junto a su número de capas), y el tiempo empleado para la inferencia, y de forma consecuente, para el entrenamiento.

Siguiendo el primer criterio, las métricas, el claro ganador en cuanto a tasa de aciertos se refiere es EfficientNet, ya que su valoración para la métrica balanced accuracy es del 81 %. Sin embargo, si atendemos al resto de valores, podemos apreciar que ResNet50 ofrece resultados bastante cercanos, requiriendo casi la mitad de tiempo para alcanzar este resultado con respecto a EfficientNet.

Modelo	Precision	Recall	Accuracy	Bal. Accuracy
EfficientNet B5	0.83	0.79	0.86	0.82
Resnet50	0.79	0.79	0.84	0.80
MobileNet V2	0.74	0.73	0.8	0.74

Tabla 5.6: Valores de las métricas de cada modelo

Además, Resnet ofrece resultados equilibrados en todas sus métricas, mientras EfficientNet aprece destaar especialmente en precision, aunque se igualan en recall.

MobileNet, por su parte, ofrece resultados distantes de los dos modelos anteriores, debido, a como ya se mencionó, su menor profundidad y capacidad expresiva para su ejecución en entornos ligeros. Aunque los resultados no son demasiado lejanos a ResNet, considero que este último es una mejor alternativa para el problema que estamos desarrollando.

Aunque a su vez, Resnet se trata de un peor modelo comparado con EfficientNet a nivel de métricas, el aprendizaje ofrecido por EfficientNet es bastante confuso por la existencia de los picos ya mostrados anteriormente. Debido a la periodicidad de los mismos, el coste computacional adicional, y los errores por falta de recursos ya producidos sin realizar cuantización, considero que no se trata de una buena alternativa, y que Resnet50 es el modelo indicado para este problema.

La complejidad adicional de EfficientNet en cuanto a sus parámetros de anchura, entrada y profundidad, así como su mayor cantidad de parámetros y capas de profundidad variable, lo convierten en un modelo demasiado complejo. Sin medimos el tamaño de cada modelo una vez exportado, obtenemos los siguientes valores:

Figura 5.6: Espacio (en MB) requerido por cada modelo

Por tanto, realizaremos el proceso de cuantizado con Resnet, ya que dará lugar a un modelo más versatil al ser cuantizado, y tiene expresividad de cálculo suficiente para el problema de clasificación multiclase. Además, ofrecerá un mejor resultado en rendimiento de inferencia que EfficientNet en dispositivos móviles.

5.2. Clasificación multiclase

Una vez distinguida si la enfermedad de entrada es benigna o maligna, debemos de indicar de qué tipo de enfermedad se trata. Como ya enunciamos en los objetivos de este proyecto, el proposito del mismo consiste en crear una arquitectura de dos modelos, donde primero, se establece si la enfermedad es benigna o maligna, y posteriormente, se clasifica la posible enfermedad de la que se trata. De esta forma, podemos modularizar el espacio, y entrenar ambos modelos por separado de forma paralela si fuera necesario.

De esta fase de entrenamiento, obtendremos dos modelos distintos, compeltamente aislados entre sí: uno de ellos, dedicado a las enfermedades malignas, y otro, únicamente a las benignas. El proceso a seguir será exactamente el mismo que con clasificación binaria: creación del dataset, entrenamiento, y evaluación del conjunto de validación.

En este caso, no se aplicará oversampling para mostrar las clases en su correspondiente proporción real. Además, debido a los grandes desequilibrios existentes entre clase, sería imposible igualar en número todas las clases a clasificar, y ocuparía un espacio en disco del cual no se dispone para la realización del proyecto.

5.2.1. Modelo de enfermedades malignas

Comenzaremos por las imágenes malignas. En este caso, al estar utilizando finalmente únicamente el conjunto de entrenamiento de ISIC, disponemos de 4 clases diferenciadas:

- Basal cell carcinoma
- Melanoma
- Melanoma metastasis
- Squamous cell carcinoma

Dichas clases poseen un desbalanceo considerable. La clase menos representada se trata de los melanomas metastásicos, que representa solo el 0.22 % del total de datos malignos. Como una clase con dicha proporción no se puede clasificar adecuadamente, aprovecharemos la existencia de la clase melanoma para introducir sus ejemplos como parte de esta clase, dando lugar a una nueva distribución (figura 5.7). Esta fusión es posible debido a que ambas clases representan la misma enfermedad, pero siendo la metastasis una fase más avanzada, y con baja tasa de supervivencia por su diseminación a otros órganos vitales.

Figura 5.7: Distribución de las clases malignas

Al realizar esta acción, aumentamos el número de ejemplares de la clase mayoritaria, pero se trata de una situación preferible sobre la exitencia de una clase minoritaria, ya que ésta penalizará considerablemente las métricas al ser todas ellas métricas equlibradas según la proproción de imágenes de cada clase.

Entrenamiento

Para el entrenamiento del modelo, haré uso de la misma configuración que la clasificación binaria: imágenes 512 x 512, tamaño de batch de 32, pero con la diferencia de no emplear oversamplin por las razones ya explicadas. Si representamos al anterior gráfico de barras 5.7 como proporciones porcentuales, obtenemos una distribución notablemente desequilibrada:

- Melanoma: 53.95 %
- Células basales: 36 %
- Células escamosas: 10.5 %

Dichas clases, a pesar del desbalanceo, poseen una representatividad suficiente como para ser clasificadas en conjunto de forma adecuada. Para la clasificación, esta vez haremos uso de la función de pérdida FocalLoss, gracias a su capacidad de

equilibrar el entrenamiento mediante el uso de sus parámetros $\alpha = 0.25$ y $\gamma = 2$, que son los valores recomendados por su paper original [39].

Tras el entrenamiento, se han obtenido las métricas referenciadas en la tabla 5.7. Podemos apreciar una valoración positiva de las métricas, ya que para Basal Cell carcinoma y los melanomas, precision y recall adquieren valores superiores a 0.8, acercándose a 0.9 en precisión y 0.92 de recall en melanomas. La clase minoritaria de las 3, Squamous cell carcinoma, tiene un recall bajo, cercano a 0.4, mientras que una precisión algo más notable de 0.66.

	Precision	Recall	F1-score	Support
Basal cell	0.81	0.85	0.83	908
Melanoma	0.88	0.92	0.90	1311
Squamous cell	0.66	0.38	0.48	240
Accuracy			0.84	19464
Macro avg	0.79	0.72	0.74	2459
Weighted avg	0.84	0.84	0.84	2459

Tabla 5.7: Informe de clasificación para validación maligna

Aunque se traten de resultados negativos para la clase minoritaria, el conjunto del modelo ofrece resultados adecuados, con una precisión de casi 80 %, y un 72 % de recall, que arrojan un resultado final de 84 % de accuracy. Si observamos las curvas de aprendizaje (figura 5.8), el comportamiento de la función de pérdida denota un correcto aprendizaje en la fase de congelado del modelo.

l

(a) Fase de congelado

(b) Descongelado

Figura 5.8: Curvas de aprendizaje para clases malignas

Podemos observar como, a diferencia del resto de modelos evaluados hasta el momento, la curva de aprendizaje para el conjunto de validación permanece por debajo

de la curva de aprendizaje. Esto no significa que el proceso de entrenamiento sea incorrecto: se debe a un efecto del aumento de datos, y la aplicación de Focal Loss. Al unir ambos mecanismos, estamos provocando que la evaluación del conjunto de entrenamiento sea más compleja que validación, ya que en validación solo se realiza un inferencia de las imágenes, pero entrenamiento es utilizado continuamente para reajustar la función, y fomentar la correcta clasificación de aquellos elementos "dificiles de clasificar". Se trata de un efecto no negativo, y el entrenamiento se ha realizado satisfactoriamente.

