UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería de la Salud

Detección de estenosis en imágenes coronariográficas aplicando aprendizaje profundo

Detection of stenosis in coronary angiography images using Deep Learning

Realizado por
Irene Romero Granados

Tutorizado por
Rafael Marcos Luque Baena
Esteban Palomo Ferrer

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, junio de 2022

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DE LA SALUD

Detección de estenosis en imágenes coronariográficas aplicando aprendizaje profundo

Detection of stenosis in coronary angiography images using Deep Learning

Realizado por
Irene Romero Granados

Tutorizado por
Rafael Marcos Luque Baena
Esteban Palomo Ferrer

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE
MÁLAGA MÁLAGA, JUNIO
DE 2022

Fecha defensa: julio de 2022

Abstract

Nowadays, heart disease is one of the main causes of death, and it is essential to create and maintain prevention protocols together with a correct diagnosis for those who suffer from cardiovascular disease. One of these diseases is coronary stenosis, a pathology which, due to the narrowing of the coronary arteries, can lead to myocardial infarction or even death. For its diagnosis, a coronary angiography is performed. This test consists of staining the arteries through the use of a catheter, which allows the state of the arteries to be visualised by means of X-rays. Coronary angiographic images are the result of this test.

In this study, regions of these coronary images will be used to train a model using a dense neural network from scratch, together with pre-trained networks such as ResNet50, EfficientNetB4 and MobileNet, and classify them as an image with stenosis or without stenosis. Cross validation will be used in the training to obtain a result that does not depend on the division of the ensemble in training and test, due to the unbalance of the classes of the ensemble, and in addition 4 different data augmentation techniques will be applied to try to improve the results of the models. These techniques are: sampling and data set augmentation, minority class augmentation, proportional data augmentation and synthetic data.)

The models obtained with the test set will be evaluated and the one with the best result will be used as a learning model integrated in a web application where any user can enter a coronary angiographic image. In the server part of the web application, the coronary image will be divided into 32x32 regions and a stenosis detection algorithm will be applied to each region using this model and the location of the stenosis will be located in the image introduced by the user.

Keywords: **Neural network, Stenosis, Coronary angiography, Pre-trained network, Cross-validation, Data augmentation**

Resumen

Hoy en día las cardiopatías son una de las principales causas de muerte, siendo esencial crear y mantener protocolos de prevención junto con un correcto diagnóstico para aquellos que sufran una enfermedad cardiovascular. Una de estas enfermedades es la estenosis coronaria, patología que debido al estrechamiento de las arterias coronarias puede llegar a producir un infarto del miocardio o incluso la muerte. Para su diagnostico se realiza una coronariografía, prueba que consiste en realizar una tinción de las arterias mediante el uso de un catéter, el cual permite visualizar el estado de las arterias mediante rayos X. Las imágenes coronariográficas son el resultado de dicha prueba.

En este estudio se utilizarán regiones de dichas imágenes coronariográficas para entrenar un modelo usando una red neuronal densa desde cero, junto con redes pre-entrenadas como *ResNet50, EfficientNetB4 y MobileNet*, y clasificarlas como imagen con estenosis o sin estenosis. Se hará uso de la validación cruzada en el entrenamiento para obtener un resultado que no dependa de la división del conjunto en entrenamiento y test, debido al desbalanceo de las clases del conjunto, y además se aplicarán 4 técnicas diferentes de aumento de datos para intentar mejorar los resultados de los modelos. Estas técnicas son : muestreo y aumento del conjunto de datos, aumento de la clase minoritaria, aumento proporcional de los datos y datos sintéticos).

Se evaluarán los modelos obtenidos con el conjunto de test y el que obtenga mejor resultado se usará como modelo de aprendizaje integrado en una aplicación web dónde cualquier usuario pueda introducir una imagen coronariográfica. En la parte del servidor de la aplicación web se dividirá la imagen coronariográfica en regiones de 32x32 y se aplicará a cada región un algoritmo de detección de estenosis usando dicho modelo y se localizará en la imagen introducida por el usuario dónde se encuentra la estenosis.

Palabras clave: Red neuronal, Estenosis, Coronariografía, Red preentrenada, Validación cruzada, Aumento de datos

Índice

1. Introducción **7**

1.1. Motivación . . . 7
1.2. Objetivos . . . 8
1.3. Estructura del documento . . . 8
1.4. Tecnologías usadas . . . 10

2. Conceptos teóricos **17**

2.1. Cardiopatía . . . 17
2.1.1. Estenosis coronaria . . . 17
2.2. Coronariografía . . . 17
2.3. Angioplastia coronaria . . . 18
2.4. Aprendizaje profundo . . . 19
2.4.1. Aprendizaje profundo aplicado a la salud . . . 20
2.4.2. Estructura . . . 20
2.4.3. Redes neuronales densas . . . 21
2.4.4. Redes neuronales Convolucionales (CNN) . . . 22

3. Metodología **25**

3.1. Conjunto de datos . . . 25
3.2. Preprocesamiento . . . 26
3.3. Arquitectura de la red . . . 26
3.3.1. Red Neuronal densa entrenada desde cero . . . 26
3.3.2. Red Neuronal Pre-entrenada . . . 27
3.3.3. *ResNet50* . . . 28
3.3.4. *EfficientNetB4* . . . 29
3.3.5. *MobileNet* . . . 29
3.4. Optimizador, función de pérdida y de activación . . . 30
3.5. Métricas . . . 31
3.5.1. *Accuracy* . . . 31
3.5.2. F1 score . . . 31
3.5.3. *Balanced Accuracy* . . . 32
3.6. Validación cruzada . . . 32
3.7. Aumento de datos . . . 33
3.7.1. *Sampling and augmentation* . . . 34

3.7.2. *Minority class augmentation* . . . 34
3.7.3. *Proportional data augmentation* . . . 34
3.7.4. *Synthetic data* . . . 35
3.8. Entrenamiento del modelo . . . 35

4. Evaluación del modelo y Resultados experimentales **37**
4.1. Resultados . . . 50

5. Aplicación web **55**
5.1. Detección de la Estenosis . . . 55
5.2. Funcionamiento . . . 56

6. Conclusiones y futuras líneas de investigación **59**
6.1. Conclusiones . . . 59
6.2. Futuras líneas de investigación . . . 60

Bibliografía **63**

Apéndice A. Manual de Instalación **69**

1 Introducción

1.1. Motivación

La prevención temprana de una enfermedad o condición es clave para el correcto tratamiento y recuperación del paciente. Las nuevas técnicas desarrolladas en el mundo de la medicina han permitido al ser humano aumentar mucho su calidad de vida así como sus años de vida. Una de esas técnicas es, para los pacientes que presentan estenosis en las arterias del corazón, la colocación de un *stent coronario*. Este dispositivo desobstruye las arterias permitiendo que la sangre fluya con normalidad. Sin la colocación de dicho dispositivo en la zona adecuada, se podría producir una angina de pecho en el caso de que la obstrucción fuera parcial, o un infarto del miocardio si la obstrucción es completa. Ambas condiciones te pueden llevar a un cuadro médico grave e incluso a la muerte.

Una de las posibles técnicas de diagnóstico es la realización de una coronariografía [1]. En dicha técnica se introduce un *catéter*, dispositivo de plástico de tamaño muy reducido (2mm de diámetro), en las arterias mediante una pequeña abertura. Cuando el catéter llega a las arterias coronarias, inyecta un líquido de contraste en ellas, tiñéndolas. De esta forma permite al *angiógrafo*, equipo médico de rayos X, mostrar el estado de las arterias. De esa prueba resultan las imágenes coronariográficas. Una correcta interpretación de dichas imágenes es imprescindible para un buen diagnóstico. Sin embargo, la detección de dicha patología es compleja ya que la interpretación de las imágenes también lo es. Un análisis computacional de las imágenes coronariográficas podría ayudar en gran medida al diagnóstico, y como consecuencia, ayudar a todas aquellas personas que padecen esta patología.

Es un hecho que en la actualidad el aprendizaje automático se está usando en todos los campos. En el campo clínico cada vez son más las empresas que se están sumando a usar esta tecnología en sus dispositivos y máquinas para reducir el posible error humano y aumentar la eficiencia del resultado y de los recursos.

Ha habido un estudio llevado a cabo por *Google* que hacía uso de aprendizaje profundo en la detección de una enfermedad llamada ***retinoplastía diabética***[2]. En dicho estudio, se ha

obtenido un 97.5 % de sensibilidad y un 96.9 % de especificidad buscando lesiones de dicha enfermedad al examinar imágenes que pertenecían al interior de la retina. Los resultados de este estudio llevó a que la **FDA** (Departamento de Salud y Servicios Humanos) aprobara en el 2018 su uso en pacientes, siendo un gran avance para el campo óptico. Lo mismo sucede en otros campos de la salud como la detección de cáncer de mama mediante el análisis de mamografías. Por lo que la motivación real de este estudio, además de ayudar al diagnóstico, es ayudar a las personas que padecen una enfermedad cardíaca producida por estenosis, proporcionándoles una *segunda opinión* y dándoles la seguridad, de que ese diagnóstico es preciso fiable.

1.2. Objetivos

El objetivo principal de este estudio es la implementación de un modelo lo suficientemente preciso como para predecir si una región de una imagen coronariográfica presenta o no estenosis. Para ello, se van a implementar diferentes modelos, usando diferentes redes neuronales (red neuronal densa entrenada desde cero y pre-entrenadas). También se van a aplicar diferentes técnicas que aumento de datos para poder entrenar mejor los modelos y poder compensar el desbalance que existe en el conjunto de datos. Por consiguiente, el problema que se va a abordar en este proyecto es de clasificación binaria, y se hará uso del aprendizaje profundo materializado en una combinación de las técnicas de aumento de datos junto con las diferentes redes neuronales. Elegir una buena métrica para evaluar el modelo es esencial, por lo que elegir la más apta y el uso de la validación cruzada para este problema de clasificación binario desbalanceado, hará que los resultados, independientemente de su división en entrenamiento y test, sean fiables.

También se quiere usar el mejor modelo obtenido para realizar la segmentación de una imagen coronariográfica completa a modo de test, que será dividida en regiones del tamaño de las regiones de entrenamiento y se predecirá para cada una de esas regiones una etiqueta. Finalmente se obtendría una imagen con las zonas que presentan estenosis y no estenosis. Todo esto se implementará en una aplicación web para que cualquier usuario que lo desee, pueda probar el modelo con una imagen coronariográfica propia y en un solo vistazo ser capaz de observar dónde, si es que la hay, se encuentra la estenosis en la imagen. De esta manera, los avances tecnológicos son más accesibles al usuario y además de una forma simple y fácil.

1.3. Estructura del documento

Esta memoria cuenta con 6 capítulos y 1 apéndice dónde la información se divide en las siguientes secciones:

- **Introducción** (Capítulo 1): En el primer capitulo se presenta el problema que se va a tratar en este proyecto, por qué es importante abordarlo y los objetivos del mismo. Así como una explicación breve de las tecnologías usadas para su desarrollo.

- **Conceptos teóricos** (Capítulo 2): En el segundo capítulo se explican conceptos básicos para entender el contexto clínico de la patología bajo estudio, las consecuencias de dicha patología y las consecuencias de la enfermedad. También se da una explicación de las redes neuronales, los tipos de ellas, y una breve explicación de lo que es el *deep learning*.

- **Metodología** (Capítulo 3): El tercer capítulo es el más extenso de todos pues es dónde se explica todo lo que se ha realizado en este trabajo. Se comenta, describe y muestra como es el conjunto de imágenes que va a ser utilizado en este estudio y el preprocesamiento de las imágenes. También se explican las diferentes redes neuronales implementadas, así como su estructura y la elección de sus parámetros. Se explican las diferentes técnicas de aumento de datos que se han abordado y se da una breve explicación de las métricas usadas para evaluar el modelo y la métrica finalmente elegida. Se da explicación breve sobre porqué el uso de la validación cruzada en este trabajo es muy importante. Finalmente se comenta como se ha realizado la división del conjunto en entrenamiento y test, y algunas técnicas para evitar el sobre-entrenamiento del modelo como pueden ser *early stopping* y *checkpoint*.

- **Evaluación y Resultados** (Capítulo 4): En el cuarto capítulo se explica como se va a realizar el proceso de la evaluación del modelo, además de la interpretación de los resultados haciendo uso de la matriz de confusión. Se comentan todos los resultados obtenidos en cada prueba realizada y se visualizan diferentes gráficas que ayudan a entender cómo ha sido el proceso de entrenamiento del modelo, deducir cuál ha sido la red neuronal con mejores resultados, así como la mejor técnica de aumento de datos.

- **Aplicación Web** (Capítulo 5): En el quinto capitulo se abordará el funcionamiento de la aplicación web, las características del proceso de detección de la estenosis. Un resumen del proceso de cliente-servidor llevado a cabo cuando un usuario introduce la imagen coronariográfica esperando obtener un resultado. Finalmente se mostrarán las vistas que tiene la interfaz gráfica de la web.

- **Conclusiones y futuras líneas de trabajo** (Capítulo 6): En este último capítulo, se hace un breve resumen de todas las fases del proyecto, así como una conclusión sobre los resultados de la evaluación y la elección del mejor modelo. También se va a hablar de la posibilidad de realizar en un futuro algoritmos de detección para finalmente realizar la segmentación de las imágenes coronariográficas. También se habla de la posibilidad

de utilizar la técnica *GANs* para aumentar el conjunto y probar si da mejores resultados que los obtenidos en este proyecto.

- **Manual de Instalación** (Capítulo 6): En él se hará un tutorial de los pasos a seguir para instalar lo necesario para iniciar el servicio de la aplicación web.

1.4. Tecnologías usadas

En esta sección se va a hablar brevemente de las diferentes tecnologías implementadas para poder realizar este proyecto.

- **Python**[3]: es un un lenguaje de programación de alto nivel, orientado a objetos, que se caracteriza por su sencillez y fácil manejo ya que utiliza una sintaxis cuyo objetivo es la legibilidad. Esto hace que su código sea más fácil de leer e incluso traducir a otros lenguajes, reduciendo el mantenimiento del programa así como de desarrollo al permitir que diferentes personas trabajen en un mismo programa y haya entendimiento en el lenguaje y forma de definir las variables. Tiene una semántica integrada que es dinámica cuyo uso principal es para el desarrollo de páginas web. Además debido a la cantidad de módulos y paquetes, que incorpora es muy funcional, añadiendo la posibilidad de escalar los paquetes a otros proyectos una vez se haya terminado de desarrollar. Además es un lenguaje de código abierto, cuya librería estándar e intérprete se encuentran disponibles en forma binaria y en forma de fuente. Todo esto convierte a *Python* en la multiplataforma que actualmente está más en auge y destaca su uso en todos los campos de la inteligencia artificial. 1.4

Figura 1: Logo de Python[3]

- **Google Colab**[4]: es un servidor web en la nube, totalmente gratuito. Pertenece a *Google* y su principal objetivo es fomentar el uso de la inteligencia artificial y del aprendizaje automático, permitiendo a cualquier usuario ejecutar scripts de *Python*, con la ventaja

de que ya vienen instaladas la mayoría de las librerías y además poner gestionar el tipo de entorno de ejecución que necesitas entre normal, GPU y TPU. Los archivos de *colab* no son más que cuadernos dónde se pueden intercambiar celdas de texto o celdas de código. Tiene la ventaja de se puede conectar a **Google Drive** permitiendo cargar archivos de tu cuenta personal. Para este proyecto, se va a usar un entorno con procesador GPU para el entrenamiento de las redes neuronales.

Figura 2: Logo de Google Colab[4]

- **TensorFlow**[5]: es una librería de código abierto diseñada para desarrollar y entrenar modelos de aprendizaje automático. Además cuanta con una gran comunidad de desarrolladores en la que poder apoyarse y aprender. Actualmente es la herramienta número uno en el aprendizaje profundo.

Figura 3: Logo de TensorFlow[5]

- **Keras**[6]: ImageDataGenerator es una librería que permite duplicar una imagen aplicándole diferentes técnicas de cambio como podría ser rotación, volteo o desplazamiento. Esta librería ha sido clave para este estudio debido al desbalanceo del conjunto de datos usado.

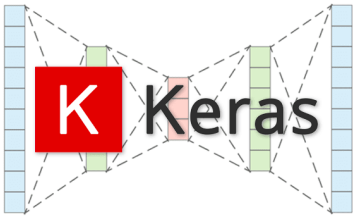

Figura 4: Logo de Keras[6]

- ***OverLeaf*** [7]: es una herramienta destinada a publicar y redactar documentos científicos, de forma colaborativa simultáneamente. Overleaf no es más que un editor de LaTeX que permite crear elaboraciones científicas de manera muy sencilla, y además permitiendo que la elaboración sea compilada paralelamente a la redacción. Cuenta con plantillas para TFG, TFM y Tesis, y permite compartir las creaciones además de hacer comentarios a las distintas secciones, aspecto muy útil en el desarrollo de un proyecto como un TFG.

Figura 5: Logo de OverLeaf[7]

- ***LaTeX***[8]: es un sistema de elaboración de textos orientado a la redacción de documentos que necesitan una calidad en su tipografía alta. Es por esto que es ampliamente utilizado para desarrollar artículos científicos. Utiliza el lenguaje de bajo nivel TeX, convirtiéndola en una herramienta práctica, de uso fácil y útil.

Figura 6: Logo de Latex[8]

- **HTML**[9]: o Lenguaje de Marcas de Hipertexto, es la base del diseño web, estableciendo

la estructura del contenido de las aplicaciones web mediante el uso de etiquetas. Define la estructura de un documento como es el título, cabecera cuerpo. Se puede combinar con lenguajes como CSS para establecer la apariencia de una web y *JavaScript* para definir el funcionamiento de una web.

Figura 7: Logo de HTML[10]

- **CSS**[11]: o estilo de hojas en cascada es un lenguaje de estilos usado para describir la apariencia de un archivo que use lenguaje marcado como una plantilla *HTML*. *CSS* está pensado para separar la parte de contenido del documente de la forma en la que este se muestra, como son el tamaño de la fuente, colores o tipología. Con esto se busca mayor accesibilidad a la información, así como que distintos documentos *HTML* tengan el mismo archivo *CSS*, y por consiguiente, el mismo estilo sin necesidad de repetir el código referente a su apariencia.

Figura 8: Logo de CSS[12]

- **JavaScript**[13]: es un lenguaje de secuencias de comandos que da funcionalidad a las páginas web. Se encarga de controlar elementos que no permanecen estáticos como son animaciones 3D o mapas interactivos. Junto con HTML y CSS, es una de las tecnologías estándar para el desarrollo web.

Figura 9: Logo de JavaScript[13]

- **Flask**[14]: es un micro servicio web desarrollado para *Python*. Permite crear aplicaciones web de forma sencilla y rápida. Sigue un patrón conocido como MVC; modelo, vista y controlador. Este sistema o patrón sirve para diferenciar los datos que van a estar integrados en una base de datos (modelo), de la página HTML (vista) y de dónde se gestionan las peticiones GET y POST de la aplicación web (controlador). Al ser un micro-servicio, Flask posee las herramientas que se necesitan para crear aplicaciones y que sean funcionales, pudiendo añadir extensiones si fuese necesario.

Figura 10: Logo de Flask[14]

- **Anaconda**: es una reconocida distribuidora de *Python* que incorpora e integra todas las herramientas de *Python*, así como sus versiones. Lo más interesante de esta herramienta es que permite crear entornos virtuales con diferentes versiones de *Python* y diferentes paquetes instalados según las necesidades o proyecto en el que se esté trabajando. EN estos entornos se tienen aplicaciones como *pyCharm*, y la más distintiva de Ananconda, *Jupyter Notebook*. *Jupyter* es al igual que *Google Colab* una plataforma que permite combinar líneas de código con líneas de texto. Ananconda será utilizada para realizar el despliegue de la aplicación web.

Figura 11: Logo de Anaconda [15]

- **PyCharm** [16]: es IDE de Python, esto quiere decir que además de ser un editor de código, cuenta con un depurador e intérprete que ayudan a la creación de programas y su exportación. El intérprete servirá de ayuda para solucionar los errores que vayan surgiendo. Además tiene una versión redistribuida para Linux, lo que hace que su uso esté más extendido entre la comunidad de desarrolladores.

Figura 12: Logo de PyCharm [17]

- **GitHub**[18]: es un portal pensado para almacenar el código de cualquier programa y aplicación contando con un control de versiones. De manera que se pueda gestionar todas las versiones de un mismo programa sin causar conflictos, pudiendo incluso llegar a versiones anteriores. Permite además la colaboración entre desarrolladores, pudiendo ver e implementar código de otros usuarios en tu propia máquina. Es un portal multiplatarforma muy usado hoy en día y que permite colaborar a muchos usuarios en un mismo proyecto sin que haya conflictos.

