ULPGC Universidad de Las Palmas de Gran Canaria | Escuela de Ingeniería Informática

Hacia la detección automática del Signo de Frank para la prevención de riesgo cardiovascular

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Sara Lis Almonacid Uribe

TUTORIZADO POR: David Sebastián Freire Obregón

FECHA: Noviembre 2022

Agradecimientos

Antes que nada, agradecer a mi tutor por darme la oportunidad de realizar este trabajo y brindarme con toda la ayuda y disponibilidad posible.

A mis padres y hermano, que me han dado el espacio y el apoyo necesario para poder realizar esta carrera.

A mi pareja, que me ha acompañado y ha sido un punto de apoyo muy importante durante este largo e intenso camino.

A mis amigos de la universidad que gracias a ellos esta experiencia ha sido de las mejores de mi vida.

Resumen

Según el estudio más reciente de la OMS y el INE, la cardiopatía isquémica sigue siendo la mayor causa de defunción en el mundo. En España, la segunda superada por el COVID-19 en 2020.

Hay estudios en el campo de la medicina que indican una posible correlación entre un pliegue diagonal de unos 45º, localizado en el lóbulo de la oreja, llamado el Signo de Frank (SF) y el riesgo de padecer enfermedades cardiovasculares.

Este proyecto tiene como objetivo detectar automáticamente este signo mediante la visión por computador, haciendo uso de redes neuronales pre-entrenadas para hacer una comparativa en el estudio y encontrar aquellas/s que sea capaz de detectar este marcador con éxito.

Abstract

According to the most recent study by the WHO and the INE, ischemic heart disease continues to be the leading cause of death worldwide. In Spain, it is the second, surpassed by COVID-19 in 2020.

There are studies in the field of medicine that indicate a correlation between a 45° diagonal crease located in the earlobe, called Frank's Sign (SF), and the risk of suffering from cardiovascular diseases.

This project aims to automatically detect this sign through Computer Vision, using pre-trained neural networks to make a comparative study and find the one/s capable of successfully detecting this marker

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

INTRODUCCIÓN

Actualmente nos encontramos en una época donde la sociedad avanza de la mano de la tecnología, prácticamente integrada en todos los aspectos de la vida. Es por ello por lo que se ha vuelto una herramienta indispensable para el avance de la sociedad. Si concretamos este aspecto en uno de los campos más importantes para el ser humano, la salud, llegamos a la conclusión de que la tecnología tiene que ser el empuje hacia nuevas líneas de investigación que permitan a los profesionales tener la capacidad de prevenir y actuar ante posibles enfermedades y patologías.

Una de las primeras causas de muerte en el mundo es por motivos cardiovasculares, la más frecuente, la enfermedad coronaria [1]. Por lo que es de vital importancia tener los mecanismos adecuados para poder detectar algún signo o indicio que permita a los profesionales actuar en consecuencia y poder prevenir cuanto antes este tipo de enfermedades que desencadenan comúnmente en ataques al corazón o accidentes cerebrovasculares.

En 1973 el neumólogo Sanders T. Frank asoció la enfermedad arterial coronaria con el surco diagonal del lóbulo de la oreja, denominándolo *Signo de Frank* [2]. Con el paso de los años se han realizado varios estudios médicos en torno a este "marcador cutáneo" de enfermedad coronaria.

Es notorio el auge que está teniendo el campo de la Inteligencia Artificial, el cual engloba muchos conceptos que se estudian en este proyecto. Un ejemplo de este auge es la fama que están adquiriendo los vehículos autónomos, sistemas inteligentes funcionales para el hogar (IoT), sistemas de detección de cánceres que superan muchos estudios de expertos en la materia, etc. Es por esto, que este campo está abriendo muchas puertas a la hora de desarrollar aplicaciones que aporten y faciliten muchas funciones en casi todos los ámbitos de nuestra sociedad (industrial, hogar, salud, académico, etc.).

Este trabajo nace con la motivación de poder desarrollar un estudio en el que mediante el aprendizaje automático se apliquen distintas arquitecturas de redes neuronales gracias a la visión por computador para encontrar aquella que sea capaz de detectar con la mayor precisión, la existencia del Signo de Frank en imágenes de orejas. Con este estudio se pretende abordar una problemática no resuelta mediante estas tecnologías, que permitan una identificación efectiva de este signo y que pueda servir como referencia para futuros estudios. Esta memoria recoge los resultados que se han obtenido y se exponen cuáles han sido los que han logrado una mejor precisión y cuales se pueden descartar o bien mejorar para incrementar su valor, así como otros factores de interés.

1.1 Objetivos

Como objetivo general de este proyecto, se plantea realizar un clasificador binario que detecte el Signo de Frank mediante técnicas de aprendizaje profundo.

Los objetivos de este TFT son los siguientes:

- **Localizar y etiquetar imágenes de orejas positivas en SF:** Se va a realizar un doble etiquetado, por un lado, la oreja entera y, por otro lado, solamente la zona del lóbulo. Ello nos proporcionará versatilidad en la fase de pruebas.
- **Aumentar el conjunto de datos**: Utilizaremos técnicas de *data augmentation* con el objetivo de incrementar el conjunto de muestras de entrenamiento.
- **Estudio de arquitecturas disponibles para realizar un *fine tuning* sobre los datos.**
- **Pruebas de rendimiento y validación**: Se aplicarán distintas arquitecturas para, posteriormente, compararlas.

1.2 Estructura del trabajo

Este trabajo se compone de 7 capítulos que se explicarán brevemente en esta sección:

- **Capítulo 1**: Se desarrolla la introducción de este trabajo, los objetivos que se han planteado para desarrollar el proyecto y la estructura de la memoria.
- **Capítulo 2**: Estado del Arte. Apartado donde se mencionan ciertos artículos relacionados con el motivo de este trabajo y la descripción del estado actual de las tecnologías usadas para llevar a cabo este proyecto.
- **Capítulo 3**: Se compone de la metodología implementada y las tareas planificadas inicialmente de este trabajo.
- **Capítulo 4:** Se describen brevemente las librerías utilizadas, el lenguaje y el entorno utilizado.
- **Capítulo 5:** Se mencionan las competencias específicas cubiertas.
- **Capítulo 6**: Desarrollo. Se compone de las diferentes fases para llevar a cabo el proyecto, se exponen los resultados que se han obtenido y se hace una evaluación de estos.
- **Capítulo 7**: Conclusiones y trabajo futuro. Se exponen las conclusiones obtenidas y se estudian posibles propuestas como continuación de este trabajo.

Capítulo 2

ESTADO DEL ARTE

2.1 Signo de Frank y problemas coronarios

Varios estudios realizados durante las últimas décadas han dejado una base en cuanto a la correlación entre el Signo de Frank y problemas de naturaleza cardíaca. Explicando el contexto de esta correlación, el Signo de Frank o DELC (*Diagonal Earlobe Crease*) es un pliegue diagonal de presencia unilateral o bilateral situado en el lóbulo de la oreja, normalmente formando un ángulo de unos 45º que va desde la incisura intertrágica a través del lóbulo hasta el borde posterior de la masa auricular, ver Figura 2-1. Este signo tiene un sistema de clasificación relacionado a la incidencia de estos eventos cardiovasculares en función de la longitud, profundidad, bilateralidad e inclinación, siendo la bilateralidad completa el caso más grave [3], véase en la Figura 2-2.

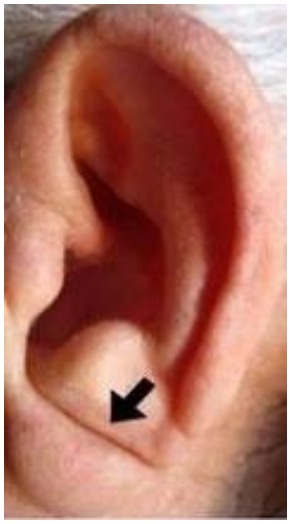


Figura 2-1.Signo de Frank

Aunque algunas veces sea complicado identificar este signo, se ha hecho un registro en base a la severidad de este. Los primeros grados pueden ser casos difíciles de clasificar, debido a que no se puede asegurar que sea el signo, pero se pueden tener en cuenta como la evolución gradual de un paciente que lo va desarrollando.

Los grados de severidad estudiados son los siguientes (Figura 2-3):

- Grado 1: arrugas
- Grado 2a: pliegue superficial a través del lóbulo (no necesariamente a través de todo el lóbulo)
- Grado 2b: pliegue que cubre más de la mitad del lóbulo, superficial o no.
- Grado 3: profunda hendidura en todo el lóbulo de la oreja.

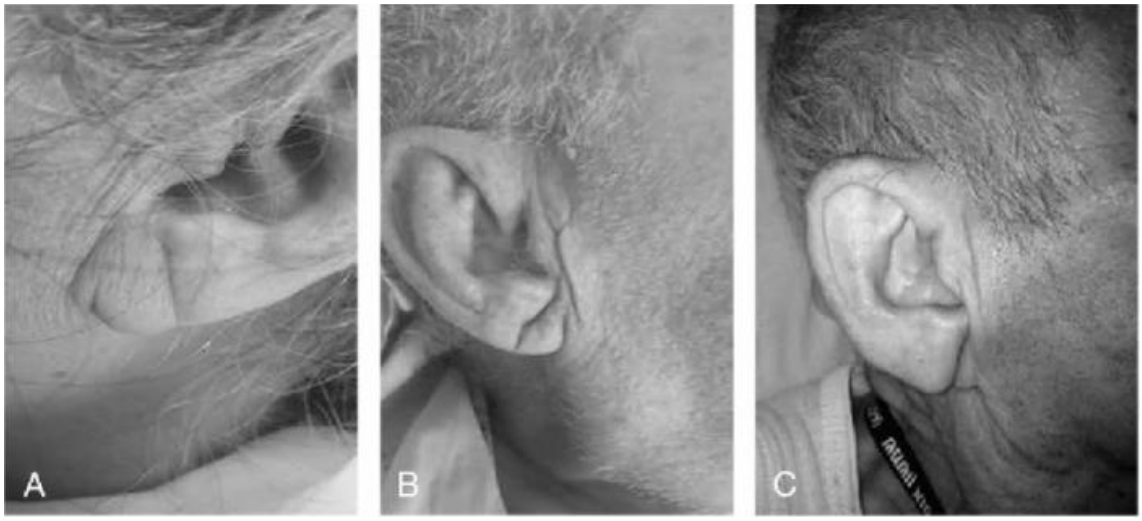


Figura 2-2 Clasificación del signo según la profundidad. Fuente: S. Nazzal [4]

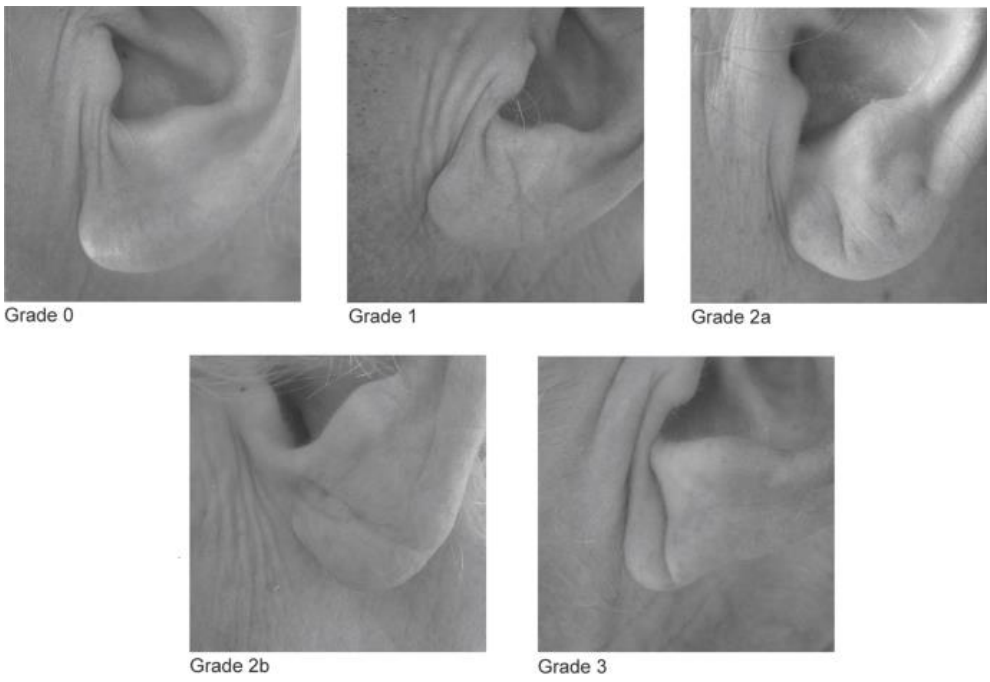


Figura 2-3 Clasificación del Signo según el grado de severidad. Fuente: C. Thilo et al. [5]

La primera hipótesis de esta correlación fue desarrollada por el neumólogo Sanders T. Frank en 1973. Hay expertos que consideran que simplemente es un signo marcador del envejecimiento fisiológico [7], sin embargo, esta hipótesis empezó a coger fuerza y se empezaron a desarrollar varios estudios donde se muestra que esta correlación puede ser realmente efectiva a la hora de poder predecir si un paciente tiene predisposición a padecer problemas cardiovasculares. En

concreto, la cardiopatía coronaria se encuentra entre las principales causas de muerte en países desarrollados [6], por lo que un diagnóstico precoz es un factor clave para poder mejorar la calidad de vida del paciente y poder evitar o reducir la mortalidad a causa de esta afección. Según un estudio realizado por George S. Stoyanov et al. [8] donde se estudiaron 45 pacientes remitidos para autopsia, se evaluó que de los 22 pacientes que presentaban historial clínico de enfermedades cardíacas, el 100% presentaba el signo de Frank. Y de los 23 pacientes sin historial clínico, 13 presentaban el signo. Estos autores concluyeron que, de la histopatología estudiada, la causa principal de muerte en los pacientes con este signo era cardiovascular.

Otro estudio realizado en 2018 por A. N. Lin et al. [9], analizó el caso de un paciente con síntomas poco habituales como opresión en el pecho y mareos. Realizándole una exploración física no se encontró ningún signo relevante excepto la existencia del Signo de Frank bilateral, cuyo paciente afirmó que tenía desde joven. A través de eso se le realizaron estudios cardiológicos diagnosticándole que padecía de enfermedad coronaria.

Un estudio un poco más amplio realizado por A. Pasternac y M. Sami en 1982 [10], evaluaron un total de 340 pacientes de los cuales 257 padecían de enfermedad coronaria. Se demostró que el 91% de los pacientes con el signo de Frank padecían de enfermedad arterial coronaria, siendo el signo más frecuente en aquellos cuyo grado de severidad de la enfermedad era más alto.

Estos ejemplos de los muchos que se han ido realizando han conseguido resultados que demuestran que el signo puede ser altamente de utilidad para poder detectar y prevenir futuras complicaciones cardíacas, sobre todo, puede servir para mantener una observación en aquellas personas que desarrollan el signo a una edad temprana o aquellas personas mayores en las que el signo se encuentre en un grado muy avanzado sin haber presentado síntomas adicionales relacionados con enfermedades cardíacas.

2.2 Aprendizaje profundo y detección de enfermedades mediante orejas

La detección del Signo de Frank en imágenes donde se muestre la oreja o el lóbulo de esta se encuentra dentro del marco teórico-práctico de reconocimiento de objetos que se puede clasificar en dos pasos. El primero es la extracción de ciertas características en el contenido de una imagen y, posteriormente, la realización de una búsqueda de objetos que se base en las características extraídas. Para llevar a cabo este proceso se pueden usar diferentes técnicas de aprendizaje, en este caso, este proyecto se centra en la técnica de aprendizaje profundo.

El aprendizaje profundo o *Deep Learning* se está implementando cada vez más en problemáticas relacionadas con el diagnóstico de enfermedades, tanto para ayudar a los especialistas a realizar un correcto diagnóstico como para agilizar uno que pueda ser crucial para la vida de un paciente [42].

Haciendo una extensa exploración del estado del arte relacionado con el aprendizaje profundo y la detección automática del Signo de Frank, no se pudo encontrar estudio alguno que resolviera esta vía de investigación, por lo que este trabajo puede servir de punto inicial para

dejar constancia de la aplicación de esta tecnología con este marcador cutáneo. Sin embargo, se puede mencionar el caso de las enfermedades en la región del oído, por el cual varios estudios han desarrollado una detección bastante precisa para las enfermedades más comunes y a la vez peligrosas como la otitis, colesteroma, acumulación de cerumen, etc.

Un estudio realizado por X. Zeng *et al* [11], hizo uso de redes pre-entrenadas tales como *ResNet, DenseNet, Inception y MobileNet* para clasificar un total de 8 enfermedades que se desarrollan en el oído. Obtuvieron una precisión de 93-95% entre todos los modelos, destacando *DenseNet* por encima de las demás.

Otro estudio realizado por Y. Wang *et al.* [12], demostró como una red de clasificación basada en la red pre-entrenada Inception-V3 para detectar Otitis crónica, obtuvo una precisión del 76,7% frente a la precisión realizada por expertos clínicos con un porcentaje inferior del 73,8%. Esto demuestra cómo la inteligencia artificial puede superar en predicción a los expertos, evidenciando la gran utilidad que está proporcionando esta tecnología.

Finalmente, un estudio reciente desarrollado por Y. Chen *et al*. [13] implementó un algoritmo de aprendizaje por transferencia (*Transfer Learning,* en inglés) [14] para la detección de enfermedades auditivas a través de un dispositivo móvil o smartphone. Resulta interesante debido a que la mayoría de los estudios realizados suelen centrarse únicamente en el desarrollo de la detección automática, sin embargo, este facilita el uso de esta inteligencia para usarla en servicios de medicina online o telemedicina, servicios que están experimentando un crecimiento de uso debido, en mayor parte, a la pandemia que se ha vivido estos últimos años.

2.3 Redes Neuronales Convolucionales (*CNN*)

En este apartado se explicarán los conceptos básicos del funcionamiento de una red neuronal convolucional [41]. Las arquitecturas que se explicarán en el siguiente apartado están compuestas por estas redes, por lo que es conveniente tener claro las bases de los modelos usados en este proyecto.

Las redes neuronales convolucionales son redes artificiales cuyo comportamiento se asemeja al de las neuronas del cerebro biológico. El comportamiento que simulan estas redes es el de extraer ciertas características en imágenes para el posterior procesamiento y clasificación de las mismas, debido a esto, son muy eficientes para trabajos que impliquen la visión artificial, como es el caso de este proyecto.

La arquitectura de estas redes se compone de una capa de entrada, un conjunto de capas ocultas (*hidden layers)* donde se aplican las convoluciones y una capa de salida, véase la Figura 2-4.

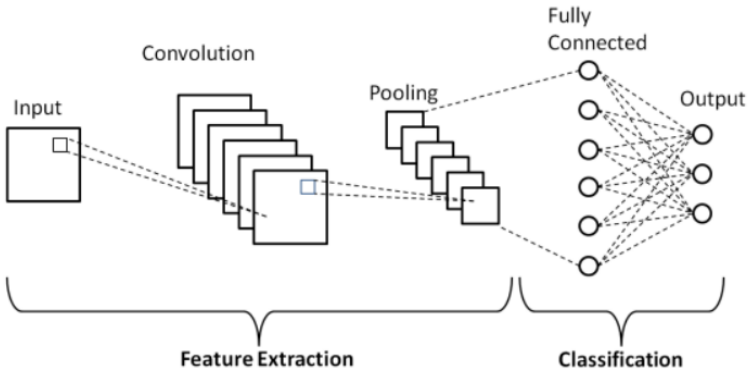


Figura 2-4. Estructura de una CNN. Fuente: Sai Balaji [30]

La capa de entrada está definida por un conjunto de imágenes donde cada imagen está compuesta por píxeles, la red toma como entrada estos píxeles y los transforma en X neuronas que serán la capa de entrada de la red. El número de neuronas de la capa se verá afectado si la imagen es a color o no. En este proyecto las imágenes a estudiar han sido a color, entonces para obtener el número total de neuronas se ha tenido en cuenta 3 canales de entrada (RGB).

Una red neuronal convolucional se caracteriza por emplear convoluciones, la Figura 2-5 representa este proceso, que consiste en realizar el producto escalar entre una matriz llamada kernel, encargada de extraer patrones de la imagen y normalmente de tamaño 3x3 o 5x5 contra una matriz del mismo tamaño tomada de la imagen de entrada. A medida que la matriz kernel se va desplazando de izquierda a derecha y de arriba abajo por la matriz de la imagen, se van obteniendo los valores que conforman la matriz resultante, conocida como capa oculta. Para las imágenes a color, el proceso es el mismo, solo que se emplean 3 filtros, uno por cada color RGB.

Este proceso va obteniendo por cada convolución lo que se conoce como un mapa de características o *feature map*, en inglés. Por tanto, la profundidad de la red dependerá de la cantidad de filtros que se apliquen para generar distintos mapas de características de la imagen, cada vez más complejas.