No podemos afirmar lo mismo de la fase de descogenlado, ya que tras realizar cinco épocas se ha alcanzando overfitting del modelo, ya que el callback de Early Stopping ha cancelado el entrenamiento antes del límite de épocas establecido (el cual, si recordamos, se trataba de un valor alto para que early stopping decidiera cuando realizar la parada). Por tanto, los resultados obtenidos en la segunda fase son prácticamente descartables, y no han ofrecido valor ninguno al modelo final.

En resumen, el modelo obtenido cumple adecuadamente los requisitos del problema. Sin embargo, como forma de evaluar si eran posibles mejores resultados, se realizó un segundo entrenamiento, aplicando pesos a la función de pérdida para penalizar en mayor medida incorrectas clasificaciones de la clase minoritaria. Basta con aplicar la siguiente operación, donde, considerando el valor c como el número de ejemplares de la clase C:

$$p_C = \left(\frac{c}{\sum_{i=0}^{num_clases} c}\right)^{-1}$$

De esta forma, haremos que cada ejemplar tenga un peso acorde a su proproción en el dataset. Si repetimos el experimento siguiendo este razonamiento, obtenemos los siguientes resultados:

	Precision	Recall	F1-score	Support
Basal cell	0.66	0.66	0.66	908
Melanoma	0.83	0.76	0.79	1311
Squamous cell	0.30	0.42	0.35	240
Accuracy			0.69	19464
Macro avg	0.59	0.62	0.60	2459
Weighted avg	0.71	0.69	0.70	2459

Tabla 5.8: Informe de clasificación para validación maligna (Pesos)

Donde, podemos observar que todas las métricas obtenidas son considerablemente inferiores a las de la versión inicial (5.7). Esto se debe a que, al insertar pesos a cada clase a clasificar, es altamente probable que se haya cancelado el efecto regularizador ofrecido por FocalLoss durante el entrenamiento.

Por tanto, podemos concluir de manera firme y justificada que el modelo entrenado anteriormente, sin ningún tipo de penalización adicional, y haciendo uso de Focal Loss, es el modelo adecuado para este subproblema.

5.2.2. Modelo de enfermedades benignas

Una vez entrenado el modelo correspondiente a las clases malignas, es el turno de construir el modelo de las clases benignas. Recordando el estudio realizado en el problema de clasificación binario, las clases benignas eran el subconjunto de datos dominante en el dataset. Esto se debe, principalmente, a que las lesiones cutáneas son frecuentemente lesiones no cancerosas. Disponemos de un conjunto de datos mucho más amplio, englobando 17 clases distintas. Como es de esperar, la condición más frecuente de este conjunto son los lunares comunes, denominados nevus en dermatología.

El desbalanceo que se produce es un caso extremo, en cual la clase mayoritaria, el lunar común, engloba el 81 % del total, y el 19 % restante se reparte entre las 16 clases que restan (figura 5.9). Esto convierte al subproblema en un reto considerable de resolver con un modelo del tamaño de Resnet50, por lo que se tratará de hallar los mejores resultados pero teniendo en cuenta el sacrificio que supone emplear modelos más ligeros que sean cuantizables y ejecutables en Android.

Dado el desbalanceo, se probarán dos alternativas distintas: la aplicación de la métrica Focal Loss sin ajuste de pesos, y la creación de un modelo basado en pesos inversamente proporcionales al número de ejemplares en dicha clase junto a CrossEntropy.

Para simplificar la notación empleada en el etiquetado de las enfermedades, se usarán las siguientes abreviaturas en las tablas de resultados:

- Nevus: NV
- Seborreic keratosis: SEK
- Actinic keratosis: AK
- Pigmented benign keratosis: PBK
- Solar lentigo: SL
- Dermatofibroma: DER
- Vascular lesion: VL
- Lichenoid keratosis: LK
- Acrochordon: ACR
- Lentigo NOS: LN
- Atypical melanocytic proliferation: AMP
- Aimp
- Wart
- Angioma: ANG
- Lentigo simplex: LS
- Neurofibroma: NF
- Scar

Figura 5.9: Conteo de clases benignas antes de agrupar

Entrenamiento

Conocemos que el desequilibrio del conjunto de entrenamiento es considerable. Pero, antes de probar otras soluciones más sofisticadas, es razonable considerar aplicar la misma técnica que la aplicada en clasificación binaria: el uso de la una métrica que tolere adecuadamente el desbalanceo, como FocalLoss.

Aplicando una configuración similar a la usada para el entrenamiento de las clases malignas, el resultado obtenido no esperanzador, y deja bastante que desear (tabla 5.9). La clase de lunares comunes es la única que cuenta con métricas excelentes, ya que el resto de clases han sido clasificadas con valores de métrica bastante bajos. Si observamos los resultados, los siguientes mejores casos provienen de los dermatofibromas, la queratosis benigna pigmenteada, las lesiones vasculares, y la queratosis liquenoide. Estas obtienen unos valores de precision de 0.75, 0.71,0.68,0.67, respectivamente. Pero en el caso de recall, los dermatofibromas son ampliamente castigados, por lo que aunque la alta precisión nos aporta que el modelo es capaz de rechazar las clases que no lo son de manera aceptable, el recall bajo nos muestra que también los casos positivos están siendo identificados como negativos en la mayoría de casos.

Esto ocurre en la mayoría de clases, aunque los peores resultados los vemos reflejados en las 6 últimas filas, donde 5 de las clases han sido completeamente ignoradas por el modelo, y establecidas a 0: la cicatrices, los neurofibromas, los lentigo simples, las verrugas y los AIMP (atypical intraepidermal melanocytic proliferation).

	Precision	Recall	F1-score	Support
NV	0.92	0.98	0.95	5864
SK	0.50	0.35	0.41	345
AK	0.53	0.58	0.56	265
PBK	0.71	0.74	0.72	222
SL	0.28	0.12	0.17	99
DER	0.75	0.18	0.29	83
VL	0.68	0.55	0.61	71
LK	0.67	0.07	0.12	61
ACR	0.46	0.24	0.32	54
LN	0.16	0.11	0.13	44
AMP	0.50	0.07	0.12	29
AIMP	0.00	0.00	0.00	24
WART	0.00	0.00	0.00	23
ANG	0.29	0.33	0.31	6
LS	0.00	0.00	0.00	5
NEU	0.00	0.00	0.00	5
SCAR	0.00	0.00	0.00	5
Accuracy			0.87	7205
Macro avg	0.38	0.25	0.28	7205
Weighted avg	0.84	0.87	0.85	7205

Tabla 5.9: Clasificación para validación benigna (Ordenadas de mayor a menor rep., sin pesos)

Estas clases, al encontrarse en absoluta minoría, no aportan suficientes caracteristicas al modelo para aprender los filtros adecuados, y por tanto, quedan descartadas por el mismo. Esto nos lleva a tomar la decisión de descartar estas clases, o bien, tratar de equilibrarlas complementando con el resto de datasets.