Figura 13: Logo de GitHub[18]

2

Conceptos teóricos

2.1. Cardiopatía

Una cardiopatía [19] es toda aquella enfermedad que afecta al corazón, sea cual sea su origen. Es una de las principales causas de muerte en el mundo puesto a que afecta a todos los géneros y razas. Hay muchos tipos de ellas, como defectos cardíacos congénitos, infarto del miocardio o la estenosis coronaria.

2.1.1. Estenosis coronaria

Es el tipo más común de las enfermedades cardíacas. Se produce cuando las arterias que suministran sangre al corazón se taponen con sustancias como el colesterol. Como consecuencia, en las arterias se produce un estrechamiento y endurecimiento. Otra de las consecuencias es la reducción de sangre que llega al corazón, lo que por consiguiente significa una reducción de oxígeno y de nutrientes. Con el tiempo esto va a producir que se debilite el musculo cardíaco, lo que puede desembocar en desarrollar otro tipo de patologías cardíacas más graves como pueden ser las arritmias o llegar a que se produzca un infarto del miocardio.

2.2. Coronariografía

Es un procedimiento médico muy poco invasivo que se realiza para la evaluación del estado de las arterias coronarias, así como de localizar obstrucciones en las mismas. Para realizar ese procedimiento se ha de introducir un *catéter* (tubo de plástico de solo 2 mm de diámetro) en las arterias mediante una pequeña abertura o punción. El catéter se va a desplazar por el sistema circulatorio hasta llegar a las arterias coronarias, dónde inyecta un líquido de contraste para teñirlas. De esta manera permite al angiógrafo, equipo médico de **rayos X**, mostrar el estado de las arterias.

De este procedimiento se obtendrá las **imágenes coronariográficas** como la observada en la Figura 14, clave para este estudio, pues serán las utilizadas para entrenar los modelos y ayudar al diagnóstico.

Figura 14: Imagen coronariográfica

2.3. Angioplastia coronaria

Si finalmente tras el procedimiento anterior se diagnostica como persona que sufre estenosis coronaria, se procederá a realizar una angioplastia coronaria[20]. La angioplastia se trata de introducir un catéter para hacer llevar a la obstrucción un pequeño globo deshinchado por un extremo, que se colocará comprimiendo dicha obstrucción y como consecuencia, la arteria se abre permitiendo que la sangre pueda circular por ella.

Además en la mayoría de casos se introduce también un *stent*. Este *stent* es un tubo hecho de malla metálica que se expande en la obstrucción y se adhiere a la pared de la arteria, permitiendo así dejar el vaso abierto. Este procedimiento se puede observar en la Figura 15

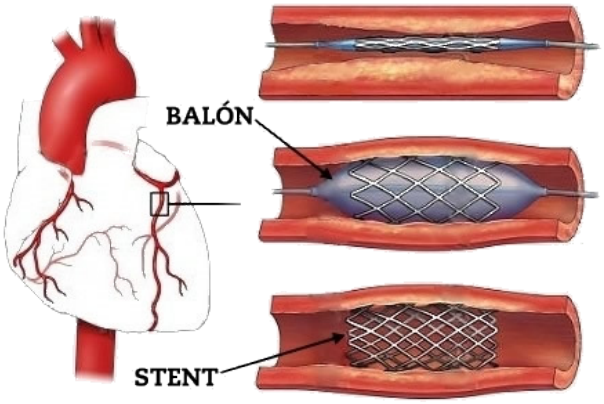

Figura 15: Procedimiento de angioplastia coronaria[21]

Es por esto que es esencial un buen diagnostico tanto de si se presenta o no estenosis, como de saber dónde se encuentra dicha estenosis, pues la colocación del stent será clave para la vida del paciente.

2.4. Aprendizaje profundo

El aprendizaje profundo es una de las bases de la **Inteligencia Artificial**, que se encuentra en auge debido a a disponibilidad de gran cantidad de datos, la era de *Big Data*, el procesamiento gráfico (**GPU**) y el procesamiento tensorial (**TPU**). Esta técnica simula el aprendizaje humano (neuronas que transmiten información a través de impulsos nerviosos) con el fin de obtener ciertos conocimientos. Es por ello por lo que destaca en el **aprendizaje no supervisado**, ya que no requiere reglas programadas previamente, sino que es capaz de aprender por sí mismo hasta llegar al resultado óptimo.

El aprendizaje profundo tiene 3 objetivos:

- Aprender por sí mismo, es decir, en la fase de entrenamiento el algoritmo va evaluando poco a poco su trabajo y añadiendo pequeñas modificaciones que hacen que se mejore.
- Obtener y procesar la información de entrada, es decir, el algoritmo debe ser capaz de reconocer diferentes representaciones de datos y procesarlos para su posterior uso.
- Utilizar el modelo de aprendizaje para reconocer **patrones** en los datos de entrada, es decir, que reconozca cierta información de interés y clasifique los diferentes tipos de objetos, como se puede ver en la Figura 16.

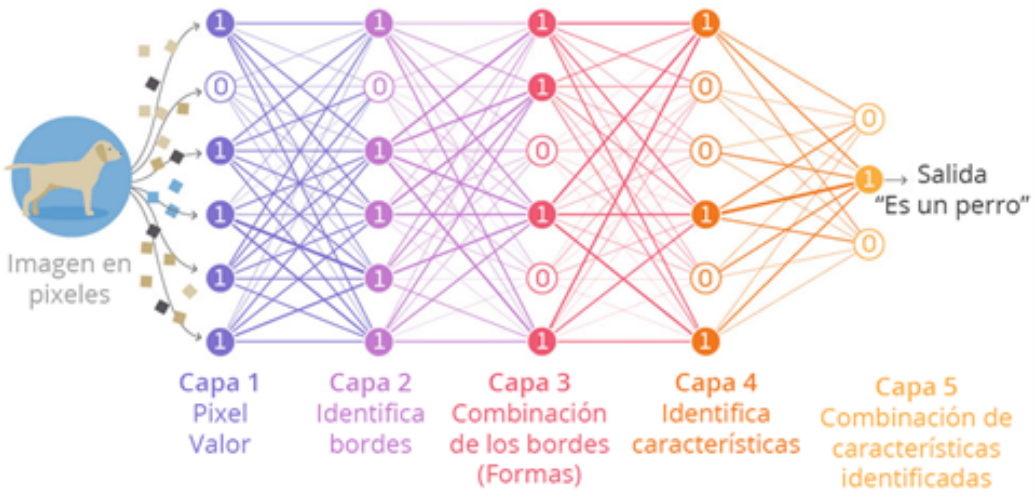

Figura 16: Funcionamiento del aprendizaje profundo[22]

2.4.1. Aprendizaje profundo aplicado a la salud

Actualmente el aprendizaje profundo se utiliza para infinidad de cosas que se usan en el día a día. Un claro ejemplo son nuestros móviles, desde traducir un texto a otro idioma a desbloquear el móvil con nuestra cara se está usando el aprendizaje profundo. Es una técnica que ha llegado a todos los ámbitos, incluido la medicina.

Uno de los principales uso en la salud del aprendizaje profundo es **ayudar al diagnóstico**; y no solo eso, sino que gracias a él se optimizan gastos, se mejora la atención a los pacientes y se acelera el sistema sanitaria, puesto a que gracias a un algoritmo de detección de cierta enfermedad, el médico podrá atender directamente a esos pacientes que hayan sido clasificados como positivos y así intervenir a tiempo, un punto esencial en la recuperación del paciente.

2.4.2. Estructura

El aprendizaje profundo se basa en redes neuronales artificiales entrelazadas para el procesamiento de información. Estas redes tienen capas no lineales en las que se encuentran las neuronas. Cada neurona de la capa está conectada con todas las neuronas de la siguiente capa, haciendo que se transmita información de unas a otras. Cada una de estas conexiones tiene asociado un peso. Cuentan con 3 tipos de capas como se observa en la Figura 17.

- **Capa de entrada (*input layer*)**: capa donde se introducen los datos de entrada, a partir de los cuales la red tomará decisiones.

- **Capas ocultas (*hidden layers*)**: es aquí donde se producen una serie de complejas operaciones matemáticas que determina el camino a seguir por la red.

- **Cada de salida (output layer)**: capa donde la red neuronal coloca el resultado final de los procesos de las capas ocultas

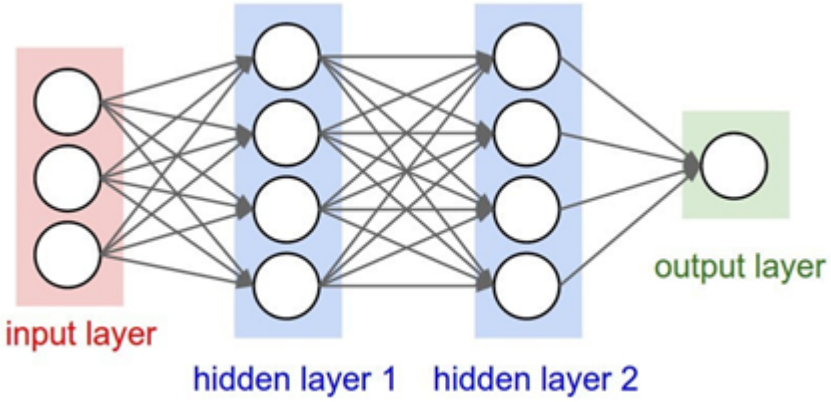

Figura 17: Capas de una red neuronal[23]

Dependiendo del problema y los datos que estemos tratando, podemos elegir el número de capas que vamos a implementar en la red neuronal, de ahí viene el concepto de profundidad de la red, '*deep*'. El aprendizaje profundo se utiliza para tareas de clasificación, regresión y optimización, pero generalmente, y concretamente en este trabajo, se emplea para la clasificación de imágenes.

2.4.3. Redes neuronales densas

Son redes neuronales clásicas dónde cada capa intermedia está conectada a todas las neuronas de la capa siguiente, así como a la anterior como se muestra en la Figura 18. Al ser una red muy conectada, es una red que aprende muchas características[27], lo que hace que su proceso de aprendizaje sea más lento que el que siguen otras redes neuronales.

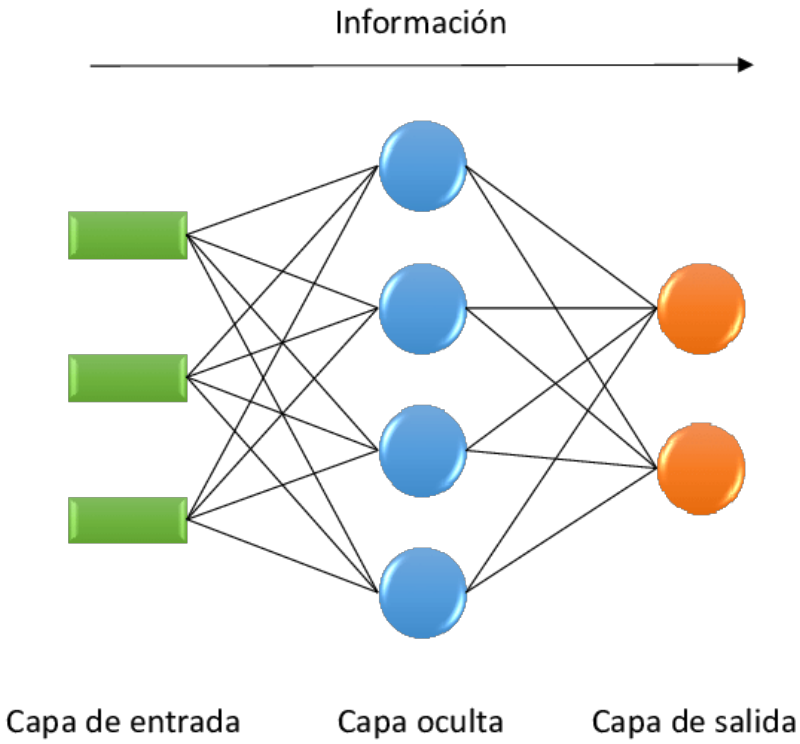

Figura 18: Red neuronal densa[28]

2.4.4. Redes neuronales Convolucionales (CNN)

Las CNN[29] son las redes más utilizadas. Son un tipo de red neuronal que procesa sus capas imitando a la corteza visual del ojo humano para poder distinguir características que en definitiva es lo que hace que pueda identificar objetos. Este tipo de modelo es muy útil a la hora de clasificar y segmentar imágenes, como para detectar y categorizar objetos.

El objetivo principal es aprender características utilizando la **operación de convolución**, que no es más que filtrar una imagen (matriz bidimensional) utilizando una máscara (matriz bidimensional más pequeña que la de entrada) a modo de filtro. Es decir, cada píxel de salida es una combinación lineal de los píxeles de entrada. Las máscaras representan las conexiones entre neuronas de capas anteriores. Cada máscara producirá un resultado diferente.

Estos modelos tienen 3 etapas: convolución, agrupación o ***pooling*** y no linealidad. Dichas etapas pueden repetirse varias veces y combinarse entre así para hacer una CNN profunda como se ve en la Figura 19.

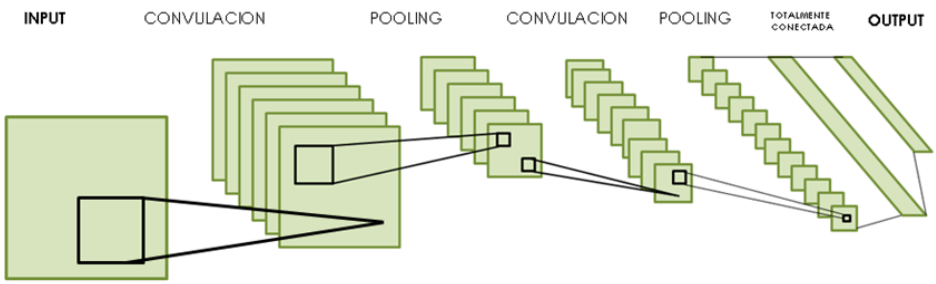

Figura 19: Etapas red neuronal[29]

Las CNN utilizan principalmente dos tipos de capas o *layers*, **convoluciones y *pooling***. La capa más importante, y la que da nombre a la red, es la capa convolucional. Esta capa utiliza unos filtros de tres dimensiones de pequeño tamaño, que va desplazándose por la imagen obteniendo las salidas de la capa. Es decir, transforma los datos de entrada. Tras este proceso, interviene la capa de *pooling*, cuya función es resumir las respuestas de las salidas cercanas. En resumen, se encarga de reducir progresivamente el tamaño espacial de los datos y después los agrupa, ayudando así a obtener una representación invariable. Este proceso se puede observar en la Figura 20.

También puede usar capas como:

- ***Dropout***: es una capa que desactiva aleatoriamente un número de neuronas. Sirve como regularizador para así intentar reducir los posibles sobreajustes que suceden en las redes neuronales.

- Flatten: es una capa que tiene que ir seguida de una capa densa. Esto es porque transforma los datos en vectores de una dimensión (1D) para que la capa densa pueda usarlo.

- **Densa**: son las capas de cálculo que conectan para una capa, cada neurona con todas las salidas de la capa previa.

Para que el modelo pueda llevar a cabo este proceso, se necesita una gran cantidad de imágenes para que la red puede recoger todas las características únicas que distinguen al objeto y así poder generalizarlo; por ejemplo, en el caso de un gato, que reconozca también como gatos a uno blanco, negro, un gato de frente o saltando.

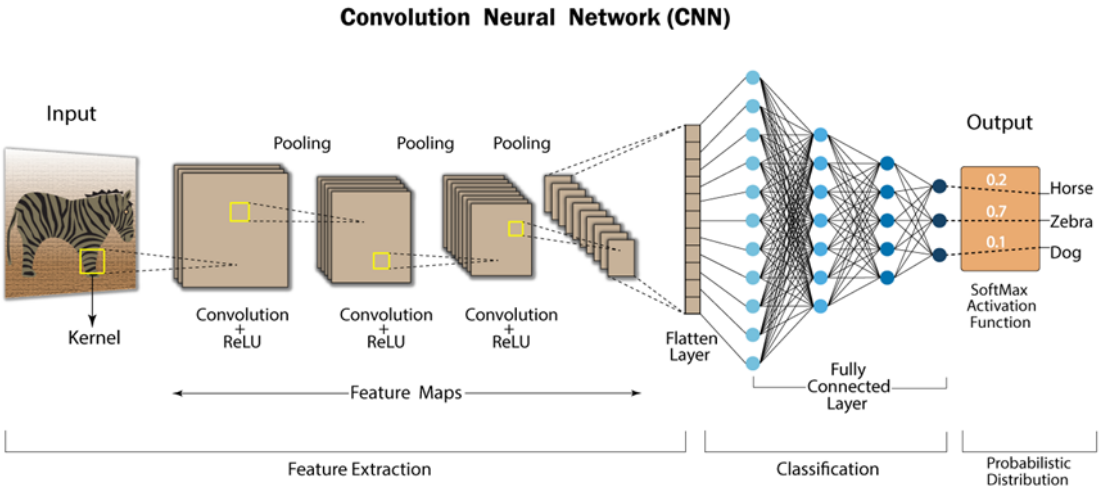

Figura 20: Red neuronal convolucional (CNN)[30]

3 Metodología

3.1. Conjunto de datos

El conjunto de datos con el cual se va a realizar este estudio pertenece a un repositorio público de *GitHub*[31]. Dicho conjunto se basa en imágenes coronariográficas de tamaño 32x32. Cuenta con dos clases distintas, imágenes que presentan estenosis e imágenes que no presentan dicha estenosis. Cada clase respectivamente cuenta con 125 y 1394 imágenes que están representadas en escala de grises. Las imágenes que se presentan en este repositorio son *pacthes* o regiones de una imagen inicial, que ha sido dividida en estas regiones de 32x32, para que posteriormente sirvieran para entrenar el modelo y evaluarlo, respondiendo al objetivo de este estudio de si una imagen coronariográfica presenta o no estenosis. Este proceso se puede observar en el esquema de la Figura 21.

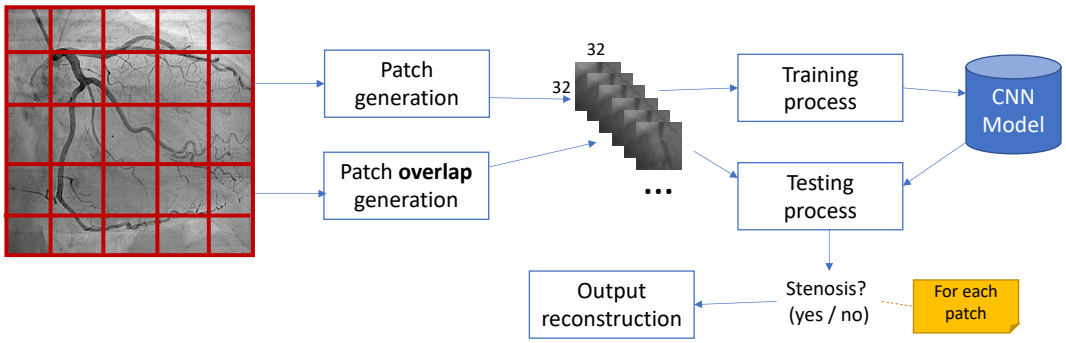

Figura 21: Proceso de división de una imagen coronariográfica en regiones para el posterior entrenamiento y evaluación del modelo

Debido a este desbalanceo de clases, al conjunto de datos se le aplicarán diferentes técnicas para conseguir equilibrar esa descompensación y tratar de mejorar los resultados del entrenamiento del modelo. En la Figura 22 se puede apreciar una de las imágenes del conjunto sin estenosis y en la Figura 23 una de las imágenes que si presenta estenosis.

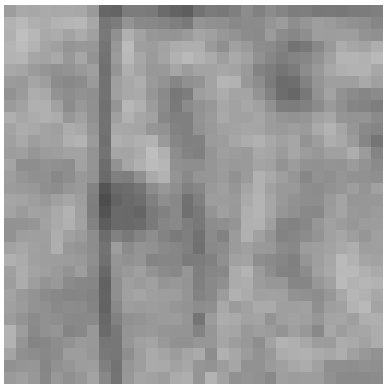

Figura 22: Imagen sin estenosis

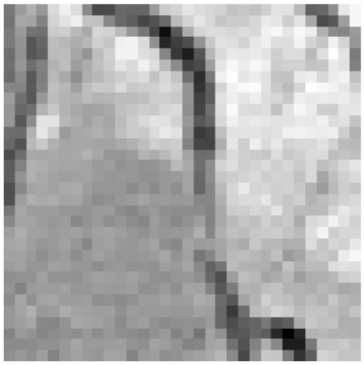

Figura 23: Imagen con estenosis

3.2. Preprocesamiento

El conjunto de imágenes del cual partimos está en formato *pickle*, una serialización de las matrices de las imágenes que contiene. Dichas matrices presentan un solo canal, es decir, están en escala de grises. Esto significa que cada píxel posee un valor que está comprendido en un rango de blanco a negro con valores en un rango comprendido entre 0 y 255.