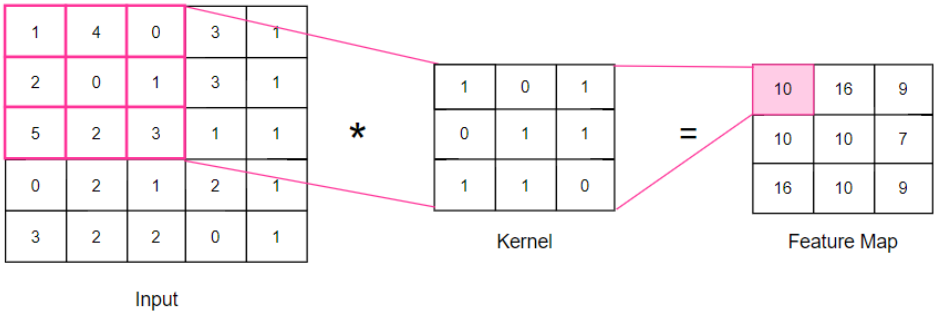


Figura 2-5. Proceso de convolución

Cabe mencionar dos técnicas que se usan en estas redes a la hora de realizar las convoluciones:

- El relleno (*padding):* esta técnica expande la matriz de entrada agregando píxeles con valor 0 en todos los bordes de la matriz. Esto se hace debido a que el proceso de convolución genera una imagen de salida de menor tamaño que la de entrada, por lo que gracias a esta técnica se consigue controlar el tamaño de la matriz resultante permitiendo que una red pueda tener mucha profundidad dado que el tamaño de la imagen se mantiene.
- Los saltos *(strides):* esta técnica es el desplazamiento que realiza el kernel por la matriz de entrada, como se explicó en apartados anteriores. El proceso de convolución puede realizarse con saltos mayores que uno, es decir en vez de desplazarse uno a la izquierda, se desplaza dos veces. Con esto se obtendría imágenes de menor tamaño, resultando útil si lo que se quiere es reducir la cantidad de datos a procesar entre capas.

Por cada capa convolutiva, se aplica una capa de agrupación (*pooling layer, en inglés*). Esta capa disminuye el tamaño del mapa de características obtenido para reducir costes computacionales manteniendo la información útil para la detección de las características estudiadas. Existen dos tipos de agrupación:

- La agrupación máxima o *max pooling:* escoge el valor más grande del mapa de características
- La agrupación promedio o *average pooling:* escoge la media calculada de los valores del mapa de características.

Por último, la capa final de la red llamada *fully-connected*. Esta capa traduce la información obtenida de las capas previas y transforma la salida en un vector del tamaño al número de clases especificado en el que cada valor del vector representa la probabilidad que tiene la imagen de pertenecer a una de las clases, ej. Clases: [0, 1]; Output = [0.9, 0.1] → 90% de probabilidades de que la imagen sea una oreja sana y 10% de que sea una oreja con el signo de Frank.

Esta capa contiene una función de pérdida que permite calcular el error a la hora de realizar la predicción haciendo que para la red sea posible identificar las imágenes e ir aprendiendo a clasificarlas correctamente.

2.4 Modelos pre-entrenados utilizados

Los modelos pre-entrenados son aquellas redes que han sido creadas y entrenadas previamente para resolver un problema de clasificación mediante un conjunto de datos formado por más de 14 millones de imágenes llamado ImageNet. Su uso se ha popularizado bastante debido a que estas redes permiten un ahorro en el consumo de recursos y tiempo, así como un buen motivo de investigación para usarlas en diferentes problemáticas de clasificación y estudiar su adaptabilidad.

Los modelos que se han implementado en este proyecto han sido redes pre-entrenadas ofrecidas por Keras Applications[1]. En la Tabla 2-1 se muestra un total de 13 modelos que serán empleados a estudio y evaluación. Las precisiones Top-1 y Top-5 que se describen en la tabla

[1] https://keras.io/api/applications/

significan el porcentaje de rendimiento que han obtenido los modelos para el conjunto de datos *ImageNet*. La profundidad se refiere a la cantidad de capas que tiene la red, su profundidad topológica. Y el tiempo por paso de inferencia hace referencia al promedio de 30 lotes y 10 repeticiones que ha realizado el modelo.

Tabla 2-1 Keras Application. Fuente: Keras [18].

Modelos	Tamaño (MB)	Top-1 Precisión	Top-5 Presición	Parámetros	Profundidad	Tiempo (ms) por paso de inferencia (CPU)	Tiempo (ms) por paso de inferencia (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB2	31	79.1%	94.4%	7.9M	186	60.2	5.6

2.4.1 InceptionV3

Inception V3 creado por C. Szegedy *et al.* [15] es la tercera versión del modelo *Inception,* esta version mejorada permite redes más profundas sin aumentar demasiado la cantidad de parámetros, siendo además un modelo con coste mucho más barato computacionalmente. En la Figura 2-6 se muestra un esquema del modelo.

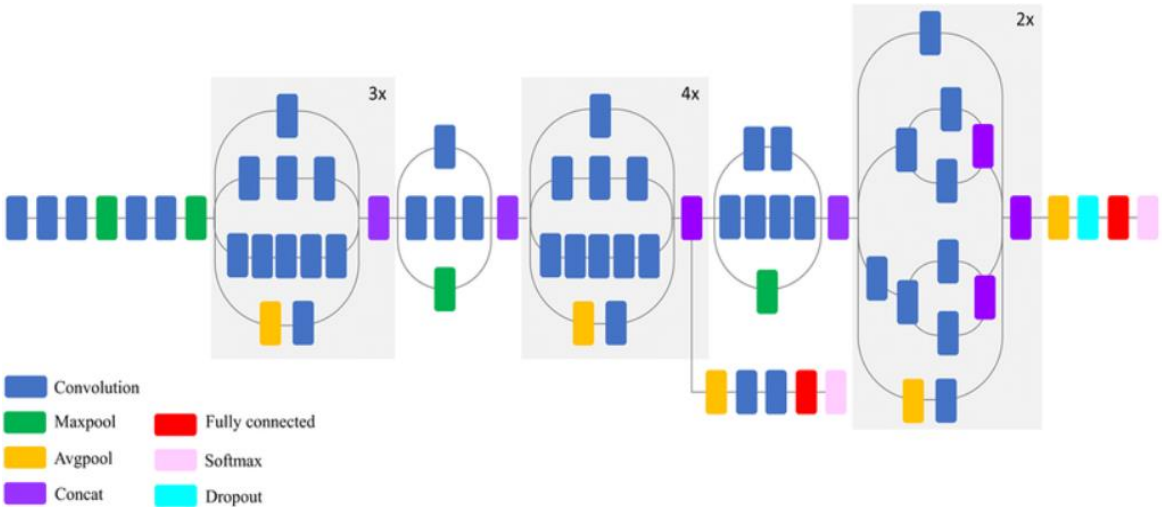


Figura 2-6. Arquitectura InceptionV3. Fuente: M. Mahdianpari [17]

2.4.2 Xception

La arquitectura Xception [16], desarrollada por François Chollet, es un modelo basado en la arquitectura Inception, de ahí que su nombre completo sea "Xtreme Inception". Este modelo representa un paso intermedio entre la convolución regular y la capa de convolución separable en profundidad a través de los módulos Inception. En la Figura 2-7 se muestra un esquema del modelo Xception.

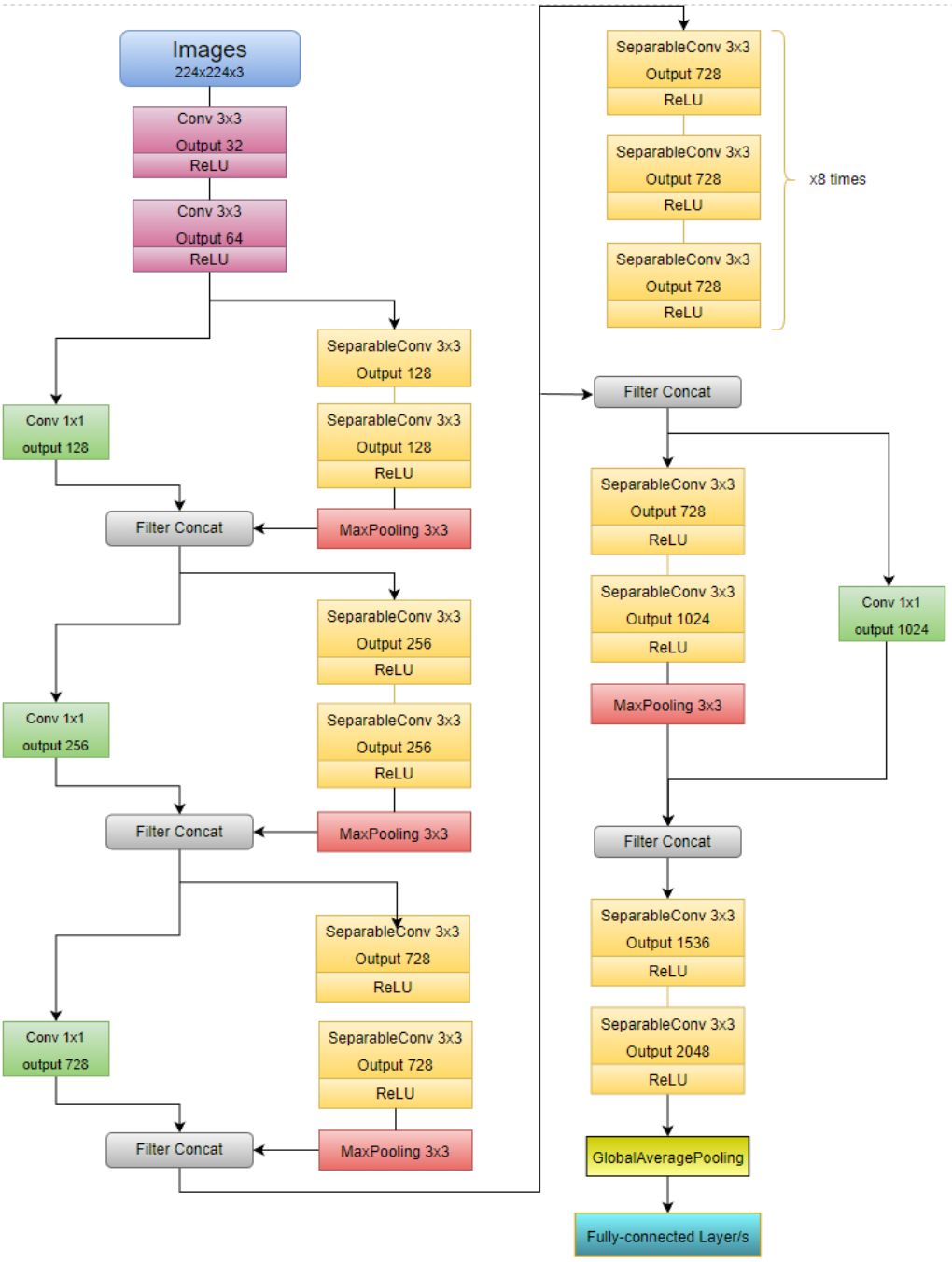


Figura 2-7. Arquitectura Xception , Figura adaptada de: Chollet [16]

2.4.3 VGG

Este modelo de red denominado VGG (***V**isual **G**eometry **G**roup*) y presentado por K. Simonyan y A. Zisserman en 2014 [19] se compone de varias capas convolucionales de tamaño 3x3 y con una agrupación máxima de 2x2 píxeles, esto refleja una simplicidad en comparación con otras arquitecturas diseñadas, sin embargo, su uso computacional es más elevado y posee unos 140M de parámetros, que es una cifra considerablemente alta. Las versiones de este modelo son VGG16 y VGG19, el número que distingue cada red representa la cantidad total de capas convolucionales y *fully-connected* que poseen. En la Figura 2-8 se puede observar el modelo VGG16.

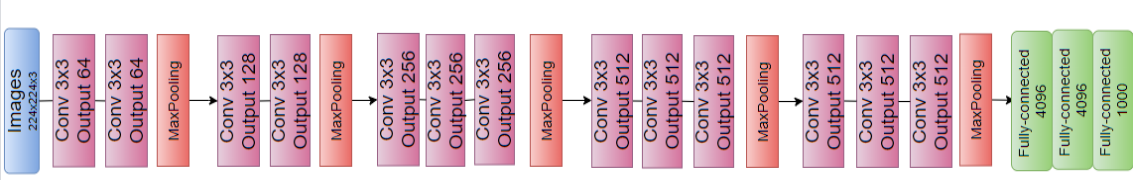


Figura 2-8. Arquitectura VGG, Figura adaptada de: Kaggle [43]

2.4.4 ResNet

Esta red propuesta por Keiming He *et al* [20], se compone de capas residuales, de ahí su nombre, **Res**idual **Net**work, ver Figura 2-9. Simulan el comportamiento biológico neuronal de manera que realizan conexiones de salto entre capas. En este proyecto se estudian los modelos ResNet50, ResNet101 y ResNet152, distinguidas por el número de capas por los que se compone cada modelo. En el entrenamiento de redes neuronales, al aumentar la cantidad de capas profundas, existe un problema llamado desvanecimiento de gradiente (*Vanishing Gradient Problem,* en inglés) [21], lo que ocurre es que las funciones de activación atenúan tanto la señal que, en una red con muchas capas, la salida de estas funciones apenas llega a alterar las capas posteriores dado que el gradiente cada vez es más cercano a 0. Sin embargo, gracias a las capas residuales, se puede obtener un entrenamiento exitoso de una red muy profunda. En la Figura 2-10 se muestra la arquitectura del bloque residual.

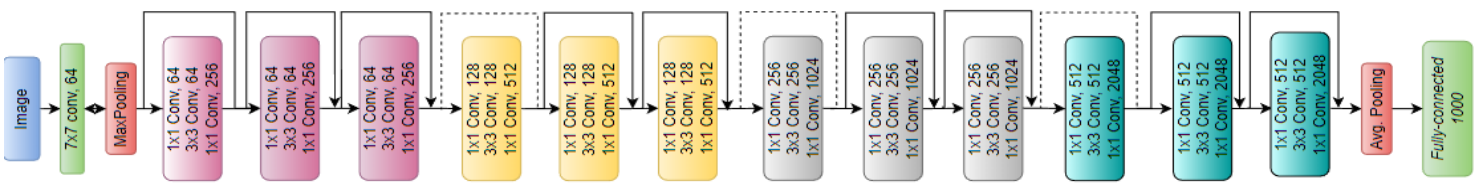


Figura 2-9. Arquitectura ResNet, Figura adaptada de: A. Rastogi [44]

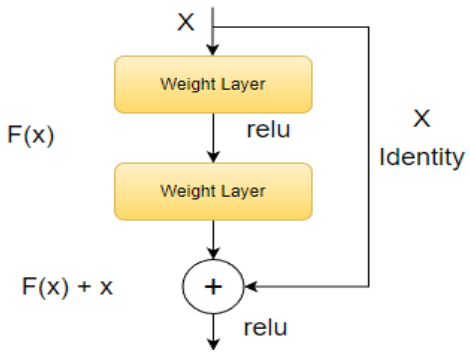


Figura 2-10. Bloque residual ResNet, Fuente: A. Rastogi [44]

2.4.5 DenseNet

Esta red fue introducida por Gao Huang *et al*. [22] en 2016, presenta una arquitectura profunda y al igual que ResNet, utiliza conexiones de salto entre capas. Pero la diferencia de la red DenseNet, es que las salidas de cada capa se concatenan con las entradas de las siguientes capas. Esto proporciona la propagación de características entre capas que se traduce en un mejor rendimiento, una considerable reducción de parámetros y reduce la aparición del problema de desvanecimiento de gradiente que puede darse en redes profundas. Las redes utilizadas en este proyecto son DenseNet121, DenseNet169 y DenseNet201, como se ha mencionado anteriormente, el número representa la cantidad de capas empleadas en cada una. En la Figura 2-11 se aprecia el modelo desarrollado.

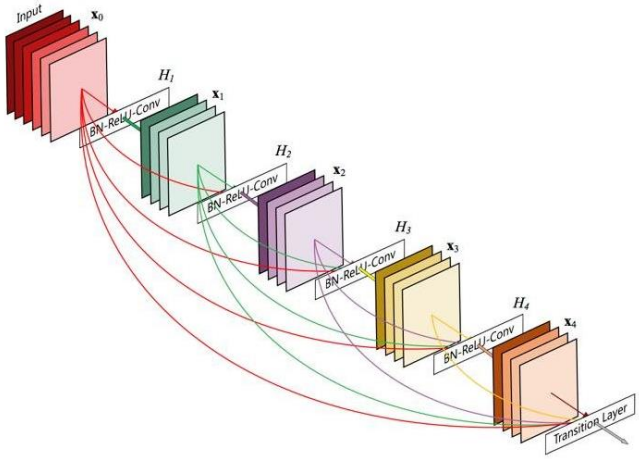


Figura 2-11. Arquitectura DenseNet. Fuente: G. Huang [21]

2.4.6 MobileNet

Esta arquitectura presentada por Andrew G. Howard *et al.* [23], ver Figura 2-12, está diseñada especialmente para aplicaciones móviles, siendo también el primer modelo de visión por computador móvil. Su modelo se basa en convoluciones separables en profundidad, lo que permite que sea una red muy profunda y a la vez ligera. Se caracterizan por su baja latencia y su velocidad de rendimiento.

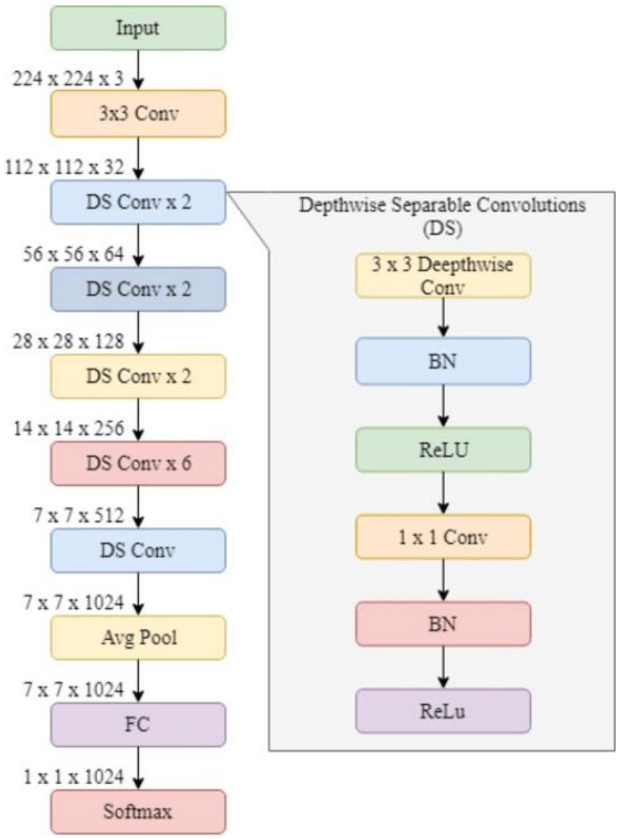


Figura 2-12. Arquitectura MobileNet. Fuente: O. Surinta [24]

2.4.7 EfficientNet

Esta red, presentada por M. Tan *et al.* [25] se caracteriza por su técnica de escalado uniforme de todas las dimensiones de profundidad-anchura-resolución mediante un conjunto de coeficientes de escalado fijo, a diferencia de las redes anteriores que realizan el escalado de manera arbitraria. Esta técnica ha permitido que esta familia de modelos (EfficientNetB0 – B7), logre la mejor precisión y un valor de parámetros mucho menor en comparación con las anteriores arquitecturas. En la Figura 2-13 se muestra el modelo de EfficientNet-B0.

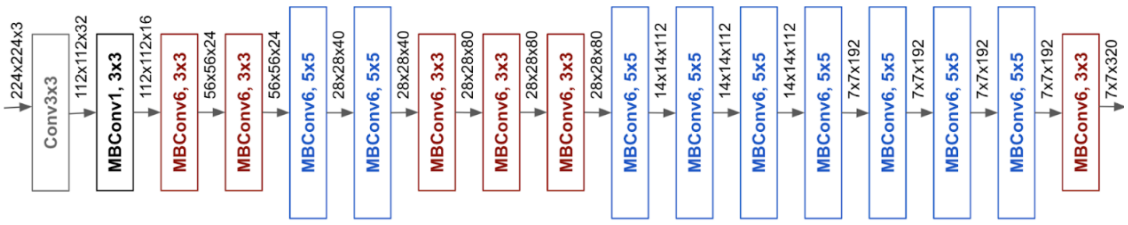


Figura 2-13. Arquitectura EfficientNet. Fuente: Google AI Blog [26]

Capítulo 3

METODOLOGÍA Y PLANIFICACIÓN

En este capítulo, se presenta la metodología utilizada a la hora de realizar el trabajo y la planificación que se planeó junto con la que se ejecutó.

3.1 Metodología

El desarrollo de este proyecto se ha basado en la metodología de prototipo o prototipado, este proceso consta en la iteración de las arquitecturas implementadas para ofrecer una mejora continua hasta obtener el mejor resultado.