No nos debemos dejar llevar por el valor del accuracy: este es del 87 %, debido a que prácticamente la completitud de la tasa de aciertos se ve aportada por la clase mayoritaria. Si atendemos a su valor medio, es del únicamente 28 %, que se trata de una cifra más representativa pero mucho más pesimista.

Para no proceder con la eliminación de clases minoritarias, trataremos de dotar al modelo de penalizaciones para las clases menos reprentadas; es decir, de forma explícita, estableceremos una serie de pesos que otorguen más importancia a clasificar ejemplares de clases minoritarias sobre las mayoritarias, calculando los pesos de la mismo forma en la que lo hicimos para el caso de las enfermedades malignas:

$$p_C = \left(\frac{c}{\sum_{i=0}^{num_clases} c}\right)^{-1}$$

De esta forma, haremos que cada ejemplar tenga un peso acorde a su proprocíón en el dataset. La diferencia con el caso empleado en enfermeades malignas, es que emplearemos como función de pérdida CrossEntropy en lugar de FocalLoss, ya que pudimos apreciar en el caso anterior que al insertarle pesos, pierde el efecto regularizador visto en el caso sin pesos, y obtenemos peores resultados.

Figura 5.10: Conteo de clases benignas tras la agrupación

De esta forma, si repetimos el experimento siguiendo este razonamiento, obtenemos los resultados de la tabla 5.10.

Los resultados han mejorado, escalando hasta el 48 % de accuracy balanceado. Con la aplicación de pesos al modelo, podemos apreciar que la distribución de las métricas es mucho más equitativa que antes, ya que aunque los valores han bajado para ciertas clases concretas, de forma global, la varianza de las métricas es menor. Sobre todo, las mejoras de rendimiento vienen vinculadas a una mejora de recall, aunque se ha producido un empeoramiento de la precisión; es decir, ahora el modelo clasifica correctamente más ejemplares pertenecientes a su correspondiente categoría, pero también diagnostica erróneamente muchos falsos positivos.

Sin embargo, este enfoque nos otorga resultados globales de mayor calidad, por lo que será el modelo seleccionado para realizar la cuantización. Esto resultados podrían ser mejorables con el empleo de técnicas de aumento de datos más sofisticadas que el aumento de resolución, y el uso de datos adicionales. Sin embargo, las limitaciones de memoria de las que se dispone en el ordenador empleado para el entrenamiento, y las limitaciones de simplicidad a la que está sometido el mo-

	Precision	Recall	F1-score	Support
NV	0.99	0.77	0.86	5864
DER	0.33	0.60	0.42	345
PBK	0.53	0.59	0.56	265
VL	0.37	0.88	0.52	222
LK	0.18	0.46	0.26	99
AK	0.31	0.63	0.41	83
SK	0.58	0.83	0.69	71
AMP	0.20	0.39	0.26	61
ACR	0.23	0.67	0.34	54
ANG	0.12	0.43	0.19	44
SL	0.23	0.48	0.31	29
LN	0.01	0.04	0.01	24
AIMP	0.40	0.26	0.32	23
WART	0.18	0.67	0.29	6
LS	0.06	0.20	0.09	5
NEU	0.00	0.00	0.00	5
SCAR	0.00	0.00	0.00	5
Accuracy			0.74	7205
Macro avg	0.28	0.46	0.33	7205
Weighted avg	0.87	0.74	0.78	7205

Tabla 5.10: Informe de clasificación para validación benigna (Ordenadas de mayor a menor rep.)

delo para ser cuantizado, nos llevan a la decisión de permanecer con este segundo modelo como hipótesis final.

6 Cuantización

La cuantización, como ya definimos en el apartado 2.2, se trata de un proceso por el cual tomamos un modelo entrenado para arquitecturas de escritorio de la forma habitual, y realizamos una serie de operaciones de simplificación o transformaciones del formato numérico para obtener un modelo cuyos requisitos de espacio y tiempos de inferencia se ven reducidos considerablemente.

Este procedimiento es el núcleo del proyecto, y el objetivo real de estudio de modelos para tecnologías móviles: experimentar cuál es la diferencia en tiempo y uso de memoria de la arquitectura tradicional sobre la optimizada.

Para realizar este proceso, emplearemos los mecanismos de Pytorch[49] dedicados a la transformación e implementación de modelos convolucionales en Android: Pytorch mobile [52]. En este punto, analizaremos las optimizaciones realizadas por este framework, y las ventajas y desventajas de dicho procedimiento.

6.1. Optimizaciones de Pytorch mobile

Pytorch mobile ofrece una serie de optimizaciones predeterminadas bastante eficientes en la optimización de modelos para arquitecturas ARM. Para comprender en detalle su funcionamiento, profundizaremos en el formato de guardado utilizado, y las optimizaciones llevadas a cabo para conseguir las ganancias en rendimiento especificadas en [52].

6.1.1. Capacidades del framework

Pytorch [49], a través de sus dependecias mobile, tiene la capacidad de transformar en unos sencillos pasos un modelo entrenado de la forma habitual a un modelo simplificado optimizado para dispositivos móviles. Incluye varios métodos relacionados con la exportación que permiten optimizar o no la salida del modelo, adecuándolo a nuestras necesidades. En su página oficial, podemos encontrar un grafo donde se describe este proceso [52]:

Figura 6.1: Optimización de modelos con Pytorch Mobile [52]

Todo este procedimiento se realiza sobre un lenguaje intermedio de alto rendimiento, perteneciente al núcleo de Pytorch, llamado Torchscript [14]. Este lenguaje permite crear modelos escalables y optimizables sin necesidad de dependencias Python en el sistema destino. Está pensado, principalmente, para su aplicación en entornos de producción, los cuales estén basados en otros lenguajes, como C o similares. La conversión del formato habitual de Pytorch, y el establecido por Torchscript es prácticamente inmediato, ya que la conversión se realiza de forma interna al aplicar el comando *torch.jit.script(model)*, siendo model el modelo que queremos transformar.

Como inconvenientes, no podremos utilizar herramientas de explicabilidad de modelos como GradCam [58], debido a la pérdida de toda la información no esencial sobre los gradientes en el proceso de exportación a producción.

La portabilidad al ser independizado de Python es el principal motivo de su uso

para realizar la transformación del modelo a dispositivos móviles, y puede ser ejecutado tanto en PC como en smartphones, facilitando así el proceso de comparativa entre ambos.

Si bien el lenguaje soporta Android y iOS, este último fue soportado oficialmente tras el comienzo de la realización de este TFG. Debido a ello, este proyecto fue concebido inicialmente para ser probado en dispositivos Android, ya que el hardware de estos dispositivos es mñas accesible, y dispone de un mejor soporte de librerías y entornos de desarrollo. iOS, al tratarse de un SO limitado a dispositivos Apple, y difícilmente emulable si no se dispone de dispositivos de la marca, requiere unas condiciones adicionales que no se pueden cumplir por limitaciones económicas, pues no disponemos de dispositivos de la marca. Por este motivo, este proyecto se limita únicamente al desarrollo sobre terminales Android, más accesibles y frecuentes en el mercado.