3.3. Arquitectura de la red

En este trabajo se van a seguir dos estrategias distintas para el entrenamiento de los datos. La primera estrategia se basa en crear desde cero una red neuronal. La segunda estrategia se basa en usar diferentes redes pre-entrenadas para así, intentar mejorar los resultados obtenidos con la primera estrategia, en este estudio se han usado *ResNet50, EfficientNetB4 y MobileNet.*

3.3.1. Red Neuronal densa entrenada desde cero

La red neuronal que se ha diseñado para este trabajo es una red densa con una estructura bastante sencilla. El hecho de que solo se utilicen capas densas implica que no se tiene en cuenta la vecindad de los píxeles.

Una imagen se introducirá en la red normalizada en una escala de [0,1], en primer lugar es vectorizada, y a continuación se le aplica un capa densa usando una función de activación ReLU. Tras esto, se aplica una capa de *dropout* que servirá para evitar el sobreajuste del modelo, estableciendo de forma totalmente aleatoria un 50 % de la entrada a 0. Para la salida se usa una capa densa con una función de activación sigmoidal puesto a que se está trabajando con un problema de clasificación binaria con las clases estenosis o no estenosis. Se puede ver un esquema de la arquitectura descrita en la Figura 24.

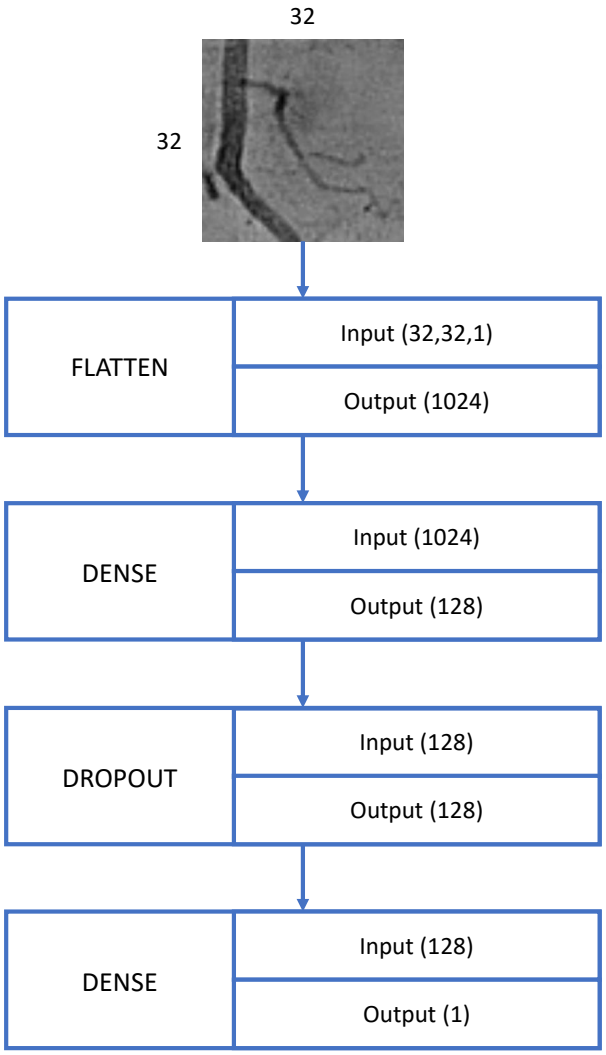

Figura 24: Arquitectura seguida por la red neuronal densa entrenada desde cero

3.3.2. Red Neuronal Pre-entrenada

Las tres redes pre-entrenadas que se van a usar constan de la misma estructura una vez se importan sus pesos al modelo y se excluye la primera capa del modelo pre-entrenado, es decir, para el proceso de entrenamiento de la red pre-entrenada, no se ha usado ni *Transfer Learning* ni *Fine Tunning*, si no que se ha entrenando la red desde cero inicializando los pesos de *imagenet*. Esto se debe a que el conjunto de datos de *imagenet* con el que fueron entrenadas estas redes, pese a tener unas 1000 clases, ninguna es similar a las que se están usando en este estudio, por lo que las características que hayan aprendido sus capas intermedias, pueden no ser necesariamente útiles

.

Cada red pre-entrenada tiene un tamaño de datos de entrada predefinido, sin embargo se pueden añadir capas que nos permitan introducir el tamaño que se desee para entrenar el modelo. Una de estas capas es **GlobalAveragePooling2D** [32] que nos permite, como se ha comentado anteriormente, introducir el tamaño de imágenes de nuestro conjunto de imágenes, en nuestro caso de 32x32, ya que esta capa produce una salida que solo depende del número de núcleos y no del tamaño del conjunto de las imágenes. Posteriormente se aplica dos veces el proceso de usar una capa densa con activación ReLU y una capa *dropout* que servirá para prever el sobreajuste del modelo. Para la salida se utiliza una capa densa con una función de activación sigmoidea debido a que es un problema de clasificación binaria. Además, las imágenes del conjunto tendrán que ser preprocesadas con la función de *Keras* **preprocess_input** antes de ser introducidas en el modelo para adaptarlas a él.

3.3.3. *ResNet50*

ResNet50[33] o red residual, se inspira en el proceso biológico humano, en el que algunas neuronas se conectan con otras que se encuentran capas que no están seguidas, haciendo un salto entre las capas que se encuentren en medio. El aumento del número de capas de una red hace que aumente su precisión, pero produce desaparición del gradiente (***vanishing gradient***). Una solución a este problema es el uso de redes residuales, que permiten construir redes muy profundas pero manteniendo su eficiencia mediante el uso de **conexiones atajo** que conectan capas que no se encuentran seguidas. ResNet50 se diferencia del resto de ResNet en que esta utiliza 50 capas de profundidad.

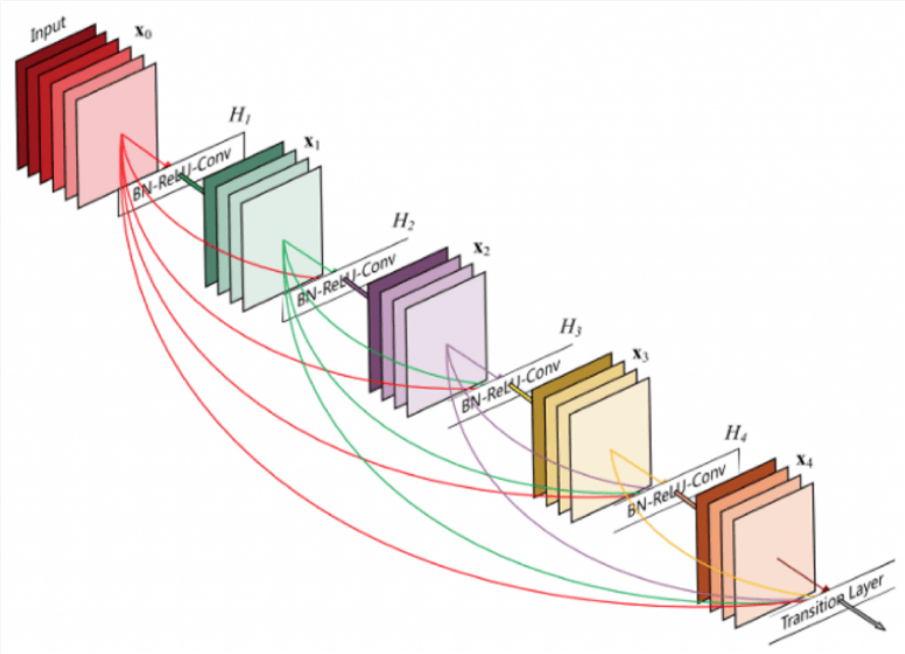

Figura 25: Arquitectura seguida por la red ResNet50[36]

3.3.4. *EfficientNetB4*

EfficientNet [38] agrupa a un conjunto de modelos que van desde la B0 hasta el B7. En este estudio se usa la B4. Estos modelos, a diferencia de la mayoría de modelos que usan *ReLU*, implementan una función de activación conocida como *Swish*. De esta manera consigue aumentar el rendimiento en los problemas de clasificación, además de obtener resultados más eficientes escalando las tres dimensiones que tiene una red neuronal convolucional (CNN): profundidad, anchura y resolución. Esto se puede ver en la Figura 26. El resto de modelos se suele centrar en tan solo una de estas capas, por lo que esto es una de las cosas que hacen que sea más eficiente.

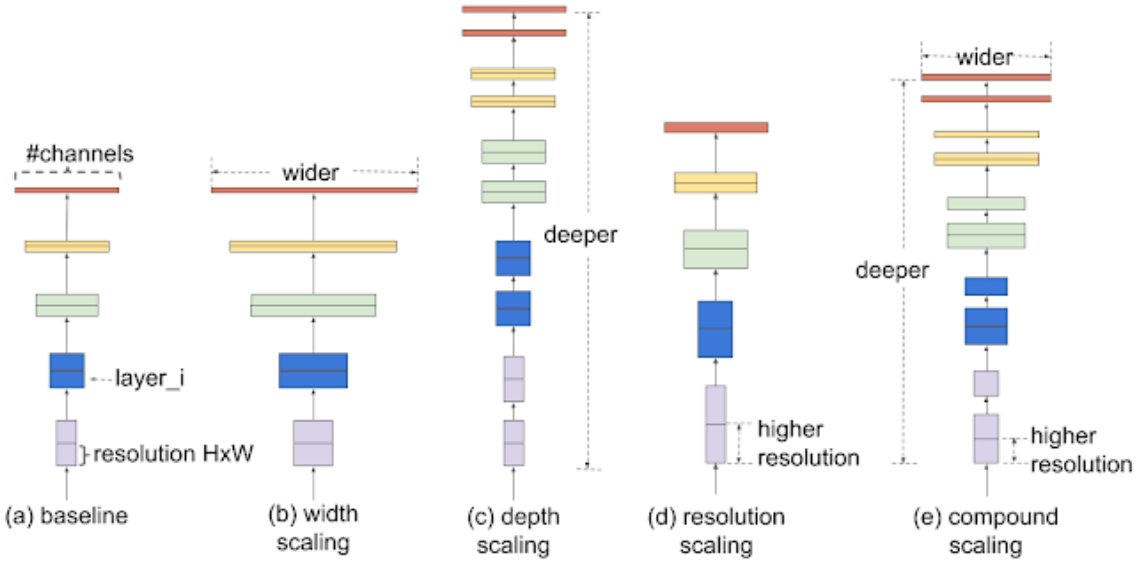

Figura 26: Arquitectura seguida por la red *EfficientNetB4*[40]

3.3.5. *MobileNet*

MobileNet[41][42] hace uso de convoluciones separables en profundidad, que se encargan de factorizar lo conocido como convolución estándar en dos salidas. Por un lado, en una convolución en profundidad que hace uso de un solo filtro por cada entrada, y por el otro lado, en una convolución puntual (1x1) que sirve para combinar los resultados obtenidos con la convolución en profundidad. El fin de hacer esto es reducir el gasto computacional reduciendo la expresión que es redundante. En la Figura 27 se puedo observar este proceso dónde pasa de una convolución estándar a una separable en profundidad y una puntual 1x1.

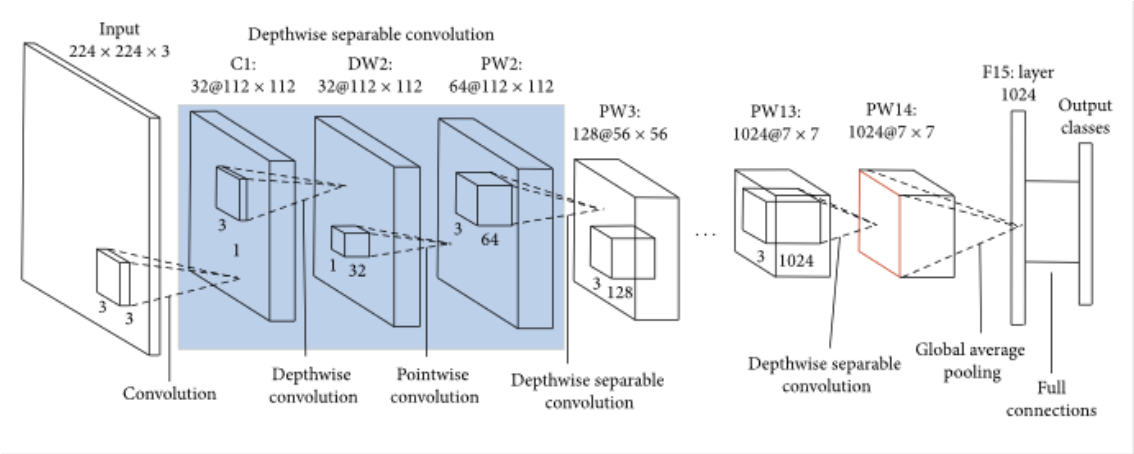

Figura 27: Arquitectura seguida por la red *MobileNet*[43]

3.4. Optimizador, función de pérdida y de activación

Para la red neuronal densa entrenada desde cero, se ha utilizado un optimizador llamado *Stochastic Gradient Descent***(SGD)**. SGD es un optimizador que de forma aleatoria selecciona una muestra dentro de cada *batch* o lote para calcular el descenso de gradiente. A esta selección aleatoria se le llama comportamiento estocástico. Necesita muchas iteraciones para llegar a los mínimos a causa de su aleatoriedad. Como parámetro se usa *momentum* el cual se encarga de acelerar el descenso de dicho gradiente.

Para las redes pre-entrenadas se intentó usar SGD, pero este se quedaba atrapado en muchos mínimos locales, por lo que finalmente se ha usado un optimizador *Adam* que también se basa en el descenso de gradiente estocástico a partir de estimaciones adaptativas de momentos. Se recomienda usar *Adam* ya que es eficiente, necesita poca memoria y da muy buenos resultados.

La función de pérdida para ambas redes será binary crossentropy, una función muy usada en los problemas de clasificación binaria como el de este trabajo.

Para ambos modelos la función de activación, que es la que define la salida del modelo, debido a que es un problema de clasificación binaria, se ha usado la **función de activación Sigmoidal**[44] la cual devuelve valores en una escala [0,1].

3.5. Métricas

Para obtener buenos resultados en la etapa de evaluación del modelo, es necesario definir unas métricas que se adecuen a nuestro problema. Lo primero a tener en cuenta para elegir dicha métrica, es saber cuales son las características del conjunto a evaluar. Dicho conjunto, en este estudio, está muy desbalanceado, teniendo una clase mayoritaria con 1300 muestras que corresponden a *negativo en estenosis* y una clase minoritaria con tan solo 125 muestras, que se corresponde a presentar *positivo en estenosis*.

3.5.1. *Accuracy*

El *accuracy* mide, tanto para las imágenes positivas como negativas, cuantas se han clasificado adecuadamente. Sin embargo, esta medida para nuestro problema de clasificación puede ser engañosa, pues se obtienen valores bastante altos de *accuracy* que únicamente corresponden a la buena clasificación de la clase mayoritaria aunque el acierto en la clase minoritaria sea muy bajo. Por lo que dicha medida queda descartada.

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}} \tag{1}$$

3.5.2. F1 score

La métrica F1 es una media armónica entre la medida *recall* y *precision*. Por lo que es necesario explicar primero las medidas de **precision** y **recall**.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2}$$

- *Recall*: mide la capacidad de un modelo de clasificar muestras positivas, sin tener en cuenta como se clasifican las negativas.

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}} \tag{3}$$

- *Precision*: mide la exactitud de un modelo de clasificar muestras como positivas.

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}} \tag{4}$$

La medida F1 suele ser una buena medida para los problemas de clasificación, sin embargo, para este caso no lo tendríamos en cuenta ya que al se un conjunto muy desbalanceado, y dicha medida solo fijarse en como se clasifican las muestras positivas, varían demasiado los resultados en función a si se considera la clase estenosis como positiva o por el contrario es la clase no estenosis la que se considera como positiva.

3.5.3. *Balanced Accuracy*

Esta medida es especialmente útil en los problemas de clasificación como el nuestro dónde las clases se encuentran muy desbalanceadas, ya que refleja la precisión del modelo para clasificar correctamente ambas clases sin que el número de muestras de cada una de ellas sea relevante. Es por esto que la usaremos como la medida para evaluar el modelo.

$$BalancedAccuracy = \frac{Sensitivity + Specificity}{2} \tag{5}$$

dónde:

- *Sensibilidad* **(TPR)**: mide la capacidad de el modelo de detectar como positivas aquellas muestras que son realmente positivas.

$$Sensitivity = \frac{True_{positive}}{True_{positive} + False_{negative}} \tag{6}$$

- *Especificidad* **(TNR)**: mide la capacidad de el modelo de detectar como negativas aquellas muestras que son realmente negativas.

$$Specificity = \frac{True_{negative}}{True_{negative} + False_{positive}} \tag{7}$$

3.6. Validación cruzada

La validación cruzada es una técnica muy extendida en proyectos de inteligencia artificial que permite evaluar, en este caso, los resultados de un modelo, asegurando que dichos resultados son totalmente independientes de la división que se le ha realizado al conjunto en entrenamiento y test. El conjunto se divide en entrenamiento y test según el número de cajas o K que se le indique. Para cada K, el conjunto de entrenamiento y test será totalmente distinto. Una vez se hayan ejecutado todas las K, se realizará una media aritmética de los resultados obtenidos de la validación para cada partición[45]. Esta técnica se puede observar en la Figura 28.

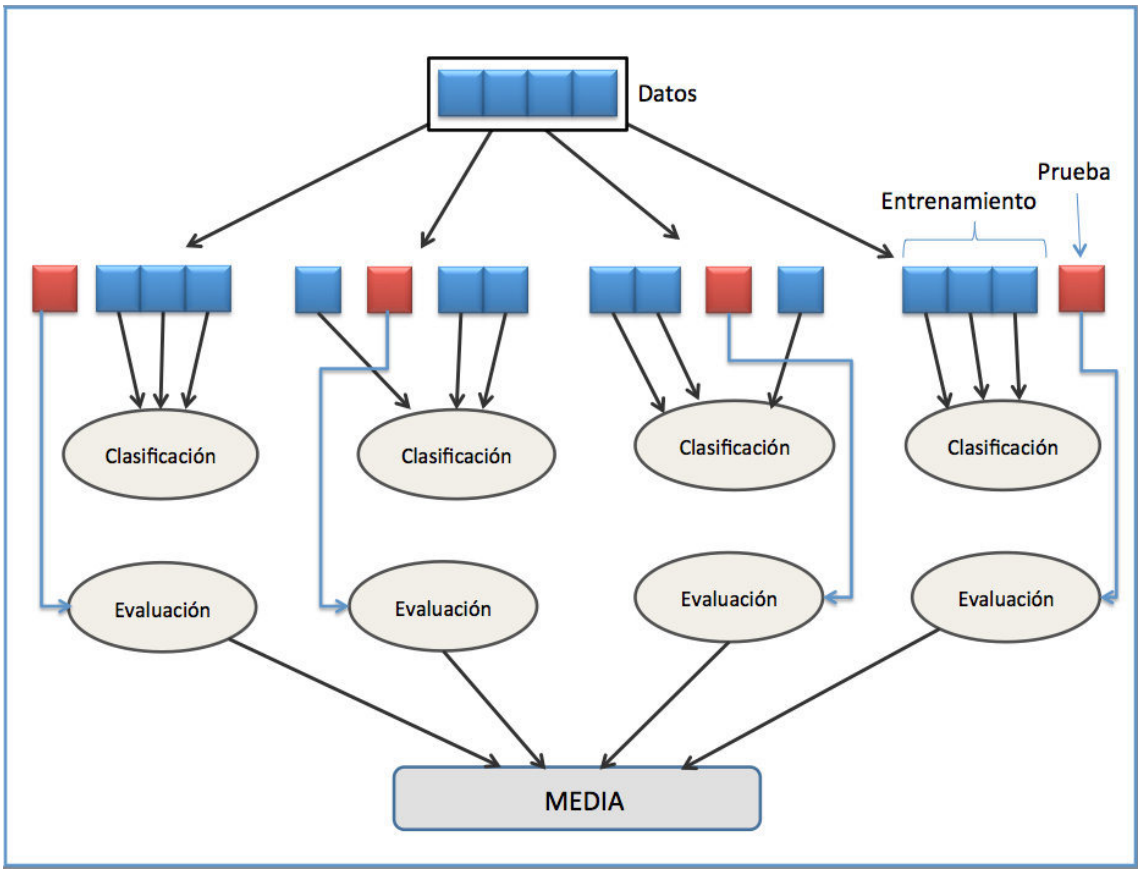

Figura 28: Validación cruzada[47]

3.7. Aumento de datos

Tener conjuntos muy desbalanceados hará que los resultados procedentes del entrenamiento no sean buenos o fiables puesto que el modelo predecirá con mucha precisión la clase mayoritaria, mientras que será muy nefasto clasificando la clase minoritaria. Para solucionar esta situación se han aplicado diferentes técnicas de aumento de datos al conjunto de entrenamiento. Además se usará **validación cruzada** para mejorar aún más los resultados.