En la Figura 3-1 se puede visionar una esquematización propia de cómo se elaboró el proceso de trabajo para este proyecto, donde claramente el proceso iterativo se centra en el entrenamiento de las redes y el objetivo final el despliegue de el/los modelo/s.

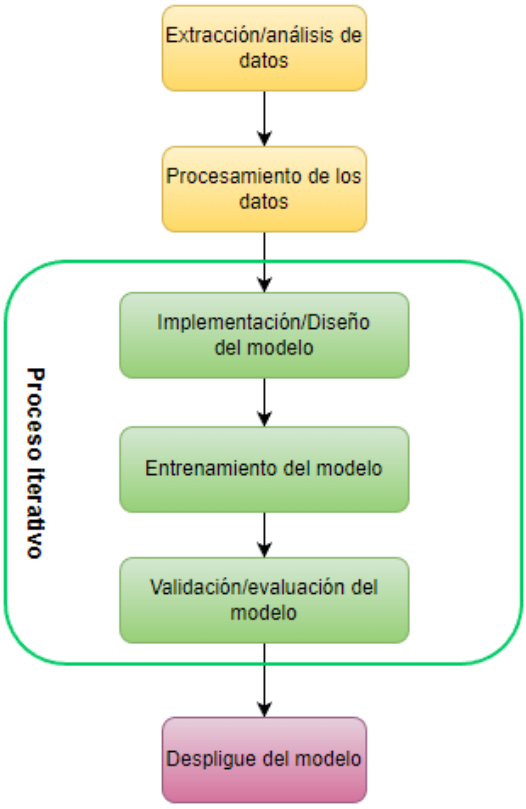


Figura 3-1. Flujo de trabajo

La primera etapa es la recopilación de datos que se necesitan para poder desarrollar el prototipo. Entiéndase como la búsqueda de un dataset que se ajuste a lo que se quiere evaluar, en este caso ha sido una recopilación manual de los datos. En esta etapa también se incluye el estudio de los conocimientos básicos en torno al Signo de Frank, realizar una búsqueda del estado del arte que engloba este proyecto, en caso de que no haya mucha información en torno a la problemática, buscar similitudes que se puedan referenciar en el área estudiada. Y, por último, el estudio previo de las tecnologías que se van a usar, así como los modelos a estudiar, su arquitectura, herramientas, etc.

Una vez concluida esta etapa, comienza la iteración en torno al desarrollo del prototipo de las redes pre-entrenadas. Se crea un primer modelo y al obtener los resultados de su entrenamiento y predicción, se evalúa la posibilidad de modificarlo, crear otro e ir comparándolo. Todo ello con la idea de obtener unos resultados óptimos para la problemática a resolver.

Terminado el diseño del pipeline o el desarrollo de los prototipos de las redes, se realiza un análisis y estudio de los resultados obtenidos para realizar comparativas y sacar conclusiones útiles para posteriores trabajos o mejoras en base al motivo general de este proyecto.

3.2 Planificación

En la Tabla 3-1, se puede observar la planificación inicial del proyecto, esta planificación es meramente orientativa, dado que la estimación de horas reales cuando se empieza el proyecto puede variar mucho en función de la disponibilidad del autor, problemas encontrados, resolución de estos, etc.

Tabla 3-1. Plan de trabajo inicial

Fases	***Duración estimada (horas)***	***Tareas***
Estudio previo / Análisis	60	Tarea 1.1: Localizar y descargar imágenes
		Tarea 1.2: Etiquetado de datos
		Tarea 1.3: Familiarización con librerías
Diseño/ Desarrollo / Implementación	140	Tarea 2.1: Aumentar el conjunto de datos
		Tarea 2.2: Preparación del conjunto de datos (split)
		Tarea 2.3: Creación de interfaz de visualización
		Tarea 2.4: Diseño del pipeline
Evaluación / Validación / Prueba	70	Tarea 3.1: Diseño de experimentos de validación
		Tarea 3.2: Evaluación de resultados
Documentación / Presentación	30	Tarea 4.1: Redacción de la memoria de TFT
		Tarea 4.2: Realización de la presentación
		Tarea 4.3: Ensayos presentación

Tabla 3-2, es la planificación real en base a la inicial, esta vez, añadiendo ciertas tareas adicionales en las fases que concretan más las actividades que se desarrollaron en el proyecto, con las horas estimadas reales y un desglose que engloba todo lo realizado en el proyecto.

Tabla 3-2. Plan de trabajo real

Fases	***Duración estimada (horas)***	***Tareas***
Estudio previo / Análisis	80	Tarea 1.1: Localizar y descargar imágenes → 50h aprox.
		Tarea 1.2: Etiquetado de datos → 10h aprox.
		Tarea 1.3: Familiarización con librerías y entorno → 5h aprox.
		Tarea 1.4: Estudio del Estado del arte → 10h aprox.
Diseño/ Desarrollo / Implementación	142	Tarea 2.1: Función segmentación de imágenes → 5h aprox.
		Tarea 2.2: Aumento de datos → 3h aprox.
		Tarea 2.3: Preparación del conjunto de datos (split) → 4h aprox.
		Tarea 2.4: Entrenamiento/reentrenamiento de los modelos → 130 h aprox.
Evaluación / Validación / Prueba	75	Tarea 3.1: Diseño de experimentos de validación → 40h
		Tarea 3.2: Evaluación de resultados → 35h aprox.
Documentación / Presentación	68	Tarea 4.1: Redacción de la memoria de TFT + Análisis y correcciones → 60 h aprox.
		Tarea 4.2: Realización de la presentación → 5 h aprox.
		Tarea 4.3: Ensayos presentación → 3h aprox.

Capítulo 4

HERRAMIENTAS UTILIZADAS

En este capítulo se describen las herramientas usadas para llevar a cabo el proyecto. Librerías, lenguajes y el entorno de desarrollo empleados para la creación del dataset, el etiquetado de imágenes, el entrenamiento de las redes, su análisis y extracción de datos.

Python

Python [31] ha sido el pilar del desarrollo de este proyecto, este lenguaje se caracteriza por ser de código abierto y estar respaldado por una gran comunidad, debido a esto, el desarrollo de nuevas librerías y su constante mantenimiento hace que sea una opción llamativa para trabajar con él. También hay que destacar su facilidad de uso y simplicidad, haciendo que sea más rápido en desarrollo que muchos lenguajes de programación.

Su código legible y conciso, permite que su uso en el campo de la inteligencia artificial resulte ventajoso, dado que tanto el Aprendizaje automático como el Aprendizaje profundo están basados en algoritmos altamente complejos e intensas tareas de cómputo, por lo que su uso permite simplificar el trabajo del desarrollador. Su creciente éxito también se debe a su gran ecosistema bibliotecario específicamente útil para este ámbito, muchas de las herramientas descritas posteriormente pertenecen a ese ecosistema.

Numpy[2]

Es una biblioteca de Python especializada en el procesamiento de arrays [32]. Está implementada en C, haciendo esto que su velocidad de cómputo sea bastante alta y sus funciones matemáticas de alto nivel permiten la realización de cálculos matemáticos complejos sobre arrays multidimensionales de manera eficiente y rápida en términos de ejecución. Es de las más importantes y básicas para comenzar proyectos de Aprendizaje automático, donde el uso eficiente de la memoria y un tiempo de cómputo óptimo es esencial. Las matrices de Numpy permiten que estas operaciones puedan realizarse en milisegundos.

Pandas[3]

Otra popular biblioteca de Python, implementada como una extensión de Numpy, proporciona potentes estructuras para la manipulación y el tratamiento de datos [33]. Su parte más importante es el Dataframe, donde los datos se reflejan en una agrupación bidimensional de

[2] NumPy: https://numpy.org/
[3] Pandas: https://pandas.pydata.org/

filas y columnas. En este proyecto se utilizó para el guardado de las predicciones de los modelos en formato CSV.

Scikit-learn[4]

De las bibliotecas más útiles de Python, contiene un amplio abanico de herramientas para el aprendizaje automático tales como la clasificación, la regresión, el agrupamiento y la reducción de la dimensionalidad [34]. En este proyecto se ha utilizado esta herramienta concretamente para realizar la división de imágenes en lotes, obtener las matrices de confusión de los modelos, las métricas de la curva ROC y su AUC.

Tensorflow [5]

Biblioteca de código abierto creada por Google Brain Team [35] y la más utilizada en el mundo del Deep Learning. Se basa en la realización de operaciones matemáticas sobre matrices multidimensionales, llamados tensores, permitiendo esto entrenar redes neuronales que permitan la detección y el descifrado de patrones, en el caso de este proyecto, el reconocimiento de imágenes.

Keras

Librería de código abierto especializada en la creación y el desarrollo de redes neuronales [18]. Keras funciona como una interfaz que permite el acceso a los frameworks tales como TensorFlow, Theano y Microsoft Cognitive Toolkit. Se ha hecho uso de esta biblioteca junto con TensorFlow precisamente para la implementación de las redes pre-entrenadas que presenta este proyecto.

Albumentations[6]

Albumentations [36] es otra librería de Python que permite trabajar con diferentes tareas de visión artificial, como clasificación, diferentes tipos de segmentación, detección de objetos, estimación de pose, etc. En este caso, se ha utilizado para la transformación y el aumento de las imágenes que permite ampliar el dataset para poder entrenar las redes.

[4] Scikit-learn: https://scikit-learn.org/stable/getting_started.html
[5] TensorFlow: https://www.tensorflow.org/?hl=es-419
[6] Albumentations: https://albumentations.ai/

Matploblib [7]

Librería especializada en la creación de gráficos 2D y 3D donde se representan los resultados de los modelos estudiados [37].

OpenCV[8]

OpenCV [38] es una biblioteca de código abierto desarrollada por Intel para la visión por computador cuyos algoritmos permiten la identificación de objetos como funcionalidad más común y otras muchas funcionalidades como extracción de modelos 3D, hacer tracking de movimiento, reconocer escenarios, etc.

Aplicación LabelImg[9]

Herramienta de anotación de imágenes gráficas escrita en Python y en QT para la interfaz de usuario [39]. Genera archivos en formatos diferentes tales como: XML, PASCAL, VOC, YOLO y CreateML. Con esta herramienta se realizó el doble etiquetado en las imágenes del dataset y se exportaron en formato XML.

Google Colab [10]

Por último, el entorno de desarrollo (IDE) utilizado en este proyecto ha sido Google Colab [40], un servicio en la nube que gracias a sus servicios de GPU's, TPU's y RAM permite el desarrollo de aplicaciones de aprendizaje automático. Estos proyectos requieren de grandes recursos tanto software como hardware, por lo que Google Colab se ha convertido en una herramienta útil y llamativa gracias a que esta tecnología los ofrece. También hay que destacar su comodidad de uso debido a que no se necesita realizar una configuración del entorno y, además, viene con todos los paquetes preinstalados mencionados anteriormente, por lo que solo basta con su importación para poder comenzar a usarlos.

[7] Matplotlib: https://matplotlib.org/
[8] OpenCV: https://opencv.org/
[9] LabelImg: https://github.com/heartexlabs/labelImg
[10] Google Colab: https://colab.research.google.com/

Capítulo 5

COMPETENCIAS ESPECÍFICAS CUBIERTAS

CP04: *Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación*

CP05: *Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.*

Capítulo 6

DESARROLLO

En este capítulo se explica el desarrollo realizado para la detección automática del Signo de Frank. Se hará un recorrido por todas las fases que ha conllevado el desarrollo de este trabajo, desde la obtención del banco de imágenes de estudio, su etiquetado, el aumento de datos, la implementación de los modelos y por último la evaluación y comparación de los resultados.

6.1 Dataset de orejas con DELC y orejas sanas

La primera fase de este trabajo fue la búsqueda de un dataset mínimamente rentable para realizar el estudio. Normalmente, para este tipo de trabajo donde se necesita de un banco de imágenes, el sitio web Kaggle tiene a disposición cientos de datasets públicos con miles de imágenes cada uno, sin embargo, se hizo una búsqueda en diferentes sitios y no se tiene conocimiento de que exista un dataset de imágenes de orejas con el Signo de Frank, por lo que esta primera fase, que normalmente toma poco tiempo, hubo que realizarla manualmente haciendo una exhaustiva exploración en múltiples motores de búsqueda, descargando las que mejor resolución tuvieran. Se pudo recolectar un total de 342 imágenes, de las cuales se calculó la media y desviación estándar necesarios para poder extraer las imágenes de orejas sanas que tengan un tamaño que se adapte a esas medias permitiendo que el conjunto de datos tenga un tamaño uniforme.

Para la obtención de un dataset con imágenes de orejas sanas o sin signos visibles, se hizo uso del dataset AWE (*Annotated Web Ears*) [29]. Este dataset contiene unas 1000 imágenes de orejas, un total de 373 se adaptaron a la media y desviación estándar obtenidos del subconjunto positivo.

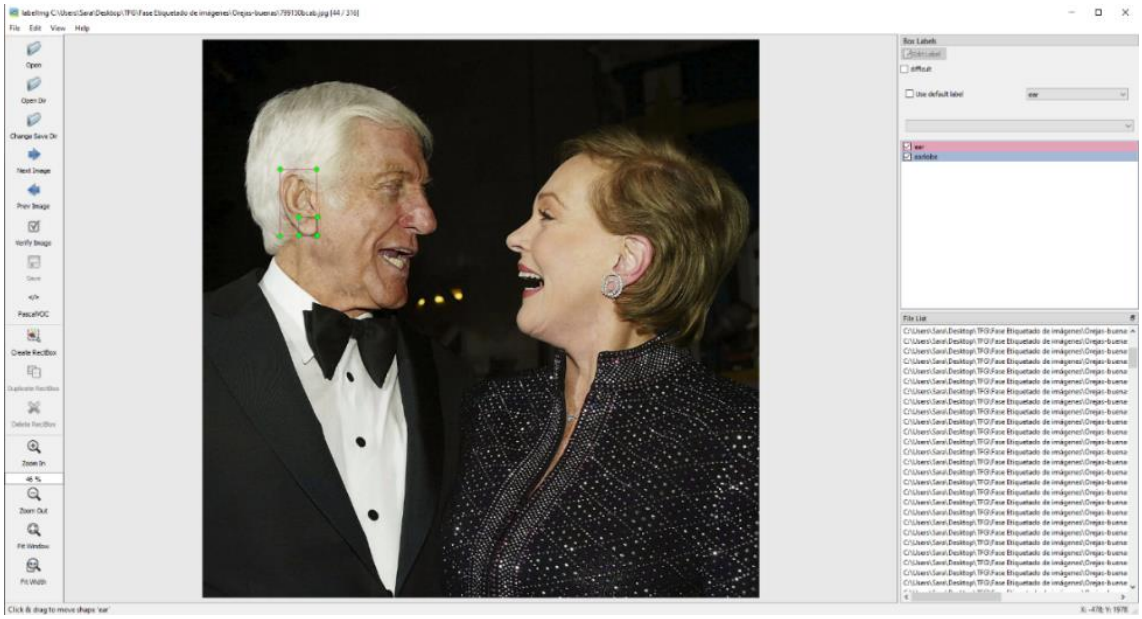
Con ambos subconjuntos obtenidos, se ha creado un dataset de muestras negativas y positivas de un total de 715 imágenes. Posteriormente se le aplica un aumento de datos que permite la ampliación del dataset, junto con distintas variaciones para el estudio posterior de las mismas.

6.1.2 Etiquetado de imágenes

La siguiente fase consiste en realizar el etiquetado o anotación de imágenes utilizando la aplicación LabelImg que se muestra en la Figura 6-1. Este paso consiste en concretar la posición en la que se encuentra el objeto mediante un cuadro delimitador y etiquetarlo con el nombre de la clase correspondiente, en la Figura 6-2 se pueden visionar un par de ejemplos. Por cada imagen etiquetada se genera su correspondiente archivo en formato XML donde se recoge toda la información de la imagen (nombre, altura, anchura, clase, etc. (ver Figura 6-3). Una vez

completada la anotación, se realizó el procesamiento de las imágenes mediante un script creado en Python, donde se leen los ficheros xml generados y a partir de las coordenadas obtenidas de cada imagen, se recortan y guardan en la carpeta que posteriormente será la que se use para el entrenamiento.

Figura 6-1. Interfaz de la aplicación LabelImg



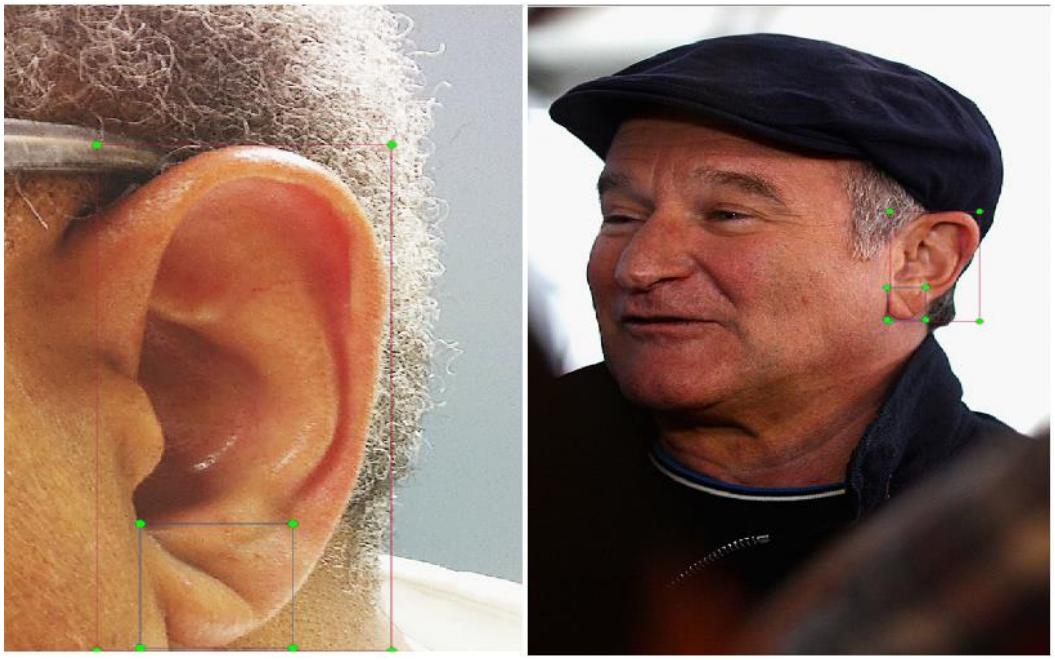


Figura 6-2. Ejemplos de etiquetado

```xml
<annotation>
    <folder>Frank_Sign_Ears</folder>
    <filename>0ccfb643a6.jpg</filename>
    <path>C:\Users\User\Documents\TFT\Frank_Sign_Ears\0ccfb643a6.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>368</width>
        <height>530</height>
        <depth>1</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>ear</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>44</xmin>
            <ymin>44</ymin>
            <xmax>307</xmax>
            <ymax>493</ymax>
        </bndbox>
    </object>
```

Figura 6-3. Fichero XML de una imagen etiquetada

6.1.3 Aumento de datos

El aumento de datos o *data augmentation* es una técnica que permite incrementar y transformar la cantidad total de imágenes que se usarán en el entrenamiento, este incremento se debe a que esta técnica realiza cierto número de copias **modificadas** de una imagen original. A la hora de entrenar un modelo, el aumento de datos permite mejorar la precisión y reducir el *overfitting,* fenómeno que puede suceder cuando el conjunto de datos es muy limitado y la red no aprende a clasificar imágenes entrantes nuevas que no sean con las que ha entrenado.

Se crearon dos configuraciones diferentes de aumento de datos para poder comparar los resultados de las redes en base a las distintas transformaciones, por cada transformación se obtuvieron 5 copias de cada imagen.

Una transformación *soft (*ver Figura 6-4) compuesta por:

- **Rotate(limit=30):** ángulo de rotación de 30º
- **RandomBrightness(limit=0.1):** aplica brillo y contraste aleatoriamente con un rango de factor cambiante de [-0.1, 0.1]
- **RandomContrast(limit=0.3):** aplica contraste aleatoriamente con un rango de factor cambiante de [-0.3, 0.3]

Una de las cinco imágenes generadas presentará estas modificaciones:

- **HueSaturationValue(p=0.5):** cambia aleatoriamente el tono, la saturación y el valor de la imagen con una probabilidad de aplicación del 50%

- **RGBShift(p=0.5):** Cambia los valores de cada canal de la imagen aleatoriamente con una probabilidad del 50%.
- **RandomBrightnessContrast(p=0.5):** aplica brillo y contraste aleatoriamente con una probabilidad del 50%

Y otra transformación *hard* (ver Figura 6-5) compuesta por:

- **RandomBrightness(limit=0.1):** aplica brillo y contraste aleatoriamente con un rango de factor cambiante de [-0.1, 0.1]
- **RandomContrast(limit=0.4):** aplica contraste aleatoriamente con un rango de factor cambiante de [-0.4, 0.4]
- **HorizontalFlip():** voltea la imagen de entrada horizontalmente alrededor del eje y.
- **ShiftScaleRotate(shift_limit=(0.05, 0.05), rotate_limit(-30, 30)):** aplica rango de escala, de rotación y cambio de altura y anchura.