6.1.2. Optimizaciones disponibles

Pytorch mobile dispone para Android de una lista de 5 optimizaciones posibles a realizar sobre el modelo. Todas ellas, seleccionables de forma independiente, o aplicadas en conjunto por defecto, cubren los siguientes aspectos:

- Fusión de las capa de convolución y normalización de batch: realiza una integración de los valores de normalización dentro de la capa convolutiva, ya que no será necesario reajustar esos valores para inferencia-
- Inserción y plegado de operaciones: utiliza la librería XNNPack, que se trata de un sistema de operaciones en coma flotante especialmente optimizado para ARM, que permite variar la precisión de los valores flotantes según el modelo. Fusiona las operaciones lineales y de convolución, de forma que se realicen en un menor tiempo.
- Fusión de la función ReLu con el conjunto de operaciones empaquetado creado con XNNPack en el paso anterior.
- Eliminación del dropout, ya que en tiempo de inferencia, se busca el mejor resultado si necesidad de reentreno.
- Optimización del grafo interno del modelo correspondiente a la convolución, para hacer que formen parte de un solo bloque raiz y mejore el tiempo de inferencia.

En total, la web [52] estima una ganancia aproximada del 60 % en tiempo de inferencia de forma teórica. Sin embargo, este valor puede oscilar en función de los datos que se estén evaluando, la arquitectura que deseamos cuantizar, y las capacidades de cálculo del dispositivo objetivo. A continuación, estudiaremos los resultados ofrecidos, y si merece la pena la aplicación de este proceso.

6.2. Creación de los modelos cuantizados

La transformación de los modelos entrenados a su versión cuantizada puede realizarse de forma inmediata, ya que en los pasos anteriores, el modelo entrenado sobre cada uno de los casos (clasificación binaria, y multiclase para benigno y maligno) fue almacenado en un fichero de formato .pt (extensión de modelo Pytorch).

Al tratarse de una conversión post-entrenamiento, no se requiere ningún ajuste de parámetros previo durante el entrenamiento, lo cual significa que el mecanismo puede aplicarse sobre los modelos entrenados de la forma habitual, sin necesidad de emplear métodos específicos como el entrenamiento apto para cuantizado (QAT, o Quantization Aware Training [38]), el cual es más complejo de ajustar, pero puede ofrecer algunas ventajas marginales en cuanto a precisión.

Para cuantizar el modelo tras su entrenamiento, basta con realizar los siguientes pasos:

1. Cargar el modelo entrenado previamente. Como en nuestro caso, hemos realizado el salvado del modelo y procedemos a realizar la cuantización en el mismo en el mismo script, podemos directamente emplear el objeto Learner que almacena el modelo completo. En caso de que se quisiese leer de disco duro, bastaría con usar la función model.load().

2. Activar el modo de evaluación del modelo. De esta forma, bloqueamos los parámetros libres de las capas de normalización de batches, y la aplicación de dropout.

3. En este punto, podemos optar por dos opciones:

   a) Exportar el modelo a Android sin ninguna optimización estructural, más allá de la adaptación de tipos de flotantes (función *save_for_lite_interpreter*)

   b) Exportar el modelo aplicando todas las optimizaciones anteriores mencionadas para obtener el modelo más eficiente posible (función *export_for_mobile*)

4. (Opcional) : exportar el modelo de escritorio en formato torch script. Se trata de un lenguaje de representación universal del modelo, que puede ser empleado con Python nativo, y ser transferido a otros frameworks de trabajo habituales.

Para realizar la comparativa de forma lo más exhaustiva posible, tomaremos el modelo de Android, y lo evaluaremos con el conjunto completo de imágenes de test, de forma que la valoración obtenida sea completamente representativa.

El pseucódigo resultante es batante sencillo, y su explicación queda prácticamente autocontenida gracias al uso de nombres de función represetativos:

Algorithm 6.1 Proceso de cuantizado de modelos a Android

```
procedure EXPORTAR_MODELO(learner: Learner Model Object, savename : String)
    model = learner.model.to('cpu') : Model  State model.eval()
    scripted_module = torch.jit.script(nn.Sequential(model, Softmax(dim=n)))
                                                    ▷ Exportación del modelo original
    scripted_module.save(savename + "Full.pt")
                                                    ▷ Exportación del modelo optimizado
    optimized_scripted_module = optimize_for_mobile(scripted_module)
    optimized_scripted_module._save_for_lite_interpreter(savenameFull+ "_an-
droidOptimized.pt")
end procedure
```

Podemos observar que el proceso es trivial, a excepción del último punto, donde se realiza la definición de un objeto de tipo sencuencial. Esto se debe que, cuando exportamos el modelo a Android, existen dos posibles alternativas para interpretar la capa de salida del mismo:

1. Interpretando el tensor de salida de la función Softmax por columnas, de forma que la clase predicha para cada imagen será el máximo valor softmax encontrado en la columna en cuestión. En Python, se le conoce como máximo en dimensión 0, al tratarse del máximo de todas las filas.

2. Interprentando el tensor de salida por filas, donde una imagen de entrenada recibe como etiqueta de salida el índice de la posición de fila cuyo valor es mayor. En este caso, se considera como dimensión 1, ya que se trata del valor máximo encontrado entre las columnas.

Por tanto, antes de exportar el modelo, será necesario especificar la dimensión de lectura. En versiones anteriores, esta dimensión era elegida de forma implícita por la función, pero ahora, debemos especificarla explícitamente para tener un correcto funcionamiento. Como en nuestro caso, solo tenemos una dimensión de etiquetas por modelos, debemos seleccionar el valor máximo por columnas, y valor a especificar es dim=1. Basta con crear un objeto de tipo Softmax, y especificar como parámetro *dim*, el valor dim = 1.

Para concatenar el modelo original, y la nueva capa específica para la función Softmax, es necesario añadir ambos pasos a un pipeline, usando para la clase Sequential de Pytorch, que no es más que la creación de un modelo por capas.

Por último, para el proceso de comparación de modelos, realizaremos dos tareas:

1. Evaluación del conjunto de test. Se realiza la inferencia completa del conjunto de test para estudiar la pérdida de rendimiento del modelo cuantizado frente al original haciendo uso un estimador insesgado, debido a la reserva de las imágenes hasta este momento.

2. Comparativa de tiempos de inferencia. Realizaremos una comparativa de ejecución de ambos modelos en Android para evaluar la ganancia en tiempo tras aplicar cuantización, y si esta compensa las pérdidas de prestaciones. Debido a las limitaciones de la arquitectura ARM, evaluaremos el tiempo sobre una muestra del total, debido a la generación de calor y el desgaste ocasionado por la evaluación del test, que al tratarse del 40 % del cómputo global de imágenes, incluye 21478 fotografías.