De manera que usando la función *StratifiedKFold* que dividirá el conjunto inicial en 5 cajas con muestras de ambas clases. Para cada K se dividirá el conjunto inicial en entrenamiento (80 %) y test (20 %) y se aplicarán diferentes técnicas de aumento de datos únicamente al conjunto de entrenamiento de la clase minoritaria (estenosis). De esta manera se consigue que para cada K el conjunto de entrenamiento, al igual que el de test, sea distinto y el modelo tenga más datos diferentes con los que aprender y con los que predecir.

3.7.1. *Sampling and augmentation*

La técnica de *Muestreo y aumento de datos* (*Sampling and augmentation*) se basa en balancear los conjuntos de estenosis y no estenosis igualando sus muestras a un punto medio, de forma que se aumenta la clase minoritaria y se reducen el número de muestras de la mayoritaria.

Para esta técnica de aumento de datos se ha creado un objeto *ImageDataGenerator*, el cual se encargará de generar imágenes con diferentes transformaciones a partir de las originales. Las transformaciones aplicadas son 15 grados de rotación, desplazamiento del 20 % a lo ancho y volteo o *flip* horizontal y vertical. Este objeto recibe las imágenes del conjunto de entrenamiento que presentan estenosis y genera 500 nuevas imágenes. A su vez se eligen de forma aleatoria 600 imágenes que no presentan estenosis del conjunto de entrenamiento. De manera que el conjunto quedaría balanceado, entrenando al modelo con un número equilibrado de muestras para cada clase, 500 generadas + 90, aproximadamente, naturales de estenosis y 600 imágenes de no estenosis.

3.7.2. *Minority class augmentation*

Para la técnica de *Aumento de la clase minoritaria* (*Minority class augmentation*), se va a realizar un balanceo más agresivo que el usado en la técnica anterior, pues se va a igualar el conjunto minoritario, al conjunto mayoritario. Para ello se parte de unas 90 imágenes que presentan estenosis del conjunto de entrenamiento y se generan 1000 imágenes nuevas que han sufrido 15 grados de rotación, desplazamiento de 20 % a lo ancho y un volteo horizontal y vertical. Estas transformaciones se realizan usando un objeto *ImageDataGenerator*. Como resultado se ha conseguido un conjunto de entrenamiento balanceado con 1000 imágenes para cada clase.

3.7.3. *Proportional data augmentation*

En la técnica *Aumento de datos proporcional* o también llamado *Proportional data augmentation* se ha aumentado el conjunto de entrenamiento y validación sin tener en cuenta las etiquetas, es decir, se han aumentado indistintamente de si presentan o no estenosis. De igual manera que en las técnicas anteriores, se ha usado un objeto de *ImageDataGenerator*. A ambos conjuntos se le ha aplicado una rotación de 10 grados, un desplazamiento de 5 % a lo ancho y un volteo horizontal. Durante el entrenamiento, para cada caja de la validación cruzada, se van generando muestras modificadas para ambas clases con los parámetros anteriores que permitirán al modelo una mayor robustez proporcionando una gran variabilidad en la entrada de datos. El problema de esta técnica es que al generar nuevas imágenes de ambas clases, la

proporción desbalanceada seguirá existiendo.

3.7.4. *Synthetic data*

Otra de las técnicas que se han usado es el uso de un algoritmo que genera imágenes sintéticas desde cero. Dicho algoritmo[48] primero dibuja gradientes aleatorios de fondo, utiliza curvas de Bézier en representación de las venas, añade ruido blanco y aplica desenfoque gaussiano a toda la imagen.

Con esta técnica es posible generar el número de imágenes que queramos para ambas clases. Dichas imágenes sintéticas se distinguen fácilmente de las imágenes naturales. Sobre todo el conjunto natural de imágenes que no presenta estenosis tiene muchas imágenes con bordes, mientras que en el conjunto sintético siempre presenta al menos una vena. Es por esto que se necesita entrenar al modelo en una primera vez con el conjunto de datos sintético y una segunda con el conjunto de entrenamiento real.

3.8. Entrenamiento del modelo

El conjunto inicial se divide en entrenamiento (80 %) y test (20 %). Como se ha comentada en la sección anterior, el aumento de datos se realizará al conjunto de entrenamiento. Una vez ya se haya obtenido el conjunto de entrenamiento aumentado se volverá a dividir ese ese conjunto en entrenamiento(80 %) y validación (20 %). Ese conjunto de entrenamiento y validación serán los usados para entrenar y validar el modelo durante 1000 épocas.

Para evitar el sobre-entrenamiento del modelo se ajustará un regularizador llamado **early stopping**[49] o parada temprana. Esto consiste en evaluar el rendimiento del modelo durante su entrenamiento tanto para el conjunto de entrenamiento como el de validación. Al principio del entrenamiento, ambos conjuntos obtendrán buenas métricas, sin embargo llegará un punto en el que el rendimiento del conjunto de validación deje de mejorar respecto al conjunto de entrenamiento, aquí es cuando el modelo se puede empezar a sobre-entrenar y por lo tanto el método de *early stopping* pararía el entrenamiento. También se usa un **checkpoint**[51] del modelo, el cuál permite guardar el modelo en un punto del entrenamiento y cargar el modelo más tarde en ese punto. Con estas dos técnicas se consigue evitar el sobreajuste por un lado y la obtención del modelo que mejores resultados obtiene para la validación.

Con este ajuste de *early stopping* y *checkpoint* se pretende entrenar el modelo para obtener el mejor rendimiento posible, sin sobre-ajuste y por lo tanto, obtener mejores resultados en la etapa de evaluación.

4

Evaluación del modelo y Resultados experimentales

La evaluación de un modelo consiste en comprobar lo fiables que son sus resultados, de manera que se comparan las etiquetas predichas para el conjunto de test, con sus verdaderas etiquetas. Se van a evaluar todos los modelos obtenidos con las distintas combinaciones de aumento de datos junto con validación cruzada y los tipos de redes seleccionados. Para ello se va a usa el conjunto de test (20 % del conjunto inicial) y debido a que se sigue la estrategia de la validación cruzada, se evalúa el modelo para cada caja obteniendo un *balanced accuracy medio* . También se obtendrá para cada caja una **matriz de confusión**. Dicha matriz muestra para cada clase el número de predicciones correctas e incorrectas. Cada columna muestra el número de predicciones realizada para cada clase, aciertos y fallos. Por el contrario, cada fila muestra el número de apariciones reales de cada clase. Lo ideal es tener en la primera diagonal el mayor número de muestras clasificadas de cada clase, ya que representan aquellas muestras que se han clasificado correctamente.

En este problema observar la matriz de confusión es imprescindible puesto a que en el conjunto de test, el desbalance es muy notorio y lo que realmente interesa a este estudio es que clasifique muy bien a la clase con estenosis que es la minoritaria, pudiendo llegar a permitir que el modelo cometa más fallos prediciendo la clase sin estenosis mientras que la clase con estenosis la clasifique bien. Esto se puede observar en la Figura 29. Los valores que se representan dentro de la matriz de confusión son cuatro:

- **TN (*True negative*)**: el modelo ha clasificado como negativo (no estenosis) una muestra que realmente era negativa.

- **FN (*False negative*)**: el modelo ha clasificado como negativo (no estenosis) una muestra que realmente era positiva (estenosis).

- **TP (*True positive*)**: el modelo ha clasificado como positiva (estenosis) una muestra que realmente era positiva (estenosis).
- **FP (*False positive*)**: el modelo ha clasificado como positiva (estenosis) una muestra que realmente era negativa (no estenosis).

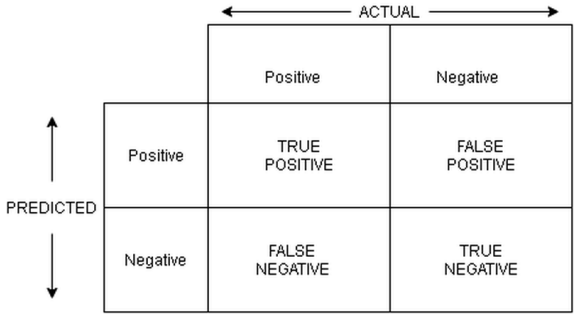

Figura 29: Matriz de confusión

A continuación se explica brevemente todas las combinaciones realizas en este proyecto y se mostrarán las matrices de confusión para dichas combinaciones junto con una gráfica que muestra el porcentaje de acierto para cada clase:

- **Red neuronal densa entrenada desde cero + Conjunto Inicial**: se ha evaluado el modelo resultante de dividir el conjunto de datos usando validación cruzada y entrenarlo con la red neuronal densa entrenada desde cero. Debido al desbalance del conjunto inicial entre la clase estenosis y no estenosis, el *balanced accuracy* para el conjunto medio de test es de 0.5, clasificando correctamente solo la clase mayoritaria, es decir, la que no presenta estenosis, y clasificando como sin estenosis aquellas muestras que la presentan. Esto se puede observar en su matriz de confusión en la Figura 30.

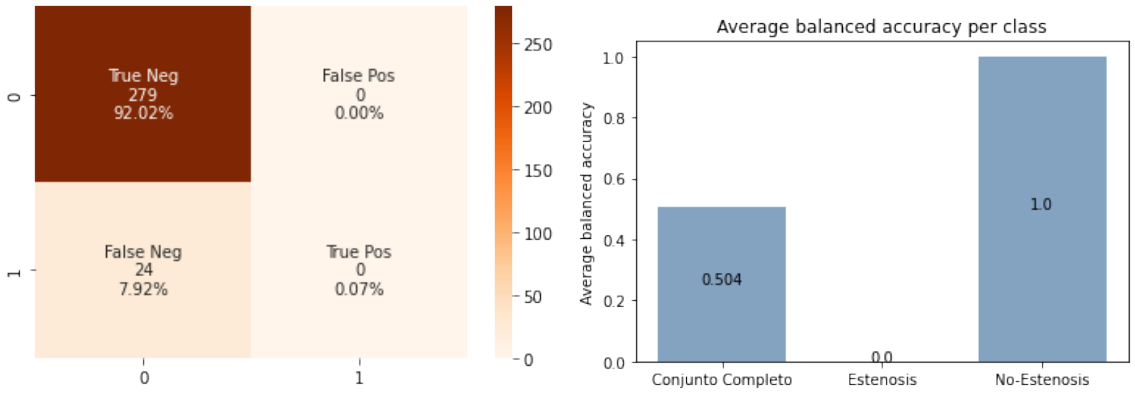

Figura 30: Matriz de confusión de la evaluación del modelo usando una red neuronal densa entrenada desde cero con el conjunto de datos inicial

- ***ResNet50* + Conjunto Inicial**: debido a los malos resultados del conjunto inicial con la red neuronal densa entrenada desde cero, se ha usado validación cruzada y entrenado el modelo con la red pre-entrenada ResNet50. El conjunto de datos debe preprocesarse previamente para poder ser introducido en el modelo y adaptarse a la entrada de la red. Se evaluará el conjunto de test obteniendo un *balanced accuracy medio* para el conjunto medio de test de 0.870, como podemos ver en la Figura 31.

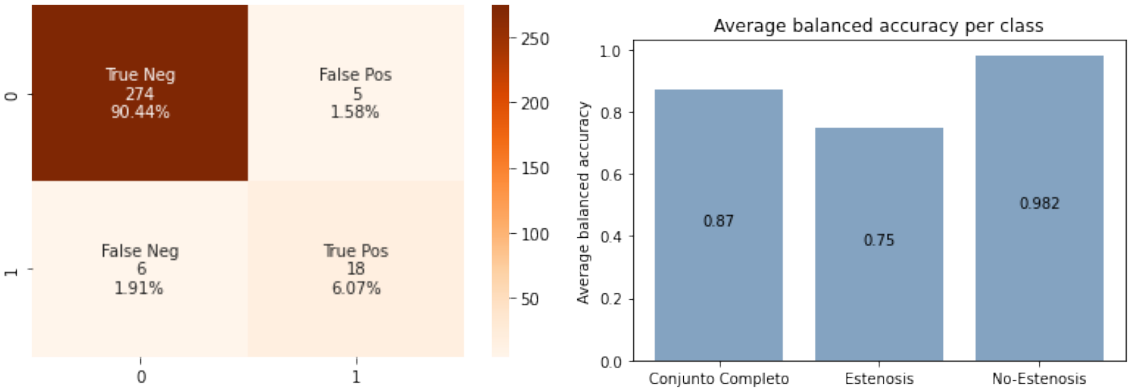

Figura 31: Matriz de confusión de la evaluación del modelo usando la red pre-entrenada *ResNet50* con el conjunto de datos inicial

- ***EfficientNetB4* + Conjunto Inicial**: con el conjunto de datos inicial, se ha probado con otra red pre-entrenada, *EfficientNetB4*. Se ha usado validación cruzada debido al desbalance, se ha entrenado el modelo para cada K y se ha evaluado con el conjunto de test, obteniendo un *balanced accuracy* medio de 0.860, la clasificación hecha 'para cada clase se puede ser en su matriz de confusión en la Figura 32.

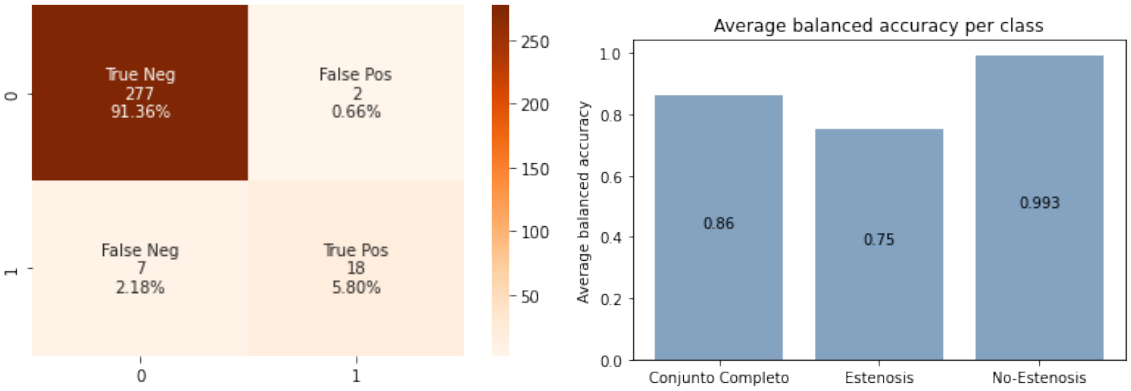

Figura 32: Matriz de confusión de la evaluación del modelo usando la red pre-entrenada *EfficientNetB4* con el conjunto de datos inicial

- ***MobileNet* + Conjunto Inicial**: la última red pre-entrenada usada para entrenar el mo-

delo inicial es *MobileNet*. De igual manera se ha usado validación cruzada para obtener unos resultados más fiables e independientes de la partición del conjunto en entrenamiento y test. Se ha entrenado el modelo y se evaluado. Se ha obtenido 0.891 de *balanced accuracy* medio. Su matriz de confusión se puede observar en Figura 33.

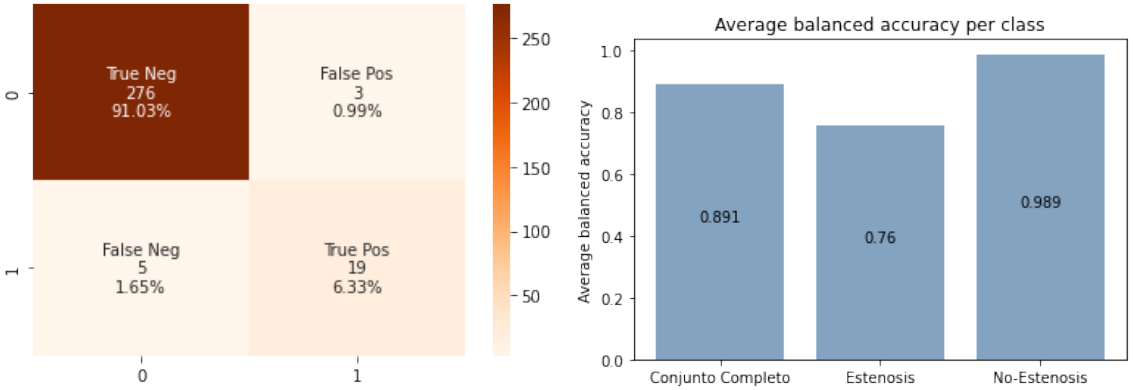

Figura 33: Matriz de confusión de la evaluación del modelo usando la red pre-entrenada *MobileNet* con el conjunto de datos inicial

- **Red neuronal densa entrenada desde cero + Muestreo y Aumento**: en este caso, para cada K se va a dividir el conjunto de datos en entrenamiento y test. Para el conjunto de entrenamiento se va a aumentar la clase minoritaria, en este caso, estenosis usando *ImageDataGenerator*, y se va a reducir la clase mayoritaria (no estenosis). Ambas clases quedan balanceadas a 600 imágenes para cada clase. Se entrena el modelo usando la red neuronal densa entrenada desde cero y tras evaluarlo, se obtiene un *balanced accuracy* medio de 0.792. Se ha conseguido mejorar muchísimo el acierto en la clase estenosis, pero como consecuencia se ha aumentado el número de falsos positivos. Esto puede verse en la matriz de confusión de la Figura 38.

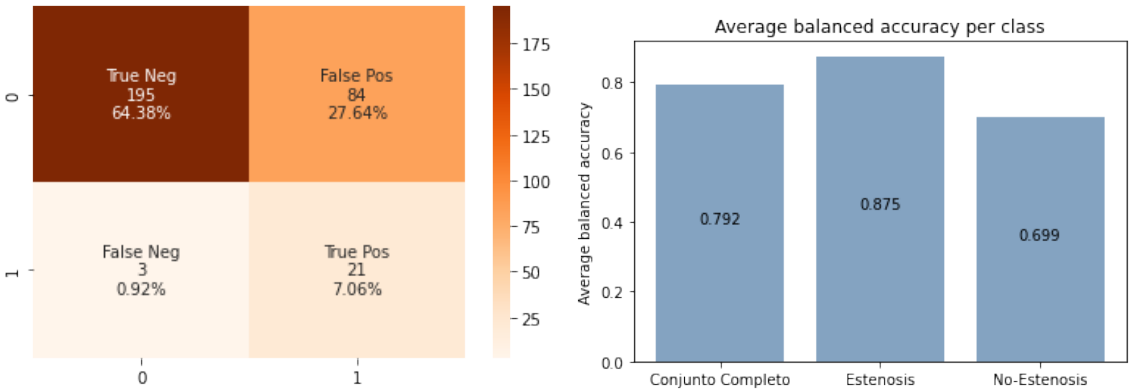

Figura 34: Matriz de confusión de la evaluación del modelo usando la red neuronal neuronal densa entrenada desde cero usando la técnica de muestreo y aumento de datos

- ***ResNet50* + Muestreo y aumento**: este proceso es igual que el anterior, de manera que se divide el conjunto en entrenamiento y test k veces. Para cada k se aumenta la clase minoritaria y se reduce la mayoritaria del conjunto de entrenamiento, teniendo un balance de 600 imágenes para cada clase. Para entrenar el modelo se usará una red pre-entrenada *ResNet50*, y debido a esto, se aplica la función de Keras *preprocess_input* al conjunto de entrenamiento y test para adaptarlos al formato del modelo. Tras la evaluación se obtiene un *balanced accuracy* medio de 0.917. Su matriz de confusión así como la precisión para cada clase se puede observar en la Figura 35.

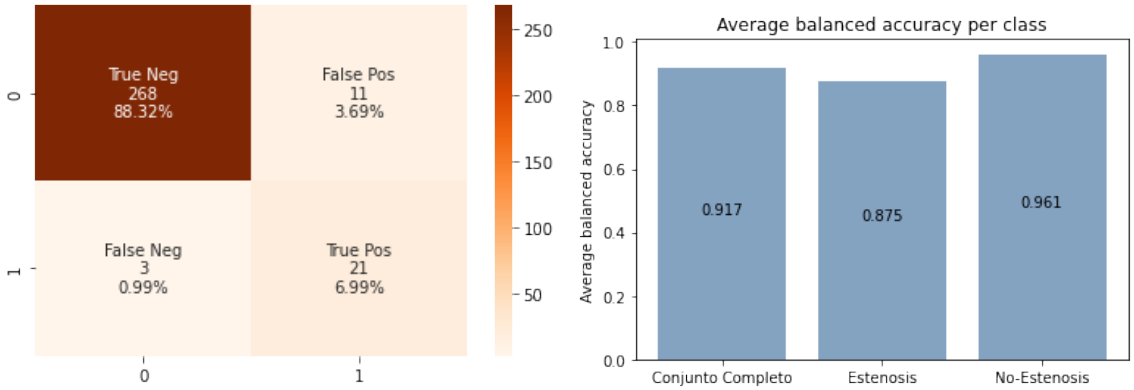

Figura 35: Matriz de confusión de la evaluación del modelo usando la red neuronal pre-entrenada ResNet50 usando la técnica de muestreo y aumento de datos

- ***EfficientNetB4* + Muestreo y aumento**: de igual manera se divide el conjunto inicial en entrenamiento y test K veces, y para cada K, se aumenta la clase minoritaria y se reduce la mayoritaria del conjunto de entrenamiento, obteniendo un balance de ambas clases a 600 imágenes. Se aplica la función de Keras *preprocess_input* al conjunto de entrenamiento y test y se entrena al modelo usando la red pre-entrenada *EfficientNetB4*. Tras la evaluación usando el conjunto de test, se obtiene 0.957 de *balanced accuracy* medio, observable en la Figura 36.