Una de las cinco imágenes generadas presentará estas modificaciones:

- **HueSaturationValue(p=0.6)**: cambia aleatoriamente el tono, la saturación y el valor de la imagen con una probabilidad de aplicación del 60%
- **RGBShift(p=0.6):** Cambia los valores de cada canal de la imagen aleatoriamente con una probabilidad del 60%.
- **RandomBrightnessContrast(p=0.6):** aplica brillo y contraste aleatoriamente con una probabilidad del 60%

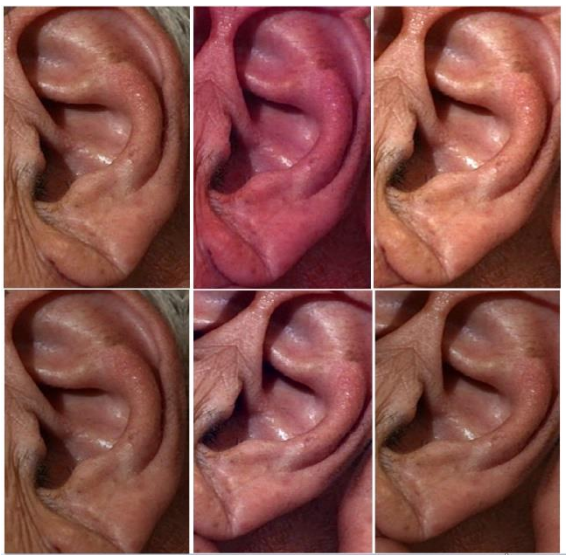


Figura 6-4. Transformación Soft

Figura 6-5. Transformación Hard

6.1.4 División del conjunto de imágenes

Una vez aplicado el aumento de datos, el siguiente paso es el proceso de agrupación de las imágenes para poder entrenar y validar los modelos. Esto se conoce como *Split* o separado de lotes y consiste en crear tres grupos llamados: lote de entrenamiento, validación y test. Estos tres lotes se extraen del total de las imágenes del dataset con el aumento de datos.

El conjunto de datos de entrenamiento es el que se utiliza para entrenar a las redes a partir de sus características y etiquetas permitiendo ajustar sus pesos. Gracias a este lote, la red aprende a detectar los patrones de identificación del objeto.

El conjunto de datos de validación el que permite confirmar si el entrenamiento se está ajustando a la realidad. Este conjunto permite la comprobación del comportamiento de la red en base a su entrenamiento y refleja si la línea de aprendizaje es óptima o no.

El conjunto de datos de prueba es el lote que se deja para el final, pues es el que se utiliza para analizar el rendimiento de una red que ya ha sido entrenada. El modelo nunca ha estudiado este conjunto de datos, por tanto, al realizar una predicción sobre este, permite comprobar el ajuste a las clases reales, pudiendo observar si realmente el modelo ha aprendido correctamente.

Teniendo un conjunto inicial de 715 imágenes, aplicando el aumento de datos se obtuvo un total de 4290. Posteriormente se realizó la división del dataset en los distintos conjuntos de datos, véase la Figura 6-6:

- Para el lote de **entrenamiento** se asignó un total de 2299 imágenes (53.60%)
- Para el lote de **validación** se asignó un total de 1416 imágenes (33%)
- Para el lote de **testeo** se asignó un total de 575 imágenes (13.40%)

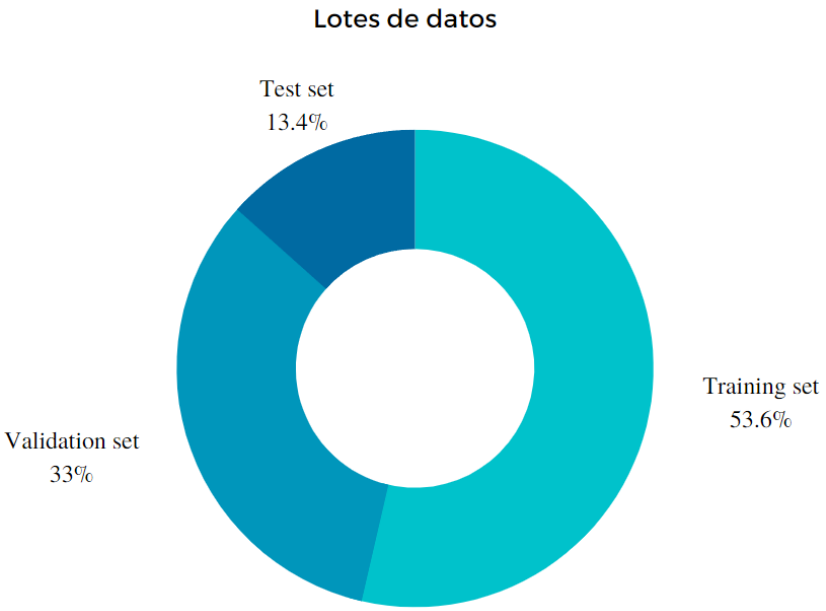


Figura 6-6. Gráfico conjunto de datos

6.2 Métodos y configuración utilizados en los modelos

Es preciso detallar los métodos que se han usado para la clasificación binaria de este proyecto que han permitido el desarrollo de los modelos pre-entrenados propuestos, se explicará el optimizador empleado, las configuraciones utilizadas y se mostrará un ejemplo de un modelo configurado.

Para que los modelos puedan desarrollar sus procesos de validación y entrenamiento, se han de compilar con unas configuraciones adaptadas al problema de estudio, en este caso, el proyecto se basa en una problemática de naturaleza binaria, pues se ha de detectar si una oreja presenta el signo de Frank o no. Por tanto, en las configuraciones de compilación se ha utilizado una pérdida probabilística llamada:

BinaryCrossEntropy:

Esta función de pérdida permite la comparación de cada probabilidad predicha con la salida de la clase real. Por lo que calcula el promedio negativo del logaritmo de las probabilidades predichas corregidas. La expresión matemática que la representa es la siguiente:

$$\boldsymbol{Loss} = -\frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i}}^{\boldsymbol{N}} \boldsymbol{y_i} \cdot \mathbf{log}\ \widehat{\boldsymbol{y}}_{\boldsymbol{i}} + (\mathbf{1} - \ \boldsymbol{y_i}) \cdot \ \mathbf{log}\ (\mathbf{1} - \ \widehat{\boldsymbol{y}}_{\boldsymbol{i}})$$

N: es el número de ejemplos

y_i: es la clase real

$\hat{y}_i$: es la clase predicha

Otra de las configuraciones de compilación importantes es el optimizador que se va a utilizar. Su función es la de optimizar los valores de los pesos para poder reducir el error que comete la red en base a la tasa de aprendizaje. En este caso se ha utilizado el optimizador:

Adam (*Adaptive Moment Estimation*):

Este algoritmo, desarrollado por D. Kingma *et al.* [27] [28], es de los más recientes creados y nace a partir de sus predecesores, llamados Momentum y RMSP. Adam engloba las ventajas de estos dos métodos de descenso de gradiente estocástico, por lo que esta función tiene un rendimiento superior. Se ha comprobado que Adam obtiene una convergencia más eficiente que otros métodos a la hora de clasificar, es fácil de usar y requiere menos memoria que otros algoritmos.

Los parámetros de configuración de Adam son los siguientes:

α: tasa de aprendizaje (por defecto: 0.001)

β_1: Tasa de caída exponencial para estimaciones del primer momento

β_2: Tasa de caída exponencial para estimaciones del segundo momento (por defecto: 0.999, recomendado no bajar ese valor en problemas de visión por computador, como es el caso de este proyecto).

epsilon: constante para evitar cualquier división por cero en la implementación. (Por defecto: 1e-7).

Función de activación *"Sigmoid":*

A la hora de configurar una red, se le pueden añadir ciertas capas para poder mejorar sus valores de entrenamiento y validación acordes a la problemática que se aborda. Una capa importante es la capa de salida, donde hay que establecer una función de activación apropiada para extraer las predicciones del modelo. También se puede hacer uso de diferentes funciones de activación, sin embargo, la última capa es la que debe tener una función adecuada.

En este proyecto, todos los modelos tienen una capa de salida con función de activación de tipo ***"Sigmoid"***, debido a que trabaja muy bien para problemas de clasificación binaria y da como resultado predicciones muy claras (0 y 1). Esta función es de las más utilizadas y matemáticamente se expresa de la siguiente forma:

$$\boldsymbol{f(x) = \frac{1}{1 + e^{-x}}}$$

6.2.1 Ejemplo de modelo pre-entrenado

Todos los modelos de este proyecto han empleado el aprendizaje por transferencia. Esto permite que el modelo no entrene desde cero, si no que ya obtiene valores a partir del entrenamiento empleado con un gran dataset de imágenes (***ImageNet***, en este caso), transfiriendo información y conocimientos a la red para el caso específico a estudiar.

Los modelos cuentan con una configuración similar, excepto en algunos donde se añadieron ciertas capas para mejorar su rendimiento. Un ejemplo de implementación de un modelo pre-entrenado se puede visionar en la Figura 6-7.