6.3. Comparativa de los modelos original y cuantizado

En este apartado, una vez comprendido el proceso de exportación y cuantización, examinaremos el rendimiento de cada uno de los 3 modelos creados para el proyecto. Para ello, evaluaremos el conjunto de test, intacto hasta esta fase, para conocer el rendimiento real de los modelos originales, y los compararemos con cada uno de los modelos cuantizados.

6.3.1. Cuantización del modelo binario

El modelo de clasificación binaria, que decide si la enfermedad se trata de un caso benigno o maligno, es la capa más importante de la arquitectura de dos niveles implementada. Su rendimiento es crítico dentro del funcionamiento del modelo, debido a que la confusión entre subtipos de enfermedades tiene un coste relativamente pequeño, pero un error entre una enfermedad cancerígena o benigna tiene un elevado coste para el paciente en consecuencias. Por tanto, es clave evaluar el modelo con el conjunto de test para conocer la bonanza del mismo, y estudiar si las ganancias en espacio y tiempo de inferencia justifican la cuantización.

Evaluaremos el conjunto de test bajo las mismas condiciones, sin ningún tipo de preprocesado, más allá de la normalización de las imágenes y el reescalado a 512 x 512. Es muy importante destacar que estos valores son los mismos que los empleados en la fase de entrenamiento, ya que recalcular la media y desviación típica podría crear mejores resultados pero no ser una prueba representativa, ya que al realizar una inferencia de un diagnóstico, esta normalmente se realiza con una única imagen, y no habría datos suficientes para hallar la media y desviación típica. Debemos trabajar siempre con los parámetros de entrenamiento para que la distribución sea la misma, verificando así las condiciones ya vista acerca de disponer de un estimador no sesgado con test.

A continuación, comparamos la inferencia de ambos modelos y la ganancia en tiempo de inferencia entre ellos.

Métricas de inferencia en test

En primer lugar, comenzamos por evaluar la pérdida de rendimiento del modelo cuantizado. Para ello, ejecutamos el proceso de inferencia con ambos modelos, y obtenemos la tabla de resultados para ambos. Los resultados de ambos modelos, original y cuantizado, podemos apreciarlos en las tablas 6.1 y 6.2 respectivamente.

	Precision	Recall	F1-score	Support
Benign	0.90	0.89	0.89	16010
Malignant	0.68	0.70	0.69	5468
		accuracy	0.84	21478
Macro avg	0.79	0.80	0.79	21478
Weighted avg	0.84	0.84	0.84	21478

Tabla 6.1: Resultados de inferencia en test para el modelo original

	Precision	Recall	F1-score	Support
Benign	0.90	0.89	0.89	16010
Malignant	0.68	0.70	0.69	5468
		accuracy	0.84	21478
Macro avg	0.79	0.79	0.79	21478
Weighted avg	0.84	0.84	0.84	21478

Tabla 6.2: Resultados de inferencia en test para el modelo cuantizado

Podemos apreciar que el resultado obtenido en el modelo original es muy similar a los obtenidos con la partición de validación, por lo que el modelo ha funcionado correctamente con datos los cuales no ha visto hasta el momento. Concretamente, en media, el modelo original en validación nos ofrecía el mismo resultado: 0.79 de precisión, 0.79 de recall, 0.79 de F1 Score y 0.84 de accuracy (5.3). Si comparamos en detalle, hemos perdido recall y precisión en la clase maligna a favor de un ligero aumento de dichas métricas en la clase benigna, arrojando el mismo resultado medio.

Si ahora estudiamos el caso optimizado y cuantizado para Android, observamos que sorprendentemente, no se han perdido prestaciones: las métricas son prácticamente idénticas a las obtenidas en el modelo original sin optimizar, a diferencia del recall medio, una centésima menor, lo cual es posible que se deba a algún ejemplo cuya valoración se ha visto erróneamente clasificada, pero cuyo impacto global ha sido. Podríamos afirmar que para este modelo, el impacto de optimización no ha afectado al rendimiento del modelo, el cual ha conservado las mismas prestaciones, ya que la diferencia en recall es prácticamente despreciable.

Esto ha sido posible gracias a las capacidades del framework desarrollado por

Pytorch, las cuales han funcionado adecuadamente, al haber elegido un modelo totalmente compatible con el proceso de cuantización y optimización [53].

En conclusión, para el modelo de clasificación binaria, obtenemos un modelo optimizado sin pérdida de rendimiento.

Tiempo de inferencia

Tras obtener unos buenos resultados en las métricas del modelo, podemos evaluar el tiempo medio de inferencia del modelo cuantizado frente al original en Android. Este factor es clave, ya que si el modelo cuantizado no ofrece resultados más eficientes en tiempo, el proceso de cuantizado no supone ninguna ventaja sobre el modelo original, sobre todo en este modelo, el cual ha demostrado no perder prestaciones en cuanto a métricas.

Los resultados se han tomado teniendo en cuenta únicamente la inferencia de 200 imágenes, debido a que la evaluación de las 21478 imágenes en un dispositivo móvil produce un sobrecalentamiento y desgaste considerable sobre los componentes. El terminal empleado hace uso de un Snapdragon 855, y 8GB de RAM, siendo estos componentes del año 2019, y arroja los resultados de la tabla 6.3.

Modelo	Tiempo total (ms)	Tiempo medio (ms)
Original	114342	571,71
Cuantizado	72352	361,76

Tabla 6.3: Tiempos de inferencia para caso binario en Android

Tras la evaluación del modelo mediante 200 imágenes, podemos apreciar una ganancia significativa en tiempo, superior a 210ms por imagen. Si calculamos la ganancia en velocidad del modelo cuantizado sobre el original, obtenemos:

$$Ganancia = \frac{t\ original}{t\ cuantizado} = \frac{571,71}{361,76} = 1.58$$

Es decir, el modelo optimizado nos aporta una ganancia de 1.58, lo cual supone aproximadamente un 40 % menos del tiempo de inferencia. Este resultado evidencia que el modelo cuantizado ofrece unas mejores prestaciones que el original. Teniendo en cuenta que el modelo cuantizado no pierde bondad de métricas al ser evaluado con el conjunto de test frente al original, queda demostrado que el modelo cuantizado supone una mejora considerable y debe ser el modelo a utilizar en un dispositivo móvil. La penalización obtenida es nula, y las ventajas, cuantiosas.

6.3.2. Cuantización del modelo multiclase maligno

El modelo de clasificación de enfermedades malignas se del primero de los casos de clasificación multiclase. A diferencia del anterior, este modelo aporta una salida más compleja, al deber distinguir entre melanomas, cáncer de células basales y cáncer de células escamosas. Esto nos deja un con vector de salida de 3 componentes, una por cada clase, que el modelo deberá aportar tanto en el modelo original como el cuantizado.

De la misma manera que con el caso de clasificación binaria, compararemos los dos modelos mediante la evaluación del conjunto de test, teniendo en cuenta que las imágenes ha de ser preprocesadas con los mismos parámetros y configuraciones del conjunto de test, y no recalcular ninguno de los valores. Este preprocesado, al igual que en entrenamiento, incluye:

1. Renombrado de etiquetas. Al igual que en el modelo de entrenamiento decidimos fusionar las clases "melanoma" y "melanoma metastasis", debemos de aplicar la misma transformación en test. Si no lo hiciésemos, el modelo funcionaría adecuadamente, pero no podríamos evaluar correctamente las métricas que nos permiten ver si el rendimiento es el esperado.