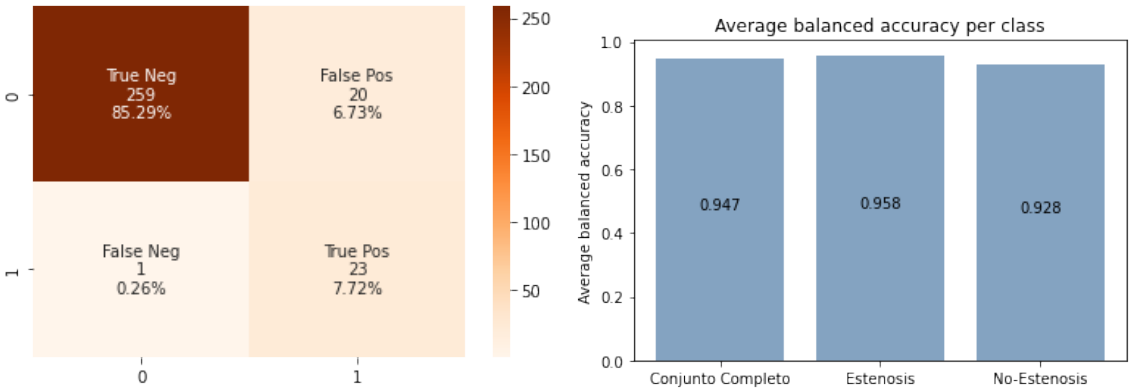

Figura 36: Matriz de confusión de la evaluación del modelo usando la red neuronal pre-entrenada *EfficientNetB4* usando la técnica de muestreo y aumento de datos

- ***MobileNet* + Muestreo y aumento**: se divide el conjunto inicial de datos en entrenamiento y test K veces. En cada K, se realiza para el conjunto de entrenamiento un aumento de la clase minoritaria a 600 imágenes y una reducción aleatoria de la clase mayoritaria a 600 imágenes, consiguiendo así el balance de clases. Se le aplica la función de Keras *preprocess_input* al conjunto de entrenamiento y test y se entrena al modelo usando la red pre-entrenada *MobileNet*. Tras la evaluación usando el conjunto de test, se obtiene 0.955 de *balanced accuracy medio*. En su matriz de confusión, Figura 41, se observar que pese a que el modelo prediga bastante bine, hay 6 muestras negativas que las clasifica como positivas, y 2 muestras positivas que las clasifica como negativas. Para el conjunto de test de las no estenosis, la proporción de solo 6 FP sobre 279 totales está muy bien, al igual que para el conjunto de test de estenosis de 25 muestras, sólo 2 están mal clasificadas.

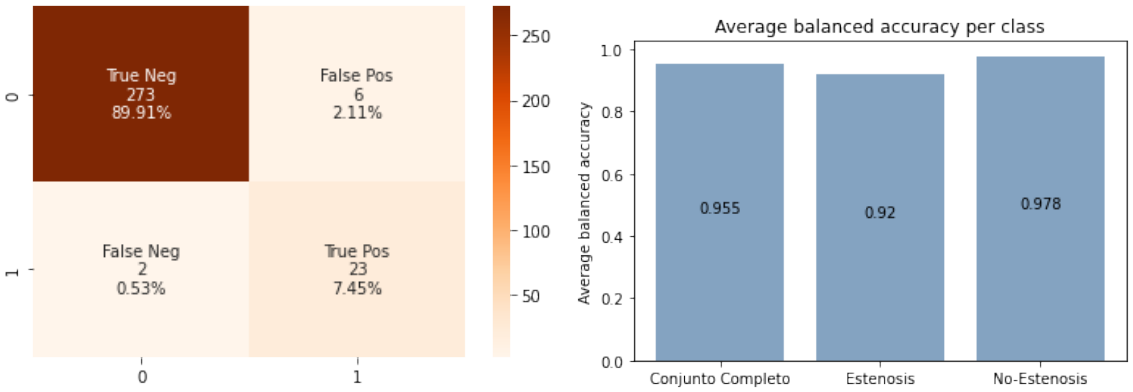

Figura 37: Matriz de confusión de la evaluación del modelo usando MobileNet y la técnica de aumento de datos *Sampling and augmentation*

- **Red neuronal densa entrenada desde cero + Aumento de la clase minoritaria**: En este caso, para cada K se va a dividir el conjunto de datos en entrenamiento y test. Para el conjunto de entrenamiento se va a aumentar la clase minoritaria, en este caso, estenosis usando ImageDataGenerator hasta igualar la clase mayoritaria (no estenosis). Ambas clases quedan balanceadas a 1000 imágenes para cada clase. Este aumento es un poco más agresivo que el de *Muestreo y aumento*. Se entrena el modelo usando la red neuronal densa entrenada desde cero y tras evaluarlo, se obtiene un balanced accuracy medio de 0.808. SU matriz de confusión se puede observar en la Figura 38.

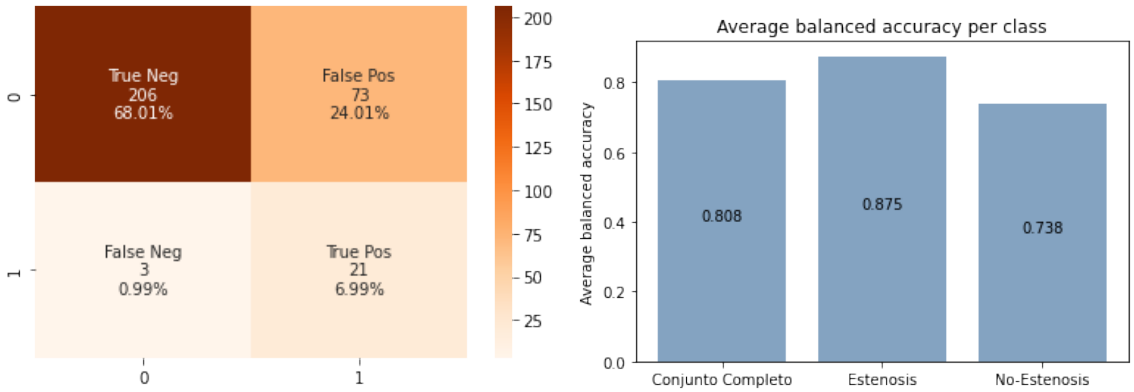

Figura 38: Matriz de confusión de la evaluación del modelo usando la red neuronal densa entrenada desde cero usando la técnica de aumento de la clase minoritaria

- ***ResNet50* + Aumento de la clase minoritaria**: Este proceso es igual que el anterior, de manera que se divide el conjunto en entrenamiento y test k veces. Para cada k se aumenta la clase minoritaria igualándola a la mayoritaria del conjunto de entrenamiento, teniendo un balance de 1000 imágenes para cada clase. Para entrenar el modelo se usará una red pre-entrenada *ResNet50*, y debido a esto, se aplica la función de Keras *preprocess_input* al conjunto de entrenamiento y test para adaptarlos al formato del modelo. Tras la evaluación se obtiene un *balanced accuracy* medio de 0.919. Si observamos su matriz de confusión en la Figura 39, se puede ser que ha clasificado muy bien las muestras para ambas clases, solo clasificando mal 3 muestras para el conjunto de estenosis y 6 para el de no estenosis.

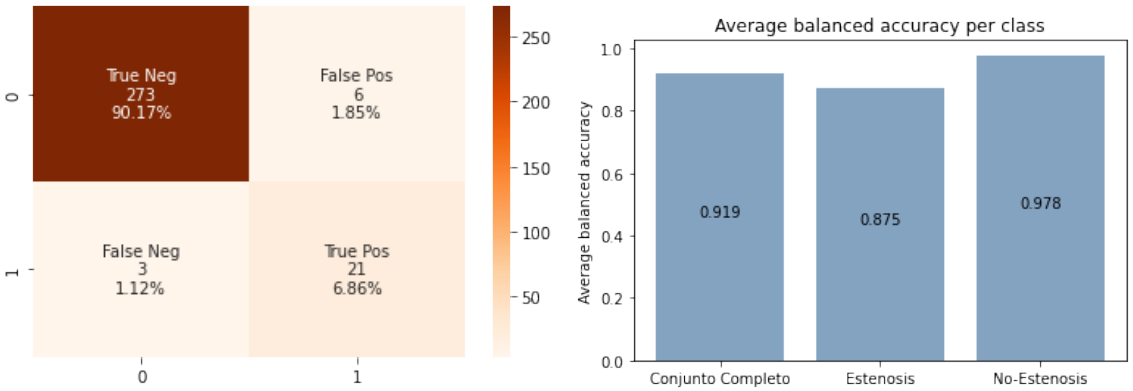

Figura 39: Matriz de confusión de la evaluación del modelo usando ResNet50 y la técnica de aumento de datos *Minority class augmentation*

- ***EfficientNetB4* + Aumento de la clase minoritaria**: de igual manera se divide el conjunto inicial en entrenamiento y test K veces, y para cada K, se aumenta la clase minoritaria igualándola a la clase mayoritaria del conjunto de entrenamiento, obteniendo un balance de ambas clases a 1000 imágenes. Se aplica la función de Keras *preprocess_input* al conjunto de entrenamiento y test y se entrena al modelo usando la red pre-entrenada *EfficientNetB4*. Tras la evaluación usando el conjunto de test, se obtiene 0.923 de *balanced accuracy* medio. Dichos resultados se pueden observar en la Figura 40.

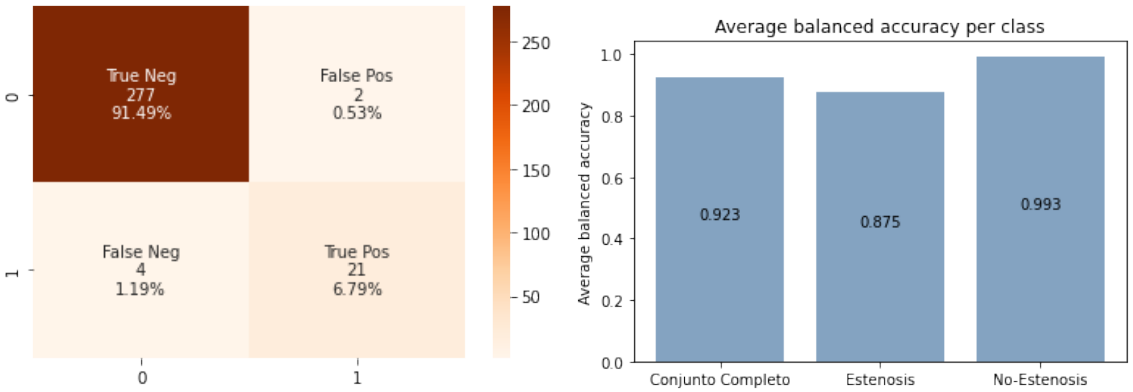

Figura 40: Matriz de confusión de la evaluación del modelo usando EfficientNetB4 y la técnica de aumento de datos *Minority class augmentation*

- ***MobileNet* + Aumento de la clase minoritaria**: Se divide el conjunto inicial de datos en entrenamiento y test K veces. En cada K, se realiza para el conjunto de entrenamiento un aumento de la clase minoritaria a 1000 imágenes para igualar las muestras a las de la clase mayoritaria, consiguiendo así el balance de clases (1000 estenosis - 1000 no estenosis). Se le aplica la función de Keras *preprocess_input* al conjunto de entrenamiento y

test y se entrena al modelo usando la red pre-entrenada MobileNet. Tras la evaluación usando el conjunto de test, se obtiene 0.951 de balanced accuracy medio pudiéndose verse reflejados estos resultados en la matriz de confusión de la Figura 41.

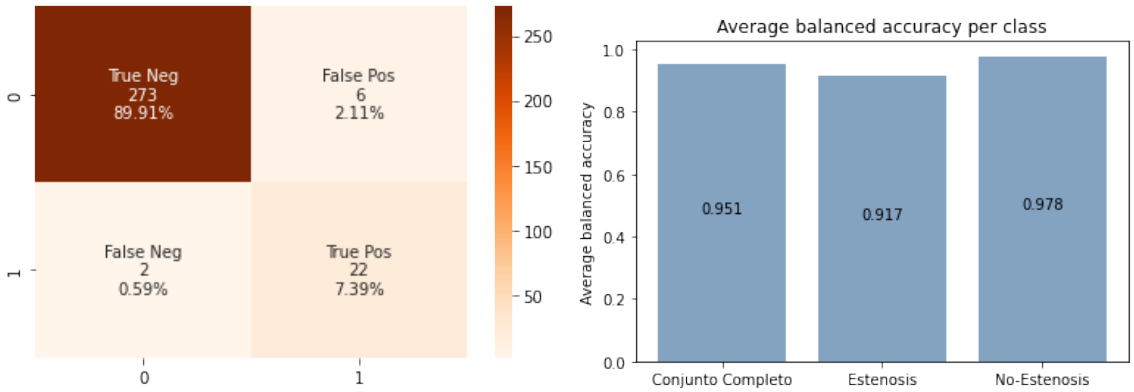

Figura 41: Matriz de confusión de la evaluación del modelo usando MobileNet y la técnica de aumento de datos *Minority class augmentation*

- **Red neuronal densa entrenada desde cero + Aumento proporcional de los datos**: en este caso, para cada K se ha dividido el conjunto de datos en entrenamiento y test y se ha aumentado el conjunto de entrenamiento al completo, es decir, de ambas clases. De esta manera aunque el conjunto sea más grande, el desbalance entre clases sigue existiendo. Se entrena el modelo usando la red neuronal densa entrenada desde cero y tras evaluarlo, se obtiene un balanced accuracy medio de 0.5. Esta mala clasificación se puede observar en la Figura 42.

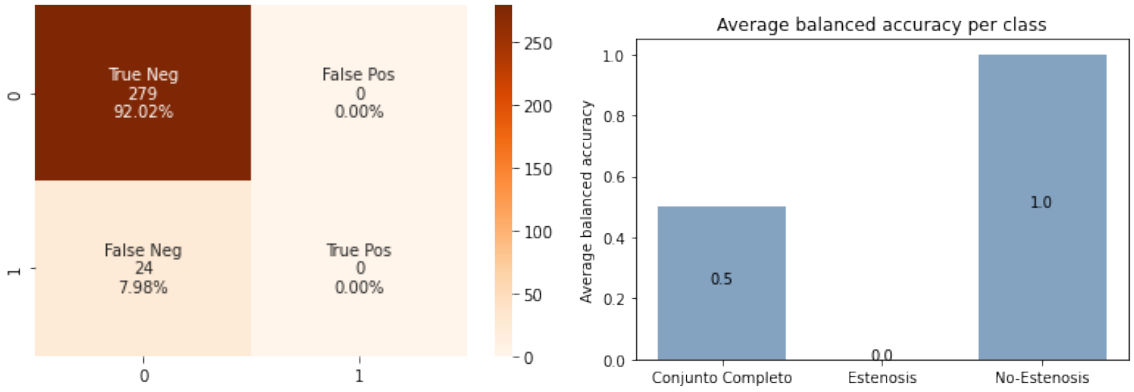

Figura 42: Matriz de confusión de la evaluación del modelo usando la red neuronal densa entrenada desde cero y la técnica de aumento de datos *Proportional data augmentation*

- ***ResNet50* + Aumento proporcional de los datos**: Usando validación cruzada se ha aumentado el conjunto de entrenamiento sin hacer distinción entre clase mayoritaria

y minoritaria. El conjunto aumenta pero el desbalance se mantiene. Se realiza el entrenamiento usando la red pre-entrenada *ResNet*, por lo que se ha aplicado la función de Keras *preprocess_input* al conjunto completo para que los datos sean afines a la entrada del modelo. Tras la evaluación usando el conjunto de test se ha obtenido un *balanced accuracy* de 0.872, observable en la Figura 43.

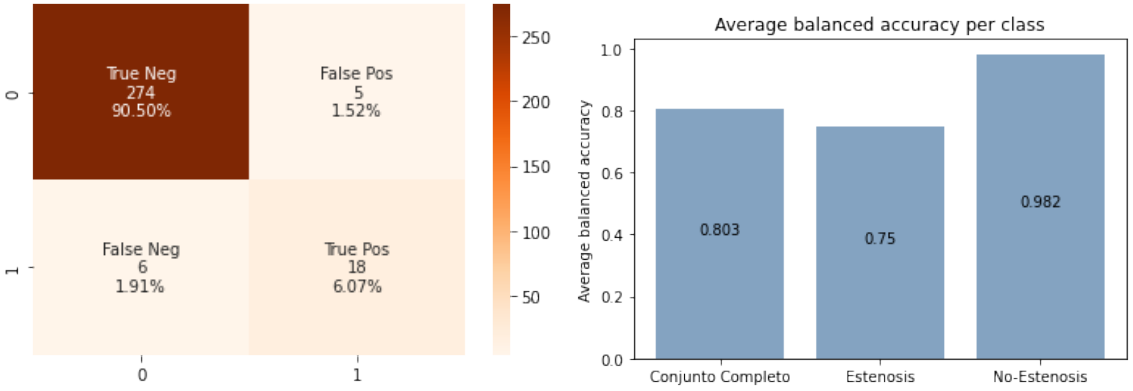

Figura 43: Matriz de confusión de la evaluación del modelo usando la red neuronal pre-entrenada ResNet50 y la técnica de aumento de datos *Proportional data augmentation*

- ***EfficientNetB4* + Aumento proporcional de los datos**: e ha usado la red pre-entrenada *EfficientNetB4* para entrenar un modelo con el conjunto de entrenamiento aumentado por completo, manteniendo las proporciones iniciales entre clases. Además, se ha utilizado validación cruzada para obtener resultados más fiables. En la etapa de evaluación, se ha obtenido un *balanced accuracy* de 0.763. Los resultados obtenidos para cada clase se pueden observar en la Figura 44.

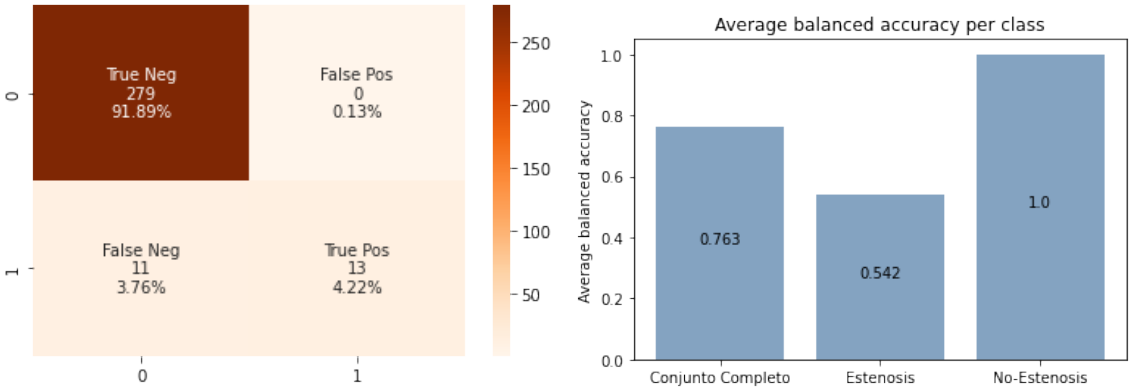

Figura 44: Matriz de confusión de la evaluación del modelo usando la red neuronal pre-entrenada EfficientNetB4 y la técnica de aumento de datos *Proportional data augmentation*

- ***MobileNet* + Aumento proporcional de los datos**: se ha aumentado el conjunto de

entrenamiento al completo para cada caja de la validación cruzada, por lo que la proporción entre clases continúa siendo la misma. Se han preprocesado los conjuntos de entrenamiento y test para adaptarlos a la entrada de una red pre-entrenada y se ha usado MobileNet para el entrenamiento. Tras la evaluación del modelo con el conjunto de test se ha obtenido un *balanced accuracy medio* de 0.803. Se pueden observar sus resultados de la evaluación en la Figura 45.