A la hora de importar el modelo, este se ha de configurar con unos parámetros de entrada. El primer parámetro, ***include_top,*** es un booleano que permite incluir o no, las capas *fully-connected*. Dado que se emplea el aprendizaje por transferencia, definido en el parámetro **weights,** lo más adecuado es no incluir esas capas, ya que están adaptadas al dataset *Imagenet* y lo que se quiere, es obtener las capas de salida específicas para este proyecto. Por tanto, se establece a ***false.*** Por último, el parámetro ***input_shape*** establece el número de píxeles y canales de los que se compone la imagen, en este caso 224x224 píxeles y 3 canales que representan el color, RGB. A continuación, se añaden ciertas capas al modelo para mejorar y equilibrar su entrenamiento y por último compilarlo con las configuraciones necesarias explicadas anteriormente.

```
Xception_model = tf.keras.applications.Xception(
    include_top=False, weights="imagenet", input_shape=(224, 224, 3)
)

model = Sequential()
model.add(Xception_model)
model.add(Flatten())
model.add(Dense(1024, activation="relu"))
model.add(Dense(1, activation="sigmoid"))


model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

Figura 6-7. Ejemplo de implementación de Modelo pre-entrenado

6.3 Resultados

En este apartado se reflejarán los resultados obtenidos por parte de los modelos pre-entrenados para la clasificación de la presencia del Signo de Frank. Con ello se representarán los valores de precisión de cada modelo y diferentes gráficas y métricas, haciendo una comparativa de cuáles han sido los mejores.

6.3.1 Resultados de la precisión (*accuracy*)

Este apartado reflejará los resultados de la precisión de los trece modelos estudiados, así como el entrenamiento y validación de estos. Primeramente, se mostrará una comparativa de todos los modelos, empleando los dos tipos de aumento de datos (véase la Tabla 6-1). La tabla abarca la precisión sobre los conjuntos de datos de validación y datos de prueba que han obtenido los modelos según la configuración de aumento de datos empleado y la mejor iteración del modelo durante su aprendizaje. Esto último se calcula gracias a la clase "*ModelCheckpoint*" proporcionada por Keras, cuya función permite ir guardando las mejores validaciones, por lo que al final se obtiene la que mejor ha dado el modelo independientemente de su trayectoria de precisión.

Del total de los 13 modelos estudiados, se han agrupado los cuatro que han obtenido los mejores porcentajes de precisión en las pruebas, entre el 95% - 98%, reconociendo estos valores como exitosos. Al tener dos configuraciones de aumento de datos distintos, la red de los modelos se ha comportado diferente, por lo que los mejores modelos de cada grupo difieren entre sí, en parte, porque ciertos modelos se han adaptado mejor a la configuración denominada "*Hard Augmentation*" y otros a la configuración *"Soft Augmentation*".

	Hard Augmentation			Soft Augmentation		
Modelos	**Best ite**	**Validation (%)**	**Test (%)**	**Best ite**	**Validation (%)**	**Test (%)**
Xception	0.979	97.76	**95.83**	0.992	95.10	94.09
VGG16	0.897	91.98	85.91	0.949	96.48	93.91
VGG19	0.902	93.50	88.35	0.944	95.15	92.70
Resnet50	0.927	88.14	84.70	0.967	98.14	**95.83**
Resnet101	0.892	93.75	88.17	0.947	97.46	94.78
Resnet152	0.892	93.45	85.74	0.964	97.81	95.13
MobileNet	0.979	95.73	**93.57**	0.994	98.74	**96.70**
Inception V3	0.955	97.06	**96.35**	0.984	98.88	**97.57**
DenseNet121	0.933	80.93	78.43	0.998	96.41	**95.48**
DenseNet169	0.932	93.15	90.96	0.974	88.65	88.17
DenseNet201	0.932	96.76	**95.13**	0.954	95.03	93.39
EfficientNetB0	0.969	52.49	56.17	0.874	52.66	56.17
EfficientNetB2	0.929	52.17	55.48	0.928	63.38	62.26

Tabla 6-1. Resultados Validation y Test accuracy

Antes de comenzar con el análisis de los modelos, es preciso explicar qué representan las gráficas que se verán a lo largo del proceso comparativo, estas gráficas son las llamadas gráficas de precisión *(accuracy)*. Representan el desarrollo de los procesos de entrenamiento y validación de la precisión de los modelos. La gráfica está compuesta por dos ejes, el eje Y representa los valores de precisión que se comprenden entre el 0.0 – 1.0 y el eje X representan las épocas o *Epoch*. Una época es básicamente un ciclo completo a través del conjunto de datos que se está entrenando. Las líneas de progreso que se ven plasmadas en estas gráficas son la representación de la precisión y la validación de esta (*accuracy* y *val_accuracy*).

Xception

Siguiendo con el orden de la Tabla 6-1, el modelo Xception es el que encabeza el análisis comparativo. Aunque tanto con el aumento de datos *Hard* como con el *Soft* se han obtenido unos resultados de predicción bastante altos, se puede apreciar como Xception ha obtenido casi un 98% de acierto en la configuración *Hard*. Sin embargo, la línea de progreso en el proceso de entrenamiento y validación que refleja el modelo usando este aumento de datos, muestra una inestabilidad significativa en algunos puntos, con picos de precisión descendentes muy pronunciados, llegando a decrecer en las épocas finales hasta un 0.4 y posteriormente, subiendo progresivamente hasta más de un 90% de precisión, por lo que su proceso de aprendizaje no es el más adecuado.

En cambio, el proceso de entrenamiento y validación que obtuvo el modelo con el aumento de datos *Soft,* aunque no llegó a tener el mejor porcentaje de precisión, de un 95%, sí tuvo una

ruta de aprendizaje más equilibrada y situándose la mayor parte del tiempo en valores aceptables.

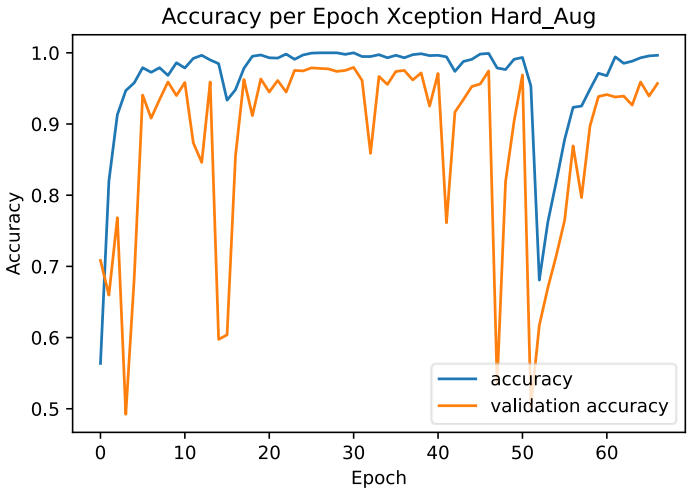


Figura 6-8. Gráfica precisión Xception Hard

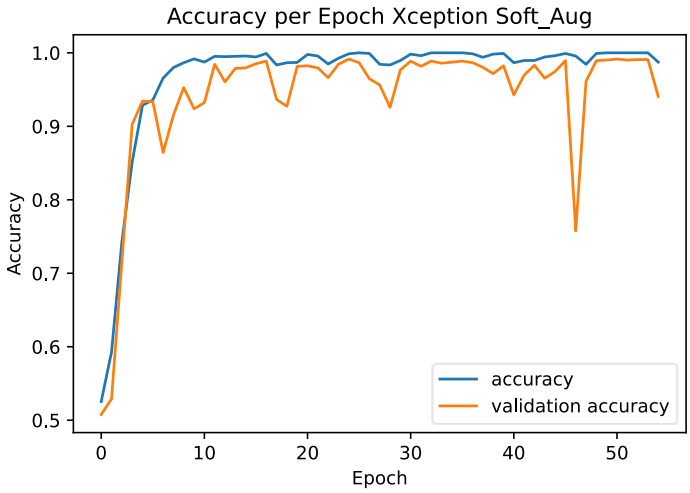


Figura 6-9. Gráfica precisión Xception Soft

VGG16 – VGG19

Normalmente, excepto algunos casos, a lo largo del proceso comparativo de los modelos, encontraremos cómo la configuración del aumento de datos *Soft* frente al *Hard,* conseguirá unos valores mayores de precisión. Esto es debido a que las imágenes sufren cambios mucho más leves permitiendo que la red tenga un aprendizaje más fácil, sin embargo, el aumento de datos *Hard,* tiene unas modificaciones un poco más variadas, permitendo que el modelo entrene con un dataset más diverso, y por consecuencia que el aprendizaje sea más robusto. Como se ha dicho anteriormente, hay casos en donde el modelo se adapta mucho mejor al dataset *Hard*.

Comenzando con el análisis de la arquitectura VGG16, se han conseguido unas precisiones de un 91.98% para el dataset *Hard* y un 96.48% para el *Soft.* Estos modelos en general se han presentado muy inestables en cuanto a la validación de la precisión. En las figuras 6-10 y 6-11 se puede observar cómo esas oscilaciones en la validación del entrenamiento son muy continuadas y con picos que van desde menos del 0.5 hasta el 0.9 en muchos casos. Esto se conoce como *overfitting,* significa que a la red le está costando reconocer los patrones de clasificación generales y tiene que aprender nuevamente produciendo así esas caídas de valores en la precisión.

Las explicaciones que se pueden sacar en base a este inconveniente pueden ser desde un cambio en la función de activación, el tamaño del dataset utilizado, que puede que sea muy pequeño para este tipo de modelo o una elección de la tasa de aprendizaje más personalizada, ya que puede que los valores por defecto no sean los más convenientes. Otra explicación es que los modelos que son más complejos (los que poseen más parámetros) tienen una tendencia a sobreajustar más que los modelos simples, esto tiene sentido teniendo en cuenta que la

arquitectura VGG es la que más parámetros tiene de todos los modelos preentrenados estudiados en este proyecto.

El modelo entrenado con el Dataset *Soft,* a pesar de las oscilaciones que sufre, consigue obtener una mejor precisión y en las últimas épocas del entrenamiento da pistas a que puede conseguirse una estabilización de la red.

Comparando ahora los modelos VGG19, en vistas generales se puede observar que al igual que los modelos VGG16, los procesos de entrenamiento y validación son demasiado inestables como para considerar que son exitosas a pesar de que tienen unos valores de precisión altos. Un 93.5% con el dataset *Hard* y un 95.15% para el *Soft,* nuevamente se puede decir que el aumento de datos *Soft* es la ganadora en precisión, sin embargo, las oscilaciones que sufre a lo largo de las épocas son más erráticas. Mientras, el modelo *Hard* presenta oscilaciones con intervalos de descensos y ascensos más cortos, dando una sensación de "estabilidad" en su ruta de aprendizaje.

Si se hace una comparación generalizada, toda la estructura VGG (ambos modelos 16 y 19), puede verse cómo los procesos de entrenamiento y validación de los modelos VGG19 son más inestables que las de VGG16, por lo que, como conclusión, esta última es la que ha ofrecido mejor relación precisión-aprendizaje de las redes.

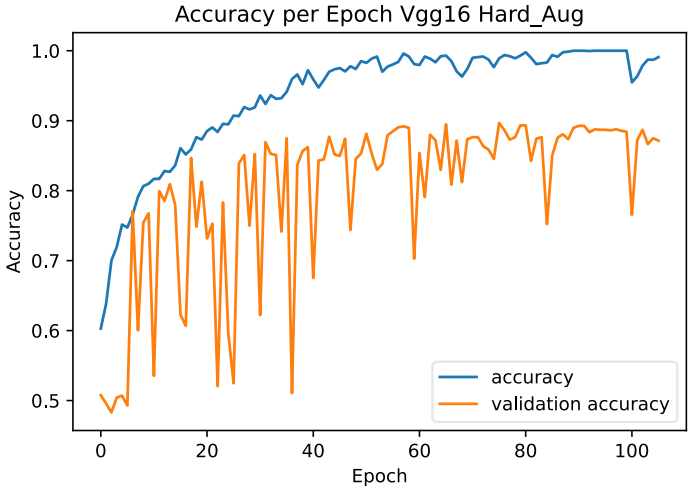


Figura 6-10. Gráfica precisión VGG16 Hard

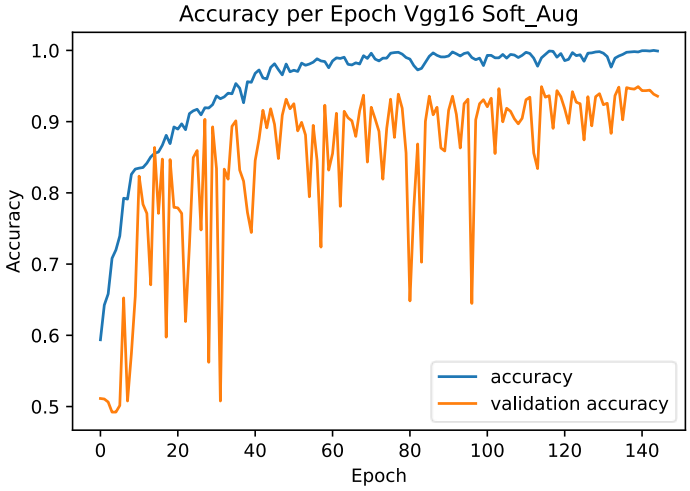


Figura 6-11. Gráfica precisión VGG16 Soft

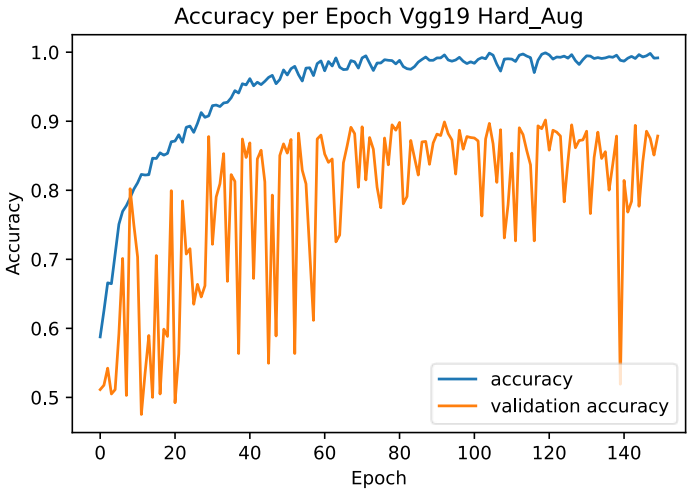


Figura 6-12. Gráfica precisión VGG19 Hard

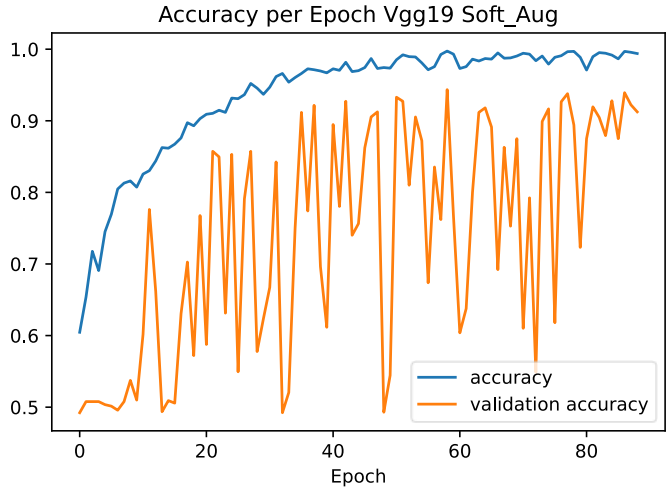


Figura 6-13. Gráfica precisión VGG19 Soft

ResNet

En este proyecto, se han estudiado varios modelos de esta arquitectura, concretamente los modelos ResNet50, ResNet101 y ResNet152. Siguiendo el orden de estos, para el dataset *Soft* se ha obtenido unos porcentajes del 98.13%, 97.46% y 97.81% respectivamente y para el dataset *Hard*, un 84.31%, 93.75% y 93.45%. Para hacer una comparación más sustancial de los modelos, se expondrán los mejores modelos de ambos datasets en base a la precisión obtenida y los mejores en base a su línea de aprendizaje.

A simple vista, se puede ver cómo los entrenamientos presentan muchas oscilaciones, no obstante, la gráfica de la Figura 6-15 presenta unas oscilaciones que no bajan del valor 0.7 excepto un pico en la época 80, a diferencia de ResNet101 (Figura 6-14), donde su precisión es mucho menos estable dado que tiene unos 3 picos drásticos que bajan hasta el 0.5. Hay que tener en cuenta que ambos modelos no tienen el mismo número de épocas desarrolladas, pero podemos comparar el rango 0-60 en la que claramente el modelo ResNet50 tiene unas oscilaciones menos pronunciadas. Luego la validación de ambas, se observan mucho más erráticas en comparación a sus precisiones, sin embargo, la que mejor se desarrolla a lo largo del entrenamiento es la red ResNet50 para el dataset *Soft,* dado que se nota cierta mejoría a partir de la etapa 80 y cuenta con menos picos de descensos bruscos que el modelo ResNet101 para el dataset *Hard.*

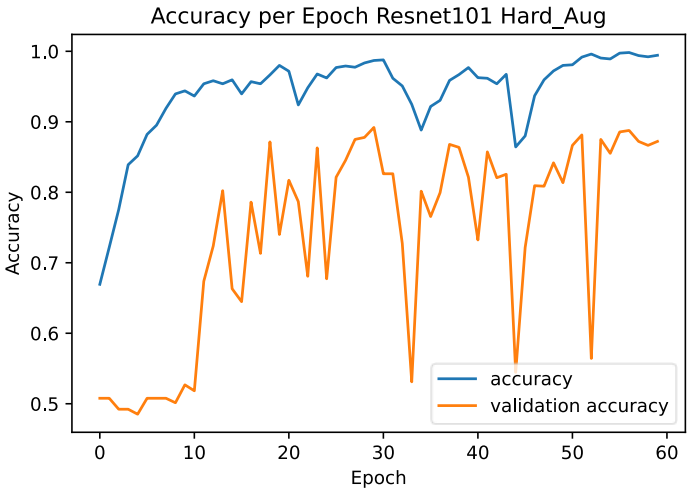


Figura 6-14. Gráfica precisión ResNet101 Hard

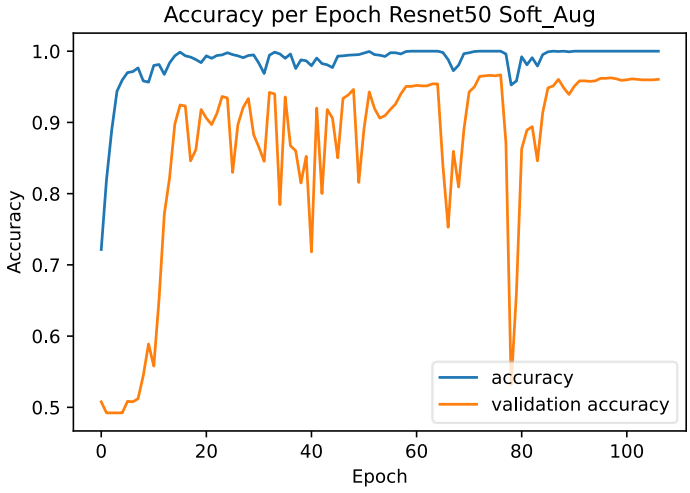


Figura 6-15. Gráfica precisión ResNet50 Soft

La gráfica que se muestra a continuación es la que ha desarrollado un aprendizaje más estable a lo largo del proceso de entrenamiento de todas las redes de la arquitectura en cuestión. Nuevamente es el modelo ResNet101 para el conjunto de datos *Soft*. Aunque la precisión no es de las más altas, esta red ha tenido un comportamiento mucho más equilibrado situándose durante un rango considerable de épocas en altos porcentajes, que las redes descritas en el párrafo anterior. Esto no quita que, aun así, todas las redes de la arquitectura ResNet hayan sufrido muchas oscilaciones en sus procesos de validación, pero esta última ha mostrado un comportamiento con tramos más estables y menos picos de descenso.

En la Figura 6-16, que representa el modelo ResNet101 se puede visionar cómo desde la época 0 hasta la 50, presenta un comportamiento inestable con presencia de picos de descensos drásticos en la época 40 que llega hasta el 0.4. Por otro lado, desde la época 50 hasta la 120, el modelo presenta un comportamiento totalmente contrario a la primera mitad de la gráfica, donde la precisión, así como su validación se mantienen muy estables, por encima del umbral del 90%, viéndose prácticamente una línea recta, algo positivo de presenciar. Como conclusión, el aprendizaje general que ha presentado este modelo a pesar de comenzar con una inestabilidad creciente termina muy estable, con posibilidad de aprender muy positivamente para obtener unos resultados muy interesantes, a conclusión de la autora, esta red es la que mejor se ha desempeñado, dado que tanto precisión como aprendizaje ha sido de las mejores.

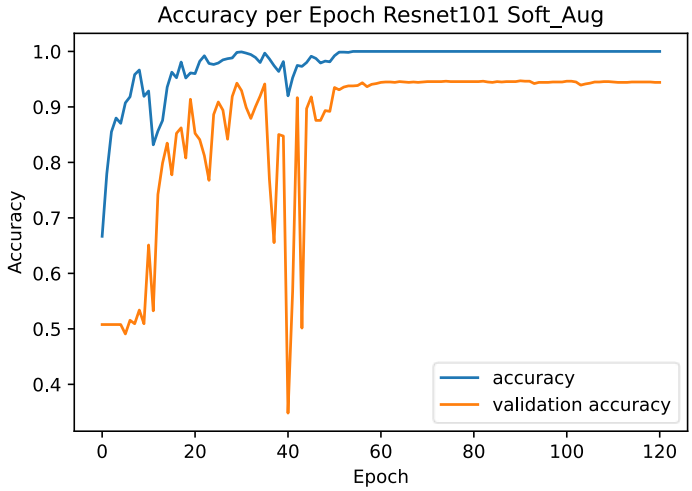


Figura 6-16. Gráfica precisión ResNet101 Soft

MobileNet

MobileNet es uno de los modelos que mejores valores ha obtenido para el dataset *Soft,* pues se encuentra en el segundo puesto con un porcentaje de precisión del 98.74%. También se encuentra entre los mejores para el dataset *Hard* pero en el puesto número cuatro, con una precisión del 95.73%. En la Figura 6-18 se observa cómo los procesos de entrenamiento y validación son bastante similares, entendiendo esto como un buen aprendizaje del modelo. Se puede ver como el aprendizaje se mantiene estable en un período de hasta 25 épocas aproximadamente situado cerca del 0.1, siendo esta de las mejores redes que se ha obtenido a lo largo del estudio. Lo único a resaltar es la oscilación que sufre en la época 50ª con una bajada de precisión hasta el 0.8, pero seguidamente se recupera y vuelve a incrementar, vemos como esta red apenas sufre oscilaciones preocupantes y se mantiene en casi todo momento por encima del 0.9.