2. Normalización y resize: se utilizan la media y desviación típica de ImageNet[16], empleadas en entrenamiento, y se aplica un resize de 512 x 512 px siguiendo la misma distribución.

Una vez realizado este preprocesado, podemos calcular la inferencia de los dos modelos.

Métricas de inferencia en test

Para evaluar la calidad del modelo cuantizado, se tomarán tanto el modelo original como el cuantizado, ambos exportados durante el aprendizaje realizado en el capitulo 5, y obtendremos la tabla resumen de métricas para realizar la comparativa. Los resultados podemos apreciarlos en las tablas 6.4 (original) y 6.5(cuantizado).

	Precision	Recall	F1-score	Support
Basal Cell C.	0.80	0.84	0.82	1969
Melanoma	0.87	0.92	0.89	2950
Squamous Cell C.	0.65	0.37	0.47	549
Accuracy			0.83	5468
Macro avg	0.78	0.71	0.73	5468
Weighted avg	0.83	0.83	0.83	5468

Tabla 6.4: Resultados de inferencia en test para el modelo original

	Precision	Recall	F1-score	Support
Basal Cell C.	0.80	0.84	0.82	1969
Melanoma	0.87	0.92	0.89	2950
Squamous Cell C.	0.65	0.37	0.47	549
Accuracy			0.83	5468
Macro avg	0.78	0.71	0.73	5468
Weighted avg	0.83	0.83	0.83	5468

Tabla 6.5: Resultados de inferencia en test para el modelo cuantizado

En este subproblema, encontramos que ambos modelos arrojan exactamente el mismo resultado, por lo que el proceso de optimización no ha conllevado ninguna pérdida para el rendimiento del modelo, lo cual es un caso totalmente ideal. Para ver su rendimiento real, podemos comparar el modelo con el resultado arrojado por el conjunto de validación: 0.79 de precisión, 0.72 de recall, y 0.74 de F1 Score (tabla 5.7). Podemos apreciar que, por tanto, el conjunto de validación fue un correcto estimador de las funciones de pérdida y métricas fuera de la muestra.

Por tanto, el modelo cuantizado, y a su vez, el original, ofrecen un muy buen resultado, acorde a lo estimado durante el entrenamiento, y sin pérdida de precisión de clasificación en el caso del modelo cuantizado.

Tiempo de inferencia

Tras observar que el modelo cuantizado no supone pérdidas frente el modelo original, un caso ideal, nos disponemos a evaluar el tiempo de inferencia por imagen, de la misma manera que se realizó con el modelo binario.

Se usará el mismo dispositivo, empleando un subconjunto para la inferencia de 200 imágenes. Los resultados se pueden apreciar en la tabla 6.6

Modelo	Tiempo total (ms)	Tiempo medio (ms)
Original	215693	1078,465
Cuantizado	136912	684,56

Tabla 6.6: Tiempos de inferencia para caso multiclase maligno en Android

A la vista de los resultados, podemos apreciar una ganancia significativa en tiempo, de alrededor de 394ms por imagen. A priori, parece que el ahorro en tiempo es mayor que en el caso binario, pero si calculamos la ganancia en velocidad, obtenemos:

$$Ganancia = \frac{t\ original}{t\ cuantizado} = \frac{1078,465}{684,56} = 1.575$$

Es decir, el modelo optimizado nos aporta una ganancia de 1.575, lo cual es proporción es prácticamente la misma ganancia que la conseguida con la cuantización en caso binario. De nuevo, supone aproximadamente un 37 % menos de tiempo de inferencia sobre el modelo original. En conclusión, la arquitectura final vuelve a ser la versión cuantizada, ya que nos permite mejorar considerablemente el rendimiento sin pérdida de prestaciones.

6.3.3. Cuantización del modelo multiclase benigno

Tras el estudio de los modelos para el caso de clasificación binaria y el caso multiclase maligno, es el turno del modelo para la detección de enfermedades benignas. Este modelo es el más complejo de los 3, ya que requiere dar como salida un grado de granularidad elevado en cuanto a diagnósticos, debiendo distinguir entre 18 enfermedades distintas.

Los resultados de esta modelo no fueron tan prometedores debido a la alta similitud de las clases, y el uso de un modelo poco profundo para acometer la clasificación. Por tanto, los resultados del modelo cuantizado podrían verse más afectados que en los dos casos anteriores.

Al igual que en la preparación de datos del modelo maligno, debemos realizar el mismo preprocesado que con las clases de entrenada de entrenamiento:

1. Renombrado de etiquetas. Realizar el ajuste de etiquetas a valores numéricos para una mayor facilidad de cálculo de las métricas de evaluación.

2. Normalización y reescalado: utilización de la media y desviación típica de ImageNet[16], y reescalado a 512 x 512 px.

Tras dichos ajustes, podemos proceder a la medición de las métricas de test y el tiempo de inferencia en el dispositivo Android.

Métricas de inferencia en test

De la misma forma que los casos anteriores, procedemos a la evaluación del conjunto de test, filtrando únicamente aquellas imágenes con lesiones de naturaleza benigna. Los resultados de los modelos original y cuantizado quedan reflejados en las tablas 6.7 y 6.8, respectivamente:

	Precision	Recall	F1-score	Support
NV	0.99	0.78	0.87	13067
DER	0.31	0.61	0.41	770
PBK	0.53	0.59	0.56	547
VL	0.39	0.87	0.54	536
LK	0.20	0.47	0.28	225
AK	0.28	0.61	0.38	168
SK	0.59	0.83	0.69	139
AMP	0.20	0.43	0.27	125
ACR	0.29	0.72	0.42	120
ANG	0.11	0.36	0.17	96
SL	0.04	0.07	0.05	56
LN	0.02	0.08	0.03	48
AIMP	0.27	0.23	0.25	48
WART	0.35	0.50	0.41	28
LS	0.19	0.53	0.28	17
NEU	0.00	0.00	0.00	10
SCAR	0.00	0.00	0.00	10
Accuracy			0.75	16010
Macro avg	0.28	0.45	0.33	16010
Weighted avg	0.87	0.75	0.79	16010

Tabla 6.7: Resultado obtenido al inferir el conjunto de test (sin cuantizado)

	Precision	Recall	F1-score	Support
NV	0.99	0.78	0.87	13067
DER	0.31	0.60	0.41	770
PBK	0.53	0.59	0.56	547
VL	0.39	0.87	0.54	536
LK	0.21	0.48	0.29	225
AK	0.27	0.61	0.38	168
SK	0.59	0.83	0.69	139
AMP	0.21	0.45	0.28	125
ACR	0.30	0.72	0.42	120
ANG	0.11	0.36	0.17	96
SL	0.04	0.07	0.05	56
LN	0.02	0.08	0.03	48
AIMP	0.28	0.23	0.25	48
WART	0.35	0.50	0.41	28
LS	0.19	0.53	0.28	17
NEU	0.00	0.00	0.00	10
SCAR	0.00	0.00	0.00	10
Accuracy			0.75	16010
Macro avg	0.28	0.45	0.33	16010
Weighted avg	0.87	0.75	0.79	16010

Tabla 6.8: Resultado obtenido al inferir el conjunto de test (cuantizado)

Los resultados son de nuevo positivos, ya que en lo que corresponde a los valores medios, no existe pérdida destacable. Si bajamos de nivel, y comparamos clase a clase, podemos ver que sí que existe alguna diferencia, como, una precisión una centésima menor en el diagnóstico de la queratosis actínica (abreviada como AK). Sin embargo, dicha pérdida de precisión es compensada por otros valores que, sorprendentemente, adquieren un mejor resultado en el modelo cuantizado que el modelo original, aunque menor a una centésima.

A grandes rasgos, y en promedio, ambos modelos son equivalentes en características en lo que a métricas se refiere. El diagnóstico, aunque ruidoso por ser un modelo más simple de lo ideal, es muy cercano al valor visto en validación (tabla 5.10), por lo que podemos confirmar que el aprendizaje se realizó de forma satisfactoria.

Tiempo de inferencia

Una vez evaluado el rendimiento de ambos modelos, podemos verificar las ganancias en rendimiento obtenidas con el procedimiento de cuantización. Al tratarse de un problema con mayor número de salidas, dicha reducción será menos notable. Empleando nuestro criterio de evaluar el promedio de 200 imágenes, obtenemos el resultado mostrado en la tabla 6.9.

Modelo	Tiempo total (ms)	Tiempo medio (ms)
Original	114972	574,86
Cuantizado	73774	368,87

Tabla 6.9: Tiempos de inferencia para caso binario en Android

A la vista de los resultados, podemos apreciar una ganancia significativa en tiempo, de alrededor de 206ms por imagen. Si calculamos la ganancia en velocidad, obtenemos:

$$Ganancia = \frac{t\ original}{t\ cuantizado} = \frac{574,86}{368,87} = 1,558$$

Por tanto, el modelo optimizado nos aporta una ganancia de 1,558, que si bien es ligeramente inferior a los dos modelos analizados anteriormente, ofrece una ventaja muy similar en rendimiento, con la ventaja de perder precisión en sus salidas de forma apreciable.

6.4. Conclusión de la cuantización

Una vez evaluados los 3 modelos, podemos dar el veredicto final de los resultados

1. La generalización de los 3 modelos al realizar la inferencia del conjunto de test ha sido adecuada, ya que las métricas obtenidas son prácticamente equivalentes a las obtenidas en validación por cada uno de los modelos. Esto demuestra la efectividad del método seguido, realizando las particiones de entrenamiento, validación y test.

2. El proceso de optimización y cuantización ofrecido por Pytorch mobile [52] ha demostrado ser efectivo: la pérdida de rendimiento es mínima, siendo inferior a 10^{-3} en todos los casos, ya que no es apreciable en el resultado de dos decimales de precisión.

3. Las ganancias en tiempo de inferencia son considerables: en los 3 casos, el factor de ganancia ha sido superior a 1.5 (visible gráficamente en la figura 6.2), lo cual indica que el modelo cuantizado sería capaz de inferir aproximadamente 3 imágenes en el tiempo en el que el modelo original infiere 2.

Las ganancia en rendimiento de inferencia, sumado a la inexistencia de pérdida de precisión en los casos estudiados, convierte a la cuantización en la solución ideal para la implementación del modelo en dispositivos móviles.

Figura 6.2: Comparativa de ganancias al emplear modelos cuantizados

Bibliografía

[1] Dermatology information system. https://www.dermis.net/dermisroot/en/home/index.htm. [Online; accessed 21-Feb-2024].

[2] Ph2. https://www.fc.up.pt/addi/ph2%20database.html, 2012. [Online; accessed 19-Feb-2024].

[3] Dermnet: all about skin. https://dermnetnz.org/dermatology-image-dataset, 2020. [Online; accessed 19-Feb-2024].

[4] The international skin imaging collaboration. https://www.isic-archive.com, 2023. [Online; accessed 20-September-2023].

[5] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*. AMLBook, 2012.

[6] Vardan Agarwal. Bye-bye mobilenet. hello efficientnet! https://towardsdatascience.com/bye-bye-mobilenet-hello-efficientnet-9b8ec2cc1a9c, 2020. [Online; accessed 10-April-2024].

[7] Samuel Albanie. Image classification architectures. https://github.com/albanie/convnet-burden, 2019. [Online; accessed 8-June-2024].

[8] Marin Benčević, Irena Galić, Marija Habijan, and Danilo Babin. Training on polar image transformations improves biomedical image segmentation. *IEEE access*, 9:133365–133375, 2021.

[9] MD: National Cancer Institute. Bethesda. Pdq tratamiento del cáncer de piel. https://www.cancer.gov/espanol/tipos/piel/paciente/tratamiento-piel-pdq, 2023. [Online; accessed 10-October-2023].

[10] Bill Cassidy, Connah Kendrick, Andrzej Brodzicki, Joanna Jaworek-Korjakowska, and Moi Hoon Yap. Analysis of the isic image datasets: Usage, benchmarks and recommendations. *Medical Image Analysis*, 75:102305, 2022.

[11] Saket S. Chaturvedi, Kajol Gupta, and Prakash S. Prasad. *Skin Lesion Analyser: An Efficient Seven-Way Multi-class Skin Cancer Classification Using MobileNet*, page 165–176. Springer Singapore, May 2020.

[12] Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, and Allan Halpern. 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), 2018.

[13] Marc Combalia, Noel C. F. Codella, Veronica Rotemberg, Brian Helba, Veronica Vilaplana, Ofer Reiter, Cristina Carrera, Alicia Barreiro, Allan C. Halpern, Susana Puig, and Josep Malvehy. Bcn20000: Dermoscopic lesions in the wild, 2019.

[14] PyTorch Contributors. Torchscript. https://pytorch.org/docs/stable/jit.html, 2023. [Online; accessed 8-June-2024].

[15] All data are Ext. Siim-isic melanoma classification, 1st solution. https://www.kaggle.com/competitions/siim-isic-melanoma-classification/discussion/175412, 2020. [Online; accessed 8-June-2024].

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[17] Mohammad Fraiwan and Esraa Faouri. On the automatic detection and classification of skin cancer using deep transfer learning. *Sensors*, 22:4963, 06 2022.

[18] Ramazan Gençay. Early stopping based on validation. https://paperswithcode.com/method/early-stopping, 2001. [Online; accessed 14-June-2024].

[19] Ioannis Giotis, Nynke Molders, Sander Land, Michael Biehl, Marcel F. Jonkman, and Nicolai Petkov. Med-node: A computer-assisted melanoma diagnosis system using non-dermoscopic images. *Expert Systems with Applications*, 42(19):6578–6585, 2015.

[20] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[21] Google. Efficientnetlite (mediapipe. https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/efficientnetlite?hl=es-419&pli=1, 2024. [Online; accessed 18-April-2024].

[22] Walaa Gouda, Najm Us Sama, Ghada Al-Waakid, Mamoona Humayun, and Noor Zaman Jhanjhi. Detection of skin cancer based on skin lesion images using deep learning. *Healthcare*, 10(7), 2022.

[23] Manu Goyal, Thomas Knackstedt, Shaofeng Yan, and Saeed Hassanpour. Artificial intelligence-based image classification methods for diagnosis of skin cancer: Challenges and opportunities. *Computers in biology and medicine*, 127:104065, 2020.

[24] David Gutman, Noel C. F. Codella, Emre Celebi, Brian Helba, Michael Marchetti, Nabin Mishra, and Allan Halpern. Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (isbi) 2016, hosted by the international skin imaging collaboration (isic), 2016.

[25] Seung Han, Ik Moon, Jung-Im Na, Myoung Kim, Gyeong Park, Seonghwan Kim, Kiwon Kim, Ju Lee, and Sung Chang. Retrospective assessment of deep neural networks for skin tumor diagnosis. 12 2019.

[26] Seung Han, Gyeong Park, Woohyung Lim, Myoung Kim, Jung-Im Na, Ilwoo Park, and Sung Chang. Deep neural networks show an equivalent and often superior performance to dermatologists in onychomycosis diagnosis: Automatic construction of onychomycosis datasets by region-based convolutional deep neural network. *PLOS ONE*, 13:e0191493, 01 2018.

[27] Seung Seog Han. Asan and Hallym Dataset (Thumbnails). 9 2017.

[28] Seung Seog Han, Myoung Shin Kim, Woohyung Lim, Gyeong Hun Park, Ilwoo Park, and Sung Eun Chang. Classification of the clinical images for benign and malignant cutaneous tumors using a deep learning algorithm. *Journal of Investigative Dermatology*, 138(7):1529–1538, 2018.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[30] Samuel Hoffstaetter. Python tesseract. https://github.com/h/pytesseractm. [Online; accessed 10-March-2024].

[31] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[32] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[33] Jeremy Howard et al. fastai. https://github.com/fastai/fastai, 2018.

[34] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[35] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size, 2016.

[36] WCRF International. Global cancer statistics for the most common cancers in the world. https://www.wcrf.org/cancer-trends/worldwide-cancer-data/, 2023. [Online; accessed 10-October-2023].

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[38] Andrey Kuzmin, Mart Van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, and Tijmen Blankevoort. Fp8 quantization: The power of the exponent, 2024.

[39] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[40] M. Llamas-Velasco and B.E. Paredes. La biopsia cutánea: bases fundamentales. parte i. *Actas Dermo-Sifiliográficas*, 103(1):12–20, 2012.

[41] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks, 2018.

[42] Samyukta Lanka. Megha Arora. Accelerating squeezenet on fpga. https://lankas.github.io/15-618Project/, 2024. [Online; accessed 21-April-2024].

[43] Teresa Mendonça, Pedro Ferreira, Jorge Marques, André Marçal, and Jorge Rozeira. Ph2 - a dermoscopic image database for research and benchmarking. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2013:5437–5440, 07 2013.

[44] Sana Nazari and Rafael Garcia. Automatic skin cancer detection using clinical images: A comprehensive review. *Life*, 13(11), 2023.

[45] Neuralscope. Mobilenets architectures benchmark. https://neuralscope.org/mobile/index.php?route=product/benchmark, 2019. [Online; accessed 8-June-2024].

[46] Andre G.C. Pacheco, Gustavo R. Lima, Amanda S. Salomão, Breno Krohling, Igor P. Biral, Gabriel G. de Angelo, Fábio C.R. Alves Jr, José G.M. Esgario, Alana C. Simora, Pedro B.C. Castro, Felipe B. Rodrigues, Patricia H.L. Frasson, Renato A. Krohling, Helder Knidel, Maria C.S. Santos, Rachel B. do Espírito

Santo, Telma L.S.G. Macedo, Tania R.P. Canuto, and Luíz F.S. de Barros. Pad-ufes-20: A skin lesion dataset composed of patient data and clinical images collected from smartphones. *Data in Brief*, 32:106221, 2020.

[47] Ian Pan. Siim-isic melanoma classification, 2nd solution. https://www.kaggle.com/competitions/siim-isic-melanoma-classification/discussion/175324, 2020. [Online; accessed 8-June-2024].

[48] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[51] Paul-Louis Pröve. Mobilenetv2: Inverted residuals and linear bottlenecks. https://towardsdatascience.com/mobilenetv2-inverted-residuals-and-linear-bottlenecks-8a4362f4ffd5, 2018. [Online; accessed 21-April-2024].

[52] Pytorch. Pytorch mobile, quantization and optimizaation. https://pytorch.org/mobile/home/, 2023. [Online; accessed 18-April-2024].

[53] Pytorch. Quantization. quantization api summary. https://pytorch.org/docs/stable/quantization.html, 2023. [Online; accessed 18-April-2024].

[54] Google Renjie Liu. Higher accuracy on vision models with efficientnet-lite. https://blog.tensorflow.org/2020/03/higher-accuracy-on-vision-models-with-efficientnet-lite.html, 2020. [Online; accessed 18-April-2024].

[55] Veronica Rotemberg, Nicholas Kurtansky, Brigid Betz-Stablein, Liam Caffery, Emmanouil Chousakos, Noel Codella, Marc Combalia, Stephen Dusza, Pascale Guitera, David Gutman, Allan Halpern, Brian Helba, Harald Kittler, Kivanc Kose, Steve Langer, Konstantinos Lioprys, Josep Malvehy, Shenara Musthaq, Jabpani Nanda, Ofer Reiter, George Shih, Alexander Stratigos, Philipp Tschandl, Jochen Weber, and H. Peter Soyer. A patient-centric dataset of images and

metadata for identifying melanomas using clinical context. *Scientific Data*, 8(1), January 2021.

[56] Quirón Salud. El 90diagnostica en fases tempranas. https://www.quironsalud.com/es/comunicacion/actualidad/90-casos-cancer-colon-puede-curar-diagnostica-fases-tempran, 2016. [Online; accessed 10-October-2023].

[57] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[58] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019.

[59] Rupali Kiran et al. Shinde. Squeeze-mnet: Precise skin cancer detection model for low computing iot devices using transfer learning, 2022.

[60] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, 2018.

[61] Xiaoxiao Sun, Jufeng Yang, Ming Sun, and Kai Wang. A benchmark for automatic visual classification of clinical skin disease images. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 206–222, Cham, 2016. Springer International Publishing.

[62] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[63] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[64] Dhiraj Jaiswal Tiancheng Xu. Dermnet-images-crawler. https://github.com/tcxxxx/DermNet-images-crawler, 2020. [Online; accessed 19-Feb-2024].

[65] Sik-Ho Tsang. Review: Shufflenet v1 — light weight model. https://towardsdatascience.com/review-shufflenet-v1-light-weight-model-image-classification-5b253dfe982f, 2019. [Online; accessed 8-June-2024].

[66] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5, 08 2018.

[67] Xintao Wang. Esrgan (enhanced srgan). https://github.com/xinntao/ESRGAN. [Online; accessed 12-April-2024].

[68] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

[69] World Health Organization (WHO). Global cancer burden growing, amidst mounting need for services. https://www.who.int/news/item/01-02-2024-global-cancer-burden-growing--amidst-mounting-need-for-services, 2024. [Online; accessed 10-February-2024].

[70] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017.