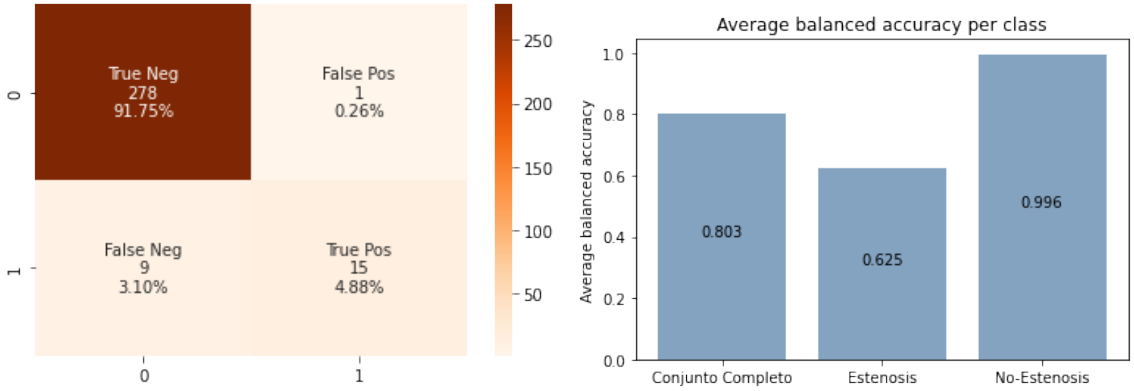

Figura 45: Matriz de confusión de la evaluación del modelo usando la red neuronal pre-entrenada MobileNet y la técnica de aumento de datos *Proportional data augmentation*

- **Red neuronal densa entrenada desde cero + Datos sintéticos**: para esta combinación, se ha importado el conjunto de imágenes generado de forma sintética usando curvas de Bézier con 5000 imágenes para cada clase. En primera instancia se ha entrenado el modelo usando una red neuronal densa entrenada desde cero y esos datos sintéticos. Posteriormente, debido al desbalance en los datos del conjunto inicial, se ha usado validación cruzada para dividir el conjunto inicial en entrenamiento y test, y se ha vuelto a entrenar ese modelo en segunda instancia. Para cada caja se ha evaluado el modelo con el conjunto de test, y se ha obtenido un *balanced accuracy medio* de 0.508. Los resultados de la mala clasificación se pueden observar en la Figura 46.

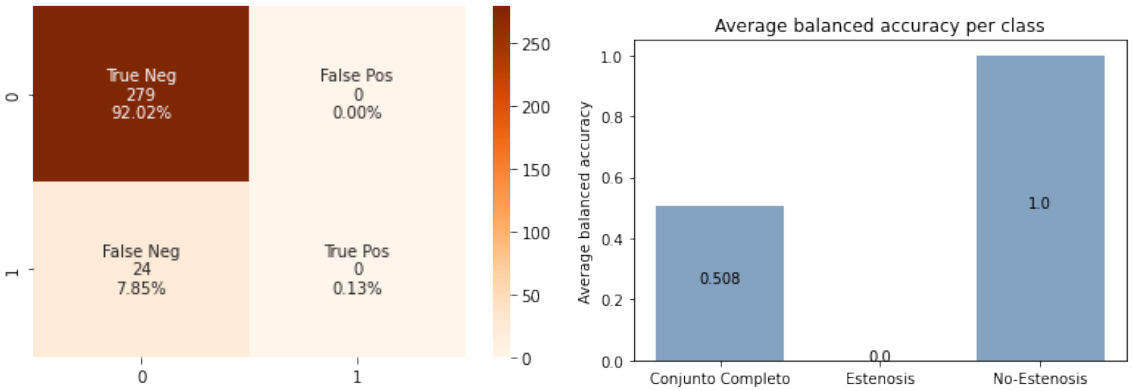

Figura 46: Matriz de confusión de la evaluación del modelo usando la red neuronal densa entrenada desde cero y la técnica de aumento de datos *Synthetic Data*

- ***ResNet50* + Datos sintéticos**: Tras importar el conjunto generado sintéticamente de 5000 imágenes para cada clase, se ha entrenado un modelo haciendo uso de la red pre-entrenada *ResNet50* y el conjunto completo sintético como conjunto de entrenamiento. Es por esto que se le ha tenido que aplicar la función de keras *preprocess_input* tanto a este conjunto como al conjunto inicial, para adaptar ambos a la entrada de la red. Tras este primer entrenamiento, se vuelve a entrenar el modelo pero esta vez con el conjunto de datos inicial y haciendo uso de la validación cruzada para obtener resultados más fiables. En la evaluación del modelo se ha obtenido 0.962 de *balanced accuracy medio*. Su matriz de confusión se puede observar en la Figura 47.

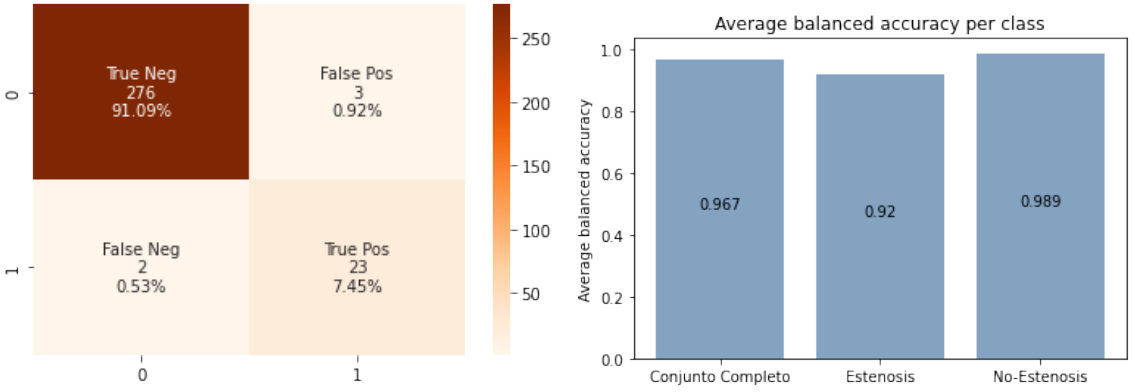

Figura 47: Matriz de confusión de la evaluación del modelo usando la red neuronal pre-entrenada ResNet50 y la técnica de aumento de datos *Synthetic Data*

- ***EfficientNetB4* + Datos sintéticos**: el modelo va a usar la red pre-entrenada *EfficientNetB4* y en primer lugar se ha entrenado con el conjunto sintético generado de 5000 imágenes por clase. Aplicando validación cruzada se ha vuelto a entrenar el modelo,

pero esta vez con el conjunto inicial de imágenes. Ambos conjuntos han tenido que ser pre-procesados para amoldarse a la entrada de la red. Tras la evaluación del modelo con el conjunto de test para cada caja, se ha obtenido un *balanced accuracy medio* de 0.966, esto quiere decir que prácticamente todas las muestras se han clasificado bien, podemos ver su matriz de confusión en la Figura 48.

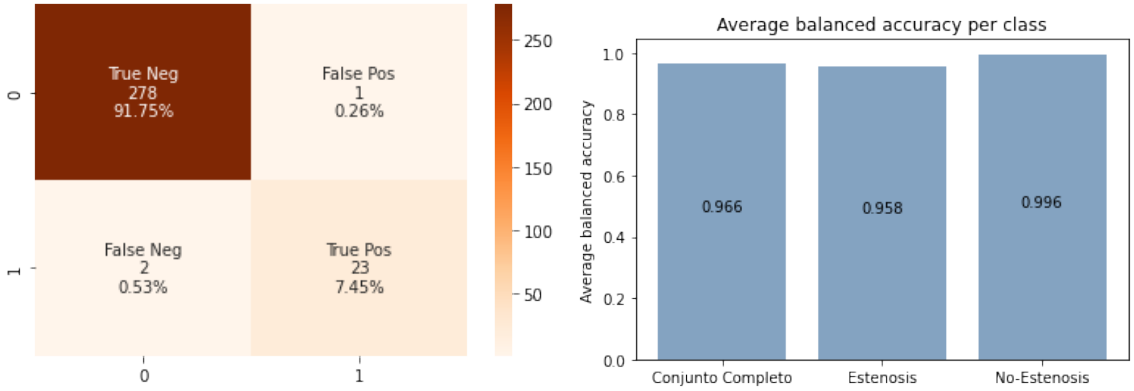

Figura 48: Matriz de confusión de la evaluación del modelo usando la red neuronal pre-entrenada EfficientNetB4 y la técnica de aumento de datos *Synthetic Data*

- ***MobileNet* + Datos sintéticos**: en esta combinación se ha usado el conjunto de imágenes sintéticas generadas con curvas de Bézier explicado en el capítulo anterior. Se ha entrenado por primera vez un modelo que usa la red pre-entrenada MobileNet y este conjunto sintético entero. Se ha vuelvo a entrenar por segunda vez el modelo con el conjunto inicial, pero esta vez usando validación cruzada para dividir el conjunto en entrenamiento y test y obtener los resultados más fiables. Tras la validación se ha obtenido un *balanced accuracy medio* de 0.967. Su matriz de confusión junto con la precisión para cada clase se puede observar en la Figura 49.

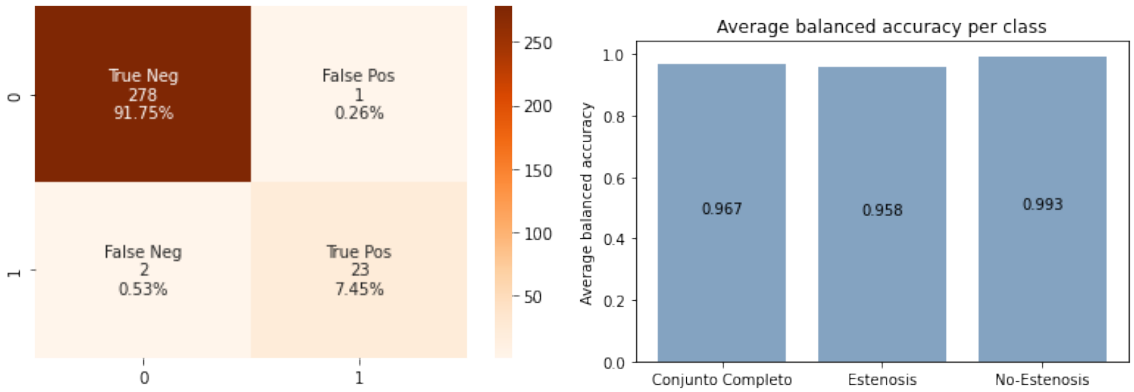

Figura 49: Matriz de confusión de la evaluación del modelo usando la red neuronal pre-entrenada MobileNet y la técnica de aumento de datos *Synthetic Data*

4.1. Resultados

Partimos de una red neuronal densa entrenada desde cero que debido al problema de desbalance de clases, no es capaz de predecir ningún caso de estenosis. Tras aplicarle diferentes técnicas de aumento de datos y utilizar validación cruzada para obtener unos resultados fiables, se ha conseguido mejorar considerablemente los resultados del modelo. La combinación que mejores resultados ha proporcionado ha sido la que realiza un aumento de la clase minoritaria, obteniendo un *balanced accuracy* de 0.808, y la que realiza un muestreo y aumento de datos obteniendo un *balanced accuracy* de 0.792.

Se ha intentado mejorar aún más los resultados haciendo uso de redes pre-entrenadas, más concretamente de *ResNet50, MobileNet y EfficientNetB4*. Estas 3 redes se han probado con los diferentes tipos de aumento de datos incluido con el conjunto inicial. Las cuatro combinaciones que mejores resultados han proporcionado ha sido el aumento de datos usando **datos sintéticos** junto con las redes *MobileNet* y **EfficientNetB4** obteniendo los mejores balanced accuracy de todos los experimentos realizados con un **0.967** y **0.966**, respectivamente. Si observamos la matriz de confusión de la combinación que usa *MobileNet* en la Figura 49, se puede ver que la predicción ha sido casi perfecta para ambas clases, obteniendo valores de 0.993 de especificidad y 0.942 de sensibilidad. Se puede observar también en la Figura 50, el *loss* o pérdida y el *balanced accuracy* durante las épocas en la fase de entrenamiento.

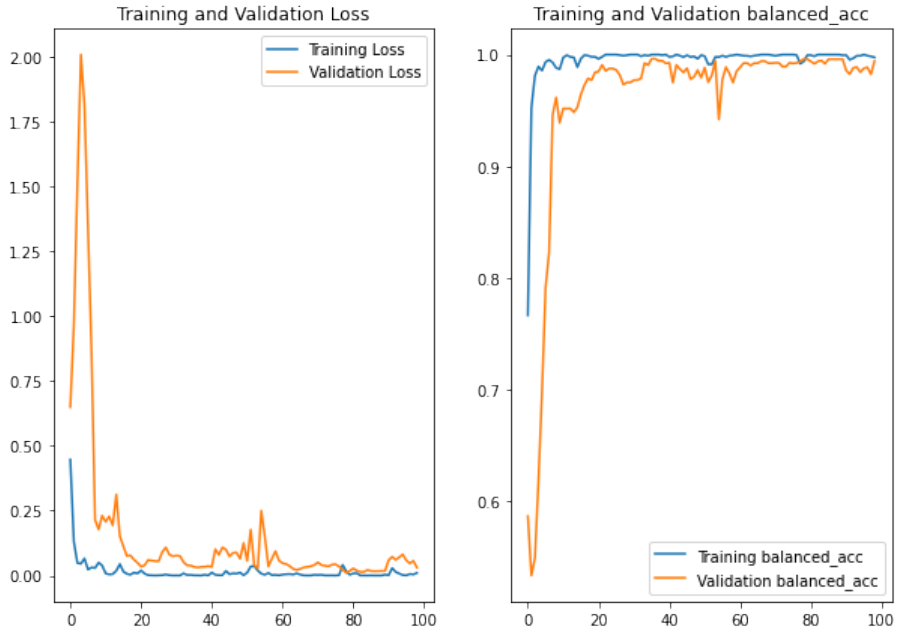

Figura 50: Curvas de entrenamiento y validación para el modelo usando MobileNet y aumento de del conjunto usando imágenes sintéticas .

Las otras dos técnicas de aumento de datos que mejores resultados han proporcionado han

sido el muestro y el aumento del conjunto con *MobileNet*, alcanzando un *balanced accuracy* de 0.955 y con el aumento de datos con datos sintéticos usando *ResNet50* se alcanza un *balanced accuracy medio* de 0.962.

Observando los resultados obtenidos en la evaluación y los resultados mostrados en la gráfica izquierda de la Figura 51, vemos que si hacemos una media de los *balanced accuracy* obtenidos para cada modelo con cada técnica de aumento de datos, las mejores redes neuronales son ***MobileNet*** y con una diferencia muy pequeña, ***ResNet50***. De igual manera en la gráfica derecha de la Figura 52 , se representa una media de los *balanced accuracy* obtenidos para cada modelo usando las diferentes técnicas de aumento de datos. En este caso, las técnicas que mejores resultados han obtenido han sido la de **aumento de la clase minoritaria** y **muestreo y aumento de datos** y datos sintéticos.

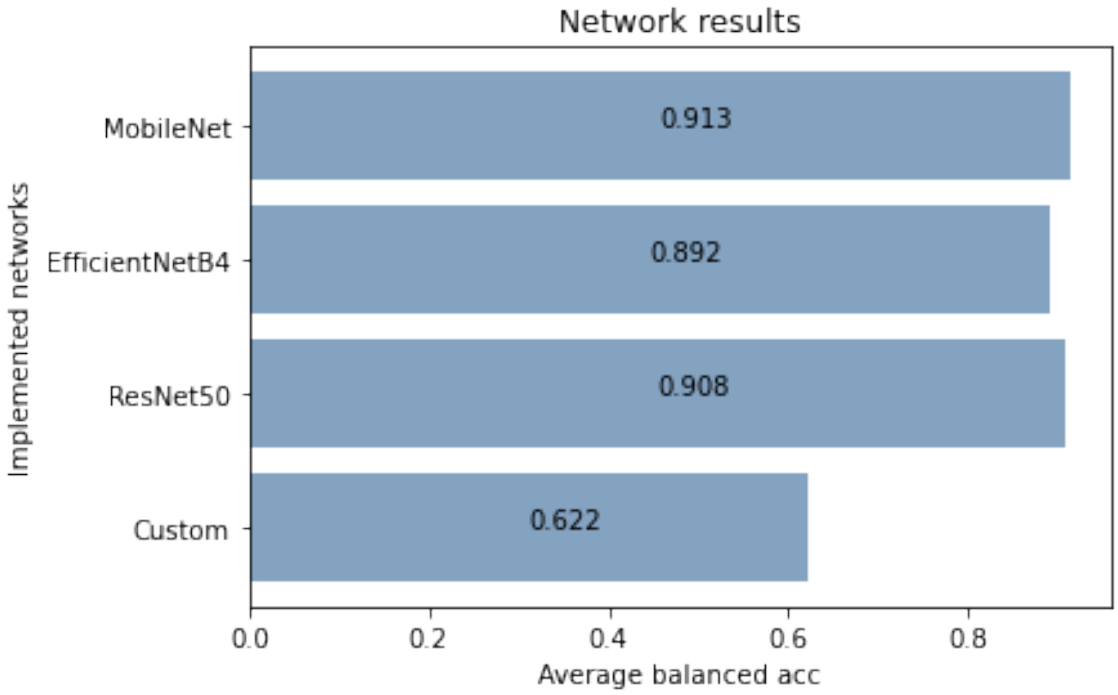

Figura 51: Gráfica comparativa de las redes en función a la *media de balanced accuracy* obtenido en la evaluación

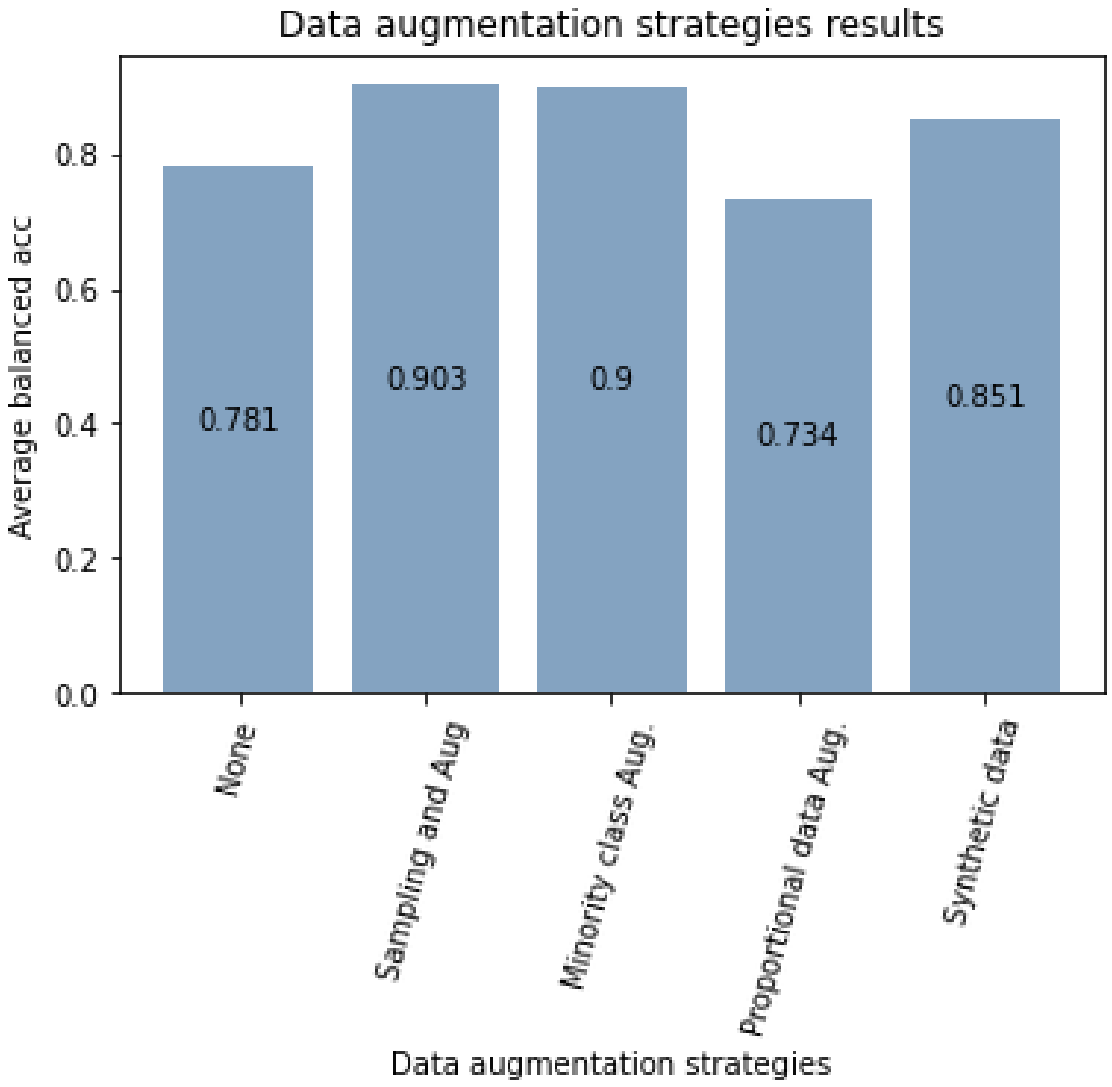

Figura 52: Gráfica comparativa de las técnicas de aumento de datos en función de la *media de balanced accuracy* obtenido en la evaluación

A modo de resumen se pueden observar en la Tabla 53, los resultados obtenidos en el conjunto de test haciendo uso de tres medidas: balanced accuracy, sensibilidad y especificidad. Cuanto más alto sea cada resultado, mejor será el rendimiento del modelo. Se muestran tanto las técnicas de aumento de datos implementadas como las redes neuronales que se han utilizado para la construcción del modelo. Por último la Tabla 54, afirma que los resultados de los modelos son óptimos, ya que no se produce sobre-entrenamiento, siendo los resultados de entrenamiento y validación muy similares a los del test.