En el caso de la Figura 6-17, se puede apreciar también un aprendizaje positivo pues los valores no bajan del 0.8, a comparación con el modelo de la Figura 6-18, se aprecia la diferencia de cantidad oscilaciones que sufre la red, así como la continuidad de estas y la cantidad de épocas que necesitó el modelo para llegar a su mejor época, no obstante, se considera que ambas

gráficas representan un aprendizaje bastante óptimo y por consecuencia de los mejores modelos estudiados.

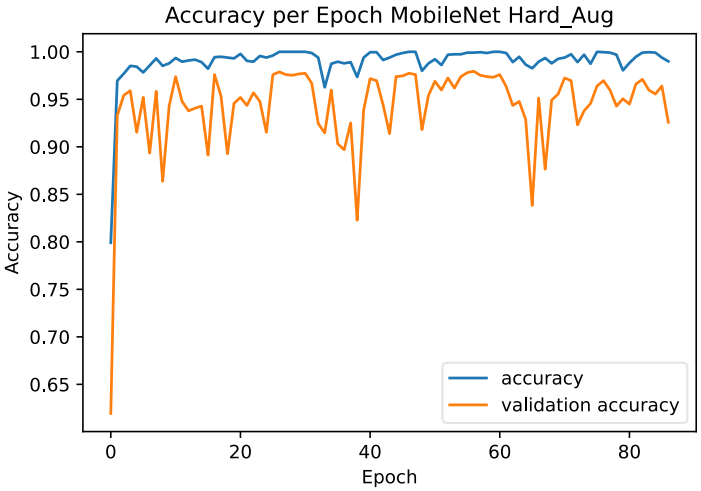


Figura 6-17. Gráfica precisión MobileNet Hard

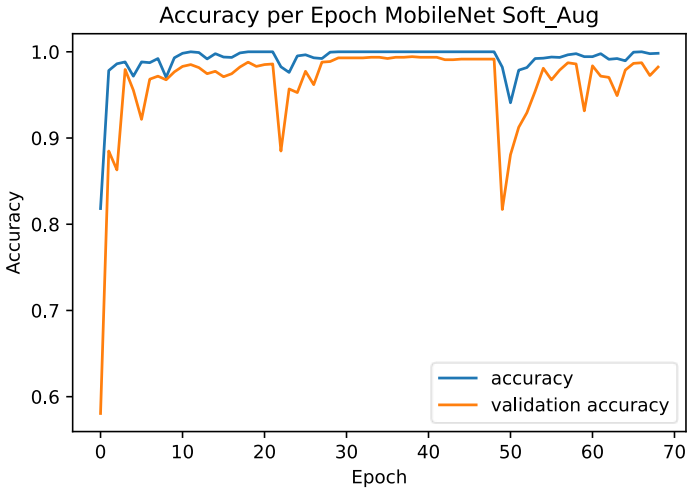


Figura 6-18. Gráfica precisión MobileNet Soft

InceptionV3

Este modelo también se encuentra entre los mejores de ambos Datasets, más concretamente, en el primer puesto para el dataset *Soft* y en el segundo puesto para el *Hard,* con unos porcentajes de precisión de un 98,88% y un 97,06% respectivamente. Como se puede observar en la Figura 6-20, la trayectoria de validación de la precisión (*val_accuracy*) es bastante prometedora pues a pesar de tener varias oscilaciones, estas no bajan del 0.7 de precisión y la ruta de aprendizaje tiende a llegar casi al 1.0 al igual que vemos con su línea de precisión (*accuracy*) que tiene una convergencia que se mantiene en todo momento cerca del 100%.

En cuanto a la Figura 6-19, que representa el aprendizaje del modelo con el dataset *Hard*, se observa como la validación de la precisión sufre sobreajustes más pronunciados, descendiendo por debajo del 0.6 hasta en dos ocasiones, cerca de la 5ª época y la 55ª época, sin embargo, si al modelo se le hubieran ampliado las épocas de entrenamiento probablemente se hubiera obtenido una precisión más alta.

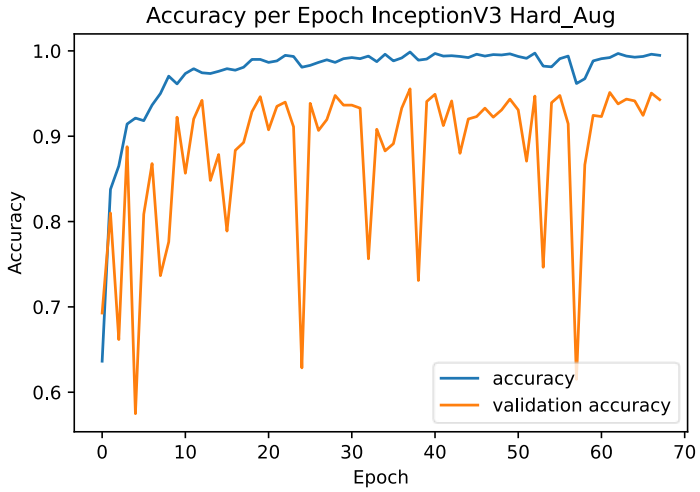


Figura 6-19. Gráfica precisión InceptionV3 Hard

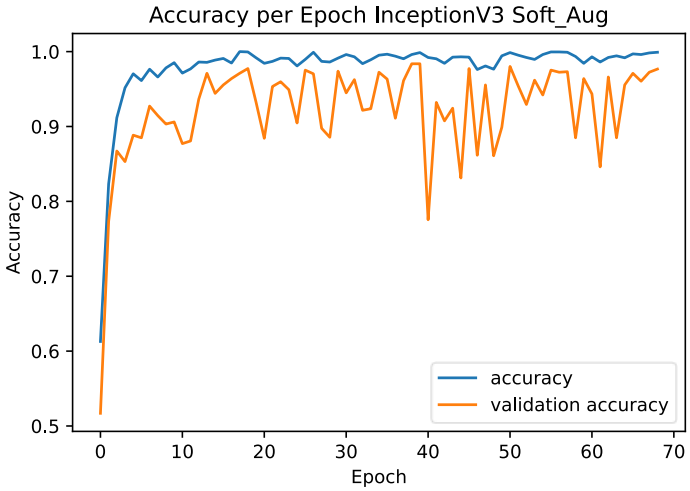


Figura 6-20. Gráfica precisión InceptionV3 Soft

DenseNet

Los modelos que se han estudiado de esta arquitectura han sido DenseNet121, DenseNet169 y DenseNet201. De estas redes se han obtenido unos porcentajes de precisión entre el 80% hasta casi el 97%. El modelo DenseNet201 utilizado para el dataset *Hard,* ha obtenido la peor precisión de todos con un 80.93%. Este se puede observar en la Figura 6-21, donde la validación de la precisión es muy irregular y distante a la precisión entrenada. Esta también dista mucho de ser una precisión aceptable pues tiene bastantes picos descendentes y no se observa ninguna tendencia a estabilizarse. Por otro lado, el mejor modelo que se obtuvo pertenece también al dataset *Hard* con un 96.76% (ver Figura 6-22). Si se comparan ambos modelos, se aprecia como DenseNet201 tiene menos picos drásticos que DenseNet121 entre las épocas 0-60.

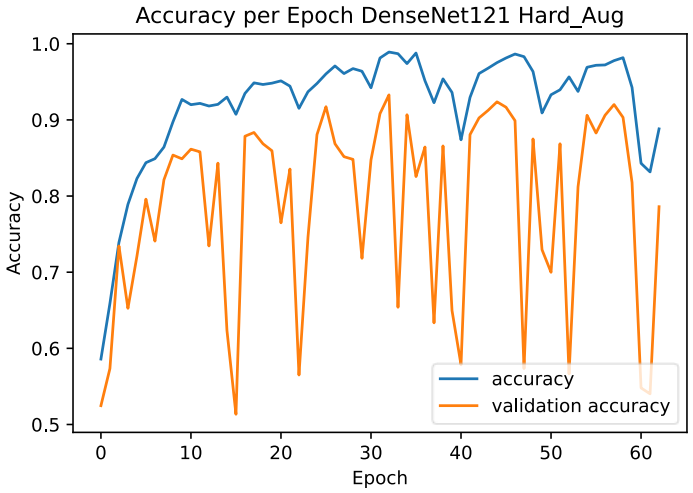


Figura 6-21. Gráfica precisión DenseNet121 Hard

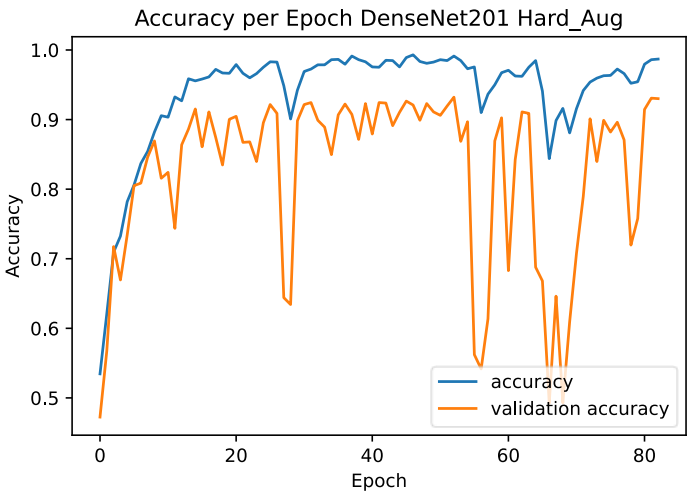


Figura 6-22. Gráfica precisión DenseNet201 Hard

Si tenemos en cuenta la ruta de entrenamiento y validación, el modelo que mejor desempeño realizó fue DenseNet201 para el conjunto de datos *Soft* (ver Figura 6-23). Se aprecia un aumento constante de la precisión y la validación a pesar de sufrir bastantes oscilaciones, conforme van pasando las épocas, los picos tienen un rango de descenso menor hasta terminar en la época 70 con una tendencia a estabilizarse.

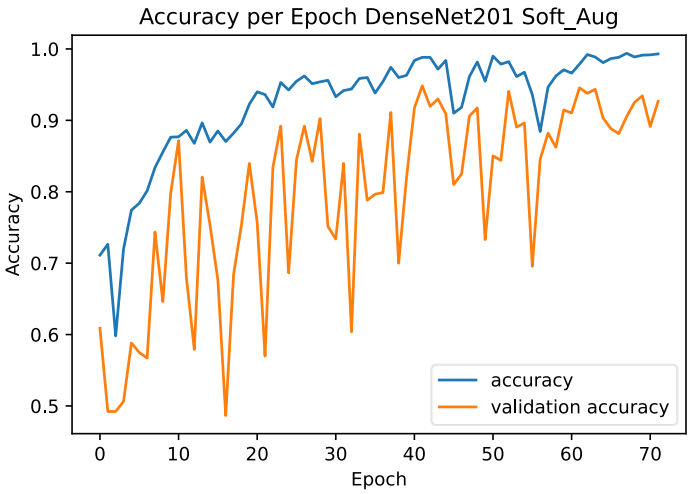


Figura 6-23.Gráfica precision DenseNet201 Soft

EfficientNet

Esta arquitectura es la última sometida a estudio y los modelos escogidos fueron EfficientNetB0 y EfficientNetB2. Estos cuentan con los porcentajes de precisión más bajos de toda la tabla de modelos estudiados. Incluso se descartan del núcleo de porcentajes aceptables, pues sus valores se engloban entre el 52% y 64%. Dado que todas las redes han dado unos resultados muy bajos, se pondrá un ejemplo con el aumento de datos *Soft* y otro del *Hard* para apreciar sus procesos de validación, aunque la predisposición de estas redes claramente será muy errática.

La Figura 6-24 muestra el modelo EfficientNetB0 con una precisión del 52.49% y el modelo EfficientNetB2 con una precisión del 63.38%. Se puede visionar como ambas gráficas tienen sus líneas de precisión y validación completamente opuestas. Esto es una representación de cómo es un modelo nada preciso dado que obtener un valor en torno al 50% de precisión indica una aleatoriedad a la hora de clasificar la problemática estudiada.

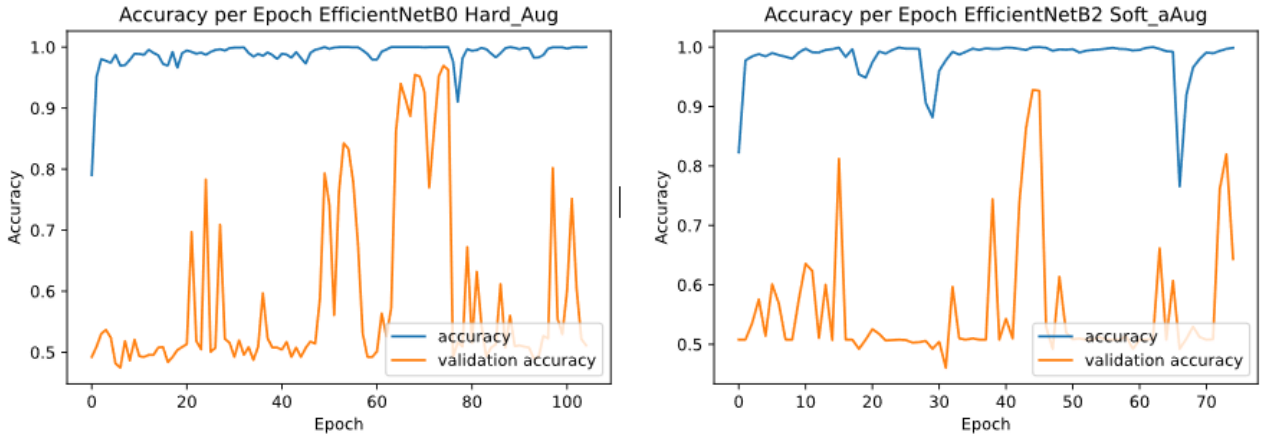


Figura 6-24. Modelos EfficientNetB0-B2 Hard y Soft

Para este análisis de la precisión, se concluye que, al haber observado el comportamiento de los modelos en torno a los conjuntos de datos estudiados, los modelos más efectivos son Inception, MobileNet y Xception, dado que estos fueron los que obtuvieron porcentajes entre un 97% y 99%. Sumando también sus procesos de entrenamiento y validación, pues fueron los más estables y constantes en valores óptimos sin apenas oscilaciones drásticas.

Seguido de ellos, las arquitecturas VGG y ResNet aunque mostraron buenos resultados, destacaron por su inestabilidad, no obstante, es probable que de estas redes se puedan obtener mejores resultados modificando sus hiperparámetros y ajustándose a un dataset más adecuado para ellas.

En la clasificación, la peor arquitectura fue EfficienNet, quedan descartadas totalmente como modelos óptimos para esta clasificación, teniendo en cuenta que el umbral de precisión aceptable para asegurar que un modelo es efectivo generalmente está en el 70%.

Finalizando esta sección, finalmente se han recogido en dos tablas los mejores modelos de ambos datasets (Tabla , Tabla) para el lote de pruebas. En las próximas secciones se explicarán distintas métricas que se han obtenido a partir de esta selección de modelos.

SOFT AUGMENTATION	
Models	**Predict %**
Inception V3	**97.57**
MobileNet	**96.70**
ResNet50	**95.83**
DenseNet121	**95.48**

Tabla 6-2. Clasificación mejores modelos Soft

HARD AUGMENTATION	
Models	**Predict %**
Inception V3	**96.35**
Xception	**95.83**
DenseNet201	**95.13**
MobileNet	**93.57**

Tabla 6-3. Clasificación mejores modelos Hard

6.3.2 Análisis de la clasificación de los modelos y su rendimiento

Para esta sección, se hará un análisis de la calidad de la clasificación del signo de Frank que han realizado los mejores modelos de este proyecto, así como una evaluación de su rendimiento empleando distintas métricas de clasificación (Matriz de confusión, Precision, Recall, F1-Score).

Matriz de confusión

La matriz de confusión es una tabla que permite visualizar el número de aciertos y errores para cada clase que produce el modelo a la hora de predecir las imágenes del conjunto de datos. El término “confusión” se debe a que, gracias a esta matriz, se puede conocer la cantidad de veces que el modelo está confundiendo una clase con otra. Por consecuencia, saber cuál es la clase que más problemas le está dando al modelo para reconocerla. Gracias a esto se puede saber la precisión exacta que tiene el modelo para predecir una clase correctamente.

En la Figura 6-25 se puede visionar cómo es una matriz de confusión y la terminología que contiene, que se explicará a continuación. La tabla se compone por el eje Y, que representa las clases reales y el eje X, representada por las clases predichas por el modelo. Dentro de la tabla

se observan unas abreviaturas que cogerán importancia durante todo este análisis, pues gracias a ellas podremos sacar diversas métricas que permitan la evaluación de los modelos estudiados.

Si definimos las clases de este proyecto como:

- **Positive:** clase que representan las orejas con el signo de Frank
- **Negative:** clase que representan las orejas sanas

Entonces, la terminología asociada se define como:

- **TP *(True Positive):*** cuando la clase real es positiva y la clase predicha es positiva
- **TN *(True Negative):*** cuando la clase real es negativa y la clase predicha es negativa
- **FP *(False Positive):*** cuando la clase real es negativa y la clase predicha es positiva
- **FN *(False Negative):*** cuando la clase real es positiva y la clase predicha negativa

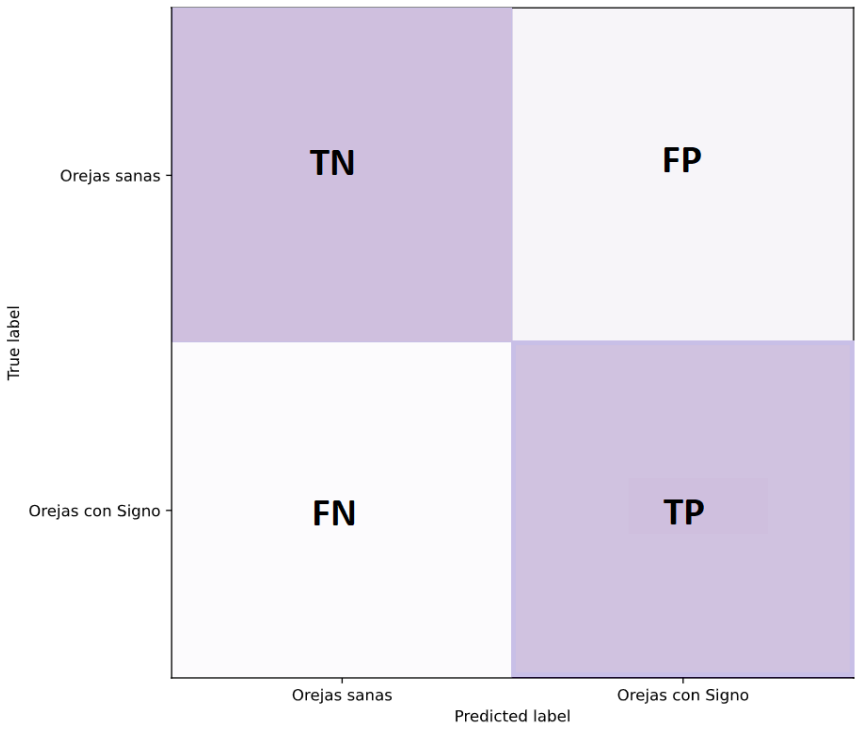


Figura 6-25. Matriz de confusión

A continuación, se representarán las matrices de confusión de los 4 mejores modelos para el conjunto de datos *Soft.* Según la descripción de la matriz de confusión explicado en el apartado anterior, la representación de la calidad de la clasificación de un modelo que sea excelente se representaría de manera que los valores TP y TN tuvieran el total de imágenes del conjunto de datos y los valores FP y FN estuvieran a cero. Esto formaría una diagonal en la matriz que se apreciaría como una buena clasificación por parte del modelo. En la Figura 6-26, Figura 6-27, Figura 6-28 y Figura 6-29 se pueden observar esas diagonales, no obstante, evidentemente no son perfectas, dado que en las celdas FP y FN hay ciertos valores que representan el número de veces que la red ha fallado a la hora de predecir ciertas imágenes.

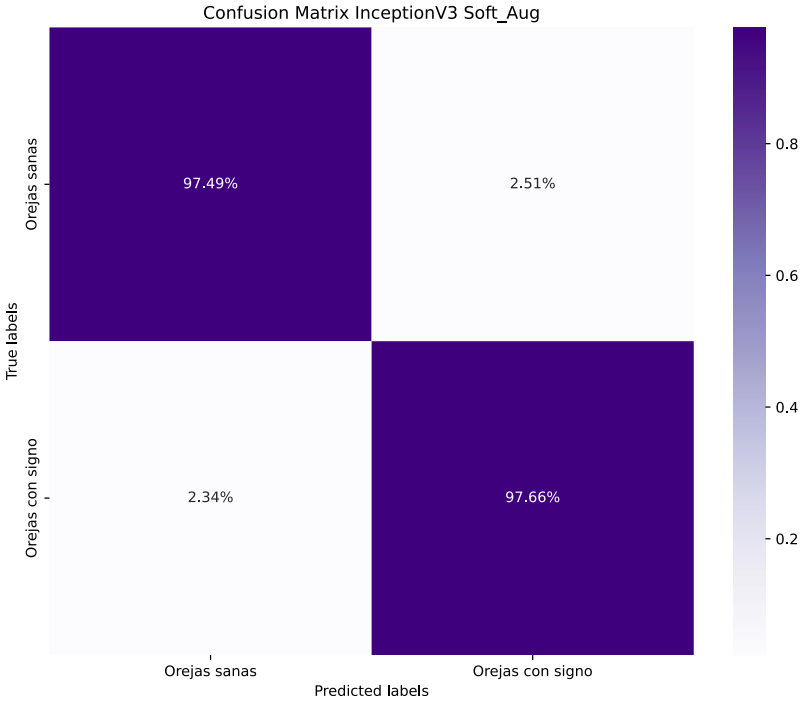


Figura 6-26. Matriz de confusión InceptionV3 Soft

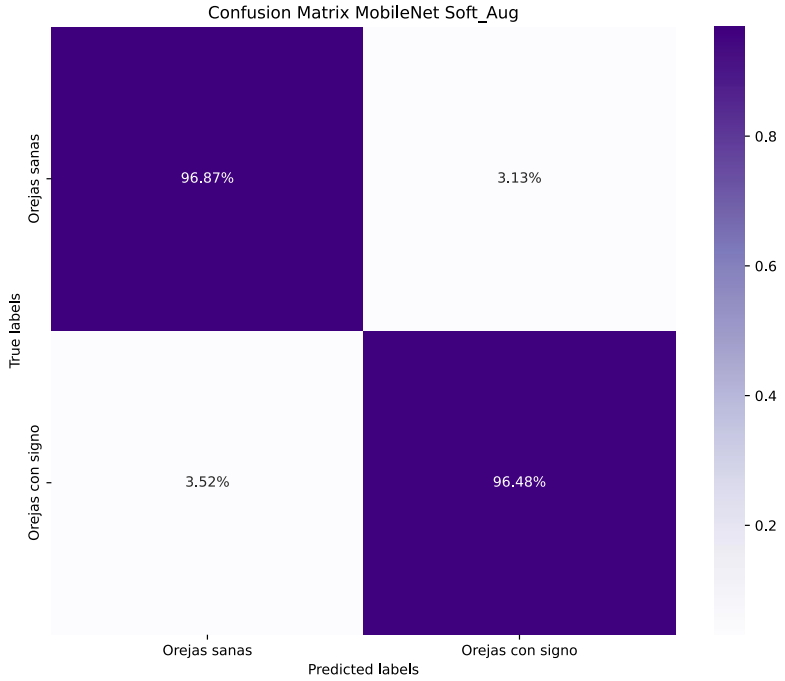


Figura 6-27. Matriz de confusión MobileNet Soft

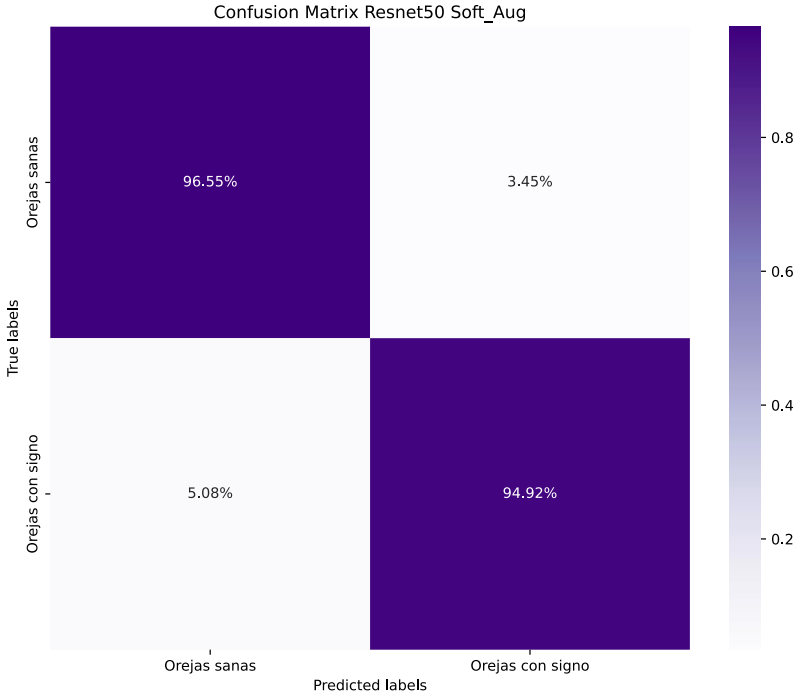


Figura 6-28. Matriz de confusión ResNet50 Soft

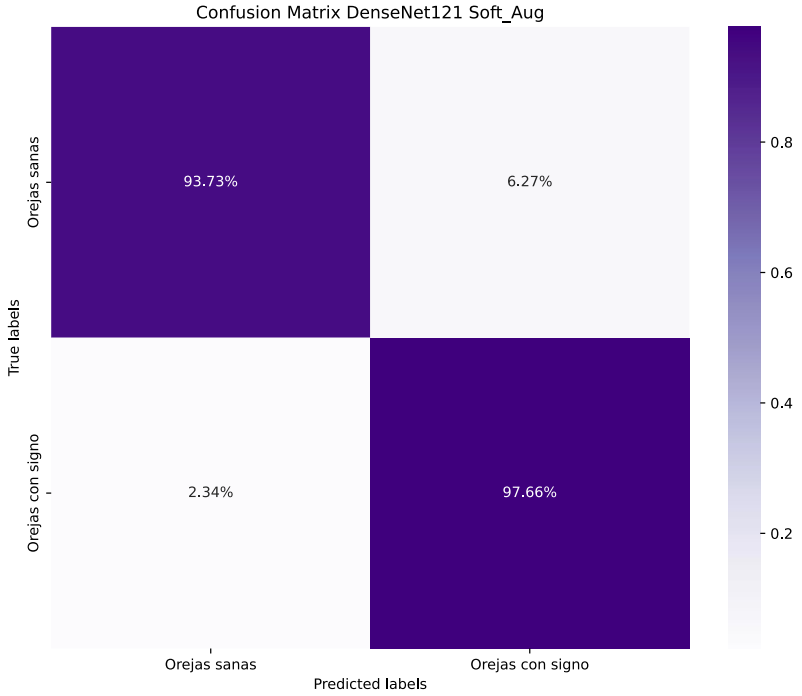


Figura 6-29. Matriz de confusión DenseNet121 Soft

Para el conjunto de datos *Hard*, se han obtenido las siguientes matrices de confusión en base a sus 4 mejores modelos. Nuevamente se han obtenido unos valores TN y TP bastante altos y unos valores FP y FN bastante bajos, eso quiere decir que los modelos han desempeñado una clasificacion exitosa, vemos como la Figura 6-30 y Figura 6-31 son las que más imágenes positivas han predicho correctamente y menos falsos positivos han detectado.

Se puede observar cómo estos modelos también han detectado correctamente un mayor numero de casos negativos que positivos, aun así la diferencia radica en un pequeño porcentaje. En general, las matrices de confusión del conjunto de datos *Hard* ha tenido valores más bajos de predicción de casos positivos en comparación con los modelos del conjunto de datos *Soft*. Aun así, en su totalidad, todos han obtenido valores muy altos de predicción correcta. Más adelante se realizarán las evaluaciones de rendimiento que permitirá diferenciar mejor la calidad de estos modelos.

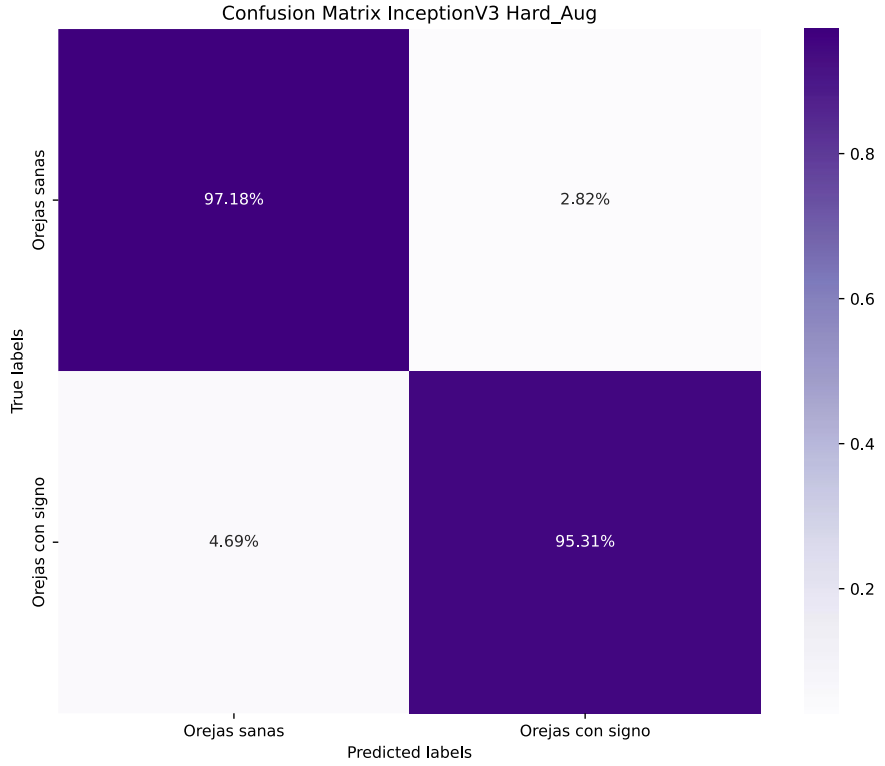


Figura 6-30. Matriz de confusión InceptionV3 Hard

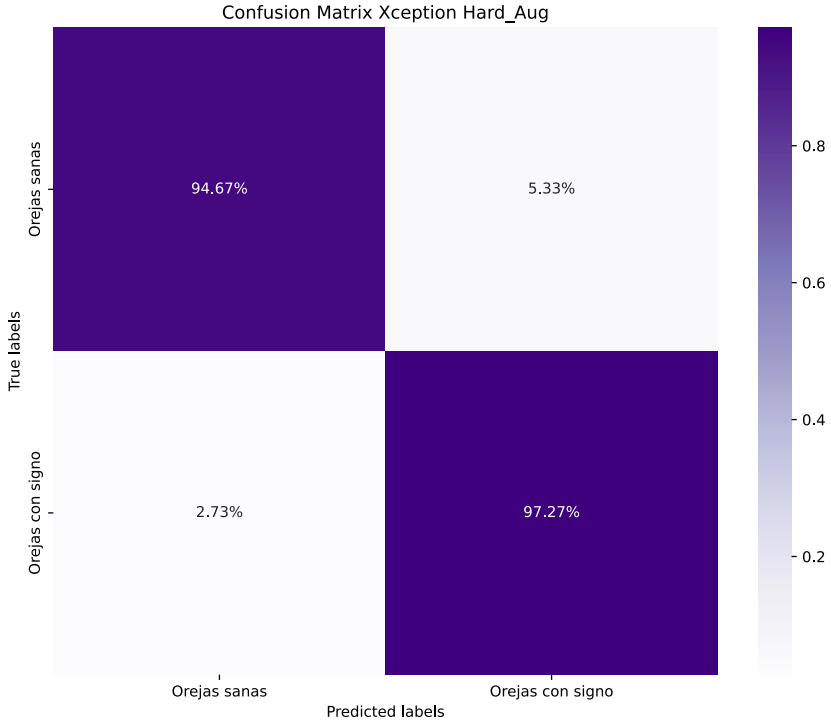


Figura 6-31. Matriz de confusión Xception Hard

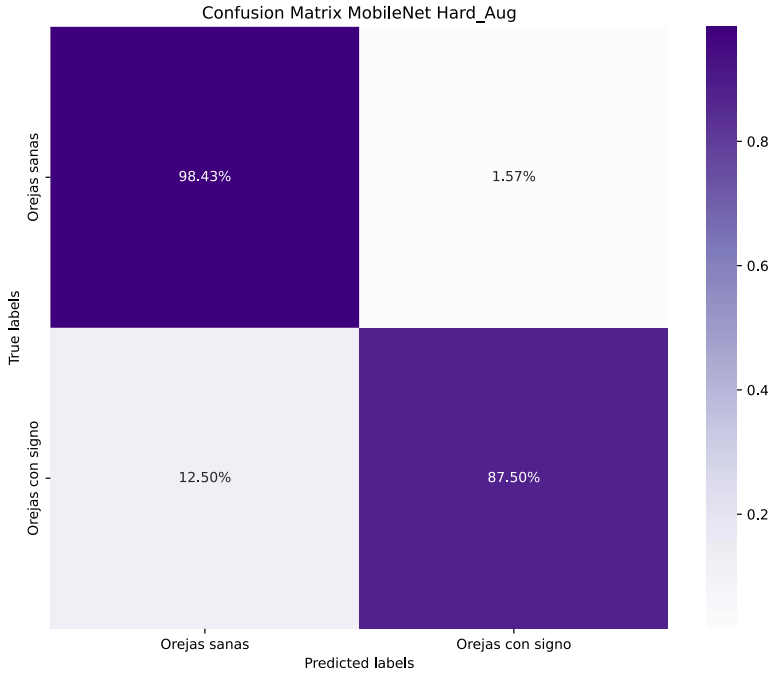


Figura 6-32. Matriz de confusión MobileNet Hard

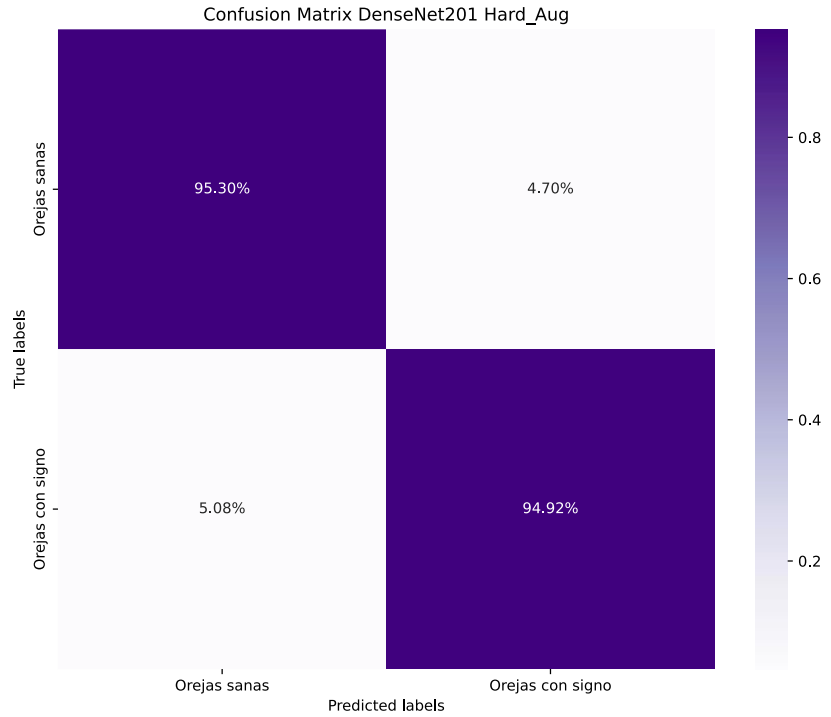


Figura 6-33. Matriz de confusión DenseNet201 Hard

Para terminar esta sección y tener una vision más clara de los resultados, se han recogido los datos de las matrices de confusión de ambos conjuntos de datos con sus modelos equivalentes y se procederá a comparar los que mejores valores han obtenido en base a la detección acertada o errónea de los elementos de los conjuntos.

La Tabla representan los valores en porcentajes obtenidos de cada modelo del dataset *Soft*. Interpretando los datos, el modelo InceptionV3 es el que más aciertos y menos errores ha cometido a la hora de predecir el conjunto de datos. Se puede observar cómo en general, los errores que producen todos los modelos se encuentra entre entre el 2% y 6.3%, valores aceptables dentro del rango de confusion de la red. También se destaca como los modelos InceptionV3 y DenseNet121 son los que mejor han detectado las imágenes positivas en comparación con las negativas. Pero los modelos InceptionV3 y MobileNet son los más equilibrados en base a la predicición de ambos tipos de imágenes, positivas y negativas.

	Aciertos		**Errores**	
Modelos	**TP (%)**	**TN (%)**	**FP (%)**	**FN (%)**
InceptionV3	97.66	97.49	2.51	2.34
MobileNet	96.48	96.87	3.13	3.52
ResNet50	94.92	96.55	3.45	5.08
DenseNet121	97.66	93.73	6.27	2.34

Tabla 6-4. Aciertos - Errores Modelos Soft

La Tabla 6-5 muestra los valores de las matrices de los modelos del conjunto de datos *Hard*. Empezando por el modelo Xception, es el que más imágenes positivas ha predicho correctamente. Este modelo es el único de los 4 que ha tenido más falsos positivos que falsos negativos, esto quiere decir que a la hora de confundirse en la predicción, ha cometido menos errores clasificando las orejas con signo como orejas sanas, esto hace que Xception sea más rentable que los demás en un problema de clasificación como este o en otros donde sea un riesgo tener un mal diagnostico de un caso positivo.

Otro modelo a destacar es MobileNet, cuya diferencia entre la detección de FP (1.57%) y FN (12.50%) es bastante amplia, esto se traduce en que este modelo puede ser de interés para aquellos casos donde se necesite una deteccion más precisa de los casos negativos que los positivos. Se confirma esa regla observando cómo su porcentaje de precision en casos negativos llega casi al 98.5% de precision frente al 87.5% que presenta para los casos positivos.

	Aciertos		**Errores**	
Modelos	**TP (%)**	**TN (%)**	**FP (%)**	**FN (%)**
Xception	97.27	94.67	5.33	2.73
InceptionV3	95.31	97.18	2.82	4.69