Data Aug	None			Sampling and augmentation			Minority class augmentation		
Network	Acc balanced	Especificidad	Sensibilidad	Acc balanced	Especificidad	Sensibilidad	Acc balanced	Especificidad	Sensibilidad
Custom	0.504 ± 0.008	0.999 ± 0.0	0.008 ± 0.017	0.792 ± 0.026	0.700 ± 0.056	0.885 ± 0.047	0.808 ± 0.027	0.739 ± 0.038	0.876 ± 0.046
ResNet50	0.870 ± 0.080	0.983 ± 0.017	0.758 ± 0.155	0.917 ± 0.028	0.960 ± 0.007	0.875 ± 0.053	0.919 ± 0.048	0.980 ± 0.008	0.858 ± 0.101
EfficientNetB4	0.860 ± 0.034	0.993 ± 0.004	0.727 ± 0.067	0.947 ± 0.041	0.927 ± 0.094	0.967 ± 0.031	0.923 ± 0.052	0.994 ± 0.002	0.851 ± 0.104
MobileNet	0.891 ± 0.025	0.989 ± 0.005	0.793 ± 0.053	0.955 ± 0.016	0.977 ± 0.009	0.934 ± 0.034	0.951 ± 0.018	0.978 ± 0.018	0.925 ± 0.049

Data Aug	Proportional data augmentation			Synthetic data		
Network	Acc balanced	Especificidad	Sensibilidad	Acc balanced	Especificidad	Sensibilidad
Custom	0.5 ± 0.0	0.999 ± 0.0	0.0 ± 0.0	0.508 ± 0.017	0.999 ± 0.0	0.016 ± 0.033
ResNet50	0.872 ± 0.033	0.983 ± 0.009	0.760 ± 0.073	0.962 ± 0.017	0.990 ± 0.01	0.934 ± 0.032
EfficientNetB4	0.763 ± 0.132	0.999 ± 0.002	0.528 ± 0.266	0.966 ± 0.035	0.997 ± 0.004	0.934 ± 0.067
MobileNet	0.803 ± 0.158	0.997 ± 0.003	0.609 ± 0.318	**0.967 ± 0.021**	0.993 ± 0.013	0.942 ± 0.042

Figura 53: Tabla comparativa de todos los resultados obtenidos para el conjunto de test en el experimento

Network	Custom			Resnet50		
Data Augmentation	Train	Validation	Test	Train	Validation	Test
None	0.515	0.5	0.504	0.971	0.932	0.870
Sampling and augmentation	0.844	0.813	0.808	0.997	0.981	0.919
Minority class augmentation	0.838	0.784	0.792	0.997	0.981	0.918
Proportional data augmentation	0.502	0.5	0.5	0.971	0.922	0.872
Synthetic data	0.508	0.518	0.508	0.978	0.928	0.962

Network	EfficientNetB4			Mobile		
Data Augmentation	Train	Validation	Test	Train	Validation	Test
None	0.959	0.941	0.860	0.997	0.968	0.891
Sampling and augmentation	0.998	0.992	0.923	0.999	0.991	0.951
Minority class augmentation	0.976	0.9623	0.947	0.999	0.989	0.955
Proportional data augmentation	0.853	0.767	0.763	0.895	0.833	0.803
Synthetic data	0.996	0.930	0.966	0.993	0.967	0.967

Figura 54: Tabla comparativa de todos los balanced accuracy medios obtenidos en el experimento para el conjunto de entrenamiento, validación y test

5
Aplicación web

5.1. Detección de la Estenosis

Una vez que se ha obtenido el modelo que proporciona los mejores resultados de *balanced accuracy*, se procede a realizar el proceso de detección de la estenosis de la imagen que se va a testear. En este proceso se va a dividir el conjunto en *patches* de 32x32 y se van a introducir en el modelo. El modelo predecirá para cada región o *patch* una clase, estenosis o no estenosis. Finalmente, se marcará en la imagen original aquellas regiones que hayan sido clasificadas como estenosis.

El proceso de detección de estenosis se base en los siguientes pasos:

- Realizar un *resize* a la imagen original de 128x128 usando la interpolación hiper cúbica.
- Dividir la imagen resultante en regiones de 32x32. Se obtendrán 16 regiones.
- Preprocesar las regiones para que tengan el formato de entrada de la red pre-entrenada que se usa en el modelo.
- Importar el mejor modelo obtenido en la fase de evaluación.
- Evaluar cada región con el modelo.
- Marcar en la imagen originar las regiones que han sido clasificadas con estenosis.

Un ejemplo del resultado de este proceso es el que se presenta en la Figura 55, a la izquierda se muestra una coronariografía etiquetada manualmente por un especialista, a la derecha se muestra la imagen resultante del proceso de detección. Como se puede observar se ha obtenido un resultado muy preciso.

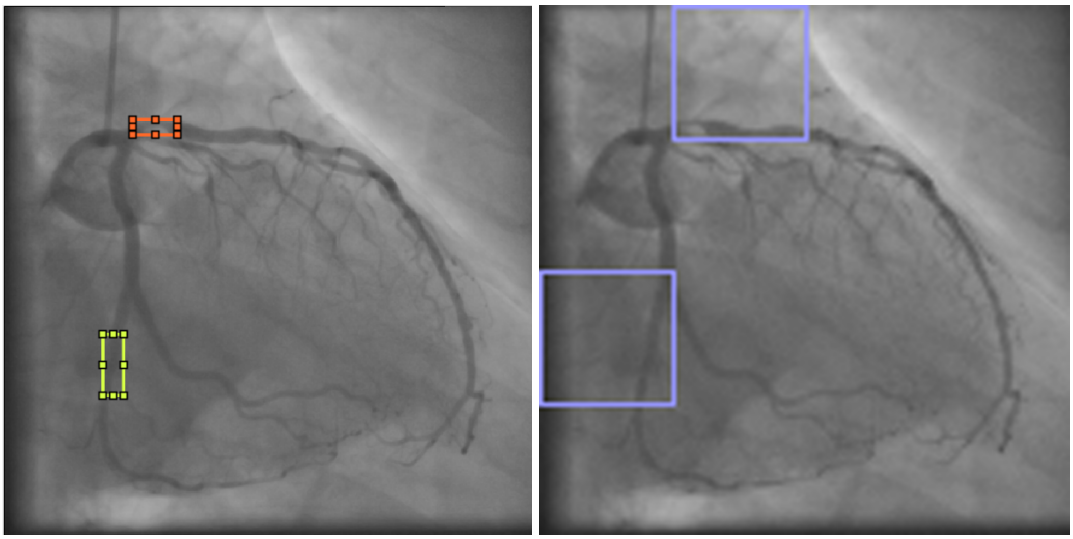

Figura 55: Imagen coronariográfica con estenosis marcada manualmente (izquierda) e Imagen coronariográfica marcada con estenosis mediante el uso de Deep learning (derecha

5.2. Funcionamiento

Se va a implementar una aplicación web para que cualquier usuario pueda introducir una coronariografía en la interfaz gráfica y pueda ver si esa imagen presenta o no estenosis y dónde la presenta. La aplicación web se basa en un modelo cliente-servidor orquestado por **Flask**, un micro *Framework* que basa su funcionalidad en el **patrón MVC**. MVC significa: vista, modelo y controlador; esto no es más que una plantilla *HTML*, los datos con los que va a contar la aplicación y el quien controla las peticiones cliente-servidor de la aplicación web.

El usuario como cliente va a acceder a la página de inicio dónde podrá subir la imagen dónde quiere detectar la estenosis, visualizarla y enviarla para procesarla. El servidor recibirá la imagen mediante una petición de tipo *POST* e iniciará el proceso de detección de la estenosis. En el servidor se producirá la división de la imagen en regiones de 32x32 y se predecirá para cada región, si presenta estenosis o no. Para ello en el servidor debe estar localizado el modelo que se va a usar para clasificar cada región. Una vez que el modelo ha predicho las etiquetas, se colorea en la imagen introducida por el usuario aquellas regiones que han sigo clasificadas con estenosis. Finalmente el servidor le devuelve al cliente la imagen introducida por el usuario con la localización de la estenosis en caso de que la hubiese, todo este proceso puede observarse en el esquema gráfico de la Figura 5.2.

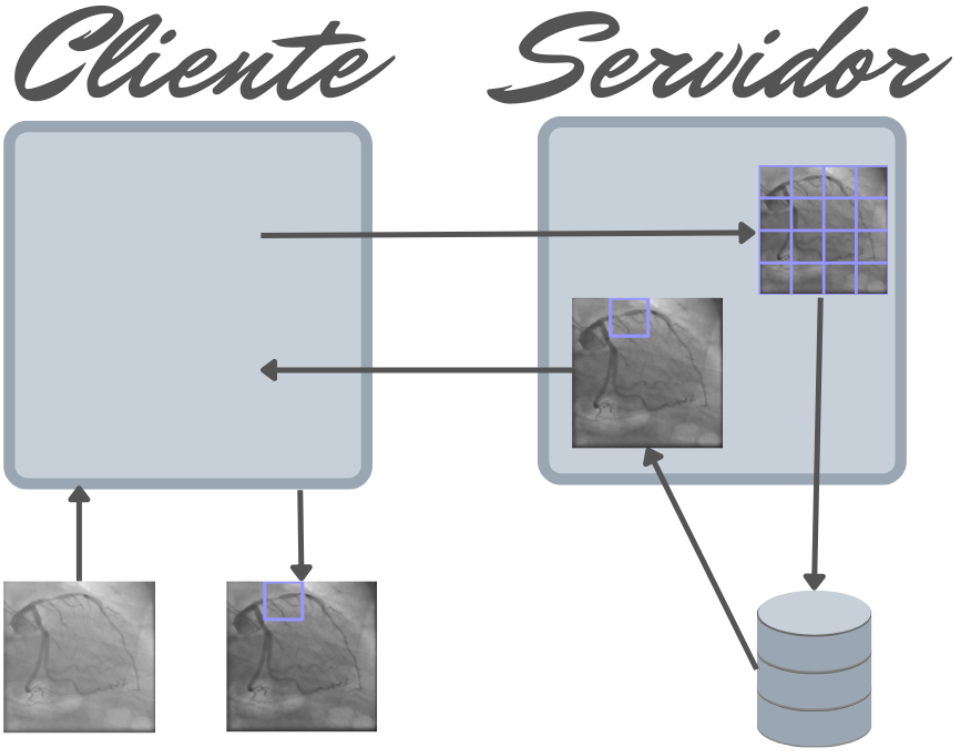

Figura 56: Proceso completo del funcionamiento de la aplicación web

Toda aplicación web cuenta con una interfaz gráfica. En este proyecto la interfaz gráfica contará con dos vistas, la primera de ellas, la página principal dónde el usuario puede seleccionar la imagen coronariográfica como se observa en la figura 5.2, visualizarla y enviarla al servidor como se observa en la Figura 5.2, y la segundo vista es la que muestra los resultados. En esta vista, se muestra la imagen resultante del proceso de detección de la estenosis, y además, para que sea más fácil la visualización, se muestra la imagen original, esto se puede observar en la Figura 5.2

Figura 57: Página de inicio de la aplicación web

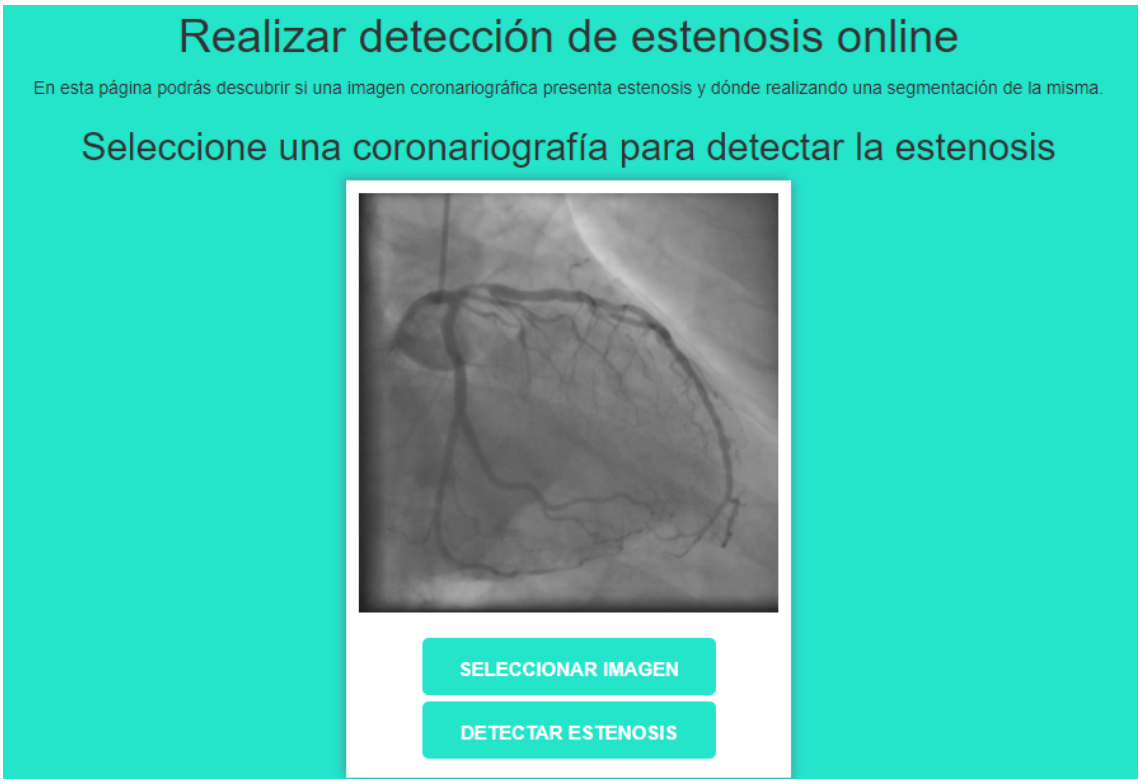

Figura 58: Página de inicio que muestra la imagen seleccionada antes de ser enviada para la detección

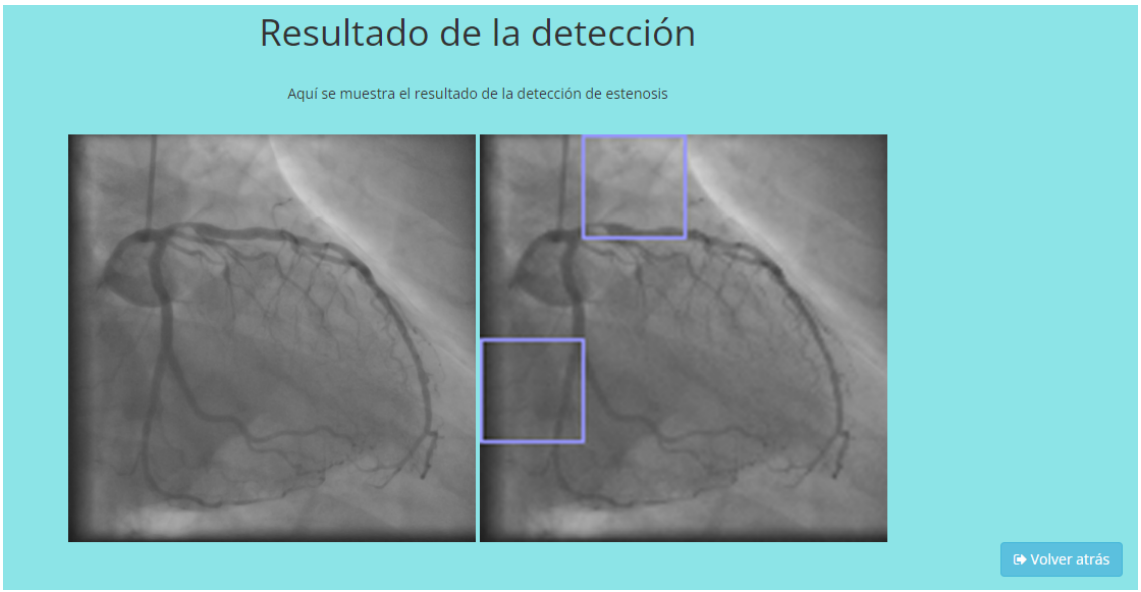

Figura 59: Página resultante tras el proceso de detección de estenosis que muestra los resultados de la misma

6

Conclusiones y futuras líneas de investigación

6.1. Conclusiones

En este trabajo se ha tratado siempre obtener los mejores resultados posibles siendo conscientes del desbalance tan abismal entre la clase estenosis y no estenosis. Este problema de clasificación binaria ha pasado por muchas fases distintas. En primer lugar, la fase de elección del conjunto de datos a usar, y lo que es más importe, el preprocesado de dicho conjunto para que fuera más fácil trabajar con él en las fases siguientes.

Seguidamente, se hizo un primer entrenamiento usando el conjunto de datos real, sin ningún tipo de preprocesamiento ni de aumento de datos con la red neuronal densa entrenada desde cero para entender y ver como se reflejaba en los resultados ese desbalanceo y poder crear modelos usando redes neuronales y técnicas de aumento de datos que mejor se adecuaran al problema. En este proceso de búsqueda, se decidió introducir el uso de redes neuronales pre-entrenadas con el objetivo de usar sus pesos para intentar obtener aún mejores resultados. El proceso ha sido largo y los resultados han sido muy dispares, llegando a descartar redes neuronales pre-entrenadas debido a que se estancaban en mínimos locales o porque no aportaban una mejoría notable. Además se ha intentado elegir tres redes neuronales pre-entrenadas que tengan características muy distintas para ver las divergencias entre ellos y cuál se adaptaba mejor al problema de clasificación.

Una vez se seleccionaron las redes a usar y las técnicas de aumento de datos, se procedió a entrenar los modelos y realizar una evaluación posterior. Para ello se utilizó la técnica validación cruzada para asegurar la fiabilidad de los resultados y comprobar que los resultados no dependen de la partición del conjunto en entrenamiento y test. Se ha llegado a obtener un *balanced accuracy* de **0.967** usando la red pre-entrenada *MobileNet* junto con aumento de

datos usando **datos sintéticos**. Este es un muy buen resultado debido a que en el modelo inicial comentado anteriormente, el resultado no era superior a 0.5 de *balanced accuracy*, lo que quería decir que solo evaluaba bien la clase sin estenosis, y para clase de estenosis, que es la que realmente es interesante para este estudio, no era capaz de predecir ninguna muestra correctamente. Por lo que con este valor obtenido de 0.967, se puede afirmar que se ha logrado obtener un modelo muy preciso capaz de predecir casi sin fallos si una región presenta estenosis o no.

Para este problema de clasificación, ***MobileNet*** es la que ha obtenido mejor media de *balanced accuracy*, muy igualada con *ResNet50*, algo que en parte ha sorprendido debido a que *ResNet50* es una red ampliamente utilizada debido a la eficiencia que demuestra tener, por lo que era un resultado a esperar. Sin embargo para *MobileNet*, al ser una red neuronal tan ligera y que reduce mucho las características aprendidas por el modelo, no se esperaban tan buenos resultados. Por otro lado, aunque los resultados obtenidos por la red *EfficientNetB4* no hayan sido malos, se esperaban mejores resultados debido a que es una red que está diseñada para modelos de clasificación binaria. Por último, la red neuronal densa entrenada desde cero queda limitada a un *balanced accuracy* máximo de 0.808, que aunque siga siendo un buen resultado, con las redes anteriores se consigue mejorar mucho más el resultado.