DenseNet201	94.92	95.30	4.70	5.08
MobileNet	87.50	98.43	1.57	12.50

Tabla 6-5. Aciertos – Errores Modelos Hard

Precisión (Precision)

La precisión es una métrica que permite medir el rendimiento del modelo a través del cálculo del porcentaje de casos positivos que detecta el modelo correctamente y así tener una medida exacta de la calidad de este a la hora de predecir únicamente las imágenes de orejas con el signo de Frank, en el caso de este proyecto. Esta métrica viene identificada por la siguiente fórmula:

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

Para entender mejor la precisión, se puede decir que un modelo con un porcentaje de precisión bajo (por debajo del 70 – 80%), puede detectar muchos casos positivos, sin embargo, puede que muchos de ellos no lo sean. Mientras, un valor alto de precisión significa que, aunque el modelo no detecte todos los casos positivos reales, aquellos que sí ha detectado tienen una probabilidad muy alta de que sean correctos.

Se puede observar en ambas tablas, cómo los porcentajes de precisión son muy buenos dado que no bajan del 92% y se tiene una media de precisión del 95.5% aproximadamente para ambos

conjuntos de datos. Concretamente, el mejor modelo en base a la precisión ha sido MobileNet para el conjunto de datos *Hard,* véase en la Tabla 6-7.

Soft Augmentation	
Modelos	***Precision (%)***
InceptionV3	96.89
MobileNet	96.11
ResNet50	95.67
DenseNet121	92.59

Tabla 6-6. Precisión Soft

Hard Augmentation	
Modelos	***Precision (%)***
Xception	93.60
Inception V3	96.44
DenseNet201	94.19
MobileNet	97.81

Tabla 6-7. Precisión Hard

Sensibilidad (*Recall*)

La sensibilidad, también conocida como Tasa de Verdaderos Positivos (*TPR*), es otra métrica de rendimiento que permite conocer el número de elementos positivos correctamente predichos entre el total de elementos de la clase real positiva. Es de las métricas más usadas junto con la precisión. La fórmula para obtener este valor es la siguiente:

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

Puede volverse un poco confuso el entender cada métrica que identifica el rendimiento y calidad de un modelo. Por ello, es mejor explicar el significado de un porcentaje alto y bajo de cada una de estas métricas.

Tener un porcentaje alto de sensibilidad, significa que el modelo identifica exitosamente todos o casi todos los casos positivos, pero esto también implica que pueda identificar ciertos casos negativos y predecirlos como positivos.

Un porcentaje bajo de sensibilidad indica que el modelo no identifica correctamente los casos positivos. Como conclusión, si se tiene un alto valor de sensibilidad, es que el modelo es de buena calidad, aunque confunda casos negativos como positivos. En el área de la salud, que es la que se encuentra este proyecto, siempre es mejor confundir pacientes sanos como enfermos que pacientes enfermos como sanos.

En la Tabla 6-8, InceptionV3 obtiene un *recall* de un 97.66%, esto significa que, en comparación a los demás modelos, es el que mejor habilidad tiene para detectar los casos positivos. Mientras, el más bajo es MobileNet en el conjunto de datos *Hard* (véase la Tabla 6-9) con un 87.50%. Aunque las comparaciones son mínimas dado que estos modelos han sido los mejores y en general todos han obtenido muy buenos valores. Se puede observar como InceptionV3 y MobileNet para el dataset *Soft* siguen siendo los mejores en el rendimiento general que desempeñan.

Soft Augmentation	
Modelos	***Recall (%)***
InceptionV3	97.66
MobileNet	96.48
ResNet50	94.92
DenseNet121	92.18

Tabla 6-8. Sensibilidad Soft

Hard Augmentation	
Modelos	***Recall (%)***
Xception	97.26
Inception V3	95.31
DenseNet201	92.59
MobileNet	87.50

Tabla 6-9. Sensibilidad Hard

Especificidad (*Especificity*)

Esta métrica, conocida también como Tasa de Verdaderos Negativos *(TNR)*, viene a ser lo contrario a la Sensibilidad descrita anteriormente, esta medida permite conocer qué tan bien el modelo puede predecir correctamente los casos negativos del total de negativos de la clase real. Su expresión matemática es la siguiente:

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

Normalmente, esta métrica se utiliza cuando en un problema de clasificación, se considere importante no confundir los casos negativos como casos positivos, es decir obtener el menor valor posible de falsos positivos. En el caso de este proyecto, se puede decir que no es necesario obtener esta métrica, pues es de mayor interés conocer aquellos casos positivos que se predicen como negativos, sin embargo, no viene mal observar los valores para un análisis profundo del desempeño de estos modelos.

De media, los modelos han obtenido más o menos un 96.4% de especificidad para este caso, destaca el modelo MobileNet de la Tabla 6-11, que ha obtenido un 98.43%. Como se observó anteriormente en el apartado de la sensibilidad o *Recall,* el modelo que menos porcentaje obtuvo fue MobileNet para el conjunto de datos *Hard.* A diferencia de los demás modelos, este ha obtenido un porcentaje mayor en especificidad que en la sensibilidad, esto significa que es el mejor modelo para clasificar orejas sanas correctamente. Para ambos conjuntos de datos, los modelos que se posicionan como mejores en especificidad son InceptionV3 y MobileNet, en este caso se puede observar como el modelo Xception para el dataset *Hard,* es el último de la clasificación en esta métrica, teniendo en cuenta que al contrario que MobileNet, obtuvo el mejor valor en *Recall*.

Soft Augmentation	
Modelos	***Especificity (%)***
InceptionV3	97.49
MobileNet	96.86
ResNet50	96.55
DenseNet121	93.73

Tabla 6-10. Especificidad Soft

Hard Augmentation	
Modelos	***Especificity (%)***
MobileNet	98.43
Inception V3	97.18
DenseNet201	95.30
Xception	94.67

Tabla 6-11. Especificidad Hard

F1-Score

Conocida también como puntuación F1, esta métrica engloba el promedio de la precisión y la sensibilidad de un modelo (*Precision and Recall*). Por tanto, gracias a esta medida se tiene en cuenta los falsos negativos y los falsos positivos (*FN, FP*). Funciona para problemas de clasificación donde el conjunto de datos no es equilibrado, a diferencia de la precisión (*Accuracy),* que únicamente es útil para datasets simétricos. La fórmula de F1-Score es la siguiente:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

En ambas tablas se puede apreciar como el promedio de la precisión y la sensibilidad de los modelos son bastante buenos, el rango se encuentra entre el 92% y el 97.3%. En la Tabla 6-12, el modelo DenseNet121 es el que ha obtenido el menor porcentaje de los 8 modelos, dando a

entender que en base a la relación *Precision-Recall*, es el que menos calidad tiene para distinguir orejas con el Signo de Frank. El modelo InceptionV3 de la Tabla 6-12 es el que sigue encabezando la clasificación, se puede apreciar también como los modelos del conjunto de datos *Soft* siguen obteniendo mejores puntuaciones.

Soft Augmentation	
Modelos	***F1 Score (%)***
InceptionV3	97.27
MobileNet	96.29
ResNet50	95.29
DenseNet121	92.38

Tabla 6-12. F1- Score Soft

Hard Augmentation	
Modelos	***F1 Score (%)***
Inception V3	95.87
Xception	95.35
DenseNet201	93.38
MobileNet	92.64

Tabla 6-13. F1 - Score Hard

Curva ROC y Área bajo la curva (AUC)

La curva ROC, llamada también Característica Operativa del Receptor (***R**eceiver **O**perating **C**haracteristic*, en inglés) y recientemente aplicada al área del aprendizaje automático, es una gráfica que permite conocer qué tan bueno es un modelo a la hora de distinguir entre dos clases, esto se llama capacidad discriminativa. Para poder medir la calidad de esa discriminación, se necesita un punto de corte (*threshold*) o umbral que establezca el límite entre los verdaderos positivos (TP) y los falsos positivos (FP). Normalmente ese umbral se establece en el 0.5 por defecto, dado que los valores de las métricas y la evaluación de un modelo se encuentran entre el 0 y 1. Para sacar la curva ROC, se necesitan los valores que los modelos han obtenido en las métricas *Recall y Especificity*, estas serán los ejes representativos del espacio ROC. La fórmula es la siguiente:

$$ROC\ Curve = \frac{Recall}{Especificity}$$

Para tener una idea más clara de cuando un modelo tiene un buen desempeño discriminatorio, se utiliza el Área bajo la curva (***A**rea **U**nder the **C**urve*, en inglés), esta es, básicamente, la línea

discriminatoria de la gráfica, véase la Figura 6-34, donde se muestra el ejemplo de comportamiento que un modelo puede presentar en base a su valor diagnóstico. Como se representa en estas tres gráficas, una discriminación perfecta se encontraría dentro de los valores 0.5 – 1.0, por lo que, en la segunda gráfica tenemos un AUC de 0.8, lo que significa que el modelo tiene un 80% de probabilidades de clasificar correctamente una oreja con el signo de Frank. Y la última gráfica presenta un valor de 0.5 que justo coincide con el punto de corte establecido, lo que indica una clasificación aleatoria, por tanto, su capacidad discriminatoria es desfavorable.

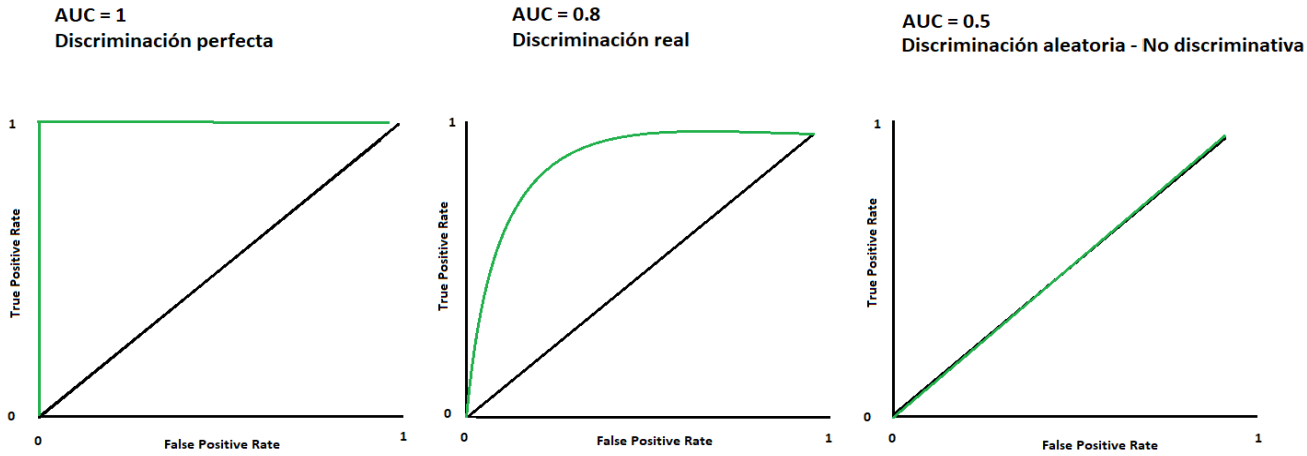


Figura 6-34. Ejemplo de gráficas Curva ROC y AUC

Para facilitar el análisis de las curvas ROC de los modelos, los intervalos AUC que se visionan en la Tabla 6-14, ofrecen una referencia para obtener una mejor precisión de clasificación de los modelos.

Intervalos AUC	Resultado
[0.5]	Aleatorio
[0.5, 0.6)	Test malo
[0.6, 0.75)	Test regular
[0.75, 0.9)	Test bueno
[0.9, 0.97)	Test muy bueno
[0.97, 1)	Test excelente

Tabla 6-14. Rangos AUC de clasificación

Para finalizar con el análisis de este capítulo, la Figura 6-35 y la Figura 6-36 que se visionan a continuación, representan las Curvas ROC de los 13 modelos estudiados. Esto da una visión general más comparativa para clasificar la calidad de los modelos y a parte se tienen ejemplos de ciertos modelos que tienen una curva ROC bastante mala. Se han recogido los datos en una tabla para cada conjunto de datos con los diferentes modelos y sus valores obtenidos de AUC para poder clasificar sus resultados.

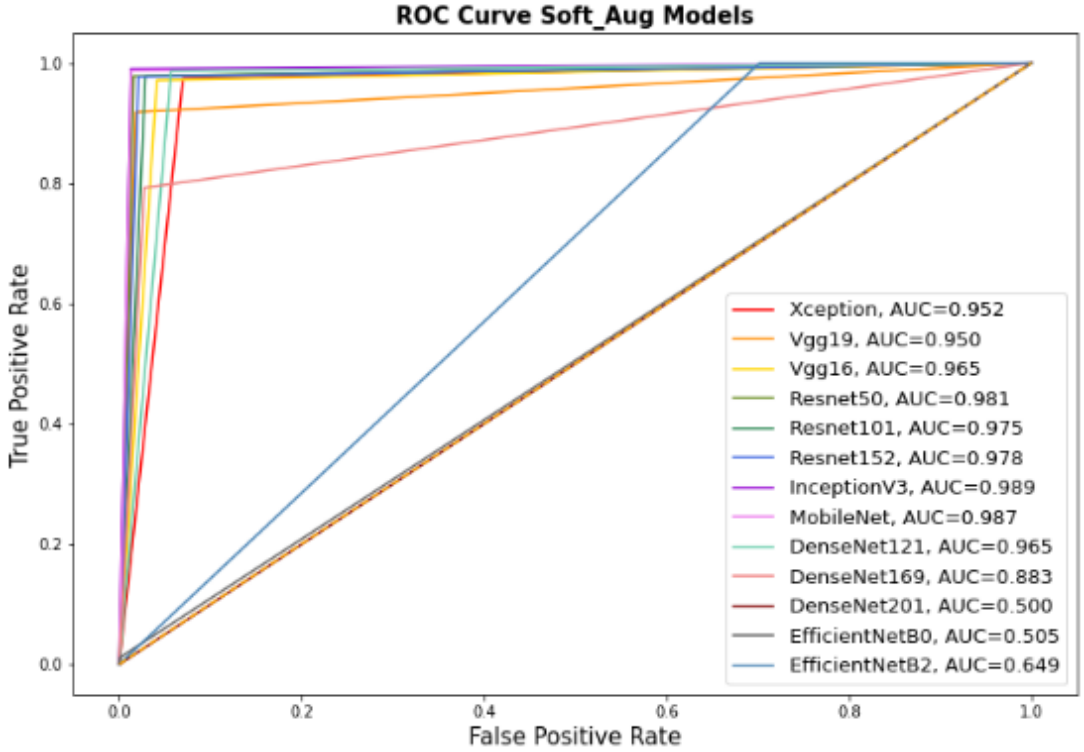


Figura 6-35. Curvas ROC Modelos Soft

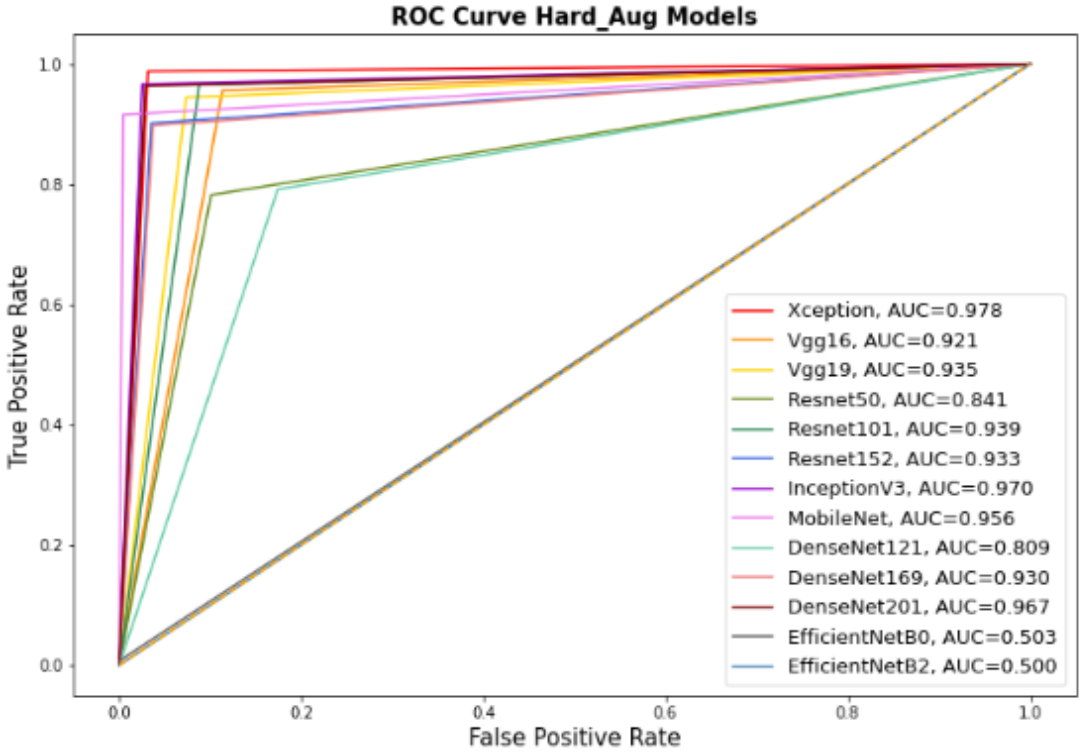


Figura 6-36. Curvas ROC Modelos Hard

Ordenados por la calidad discriminatoria que se reflejan en las Curvas ROC, se puede apreciar en la Tabla 6-15, cómo el modelo DenseNet201 es el que ha obtenido un AUC de 0.5 justo, por lo que se concluye que este modelo no sirve para clasificar en este proyecto, pues no se puede tener una certeza de que lo que clasifique sea correcto o incorrecto.

El opuesto de este, viene a ser InceptionV3, reflejada en la Figura 6-35 de color morado, su TPR (True Positives) es casi perfecto, se observa un angulo recto respecto a la AUC por defecto. Otro modelo que destaca en la gráfica es EfficienNetB2, su recorrido forma una clara diagonal muy cercana al umbral en su trayectoria respecto a los TP. Lo contrario le sucede a DenseNet169, de color salmón, tiene una trayectoria en angulo recto como los demas modelos, no obstante, se nota una clara diagonal apartir del valor 0.8 respecto a los FP.

Según los rangos AUC de clasificación vistos anteriormente, se pueden clasificar como excelentes, los modelos InceptionV3, MobileNet y los tres ResNets.

SOFT AUGMENTATION		
Models	**AUC**	**Resultados**
Inception V3	0.989	Excelente
MobileNet	0.987	Excelente
ResNet50	0.981	Excelente
ResNet152	0.978	Excelente
ResNet101	0.975	Excelente
VGG16	0.965	Muy bueno
DenseNet121	0.965	Muy bueno
VGG19	0.950	Muy bueno
Xception	0.952	Muy bueno
DenseNet169	0.883	Bueno
EfficientNetB2	0.649	Regular
EfficientNetB0	0.505	Malo
DenseNet201	0.500	Aleatorio

Tabla 6-15. Clasificación AUC Modelos Soft

La Tabla 6-16, representa los valores AUC de los modelos para el conjunto de datos *Hard,* se observa que solo los modelos Xception e InceptionV3 han sido los que han ganado la clasificación en base a sus valores AUC. En la Figura 6-36, el modelo Xception representado con el color rojo, forma un angulo recto casi perfecto, no obstante, el modelo MobileNet, de color rosa, aunque tenga un valor más bajo de TP, forma una línea mucho más recta que Xception. Los peores han sido los modelos EfficientNet, con un comportamiento aleatorio por parte de ambos, apenas visibles (véase en la Figura 6-36), dado que están justo en el umbral de referencia.

A diferencia de la clasificación del modelo DenseNet201 para el conjunto de datos *Soft,* que se posicionó como el peor modelo, en este caso, ha obtenido el tercer puesto en la clasificación, con una calidad muy buena, situandose casi en el 0.97.

HARD AUGMENTATION		
Models	**AUC**	**Resultados**
Xception	0.978	Excelente
Inception V3	0.970	Excelente
DenseNet201	0.967	Muy bueno
MobileNet	0.956	Muy bueno
ResNet101	0.939	Muy bueno
VGG19	0.935	Muy bueno
ResNet152	0.933	Muy bueno
DenseNet169	0.930	Muy bueno
VGG16	0.921	Muy bueno
ResNet50	0.841	Bueno
DenseNet121	0.809	Bueno
EfficientNetB0	0.503	Malo
EfficientNetB2	0.500	Aleatorio

Tabla 6-16. Clasificación AUC Modelos Hard

Finalizando el análisis que se ha desarrollado a lo largo de este capítulo, se concluye que los 4 mejores modelos de ambos datasets han obtenido porcentajes que superan con creces el umbral de aceptación, tanto en la precisión como en las métricas de rendimiento.

Para el conjunto de datos ***Soft*** se ha obtenido:

- **Precisión media:** 95.32%
- ***Recall* medio:** 95.24%
- ***Specificity* medio:** 96.15%
- **F1 – Score medio:** 95.30%

Para el conjunto de datos ***Hard*** se ha obtenido:

- **Precisión media:** 95.51%
- ***Recall* medio:** 93.16%
- ***Specificity* medio:** 96.39%
- **F1 – Score medio:** 94.31%

Al observar las medias obtenidas de las métricas estudiadas, se puede apreciar la calidad que estos ocho modelos han desempeñado en su tarea de clasificación. Tanto la precisión como la calidad de discriminación de estos representan unos valores óptimos para dejar concluido el análisis realizado para este proyecto académico con una visión optimista de las redes desarrolladas. No obstante, los valores son mejorables, pues estos últimos modelos tienen

potencial para poder superar el umbral del 97-98% de media en las métricas. Realizando ajustes más personalizados en las redes, es probable que se pueda obtener valores más altos, sin embargo, dado que este trabajo se encarga de hacer un análisis comparativo, es mejor estandarizar una configuración para todas las redes o la mayoría y poder ver el comportamiento que han ejercido sobre los distintos conjuntos de datos.

Capítulo 7

CONCLUSIONES Y TRABAJO FUTURO

Realizando un breve recordatorio de los objetivos planteados inicialmente se han cumplido los siguientes:

- Recopilación manual de imágenes que conforman los conjuntos de datos a estudiar, intentando obtener las que mejor resolución tuvieran.
- Anotación manual de las imágenes mediante un cuadro delimitador y generación de los ficheros xml correspondientes.
- Creación de un script para cortar las imágenes mediante el cuadro delimitador (*ground-truth)* y así quitar las partes irrelevantes o ruido de la imagen.
- Aplicación de la técnica de aumento de datos para ampliar el conjunto de imágenes y obtener variaciones de ellas.
- Familiarización con las librerías correspondientes al entorno de desarrollo.
- Preparación del conjunto de datos para el entrenamiento y validación de los modelos.
- Estudio de las arquitecturas disponibles y posterior evaluación de estos.

Ahora, teniendo en cuenta el objetivo general de este proyecto, que era el de realizar un estudio de un total de 13 modelos pre-entrenados ofrecidos por Keras para la detección del signo de Frank mediante clasificación binaria y a partir de este estudio, extraer los modelos que mejor se comportasen para esta problemática mediante la evaluación de las métricas aplicadas a este tipo de campo de investigación, puede decirse que se ha cumplido satisfactoriamente, pues a pesar de ciertas dificultades, como la de tener que construir el conjunto de datos desde cero dado que no existía un dataset que se ajustara a este proyecto, se ha conseguido obtener unas redes con más del 90% de precisión.

No todas consiguieron superar ese umbral y algunas que sí lo superaron, sus validaciones eran muy inestables haciendo reflexionar sobre posibles cambios en esos modelos que podrían abarcar desde la configuración de las capas, hasta el optimizador empleado. Cabe destacar que, para este estudio, se aplicó la misma configuración en las redes, excepto en la arquitectura VGG, que tuvo ligeros cambios en las capas para ver si era posible obtener unos mejores resultados (y así lo hizo). Pero la intención era mantener en todos los modelos la misma configuración para poder hacer una comparativa lo más equilibrada posible y así poder evaluar el comportamiento de cada arquitectura estudiada frente a esta problemática.

Las redes VGG, como se dijo en el capítulo anterior, tienen mucho potencial para entrar en la clasificación como las mejores, no obstante, se necesitan de ciertos requisitos como un conjunto de datos bastante amplio, que este proyecto no ha podido cumplir. Por lo que seguramente una

buena configuración de la red y un dataset lo suficientemente grande, permitiría obtener unos valores bastante satisfactorios.

Los modelos en común que mejor se comportaron en ambos conjuntos de datos, fueron InceptionV3 y MobileNet. Estas arquitecturas consiguieron tanto un buen porcentaje de precisión (*accuracy*), como un aprendizaje equilibrado y bastante óptimo. Concretamente, el mejor porcentaje obtenido fue de un 97.81%, proporcionado por **MobileNet.** Estos dos modelos podrían ser los elegidos para una continuación y mejora en base a la clasificación de este marcador cutáneo.

Como trabajos futuros respecto a esta línea de investigación, se podría continuar el estudio, profundizando en cada arquitectura para adaptarla y probarla en un entorno médico real con pacientes, para testar las bondades y dificultades del sistema.

Otra opción de mejora de este proyecto es la expansión del dataset. Aprovechando la idea propuesta en el anterior párrafo de probar el modelo en un entorno real tratar de crear un conjunto de datos mucho más amplio que sirva para un mejor rendimiento de las redes. Se ha concluido que la calidad de las imágenes influye mucho en la detección del signo, dado que de ella depende la correcta extracción de información sobre los rasgos que puede aportar al modelo. También, la dirección de la imagen es otro factor importante, la oreja es una parte del cuerpo difícil de fotografiar correctamente pues normalmente, la cara es el foco principal de la imagen y la oreja suele quedar escondida. Las fotografías donde se recoge el perfil de una persona es el enfoque ideal para obtener una buena imagen de las orejas.

Otro posible estudio mucho más detallado, es el de realizar una clasificación binaria que permita distinguir el signo de Frank, de la arruga que normalmente se genera por el uso de pendientes, creando un conflicto muchas veces en el diagnóstico correcto del signo. Para ello se necesitaría obtener un conjunto de datos que contenga imágenes de orejas con la arruga originada por el pendiente y orejas que tengan el signo de Frank bien marcado.

Por último, enlazando con la idea propuesta anteriormente, una buena continuación de este proyecto es el de desarrollar una aplicación móvil donde se aplique la detección automática del signo de Frank *in situ* y con ello facilitar el reconocimiento de este marcador cutáneo a los profesionales para tenerlo en cuenta en el historial del paciente.

BIBLIOGRAFÍA

[1] "Defunciones según la causa de muerte. Año 2020". *Instituto Nacional de Estadística. INE.* 2021. https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176780&menu=ultiDatos&idp=1254735573175.

[2] Wikipedia Contributors. "Frank's sign". *Wikipedia*, 2022. https://en.wikipedia.org/wiki/Frank%27s_sign#:~:text=Frank's%20sign%20is%20a%20diagonal,Frank.

[3] "Frank's Sign (Diagonal Earlobe Crease)," *Stanford Medicine 25*. https://stanfordmedicine25.stanford.edu/blog/archive/2015/what-is-the-name-of-this-sign.html, July 2015.

[4] S. Nazzal and A. Blum. "Association Between the Frank Sign and Cardiovascular Events"*, Southern Medical Journal*. Vol. 111, no. 8, pp. 504-509. Aug. 2018, doi: 10.14423/SMJ.0000000000000845.

[5] C. Thilo, C. Meisinger, M. Heier, W. von Scheidt, and I. Kirchberger*, "*Diagonal earlobe crease and long-term survival after myocardial infarction". *BMC Cardiovasc Disorders,* vol. 21**,** no. 1. Dec. 2021, doi: 10.1186/s12872-021-02425-4

[6] F. Sanchis-Gomar, C. Perez-Quilis, R. Leischik and A. Lucia. "Epidemiology of coronary heart disease and acute coronary syndrome", *Annals of Translational Medicine,* Vol. 4, no. 13. pp. 256-256 Jul. 2016. doi: 10.21037/atm.2016.06.33

[7] T. E. Mallinson and D. Brooke. "Limited Diagnostic Potential of Diagonal Earlobe Crease". *Annals of Emergency Medicine*. Vol. 70**,** no. 4, pp. 602-603, Oct. 2017, doi: 10.1016/j.annemergmed.2017.06.013

[8] G. S. Stoyanov, D. Dzhenkov, L. Petkova, N. Sapundzhiev and S. Georgiev. "The Histological Basis of Frank's Sign*". Head and Neck Pathology.* Vol. 15, no. 2, pp. 402-407, Jul. 2020, doi: 10.1007/s12105-020-01205-4

[9] A. N. Lin, K. Lin, H. Kyaw, and J. Abboud. "A myth still needs to be clarified: a case report of the Frank's sign". *Cureus,* Vol. 10, no. 1, Jan. 2018, doi: 10.7759/cureus.2080.

[10] A. Pasternac and M. Sami. "Predictive value of the ear-crease sign in coronary artery disease*". Canadian Medical Association Journal*, vol. 126, no. 6, pp. 645-649, Mar. 1982, PMID:7066824

[11] X. Zeng *et al.,* "Efficient and accurate identification of ear diseases using an ensemble deep learning model". *Scientific Reports,* vol. 11**,** no. 1, May 2021, doi:10.1038/s41598-021-90345-w.

[12] Y.-M. Wang, *et al.* "Deep Learning in Automated Region Proposal and Diagnosis of Chronic Otitis Media Based on Computed Tomography", *Ear & Hearing,* vol. 41, no. 3, pp. 669-677**,** Sep. 2019**.** doi: 10.1097/aud.0000000000000794.

[13] Y.-C. Chen, *et al. "*Smartphone-based artificial intelligence using a transfer learning algorithm for the detection and diagnosis of middle ear diseases: A retrospective deep learning study". *eClinicalMedicine,* vol. 51, p. 101543, Sep. 2022, doi: 10.1016/j.eclinm.2022.101543.

[14] Wikipedia Contributors, "Transfer Learning", *Wikipedia*, Jun 2019. https://en.wikipedia.org/wiki/Transfer_learning.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818-2826, Jun. 2016, doi: 10.1109/cvpr.2016.308.

[16] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions". *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800-1807, 2017, doi: 10.1109/cvpr.2017.195.

[17] M. Mahdianpari, B. Salehi, M. Rezaee, F. Mohammadimanesh and Y. Zhang. "Very Deep Convolutional Neural Networks for Complex Land Cover Mapping Using Multispectral Remote Sensing Imagery", *Remote Sensing,* vol. 10, no. 7, pp. 1119, Jul. 2018, doi: 10.3390/rs10071119.

[18] K. Team. "Keras documentations: Keras Applications"., *keras.io,* https://keras.io/api/applications/

[19] K. Simonyan, A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, doi: 10.48550/arXiv.1409.1556

[20] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2015, doi: 10.48550/arXiv.1512.03385.

[21] Wikipedia Contributors, "Vanishing gradient problem", *Wikipedia,* 2019, https://en.wikipedia.org/wiki/Vanishing_gradient_problem

[22] G. Huang *et al*, "Densely Connected Convolutional Networks". *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, doi: 10.48550/arXiv.1608.06993

[23] A.-G. Howard *et al. "*MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2017, doi: 10.48550/arXiv.1704.04861.

[24] S. Phiphitphatphaisit, O. Surinta. "Food Image Classification with Improved MobileNet Arquitecture and Data Augmentation". *ICISS,* 2020 pp. 51-56, doi:10.1145/3388176.3388179.

[25] M. Tan, Q. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". *In Proceedings of the 36th International Conference on Machine Learning, ICML.* 2019, pp. 6105-6114, 2019, doi: 10.48550/arXiv.1905.11946.

[26] "EfficientNet: Improving Accuracy and Efficiency through AutoMl and Model Scaling", *Google AI Blog,* 2019, https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

[27] D. P. Kingma, J. Ba. "Adam: A method for Stochastic Optimization". *3rd International Conference for Learning Representations, ICLR 2015,* doi: 10.48550/arXiv.1412.6980.

[28] A. Ajagekar. "Adam". *Cornell University SYSEN 6800 Fall 2021*.

[29] Ž. Emeršič, V. Štruc, P. Peer. "Ear recognition: More than a survey". *Neurocomputing,* Vol. 255, 2017, pp. 26-39, doi: 10.1016/j.neucom.2016.08.139.

[30] S. Balaji. "Binary Image classifier CNN using TensorFlow". *Medium*, Aug 2020, https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697

[31] G. Van Rossum, and F. L. Drake. *Python 3: reference manual,* United States: Sohobooks, 2009.

[32] C.R. Harris *et al*., "Array programming with NumPy". *Nature* vol. 585, no. 7825, pp. 357-362, Sep. 2020. doi: 10.1038/s41586-020-2649-2.

[33] W. McKinney. "Data structures for statistical computing in python". *Proceedings of the 9th Python in Science Conference,* vol. 445, pp 51-56, 2010, doi: 10.25080/Majora-92bf1922-00a.

[34] F. Pedregrosa *et al*. "Scikit-learn: Machine Learning in Python", Journal *of Machine Learning Research.* Vol. 12, pp. 2825-2830, 2011, doi: 10.48550/arXiv.1201.0490.

[35] M. Abadi *et al.*, "TensorFlow: Large-scale Machine Learning on Heterogeneous Systems". *OSDI 16.* pp 265-283. 2016

[36] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin. "Albumentations: fast and flexible image augmentations", *Information,* vol. 11, no. 2, pp. 125, Feb. 2020, doi:10.3390/info11020125.

[37] J. D. Hunter. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering.* vol. 9, No.3, pp. 90-95, 2007, doi: 10.1109/mcse.2007.55.

[38] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools.* 2000.

[39] Tzutalin. LabelImg. *Free Software: MIT License,* 2015. https://github.com/tzutalin/labelImg

[40] Google. "Google Colaboratory", *colab.research.google.com,* https://colab.research.google.com/?hl=es

[41] K. O'Shea, R. Nash. "An Introduction to Convolutional Neural Networks". *Neural and Evolutionary Computing*. 2015, doi: 10.48550/arXiv.1511.08458.

[42] D. Dansana *et al., "*Early diagnosis of COVID-19-affected patients based on X-ray and computed tomography images using deep learning algorithm", *Soft Computing,* Aug. 2020, doi: 10.1007/s00500-020-05275-y.

[43] "VGGNet-16 Architecture: A Complete Guide", *Kaggle.com,* https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide/notebook.

[44] A. Rastogi, "ResNet50", *Medium,* 2022, https://blog.devgenius.io/resnet50-6b42934db431