La técnicas de aumento de datos que mejores resultados han dado han sido la que aumenta la clase minoritaria y la que realiza un muestreo y un aumento del conjunto. Era algo de esperar ya que son técnicas que realmente duplican el conjunto de datos inicial manteniendo su formato inicial como es el brillo, el zoom o la forma de las venas, y además balancean las clases del conjunto permitiendo que el modelo aprenda correctamente a predecir ambas clases.

Por consiguiente, se ha logrado llegar al objetivo de obtener el modelo que proporcione los mejores resultados combinando diferentes técnicas de aumento de datos y diferentes redes neuronales. Para ello ha sido esencial el uso de *Google Colab*, una herramienta que ha demostrado ser muy útil y que permite dar acceso a recursos que no todos los ordenadores integran como es el uso de la GPU, procesador obligatorio para entrenar las redes neuronales.

6.2. Futuras líneas de investigación

Sería interesante en un futuro implementar un algoritmo de **redes GANs** para aumentar los datos del conjunto y ver como se comporta en comparación a las otras técnicas ya implementadas en este estudio. GANs[52] (redes generativas antagónicas) es una técnica muy precisa para generar nuevos datos. Se basa en la utilización de dos redes, una la que genera

el conjunto de datos sintéticos y otra red, la que lo evalúa. La red discriminadora es entrenada con los datos auténticos, por lo que el objetivo de la **red generadora** es engañar a la **red discriminadora** produciendo imágenes sintéticas tan precisas que las clasifique como auténticas. Ambas redes usan la **retropropagación**[53] o *backpropagation*. Esto es que cuando la red recibe un estimulo, este se propaga por toda la red hasta generar la salida, que es comparada con la salida deseada. Se genera una señal de error por cada salida y se propaga hacia las neuronas anteriores hasta llegar a todas ellas. Con este proceso se consigue que a medida que se entrena una red, las capas intermedias sean capaces de distinguir las características de la entrada.

Gracias a que ambas redes utilizan la retropropagación, la red generadora cada vez crea imágenes mejores, a la vez que la red discriminadora es más precisa distinguiendo esas imágenes sintéticas. Con esta técnica hoy en día se ha llegado a generar caras humanas falsas que parecen totalmente reales, por lo que esta metodología sería capaz de generar imágenes con y sin estenosis iguales o prácticamente iguales a las naturales.

Otra de las lineas que podrían seguirse sería la adaptación de este problema de clasificación en un problema de detección. Para ello necesitaríamos el conjunto de datos bien etiquetado. La principal diferencia entre un problema de clasificación y uno de detección[54], es que el problema de clasificación se limita a generar etiquetas con la probabilidad para cada clase mientras que los de detección detectan los diferentes objetos en una imagen y los clasifican.El siguiente paso tras la detección de objetos es la segmentación, que se basa en predecir la etiqueta para cada píxel. Una imagen se puede dividir en agrupaciones de píxeles para su posterior predicción y clasificación. De esta manera la clasificación de la imagen es mucho más simple y precisa que si hiciéramos la predicción de la imagen completa.

Entonces, el estudio siguiente sería, teniendo el conjunto de imágenes coronariográficas bien etiquetado, implementar un algoritmo de detección que sea capaz de distinguir arterias, y entrenarlo para que sea capaz de distinguir la parte con estenosis y sin estenosis. Posteriormente se realizaría la segmentación de la imagen para que el análisis de la misma sea mucho más fácil y liviano. Las técnicas de las que se ha hablado se muestran en la Figura 6.2.

Figura 60: Imagen comparativa entre la técnica de clasificación, detección y segmentación[55]

Bibliografía

[1] Colegio Argentino de Cardio angiólogos intervencionistas, *Coronariografía– Colegio Argentino de Cardio angiólogos intervencionistas CACI*. dirección: http://www.caci.org.ar/post/coronariografia#:~:text=Se%20trata%20de%20un%20procedimiento,(Figuras%201%20y%202)..

[2] V. Gulshan, L. Peng, M. Coram y col., "Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs," *JAMA*, vol. 316, n.º 22, págs. 2402-2410, 2016, ISSN: 0098-7484. DOI: 10.1001/jama.2016.17216. eprint: https://jamanetwork.com/journals/jama/articlepdf/2588763/joi160132.pdf. dirección: https://doi.org/10.1001/jama.2016.17216.

[3] A. Robledano, *Qué es Python: Características, evolución y futuro*, 2019. dirección: https://openwebinars.net/blog/que-es-python/.

[4] thiago-gsantos03, *Google Colab: ¿qué es y cómo usarlo?* 2020. dirección: https://www.aluracursos.com/blog/google-colab-que-es-y-como-usarlo.

[5] Isaac, *Cómo instalar TensorFlow Machine Learning*. dirección: https://www.linuxadictos.com/como-instalar-tensorflow-machine-learning-system-ubuntu-20-04.html.

[6] A. Rosebrock, *Autoencoders with Keras, TensorFlow, and Deep Learning*, 2020. dirección: https://pyimagesearch.com/2020/02/17/autoencoders-with-keras-tensorflow-and-deep-learning/.

[7] A. Nerja, *Qué es overleaf*. dirección: https://www.adrinerja.com/que-es-overleaf/.

[8] Wikipedia, *Latex - Wikipedia: La enciclopedia Libre*, 2022. dirección: https://es.wikipedia.org/wiki/LaTeX.

[9] G. B., *¿Qué es HTML? Explicación de los fundamentos del Lenguaje de marcado de hipertexto*, 2022. dirección: https://www.hostinger.es/tutoriales/que-es-html.

[10] Freepik, *Html 5 icono*. dirección: https://www.flaticon.es/icono-gratis/html-5_174854.

[11] D. Santos, *Introducción al CSS: qué es, para qué sirve y otras 10 preguntas frecuentes*, 2021. dirección: https://blog.hubspot.es/website/que-es-css.

[12] b3nj5m1n, *Logo CSS*, 2017. dirección: https://pixabay.com/es/illustrations/logo-css-css3-icono-2582747/.

[13] Wikipedia, *JavaScript — Wikipedia, La enciclopedia libre*, 2022. dirección: https://es.wikipedia.org/w/index.php?title=JavaScript&oldid=144235618.

[14] J. D. Muñoz, *Qué es Flask*, 2017. dirección: https://openwebinars.net/blog/que-es-flask/.

[15] Anaconda, *Data science technology for groundbreaking research.a competitive edge.a better world*, 2022. dirección: https://www.anaconda.com/.

[16] J. García, *PyCharm, un potente IDE para crear programas con Python.* dirección: https://www.linuxadictos.com/pycharm-un-potente-ide-para-crear-programas-con-python.html.

[17] kolmak12, *PyCharm entorno de desarrollo Integrado JetBrains IntelliJ IDEA de Python*, 2022. dirección: https://www.freepng.es/png-d9wawp/.

[18] Y. Fernandez, *Qué es Github y qué es lo que le ofrece a los desarrolladores*, 2019. dirección: https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores.

[19] Felman, Adam y Oen-Hsiao, Joyce, "Todo lo que debes saber acerca de las enfermedades del corazón," *Medical News Today*, 2019. dirección: https://www.medicalnewstoday.com/articles/es/327293#tipos.

[20] Massó Asencio, Josep, "Coronariografia: en qué consiste, cómo y cuándo se realiza," *Blog de cardiología de Hospital Quirónsalud Barcelona*, 2021. dirección: https://www.quironsalud.es/blogs/es/corazon/coronariografia-consiste-realiza.

[21] ICBA, *Angioplastia - ¿Qué es la angioplastia?* Dirección: https://www.icba.com.ar/pacientes/educacion-para-pacientes/enciclopedia/angioplastia.

[22] Aurora, *Qué es el Deep Learning y sus aplicaciones en la actualidad*, 2021. dirección: https://iddigitalschool.com/bootcamps/que-es-el-deep-learning-y-sus-aplicaciones-en-la-actualidad/.

[23] A. C. García, *Estructura Red neuronal Densa.* dirección: https://www.researchgate.net/figure/Estructura-Red-neuronal-Densa-Cuenta-con-la-primera-capa-de-entrada-dos-capas-ocultas_fig4_343450066.

[24] M. Bakator y D. Radosav, "Deep learning and medical diagnosis: A review of literature," *Multimodal Technologies and Interaction*, vol. 2, n.º 3, pág. 47, 2018. DOI: https://doi.org/10.3390/mti2030047. dirección: https://books.google.es/books?hl=es&lr=&id=ju1mDwAAQBAJ&oi=fnd&pg=PA78&dq=deep+learning+que+es+articulo&ots=k_SeNC32UR&sig=Bn5IH3aXXTAA6vq5Vr95u63WS6w.

[25] Sarmiento-Ramos, José Luis, "Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica," *UIS Ingenierías*, vol. 19, págs. 1-18, 2020. DOI: https://doi.org/10.18273/revuin.v19n4-2020001. dirección: https://www.redalyc.org/articulo.oa?id=553768213002.

[26] J. Torres, *DEEP LEARNING Introducción práctica con Keras*. Lulu. com, 2018. dirección: https://torres.ai/deep-learning-inteligencia-artificial-keras/.

[27] M. G. Villanueva y L. R. Muñoz, "Diseño de una Arquitectura de Red Neuronal Convolucional para la clasificación de objetos," *Ciencia Nicolaita*, n.º 81, págs. 46-61, 2021. dirección: https://www.cic.cn.umich.mx/cn/article/download/517/410/2388.

[28] A. C. Franco, *DEEP LEARNING*. dirección: https://idus.us.es/bitstream/handle/11441/90004/Centeno%20Franco%20Alba%20TFG.pdf?sequence=1&isAllowed=y.

[29] J. Brownlee, *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*, 2019. dirección: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

[30] S. K. E., *Convolutional Neural Network | Deep Learning*. dirección: https://developersbreach.com/convolution-neural-network-deep-learning/.

[31] Antczak, Karol, "Source codes for training and testing deep CNN for stenosis detection.," *MATEC Web Conf.*, vol. 210, pág. 04 001, 2018. dirección: https://github.com/KarolAntczak/DeepStenosisDetection.

[32] paulo116, *how to use pretrained models on low resolution images like tiny imagenet dataset - Stackoverflow*, 2021. dirección: https://stackoverflow.com/questions/66949080/how-to-use-pretrained-models-on-low-resolution-images-like-tiny-imagenet-dataset.

[33] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos y P. De Geus, "Malicious software classification using transfer learning of resnet-50 deep neural network," en *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2017, págs. 1011-1014. DOI: 10.1109/ICMLA.2017.00-19.

[34] D. Team, *Redes neuronales residuales – Lo que necesitas saber (ResNet)*. dirección: https://datascience.eu/es/aprendizaje-automatico/una-vision-general-de-resnet-y-sus-variantes/.

[35] K. He, X. Zhang, S. Ren y J. Sun, "Deep Residual Learning for Image Recognition," en *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, págs. 770-778. DOI: 10.1109/CVPR.2016.90.

[36] V. Feng, *An Overview of ResNet and its Variants*, 2017. dirección: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035.

[37] M. Tan, *EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling*, 2019. dirección: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html.

[38] Ü. Atila, M. Uçar, K. Akyol y E. Uçar, "Plant leaf disease classification using EfficientNet deep learning model," *Ecological Informatics*, vol. 61, pág. 101 182, 2021. DOI: 10.1016/j.ecoinf.2020.101182.

[39] L. T. Duong, P. T. Nguyen, C. Di Sipio y D. Di Ruscio, "Automated fruit recognition using EfficientNet and MixNet," *Computers and Electronics in Agriculture*, vol. 171, pág. 105 326, 2020. DOI: 10.1016/j.compag.2020.105326.

[40] M. Tan y Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," en *International conference on machine learning*, PMLR, 2019, págs. 6105-6114. DOI: https://doi.org/10.48550/arXiv.1905.11946.

[41] L. G. Falconí, M. Pérez y W. G. Aguilar, "Transfer Learning in Breast Mammogram Abnormalities Classification With Mobilenet and Nasnet," en *2019 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2019, págs. 109-114. DOI: 10.1109/IWSSIP.2019.8787295.

[42] A. G. Howard, M. Zhu, B. Chen y col., *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, 2017. DOI: 10.48550/arXiv.1704.04861. dirección: https://arxiv.org/abs/1704.04861.

[43] A. Pujara, *Image Classification With MobileNet*, 2020. dirección: https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470.

[44] Calvo, Diego, *Función de activación – Redes neuronales*. dirección: https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/#:~:text=La%20funci%C3%B3n%20sigmoide%20transforma%20los,de%20manera%20asint%C3%B3tica%20a%200.&text=Caracter%C3%ADsticas%20de%20la%20funci%C3%B3n%20signoide,Satura%20y%20mata%20el%20gradiente..

[45] A. Lopez-del Rio, A. Nonell-Canals, D. Vidal y A. Perera-Lluna, "Evaluation of cross-validation strategies in sequence-based binding prediction using deep learning," *Journal of chemical information and modeling*, vol. 59, n.º 4, págs. 1645-1657, 2019. DOI: https://doi.org/10.1021/acs.jcim.8b00663.

[46] M. W. Browne, "Cross-validation methods," *Journal of mathematical psychology*, vol. 44, n.º 1, págs. 108-132, 2000. DOI: https://doi.org/10.1006/jmps.1999.1279.

[47] Wikipedia, *Validación cruzada — Wikipedia, La enciclopedia libre*, 2020. dirección: https://es.wikipedia.org/w/index.php?title=Validaci%C3%B3n_cruzada&oldid=124047241.

[48] Antczak, Karol y Liberadzki, Lukasz, "Stenosis Detection with Deep Convolutional Neural Networks," *MATEC Web Conf.*, vol. 210, pág. 04 001, 2018. DOI: 10.1051/matecconf/201821004001. dirección: https://doi.org/10.1051/matecconf/201821004001.

[49] Burrueco, Daniel, *Parada temprana - Interactive Chaos*. dirección: https://interactivechaos.com/es/manual/tutorial-de-machine-learning/parada-temprana#:~:text=Una%20de%20estas%20formas%20de,como%20con%20el%20de%20evaluaci%C3%B3n..

[50] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural Networks*, vol. 11, n.º 4, págs. 761-767, 1998. DOI: https://doi.org/10.1016/S0893-6080(98)00010-0.

[51] *TensorFlow: tf.keras.callbacks.ModelCheckpoint*, Software available from tensorflow.org, 2022. dirección: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint.

[52] Wikipedia, *Red generativa antagónica — Wikipedia, La enciclopedia libre*, 2022. dirección: https://es.wikipedia.org/wiki/Red_generativa_antag%C3%B3nica.

[53] ——, *Retropropagación — Wikipedia, La enciclopedia libre*, 2022. dirección: https://es.wikipedia.org/wiki/Propagaci%C3%B3n_hacia_atr%C3%A1s.

[54] S. Kolungada, *Object Detection, Image Classification and Semantic Segmentation using AWS Sagemaker*, 2020. dirección: https://medium.com/@kolungade.s/object-detection-image-classification-and-semantic-segmentation-using-aws-sagemaker-e1f768c8f57d.

[55] B. Dickson, *New deep learning model brings image segmentation to edge devices*, 2021. dirección: https://venturebeat.com/2021/05/14/new-deep-learning-model-brings-image-segmentation-to-edge-devices/.

Apéndice A
Manual de Instalación

Lo primero que se necesita para poder instalar la aplicación es descargar Anaconda. Anaconda es el distribuidor de Python que se va a usar para poder tener la versión de Python necesaria para el funcionamiento de la app. La descarga se puede hacer desde la página oficial[1]. Se descargará con todas las configuraciones predeterminadas.

Una vez se haya instalado, abriremos la aplicación y se verá tal un menú como el de la Figura 62 dónde se podrá instalar y lanzar las aplicaciones que se encuentran en la plataforma. En este caso, instalaremos y usaremos PyCharm. PyCharm será el entorno de desarrollo de la aplicación web.

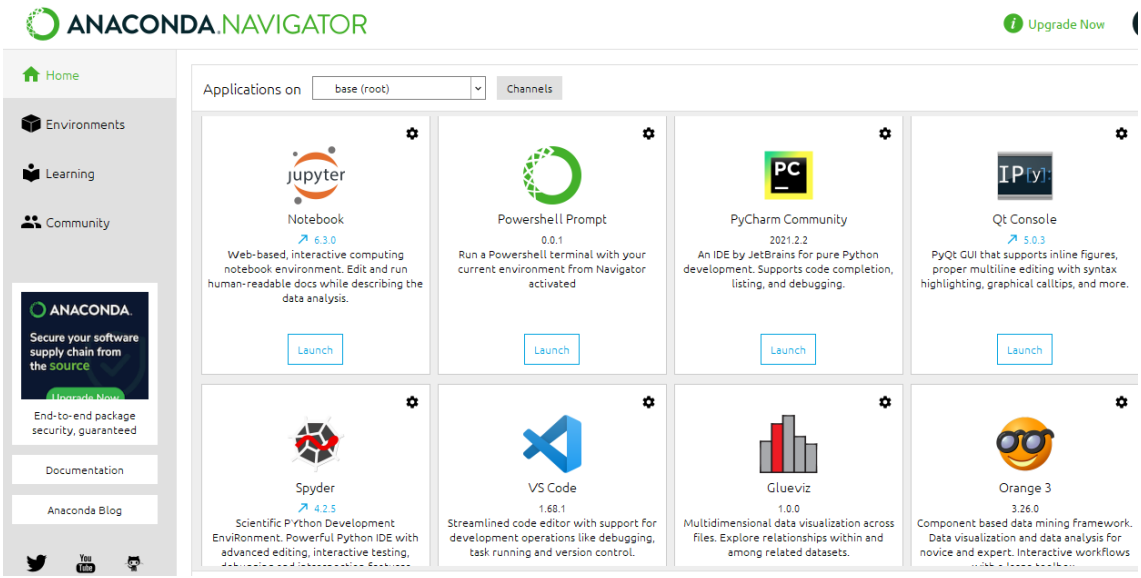

Figura 61: Interfaz de de inicio de Anaconda

Lo siguiente es crear un entorno con los paquetes necesarios para poder lanzar la aplicación. Para ello abrimos la terminal de Anaconda *(Anaconda Prompt)*. Para ello escribiremos el siguiente comando, dónde en nombre_entorno se debe poner el nombre que se desee. La **versión** de *Python* debe ser la **3.8** ya que *TensorFlow*, una librería usada en el desarrollo de la

[1] https://www.anaconda.com/products/distribution

aplicación no está soportada por versiones de *Python* superiores.

```
conda create -n nombre_entorno python=3.8
```

Cuando escribamos el comando nos preguntará si queremos instalar paquetes básicos cómo *pip* o *python*, le decimos que sí. Una vez hecho esto, se descargarán los paquetes y se creará el entorno. Para trabajar en él debemos activar el entorno usando *conda activate*, como se muestra en el siguiente comando:

```
#activar entorno
conda activate nombre_entorno
#desactivar entorno
conda deactivate
```

Tras activar el entorno, hay que instalar los paquetes necesarios para el correcto funcionamiento de la aplicación web. Para instalar los paquetes se podrá usar **pip install** o **conda install**. Se puede usar el de su preferencia, aunque en este caso se usará *pip install*. Es necesario ejecutar los siguientes comandos para instalarlos:

```
pip install flask
pip install tensorflow
pip install opencv-python
pip install pillow
```

Una vez instalado todo, ya está listo para usar la aplicación. Para ello deberá acceder a la carpeta del proyecto usando el siguiente comando, donde en *path* deberá poner la ruta de su ordenador dónde se encuentre el proyecto. El proyecto se verá como en la Figura 62. Dentro de *static* se encuentran las hojas de estilo y una carpeta de imágenes dónde se guardarán las imágenes que se procese, además de el modelo implementado. En la carpeta *templates* se encuentran los ficheros *HTML* usados para la creación de esta aplicación.

```
cd path
```

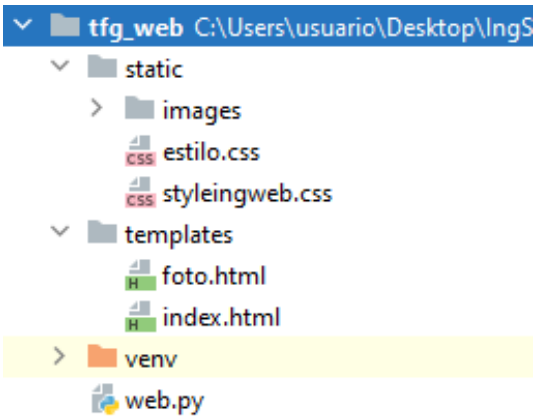

Figura 62: Archivos y carpetas que se encuentran dentro de la carpeta del proyecto

Una vez tengamos el entorno activado y nos encontremos dentro de la carpeta del proyecto, para iniciar la aplicación solo se deberá ejecutar el siguiente comando:

```
python web.py
```

Se iniciará el servicio en la dirección http://127.0.0.1:5000/.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA