TÍTULO

ANALISIS DE REDES NEURONALES CONVOLUCIONALES PARA EYE-TRACKING

AUTOR

Antonio Jesús Miranda Torres

Esta edición electrónica ha sido realizada en 2023

Tutores	D. Diego Marín Santos; D. Manuel E. Gegúndez Arias
Instituciones	Universidad Internacional de Andalucía
Curso	*Máster en Big Data (2021-2022)*
©	Antonio Jesús Miranda Torres
©	De esta edición: Universidad Internacional de Andalucía
Fecha documento	2023

MÁSTER EN BIG DATA
Universidad Internacional
de Andalucía

Rectorado
Sede de la Cartuja, Sevilla
www.unia.es/masterbigdata

UNIVERSIDAD INTERNACIONAL DE ANDALUCÍA

TRABAJO FIN DE MÁSTER

Análisis de Redes Neuronales Convolucionales para Eye-Tracking

Autor:
Antonio J. MIRANDA TORRES

Supervisor:
Diego MARIN SANTOS
Manuel E. GEGUNDEZ ARIAS

Trabajo presentado para completar los requerimientos para el Máster en Big Data

24 de febrero de 2023

UNIVERSIDAD INTERNACIONAL DE ANDALUCÍA

Abstract

MSc in Big Data

Convolutional Neural Networks Analysis for Eye-Tracking

by Antonio J. MIRANDA TORRES

Eye-tracking or gaze detection has historically been a problem widely analyzed by Artificial Intelligence methods due to its numerous possible integrations into everyday systems. However, technologies that have shown good enough results to be applied in the real world are expensive and require additional equipment. In this work, the results are analyzed on two trained architectures of the state of the art of Deep Learning such as VGG-16 and ResNet-50, comparing their results with respect to a new, simpler architecture proposed, which obtains such good results as the best of the previous ones, achieving a precision of 99.86 % on a *dataset* made up of 14,400 images belonging to four categories (eye closed, forward looking, left looking and right looking), which has been divided into 80 %, 10 % and 10 % for training, validation and testing.

Based on the results obtained in the experimentation, the extension of the problem to any environment of similar images is proposed, in order to generate a technology that is capable of classifying images in real time. Nonetheless, the results reached for this second objective are not so promising, and are indicators that it would be necessary to train the models on a set of images taken by a device with the same characteristics as the final output in order to achieve optimal results in an application such as the one proposed.

UNIVERSIDAD INTERNACIONAL DE ANDALUCÍA

Resumen

Máster en Big Data

Análisis de Redes Neuronales Convolucionales para Eye-Tracking

por Antonio J. MIRANDA TORRES

El *Eye-tracking* o detección de la mirada ha sido históricamente un problema ampliamente analizado por métodos de Inteligencia Artificial debido a sus numerosas integraciones posibles en los sistemas cotidianos. Sin embargo, las tecnologías que han mostrado resultados suficientemente buenos para ser aplicadas en el mundo real son costosas y requieren de equipamiento adicional. En este trabajo se analizan los resultados sobre dos arquitecturas entrenadas del estado del arte de *Deep Learning* como son VGG-16 y ResNet-50, comparando sus resultados con respecto a una nueva arquitectura propuesta más sencilla, que obtiene resultados tan buenos como la mejor de las anteriores, consiguiendo una precisión del 99,86 % sobre un *dataset* formado por 14.400 imágenes pertenecientes a cuatro categorías (ojo cerrado, mirando hacia adelante, mirando hacia la izquierda y mirando hacia la derecha), que se ha dividido en 80 %, 10 % y 10 % para entrenamiento, validación y prueba.

En base a los resultados obtenidos en la experimentación, se propone la extensión del problema a cualquier entorno de imágenes similares, con el fin de generar una tecnología que sea capaz de clasificar imágenes en tiempo real. No obstante, los resultados alcanzados para este segundo objetivo no son tan prometedores, y son indicadores de que sería necesario entrenar los modelos sobre un conjunto de imágenes tomadas por un dispositivo de las mismas características que la salida final para lograr resultados óptimos en una aplicación como la propuesta.

Universidad Internacional de Andalucía, 2023

Índice general

Índice de figuras

Índice de cuadros

List of Abbreviations

IA	**I**nteligencia **A**rtificial
CNN	**C**onvolutional **N**euronal **N**etwork
ILSVRC	**I**mageNet **L**arge-**S**cale **V**isual **R**ecognition **C**hallenge
FC	**F**ully-**C**onnected
TP	**T**rue **P**ositive
TN	**T**rue **N**egative
FP	**F**alse **P**ositive
FN	**F**alse **N**egative
GPU	**G**raphics **P**rocessing **U**nit

A aquellos profesores y compañeros que me han servido de inspiración y han motivado mi especialización en Ciencia de Datos e Inteligencia Artificial.

Capítulo 1

Propuesta del TFM

1.1. Motivación

La Inteligencia Artificial (IA) es un concepto que ha adquirido numerosas determinaciones de acuerdo a distintos enfoques, sin embargo, podría definirse desde un punto de vista práctico como la rama de las ciencias computacionales encargada de estudiar modelos de cómputo capaces de realizar actividades propias de los seres humanos en base a dos de sus características primordiales: el razonamiento y la conducta [Takeyas, 2007].

Los orígenes de la IA se remontan a la antigüedad, cuando se comenzó a reflexionar sobre la posibilidad de crear máquinas capaces de similar el comportamiento humano y su razonamiento lógico, aunque no fue hasta mediados del siglo XX cuando esta comenzó a desarrollarse de manera sistemática, convirtiéndose en un campo en constante evolución que ha ganado gran importancia en los últimos años por su potencial para resolver problemas complejos, habiéndose extendido su uso por áreas muy diversas desde la robótica hasta la medicina.

Dentro de este mayor desarrollo nace una subrama de la IA conocida como Aprendizaje Profundo (o *Deep Learning*, según su notación en inglés), que se basa en sucesiones de redes neuronales diseñadas para imitar la forma en la que el cerebro humano procesa la información, aprendiendo de la experiencia a medida que se expone a más datos, llegando a poder aprender patrones de forma autónoma. Estos algoritmos de *Deep Learning* han logrado resultados impresionantes en tareas complejas como el procesamiento de lenguaje natural, el reconocimiento de voz o el reconocimiento de imágenes.

La motivación a la realización de este proyecto nace de la idea de aprovechar y poner en valor algunas de las técnicas y algoritmos de *Deep Learning* desarrollados durante los últimos años para poder crear un sistema autónomo que a través del reconocimiento y aprendizaje de patrones provenientes de un conjunto de imágenes catalogadas en diferentes categorías, sea capaz de clasificar imágenes nuevas en alguna de esas categorías atendiendo a la similitud de sus características con las que ya ha utilizado previamente para aprender.

Más concretamente se pretende conseguir un algoritmo autónomo que permita clasificar imágenes de ojos humanos en diferentes categorías según la posición de las pupilas para conseguir trazar hacia donde se dirige la mirada del individuo en cuestión y dar una aproximación a un sistema que pueda realizar esta detección en tiempo real, el cual podría usarse, por ejemplo, como medida de seguridad vial integrada en un vehículo para establecer cuándo el conductor se encuentra mirando al

frente, y establecer algún tipo de aviso si se detecta que no lo hace durante un tiempo determinado, para que el conductor del vehículo pueda recuperar la atención.

1.2. Objetivos generales y específicos

Como ya se ha adelantado en el apartado anterior el **objetivo principal** de este proyecto será el de diseñar un algoritmo autónomo basado en redes neuronales que sea capaz de recibir como entrada imágenes de ojos humanos y realizar un reconocimiento y clasificación de ellas en distintas categorías con la máxima precisión posible, a través del aprendizaje adquirido a partir de un conjunto de imágenes inicial que se haya utilizado para entrenar al modelo diseñado.

El objetivo se puede catalogar de este modo como la resolución de un problema de Aprendizaje Supervisado de clasificación multiclase, para cuya resolución es necesaria la sucesión de una serie de pasos que se resumen a continuación:

1. Obtención de un conjunto de datos con las características adecuadas y su división en: un conjunto de entrenamiento que servirá para hacer aprender a los modelos, un conjunto de validación para medir el comportamiento del proceso de aprendizaje, y un conjunto de test para analizar su rendimiento final y comparar los modelos entre sí.
2. Diseño de los diferentes modelos de redes neuronales que se utilizan como propuesta.
3. Entrenamiento de dichos modelos durante un número de épocas suficiente para que lleguen a alcanzar un comportamiento óptimo.
4. Comparación de los resultados obtenidos por los modelos entrenados sobre el conjunto de test.
5. Elección del modelo más adecuado según la precisión obtenida en la evaluación de los resultados sobre el problema de clasificación para el cuál han sido creados.

Una vez superado el objetivo principal y teniendo, en consecuencia, como resultado un algoritmo lo suficientemente preciso para catalogar nuevas imágenes de ojos humanos en los diferentes grupos considerados como posibles salidas del problema, se plantea como **objetivo secundario** la maquetación de un boceto de aplicación que permita obtener una respuesta en tiempo real del modelo entrenado previamente a través de imágenes entrantes por medio de una cámara web.

1.3. Estructura del documento

Este escrito está divido en seis capítulos principales cuyo contenido se resume a continuación:

- Capítulo 1: en el que el lector se encuentra inmerso y sirve para poner en contexto la realización de este trabajo, exponiendo su motivación, estructura y los objetivos que persigue.

- Capítulo 2: se centra en, tras una primera introducción, describir el estado del arte en el campo de estudio del proyecto, planteando el problema y citando algunos otros trabajos de temática similar que han servido de motivación a este proyecto.
- Capítulo 3: está dedicado a la presentación de los materiales utilizados para la realización del estudio, es decir, los datos originales usados y la descripción de su tratamiento durante el desarrollo.
- Capítulo 4: presenta la metodología seguida en el proyecto, las redes neuronales empleadas y su adaptación al problema planteado anteriormente, así como todo lo relativo al entrenamiento de los modelos.
- Capítulo 5: centrado en la exposición y en el análisis de los resultados obtenidos sobre el conjunto de prueba de los modelos entrenados en el capítulo precedente, atendiendo a una serie de métricas explicadas.
- Capítulo 6: sirve de cierre mediante la presentación de la valoración de los resultados, las conclusiones obtenidas y la propuesta de trabajos futuros relacionados.

Capítulo 2

Introducción y estado del arte

El método de resolución del problema presentado en el capítulo anterior estará basado en el empleo de redes neuronales convolucionales (CNN), unas estructuras de aprendizaje profundo capaces de, a través de la aplicación de varias capas de filtrado, extraer características de unos datos de entrada. Estas arquitecturas han resultado especialmente eficaces en su utilización para el tratamiento de imágenes y han sido extensamente usadas en aplicaciones de visión artificial por computador. En este apartado se describe brevemente el desarrollo de esta tecnología a lo largo del tiempo atendiendo a algunos de los hitos más importantes en su historia, para después terminar citando algunos trabajos similares al propuesto que se tomarán como referencia.

2.1. Contexto histórico

No puede entenderse el desarrollo de la implementación de redes neuronales convolucionales para la detección y clasificación de imágenes sin la creación, por parte de Universidad de Stanford junto al equipo de investigación del laboratorio de inteligencia artificial de Google, del conjunto de datos **ImageNet** [Deng et al., 2009]. Una base de datos que contiene más de catorce millones de imágenes diferentes categorizadas jerárquicamente entre más de veinte mil clases distintas en el nivel superior y más de quinientas mil etiquetas en el inferior de los niveles. Su aparición supuso un cambio de paradigma en el entrenamiento de modelos de aprendizaje automático, ya que pretendía la mejora de sus rendimientos a través de la utilización de un mayor conjunto de datos de entrenamiento, en lugar de centrarse únicamente en mejorar los propios modelos, como se había estado haciendo hasta entonces.

A partir de subconjuntos de esta base de datos se inicia en 2012 un desafío anual conocido como *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [Russakovsky et al., 2015] para fomentar el desarrollo y la evaluación comparativa de algoritmos de aprendizaje automático de última generación sobre, entre otros problemas, la clasificación de imágenes.

La primera edición de este concurso fue ganada con una amplia diferencia respecto a sus competidores por una red conocida como **AlexNet** [Krizhevsky, Sutskever e Hinton, 2017] que a través de uso de una CNN de ocho capas (cinco de ellas convolucionales) demostró que las características o *features* aprendidas de forma autónoma con un paradigma de *Deep Learning* podían transcender a las *features* diseñadas de forma manual a través de un enfoque de *Machine Learning*. Este hito provocó que el *Deep Learning* se posicionara en primer plano de la IA. Este algoritmo se aprovechó además del desarrollo de las GPUs, que hicieron posible entrenar

una red con millones de parámetros en un par de días, algo impensable en la anterioridad. Consiguió una precisión del 83,6 % en la clasificación de imágenes de ImageNet, introduciendo la función de activación *ReLU*, el aumento de datos para entrenamiento y el concepto de *Dropout* para regularización.

En 2014 aparecen las arquitecturas **VGGNet** [Simonyan y Zisserman, 2014] en el ILSVRC, que introducen una forma nueva de generar redes convolucionales, caracterizadas por su simplicidad arquitectónica y por el refuerzo de la idea de mejorar el rendimiento a través del aumento de la profundidad de las CNN. Se basan en la construcción de un bloque básico formado por una capa convolucional con el mismo *padding* para mantener la resolución, una función de activación *ReLU* y una capa de *pooling* para reducir la dimensionalidad, que se apilan de forma consecutiva. Las capas convolucionales tienen asociado un *kernel* óptimo que se fija para todos los bloques del proceso de apilamiento. Esta arquitectura en su versión de 16 bloques (VGG-16) consiguió mejorar el *accuracy* sobre el problema de clasificación de ImageNet hasta un 92,7 %.

Los algoritmos de CNN presentados hasta ahora en este escrito tenían en común que extraen las características a través de una secuencia de capas de convolución y de *pooling*, y post-procesan estas características a través de capas densas al final para realizar la clasificación. La arquitectura **NiN** [Lin, Chen y Yan, 2013] propone que estas capas densas se integren también en medio del procesamiento. Con esta idea implementa una analogía a las capas densas que funciona como una convolución 1x1 para conservar la estructura espacial de la imagen. De esta forma se puede ir obteniendo información sobre la decisión durante el propio proceso de entrenamiento, pudiendo eliminar las capas densas finales, reduciendo el número de parámetros necesarios. Esta arquitectura, a pesar de no obtener resultados demasiado buenos, sirvió de inspiración a algoritmos posteriores.

La arquitectura **GoogleNet** [Szegedy et al., 2015] resultó ganadora en 2014 del reto de clasificación de imágenes del ILSVRC mediante la propuesta de una estructura que combinaba las fortalezas de NiN y el paradigma de la repetición de bloques. Mientras se discutía cuál era el tamaño de *kernel* óptimo, los creadores de este algoritmo propusieron la implementación de bloques *Inception* que consistían en una combinación de *kernels* de distinto tamaño en caminos paralelos, diversificando las *features* y concatenándolas al final. De esta forma a la salida del bloque se conseguía tener información obtenida a diferentes escalas, generándose así *features* más expresivas. Además, consiguieron mediante el uso de estos bloques reducir de una forma considerable el número de parámetros y de cálculos requeridos. Consiguió un resultado de 93,3 % de precisión en la clasificación de imágenes de ImageNet.

Adicionalmente, GoogleNet populariza el uso de **Batch Normalization** o normalización por lotes. Este concepto consiste en, dado que tras cada capa los datos se deforman, normalizar la entrada de cada capa según los datos disponibles del *dataset*, que es procesado en forma de lotes. Es decir, se aplica una normalización teniendo en cuenta los datos del lote que se está procesando en cada paso del entrenamiento. Se implementa en forma de una capa adicional y se incluye un cierto ruido para regularizar el entrenamiento y evitar un *overfiting* de la red.

La triunfadora del desafío ILSVRC en 2015 fue la arquitectura **ResNet** [He et al., 2016] que consiguió un *accuracy* de 96,4 % sobre ImageNet y se consolidó como el

trabajo más innovador en la visión artificial desde AlexNet. Su importancia radica en que hasta ahora introducir más capas no siempre mejoraba el resultado (redes más complejas no eran estrictamente más expresivas), pero con su inclusión del *Bloque Residual* se garantizaba un mejor resultado añadiendo más bloques (mejora de la expresividad). La idea de estos bloques es que puedan transformarse en una función identidad, en cuyo caso es como si no estuvieran y sería equivalente a la red anterior y por tanto la red de n bloques contendrá a la red de $n-1$ bloques; luego el nuevo modelo será como mínimo igual de efectivo que el anterior. Para ello, los bloques contienen una conexión residual que suma a la salida del bloque su entrada de manera que el aprendizaje tiene lugar a partir del residuo (la diferencia entre su salida y su entrada). Si el bloque se hace nulo, la salida es igual a la entrada lo que implica que actúe como una función identidad, haciendo que añadir más capas no pueda perjudicar al resultado del aprendizaje. Se lograba de esta forma poder entrenar redes de cientos y miles de capas (a costa de un mayor tiempo de computación y de entrenamiento), consiguiendo resultados estrictamente mejores a medida que se incluyeran más capas. Además, el uso de las conexiones residuales implicaba una mejora en el flujo de gradiente.

El último gran trabajo reconocido en el ámbito del reconocimiento de imágenes por computador tuvo lugar en 2017, la arquitectura **DenseNet** [Huang et al., 2017] que no llegó a participar en el ILSVRC, ya que para entonces el certamen se había cancelado, y que surge como la evolución de ResNet. Su idea principal es la de componer la red en bloques densos en los que sus capas estén densamente conectadas entre sí de modo que la salida de un bloque recibe una supervisión por parte de una concatenación de las salidas de cada uno de los bloques anteriores. De esta manera, como cada capa recibe mapas de activación de todas las capas anteriores, la red puede ser más delgada y compacta, esto es, reducir su número de canales. Se consiguen a través de este método un flujo de gradiente aún mejor ya que las capas están interconectadas entre sí, una mayor eficiencia computacional y la diversificación de las *features* dada por la reutilización de la salida de cada bloque.

Como punto final de este apartado es conveniente introducir el concepto de **Transfer learning**, que hace referencia a la reutilización de un modelo de aprendizaje ya entrenado para resolver un problema relacionado. Este es un método comúnmente empleado en tareas de visión por computadora, ya que permite reusar alguno de los modelos presentados que ya han sido entrenados previamente sobre enormes conjuntos de datos y aprovechar su conocimiento ya adquirido para inicializar los parámetros de un nuevo modelo que sirva para resolver un problema relacionado mediante un nuevo entrenamiento sobre un *dataset* más reducido y específico para la nueva tarea. Se disminuye así la cantidad de datos y de cálculos necesarios para llevar a cabo un nuevo entrenamiento.

2.2. Trabajos relacionados

Son numerosos los artículos académicos publicados durante los últimos años que han tratado problemas similares al que se plantea en este trabajo sobre el reconocimiento y clasificación de imágenes por computador a través del empleo de redes neuronales, tanto a partir de la creación de CNN propias, como de la utilización de alguno de los modelos clásicos presentados en el apartado anterior como base del entrenamiento a partir de transferencia de conocimiento (*Transfer learning*)

para adaptarlos a un conjunto de datos concreto característico del problema que se pretenda abordar. A continuación, se describen algunos de los trabajos que se han tomado en consideración como motivación o como referencia para este trabajo.

En el primero de ellos [Chinsatit y Saitoh, 2017], los autores desarrollan un método para detección del centro de las pupilas que está basado en dos modelos de CNN. El primero de ellos se usa para clasificar imágenes en tres categorías según el estado del ojo (entreabierto, cerrado o semiabierto), y el segundo toma las imágenes no clasificadas como ojo cerrado y aplica una regresión para dar la posición de la pupila. Se emplea un dataset de 19.600 imágenes de ojos tomadas sobre diez individuos, todas sin gafas, divididas entre las tres categorías citadas. Para resolver el primer problema de clasificación emplean una CNN formada por cinco capas de convolución y dos capas densas regularizadas con *Dropout*, que había sido preentrenada usando el *dataset* ImageNet, consiguiendo un acierto en la clasificación entre las tres categorías de un 82,6 % en una evaluación del tipo *leave-one-out cross-validation*, resultados que se consideraron aceptables en esta aproximación.

Con la motivación de desarrollar una aplicación de detección *"low-cost"* de dirección de la mirada en tiempo real que pudiera ayudar a personas con trastornos motrices a dirigir sus sillas de ruedas en [Dahmani et al., 2020] los autores desarrollan un modelo de CNN para compararlo con modelos clásicos de *Machine Learning* basados en la similitud a unas plantillas. Para ello recopilaron dos mil imágenes de cada uno de los ocho individuos que participaron en el experimento, divididas equitativamente en cuatro categorías: mirada al frente, derecha, izquierda y ojo cerrado. Estas imágenes tenían tres canales de profundidad y una dimensión de 64x64 píxeles. La red utilizada constaba de dos bloques formados cada uno por una capa convolucional, una capa de *pooling* para reducir la dimensionalidad y una capa densa. Aplicando *5-fold cross-validation* para el entrenamiento se obtuvo una *accuracy* global sobre el conjunto de test de 99,3 %, siendo del 100 % para cinco de los ocho individuos estudiados, y del 96,875 % sobre el individuo con peor resultado. El experimento se consideró un éxito ya que el modelo basado en CNN superó el rendimiento de los modelos más tradicionales.

En [Ansari, Kasprowski y Obetkal, 2021] nuevamente se plantea una alternativa barata a una aplicación de detección de posición de la mirada en la interacción *"humano-máquina"* a partir de la cámara web de un ordenador portátil empleado un modelo de CNN sobre imágenes que no tienen alta resolución. Estas imágenes se recolectaron por medio de una aplicación que toma imágenes sobre un individuo al que hace mirar sobre diferentes puntos de la pantalla para capturar sus ojos en distintas posiciones oculares. El estudio se hace sobre dos *datasets* distintos, el primero consta de unas seis mil imágenes sobre un mismo individuo focalizando su mirada en veinte puntos de la pantalla distintos, considerados como categorías en la clasificación, llegando a unas trescientas imágenes por punto de atención y siendo imágenes de ojo izquierdo y derecho, así como de la cara completa. El segundo *dataset* contiene características similares, pero sobre cuatro individuos (con y sin gafas) por lo que su tamaño es el cuádruple. Se realizan tres experimentos: considerar los ojos por separado, considerar la cara completa y considerar una concatenación de ambos ojos. En cualquiera de los casos se emplean CNNs con tres tipos de componentes: capas de convolución, capas de *pooling* y capas densas, en diferentes combinaciones adecuándose a la complejidad de cada experimento. Los resultados arrojan un

comportamiento mejor para el primer conjunto de datos (como es lógico al centrarse entrenamiento y evaluación sobre un único individuo) entre el 77 % y el 81 % de precisión en la clasificación entre las veinte categorías, y de entre 72 % y 80 % para el segundo *dataset*. Concluyendo en un buen resultado dada la complejidad de un problema que en principio sería más habitual resolver como una regresión.

Por último, aunque se escape del ámbito de estudio de este trabajo (*Eye-tracking*) se hace mención al artículo [Marin-Santos et al., 2022], ya que la CNN presentada se replicará de forma simplificada en este proyecto y se compararán sus resultados con los obtenidos sobre algunas de las CNN clásicas preentrenadas sobre ImageNet, de forma similar a como se hace en este escrito en el que los autores desarrollan un modelo de CNN de aplicación médica para detectar la presencia de la enfermedad de Crohn en imágenes endoscópicas disponibles en un *dataset* de 15.972 imágenes recopiladas sobre pacientes afectados por esta enfermedad, en la que aproximadamente la mitad de las imágenes muestran lesiones relacionadas con la enfermedad. Finalmente, se concluye que el modelo propuesto es capaz de clasificar correctamente el 99 % de las imágenes con lesión de un conjunto de test, y lo hace en términos de eficiencia mejor que las otras arquitecturas de referencia en el estado del arte (EfficientNet-B5, VGG-16, Xception o ResNet).

2.3. Definición del problema

Como ya se citó en el apartado de descripción de objetivos se pretende resolver un problema de Aprendizaje Supervisado consistente en una clasificación multiclase, en el que concretamente se plantea un contexto similar al de los artículos descritos en el apartado anterior. El fin en este caso es conseguir clasificar con la mayor precisión posible elementos de una colección de imágenes de ojo humano en cuatro categorías diferenciadas según su estado: mirada al frente, mirada a derecha, mirada a izquierda y ojo cerrado.

Con este objetivo se hará un estudio comparativo entre tres arquitecturas para tomar la de mejor rendimiento según unas métricas de evaluación consideradas. Las arquitecturas que se van a analizar son una simplificación a la red descrita en [Marin-Santos et al., 2022] y dos de los modelos de referencia en el estado del arte, preentrenados sobre el conjunto ImageNet: VGG-16 y ResNet-50. Sobre todos estos sistemas se realizarán pequeñas transformaciones para adaptarlos al problema presentado.

Capítulo 3

Materiales

3.1. Base de datos

El conjunto de datos [Shah, 2020] que se va a tratar para alcanzar los objetivos definidos en este trabajo es un *dataset* de dominio público disponible en la plataforma `Kaggle`. Consiste en una recopilación de imágenes ideada originalmente para conseguir automatizar el movimiento de una silla de ruedas a través de un control ocular basado en la dirección de la mirada.

Estas imágenes fueron coleccionadas a través de `Google Images` y editadas posteriormente para conseguir una adecuada posición angular. Sobre esta colección inicial de imágenes se aplicaron métodos de aumento de datos hasta llegar a un conjunto final de unas 14.400 imágenes con tres canales de color y separadas entre las cuatro siguientes categorías:

- ***left_look*:** imágenes del ojo mirando hacia la izquierda
- ***right_look*:** imágenes del ojo mirando hacia la derecha
- ***close_look*:** imágenes del ojo cerrado
- ***forward_look*:** imágenes del ojo mirando al frente

Es preciso mencionar que en la base de datos aparecen indistintamente imágenes de ambos ojos (izquierdo y derecho) sin ninguna diferenciación adicional. Además, a la hora de clasificar las imágenes se toma como referencia el punto de vista de la cámara, es decir, si en la imagen el ojo aparece mirando a la izquierda se clasifica como *left_look*, y viceversa, aunque desde el punto de referencia del humano fotografiado la dirección de la mirada fuera la contraria. De esta forma el número total de imágenes que forma cada uno de los cuatro subconjuntos o categorías se detalla en el Cuadro 3.1.

left_look	*right_look*	*close_look*	*forward_look*
3498	3601	3828	3457

CUADRO 3.1: Número de imágenes total por categoría

Aunque se aprecian ciertas diferencias entre el número de imágenes pertenecientes a cada agrupación, se podría considerar que el conjunto de datos está aproximadamente balanceado por lo que no deberían producirse errores en el entrenamiento por la presencia de alguna clase dominante sobre el resto. En la Figura 3.1 puede verse una pequeña muestra formada por cinco imágenes pertenecientes a cada una

de las cuatro categorías analizadas en el *dataset* tratado. El tamaño de las imágenes es variable como puede comprobarse en la representación por lo que será necesario para el procesamiento escalar las imágenes a unas dimensiones de largo y ancho comunes, lo que se hará de acuerdo a la entrada del modelo de red neuronal diseñado en cada caso.

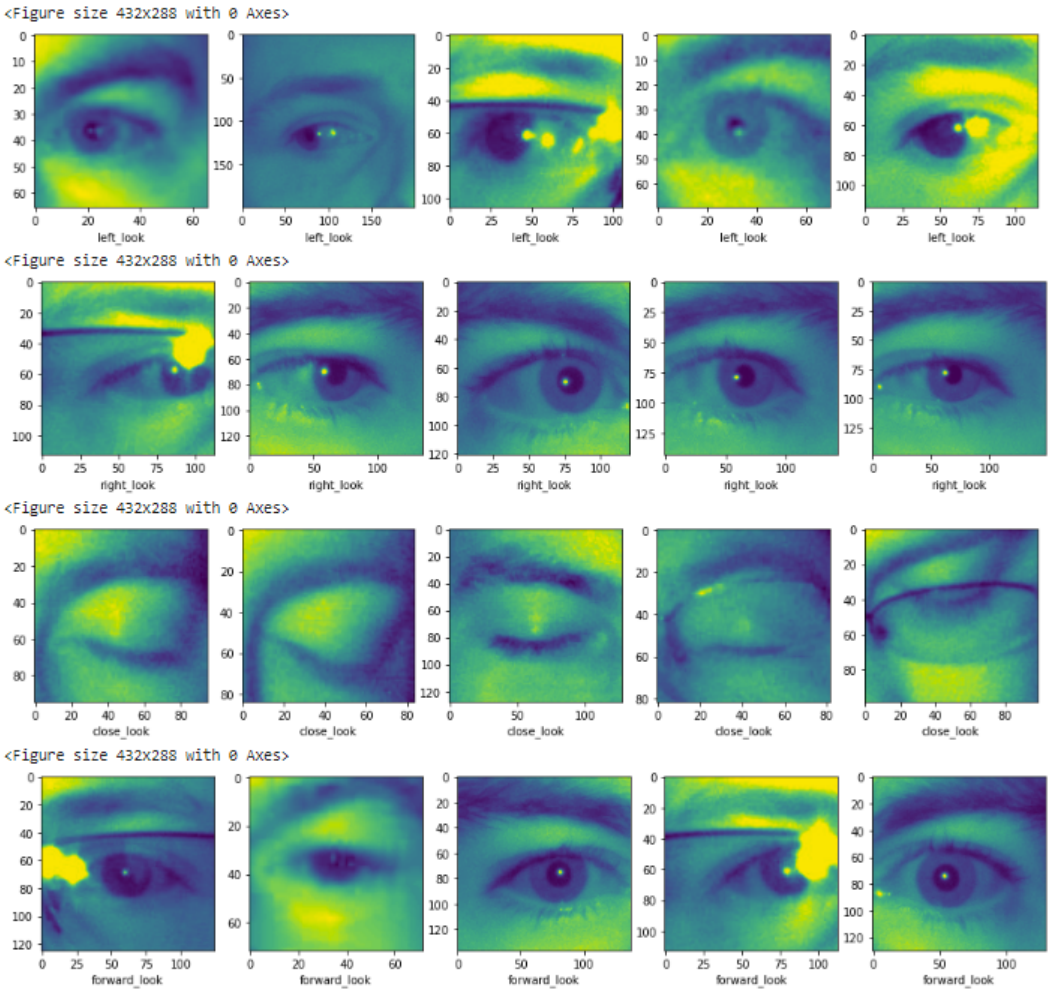

FIGURA 3.1: Muestra de cinco imágenes de cada una de las categorías presentes en el conjunto de datos analizado

3.2. Generación de datos entrada/salida del modelo

Con el fin de poder entrenar y probar correctamente los modelos propuestos en posteriores capítulos de este escrito es importante realizar una separación de la base de datos seleccionada en distintos subconjuntos. En este caso se hará una división de los datos de cada una de las categorías en un 80/10/10 entre tres conjuntos de entrenamiento/validación/prueba cuyos objetivos se detallan a continuación.

- ***Training*:** el subconjunto de entrenamiento estará compuesto por el 80 % de los datos de cada categoría y servirá como entrada a los modelos para su entrenamiento. Sobre este conjunto se realizarán operaciones de aumento de datos

para que en cada época las imágenes sobre las que se produce el entrenamiento sean distintas y no se produzca sobreaprendizaje respecto a ellas, ya que, sobre este conjunto se realiza el ajuste de los pesos de la red durante su proceso de optimización. Se evita de esta forma que los parámetros de las redes entrenadas queden sobreajustados a las imágenes del conjunto sin modificar.

- ***Validation*:** el subconjunto de validación estará formado por un 10 % de datos de cada clase y su fin es el de evaluar el entrenamiento tras cada época. Este conjunto, por tanto, permanecerá invariante para no condicionar los resultados de la evaluación a imágenes deformadas por procesos de aumento de datos. De esta forma las evaluaciones tomarán como referencia imágenes distintas a las que se hayan utilizado para el aprendizaje, y permitirán elegir el modelo con mejor rendimiento sobre ellas.

- ***Test*:** el conjunto de prueba lo formarán el 10 % restante de datos de cada categoría y tendrá como objetivo evaluar los modelos finales seleccionados según el mejor rendimiento sobre el conjunto de validación, sobre imágenes distintas a ambos conjuntos anteriores y de esta forma poder comparar los diferentes modelos implementados para la clasificación de imágenes. Las imágenes de este conjunto también permanecen invariantes durante todo el proceso por motivos análogos a la validación.

La cantidad de imágenes contenida en cada uno de los tres subconjuntos de datos descritos queda resumida en el Cuadro 3.2, en el que puede comprobarse que se cumple la regla del 80/10/10 en cada una de las categorías por separado para así mantener los tres conjuntos finales balanceados.

	left_look	*right_look*	*close_look*	*forward_look*
Training	2798	2880	3062	2765
Validation	350	360	383	346
Test	350	361	383	346

CUADRO 3.2: Número de imágenes de los subconjuntos de entrenamiento/validación/prueba por categoría

Las imágenes de los tres conjuntos se procesan en tres canales de color, lo que es adecuado para las redes que se van a estudiar, sin embargo, es necesario realizar un procesamiento previo de las imágenes que consiste en el redimensionamiento de la altura y anchura de las imágenes de forma que el formato de estas coincida con las dimensiones admitidas por la capa de entrada de las arquitecturas de redes neuronales analizadas. En este sentido, es necesario un redimensionamiento de 224x224x3 sobre todas las imágenes para ser procesadas por las redes propuestas VGG-16 y ResNet-50, mientras que se adaptará el formato a 320x320x3 para el caso de la arquitectura de CNN propuesta en [Marin-Santos et al., 2022]. Además, para mejorar el rendimiento de los modelos se realiza un escalado de los píxeles que conforman cada uno de los tres canales de las imágenes para que tomen valores normalizados entre 0 y 1. Por su lado las etiquetas de clase asociadas a cada imagen se procesan con formato *one-hot*, es decir, como un vector de cuatro componentes en el que la posición correspondiente a la clase a la que pertenece la imagen toma valor 1 y el resto tienen valor 0.

Adicionalmente, como ya se ha comentado sobre el conjunto de entrenamiento se realizarán antes de cada época de entrenamiento una serie de transformaciones de

aumento de datos que consistirán en giros, recortes de los márgenes y acercamientos de forma aleatoria sobre la imagen. Estas transformaciones tienen el fin de que las imágenes usadas en cada época de entrenamiento sean en cierto modo distintas y los parámetros del aprendizaje no se sobreajusten a unas imágenes concretas, en un fenómeno de sobreaprendizaje que provoque que el modelo después no sea capaz de clasificar de forma correcta nuevas imágenes.

En cuanto a la salida del modelo, las tres redes analizadas serán adaptadas de modo que tengan como salida para cada imagen procesada un vector de cuatro componentes (igual al número de clases en las que está divida el *dataset*), que representarán la probabilidad que tiene la imagen de pertenecer a cada una de las clases (la función *soft-max* devuelve un vector para el cual la suma de sus componentes es igual a 1). De este modo, calculando el valor máximo del vector y su posición se puede obtener la clase a la que la imagen pertenece según la predicción del modelo, transformando este vector a formato *one-hot* para poder compararlo con la etiqueta real de la entrada asociada.

Capítulo 4

Metodología

4.1. Propuesta metodológica

Se analizarán en este proyecto tres arquitecturas distintas de red neuronal para, tras entrenarlas y comparar sus resultados en la evaluación del problema de clasificación de imágenes planteado, terminar seleccionado para su aplicación a la que genere un mejor rendimiento sobre el conjunto de datos de prueba reservado. La metodología seguida para los tres casos será la misma y se detalla a continuación.

En primer lugar, se generan los modelos. Dos de ellos están basados en arquitecturas del estado del arte en reconocimiento de imágenes por visión artificial, concretamente en las arquitecturas VGG-16 y Resnet-50. En ambos casos, se carga la arquitectura disponible en las librerías de `Tensorflow` pero es necesario adaptar su salida que está condicionada para un problema de 1000 clases de la base de datos de `ImageNet`, al caso concreto estudiado en el que el vector de salida tendrá que contemplar solamente cuatro clases. En el tercero de los casos, el modelo se crea desde cero en base a la red neuronal propuesta en [Marin-Santos et al., 2022], también adaptándola consecuentemente al problema específico.

Una vez definidos los modelos se compilan de acuerdo con una función de pérdida que se define para calcular la desviación de las predicciones durante el entrenamiento, a un optimizador que irá actualizando y adecuando progresivamente los pesos de la red, y a una métrica de evaluación que se usará para analizar el rendimiento del modelo durante el proceso de validación que tiene lugar tras cada etapa de entrenamiento.

En paralelo, se crean para el entrenamiento dos generadores de datos que agrupan los conjuntos de entrenamiento y de validación de forma aleatoria para cada época de entrenamiento en lotes de 32 imágenes para facilitar el procesamiento y permitir *Batch Normalization* en los modelos en los que pueda ser requerido, mejorando su rendimiento. Sobre el generador asociado al conjunto de entrenamiento se aplican las operaciones de aumento de datos indicadas en el capítulo previo para evitar que los pesos del modelo se sobreajusten a un conjunto de imágenes estáticas determinadas y sea incapaz de interpretar de forma adecuada imágenes nuevas, cuando así sea requerido.

El siguiente paso consiste en el entrenamiento de cada uno de los modelos planteados utilizando como entrada los datos agrupados en lotes por parte de los generadores definidos. En cada etapa, constituida por el procesamiento de las imágenes de cada lote, se ajustan los pesos de la red según la función de pérdida definida para los datos de entrenamiento y después se evalúa el modelo con los datos de validación,

según la métrica seleccionada. El entrenamiento tendrá lugar durante un número de épocas (una época hace referencia al procesamiento por parte del modelo de la totalidad los datos de entrenamiento disponibles) lo bastante alto para llegar a unos resultados suficientemente buenos.

Por último, se guarda el modelo con parámetros ajustados correspondiente a la época en la que el resultado de la validación ha sido el mayor, de forma que pueda ser utilizado para predecir y comparar los resultados respecto al conjunto de *test* que había quedado reservado a lo largo de este proceso. La evaluación sobre este tercer conjunto permite comparar las diferentes arquitecturas entre ellas atendiendo a unas métricas de evaluación definidas y obtener conclusiones al respecto.

4.2. Modelos de redes utilizados

Este apartado está dedicado a la descripción de la arquitectura de cada uno de los modelos cuya utilización se ha propuesto analizar para resolver el problema de reconocimiento y clasificación de imágenes planteado en este trabajo.

4.2.1. VGG-16

El primero de los modelos propuestos está basado en una de las arquitecturas clásicas de detección artificial de imágenes conocida como VGGNet [Simonyan y Zisserman, 2014] que resultó ganadora del desafío de reconocimiento de imágenes sobre el conjunto de datos ImageNet (ILSVRC) en el año 2014, en su versión VGG-16.

En la Figura 4.1 queda representada la arquitectura de esta primera red neuronal analizada que está caracterizada por su sencillez de construcción basada en la repetición de un bloque común. El dieciséis hace referencia al número de capas que tienen pesos asociados, o dicho de otra forma, las que tienen capacidad de aprendizaje que resultan ser 13 capas convolucionales y 3 capas densas o *Fully-Connected* (FC).

El vector de entrada a la red será de tamaño 224x224 con 3 canales de color RGB. Una vez la imagen se introduce en la red se encuentra con los 5 bloques característicos de la red formados por 2 capas convolucionales los 2 primeros bloques y por 3 de estas los 3 bloques siguientes. Todas estas capas convolucionales tienen un *kernel* fijo de 3x3 y el mismo *padding*, por lo que no reducen la dimensionalidad, ya que con este objetivo se colocan al final de cada uno de los cinco bloques una capa de *pooling* con un filtro 2x2 y un *stride* igual a 2, que reducen la dimensión del vector a la mitad para proporcionárselo como entrada al siguiente bloque. La totalidad de las capas convolucionales incluyen una función de activación *ReLU* y un número de filtros que se duplica en cada bloque desde 64 en el bloque inicial hasta 512 en el cuarto bloque, manteniéndose en 512 también en el quinto y último bloque.

Tras los 5 bloques convolucionales, una capa de *flatten* transforma el vector saliente a una única dimensión y el vector unidimensional resultante pasa por otro bloque de tres capas densas (en el modelo original), dos de ellas con 4096 canales cada una y función de activación *ReLU*, y una tercera con 1000 canales (tantos como clases distintas tiene el problema original de ImageNet) y una función de activación *soft-max* que genera el vector final de salida del sistema. El número total de

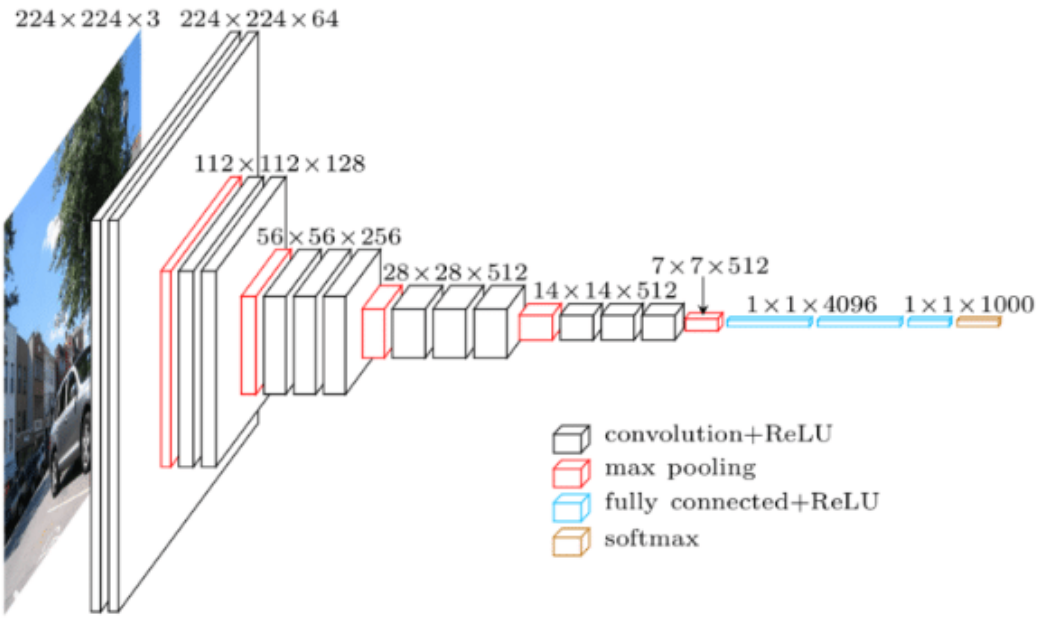

FIGURA 4.1: Arquitectura del algoritmo VGG-16 [Sugata y Yang, 2017]

parámetros entrenables teniendo en cuenta todas las capas citadas es superior a 138 millones.

Para la adaptación de esta arquitectura al problema estudiado se tiene que realizar una modificación sobre la capa de salida, ya que las posibles clases del conjunto del problema en cuestión es 4. Para ello, se elimina esta última capa del modelo y se sustituye por una de las mismas características, pero ajustando el número de canales a 4, lo que hace que el número de parámetros entrenables posibles quede en 134.276.932. Además, se añade una capa de *Dropout* con valor 0,3 antes de esta capa de salida para frenar el sobreajuste a los datos de entrenamiento, ya que en pruebas iniciales podía apreciarse una tendencia de sobreaprendizaje.

Por último, se decide que de todas las posibles capas de entrenamiento se congelarán todas menos la capa de salida añadida para aprovechar el aprendizaje realizado sobre el conjunto ImageNet (siguiendo el concepto de *Transfer Learning*) y aplicarlo directamente sobre el conjunto de datos del problema concreto estudiado. Esto hace que el número final de parámetros entrenados en el proceso de aprendizaje para este problema se reduzca a 16.388.

4.2.2. ResNet-50

La segunda propuesta de modelo analizado toma como base la arquitectura ResNet [He et al., 2016], otra de las arquitecturas clásicas del estado del arte, que triunfó en el desafío ILSVRC en 2015, introduciendo la idea de los bloques residuales que supusieron una gran innovación implicando que añadir un número mayor de capas convolucionales solo pudiera suponer una mejora en el rendimiento de los modelos. La versión de esta arquitectura usada como referencia es aquella con 50 capas convolucionales entrenables, ResNet-50.

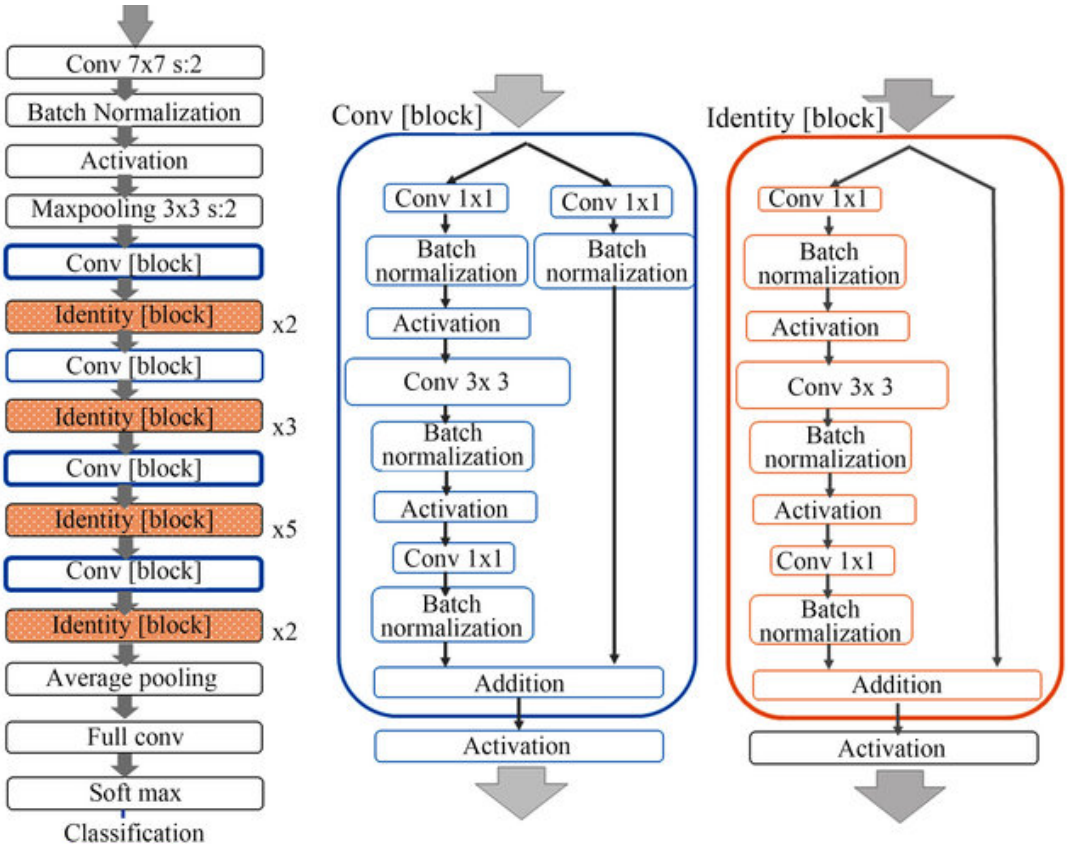

FIGURA 4.2: Arquitectura del algoritmo ResNet-50 [Matsuyama, 2020]

La arquitectura de este algoritmo, que se refleja en la Figura 4.2, es bastante más compleja que la VGG-16, La inclusión de bloques residuales hace que tenga sentido implementar redes mucho más profundas y, gracias a otros avances paralelos en el arte, esta red incluye *Batch Normalization* y emplea capas de convolución 1x1 a lo largo del proceso que actúan de forma similar a las capas densas ayudando a la toma de decisión durante el procesamiento. El uso de estas capas presentes de forma intercalada en la arquitectura permite reducir el uso de capas densas con muchísimos parámetros justo antes de la salida del modelo, lo que habilita que el entrenamiento sea mucho más rápido. Tras cada capa convolucional se ejecuta *Batch Normalization* sobre los datos del lote procesado y una función de activación *ReLU*.

En más detalle, en la Figura 4.3, puede notarse que tras la entrada de una imagen, que de nuevo está en formato 224x224 y con 3 canales de color RGB, se sucede una serie formada por los elementos siguientes:

- Una primera capa de convolución de kernel 7x7 con 64 canales y un *stride* igual a 2 que reduce la dimensión a la mitad.
- Una capa de *pooling* con *stride* de 2, que vuelve a reducir a la mitad la dimensionalidad.
- Un bloque formado por 3 capas convolucionales de *kernel* 3x3 con 64 canales, *kernel* 1x1 con 64 canales y *kernel* 1x1 con 256 canales; que se repite 3 veces.
- Un nuevo bloque con otras 3 capas convolucionales que se repite ahora 4 veces: *kernel* 1x1 con 128 canales, 3x3 con 128 canales y 1x1 con 512 canales.

- Otro bloque de 3 capas convolucionales que en este caso se repite 6 veces: *kernel* 1x1 con 256 canales, 3x3 con 256 canales y 1x1 con 1024 canales.
- Un último bloque con otras 3 capas convolucionales repetido 3 veces: *kernel* 1x1 con 512 canales, 3x3 con 512 canales y 1x1 con 2058 canales.
- Una capa de *Average pooling* seguida de una capa dense de 1000 nodos (en el problema original de ImageNet) con activación *soft-max*.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

FIGURA 4.3: Detalle de la arquitectura del algoritmo ResNet-50 [He et al., 2016]

Teniendo en cuenta el número de redes convolucionales de cada bloque y sus repeticiones más la capa densa final se tiene un modelo de 50 capas entrenables que contienen 25.568.360 parámetros de entrenamiento posibles, un número más de 5 veces menor al de la arquitectura VGG-16.

Para la adaptación al problema concreto se toma un procedimiento similar al modelo anterior. Se carga la red sin su última capa densa de salida, que es sustituida por una red FC de 4 nodos para hacerla concordar con las clases de salida posibles y se le asigna la función de activación *soft-max*. Se añade igualmente un Dropout de 0,3 antes de esta última capa para tratar posibles problemas de sobreajuste, junto con un *pooling* 2x2 para reducir la dimensionalidad de la matriz de salida de la red original y una capa de *Flatten* para aplanar la matriz a un vector de una sola columna y que pueda ser procesado por el *Dropout* y la capa densa final. El número de parámetros entrenable es de 23.519.360 tras esta transformación. En este caso, se decide congelar las capas de entrenamiento hasta la sexta repetición del cuarto bloque de convolución, quedando un número final de parámetros a entrenar de 16.119.812.

4.2.3. CNN Propuesta

El último modelo que se propone analizar para la resolución del problema estudiado se basa en la arquitectura presentada en [Marin-Santos et al., 2022] que queda reflejada en la Figura 4.4. Esta red consiste en seis bloques formados cada uno por 3 redes convolucionales de *kernel* 3x3 y *stride* de 1 sobre las que se aplican *Batch Normalization* y funciones de activación *ReLU*, y una capa de *max pooling* 3x3 con *stride* de 2 para reducir la dimensionalidad; excepto en el último de los seis bloques donde la capa de *max pooling* se sustituye por una de *global average pooling*. La diferencia

entre bloques reside en el número de convoluciones por capa que va aumentando progresivamente desde 32 hasta 96.

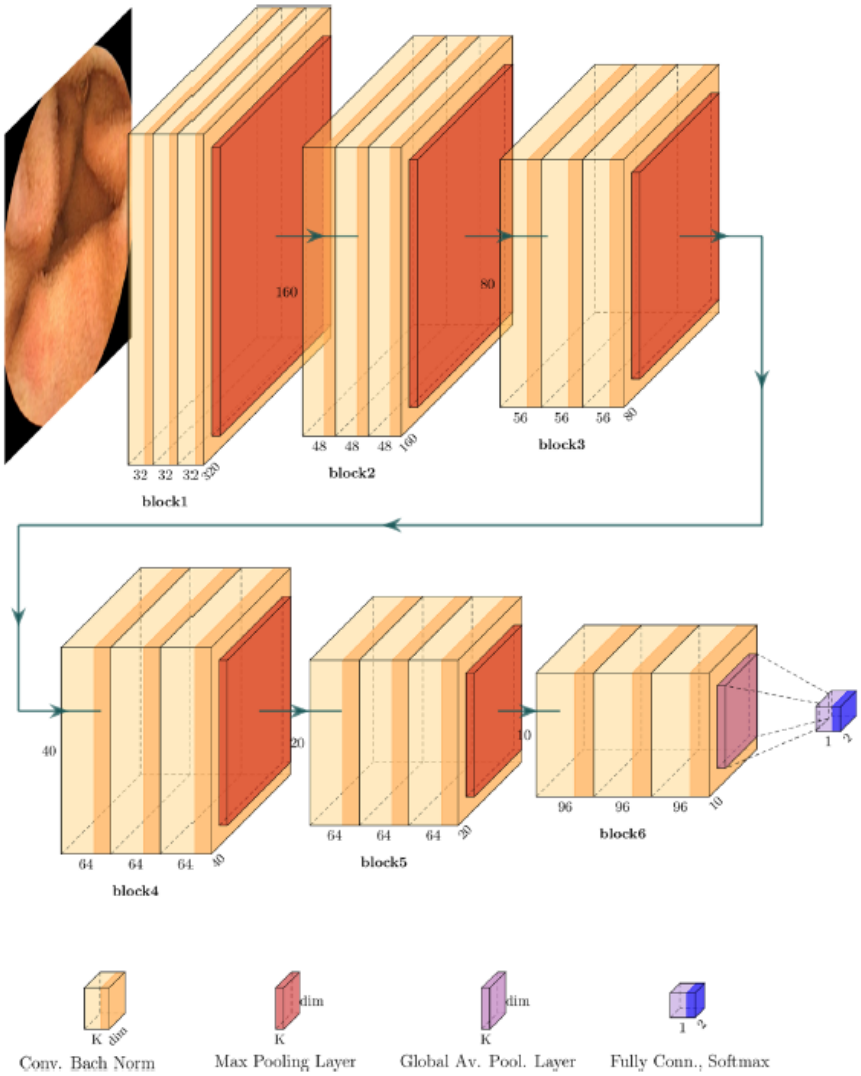

FIGURA 4.4: Detalle de la arquitectura propuesta en [Marin-Santos et al., 2022]

Esta arquitectura presentada se creará de forma simplificada para su adaptación, ya que, a diferencia del problema original de detección de enfermedades, el problema concreto planteado es de carácter mucho más simple por lo que tiene sentido reducir la complejidad de la red. Es por esto que manteniendo la idea de los seis bloques, se reducirá en cada uno de ellos el número de capas convolucionales a solo una manteniendo el resto de parámetros (*kernel* 3x3, *stride* de 1), eliminando también el uso de *Batch Normalization* a la salida de ellas. Las capas de *pooling* se mantienen idénticas y la capa densa final se modifica para que tenga cuatro clases de salida. El número de convoluciones también se mantiene en cada bloque, tomando progresivamente los valores: 36, 48, 56, 64, 64 y 96.

Aplicando todas estás simplificaciones para una entrada que se deja también constante respecto a la arquitectura de referencia con unas dimensiones de 320x320

y 3 canales RGB de color, se pasa de una red con 596.620 parámetros entrenables a otra de 164.764. Quedará como objetivo determinar si con estas simplificaciones la red es suficientemente expresiva para resolver el problema de clasificación con un nivel de precisión adecuado.

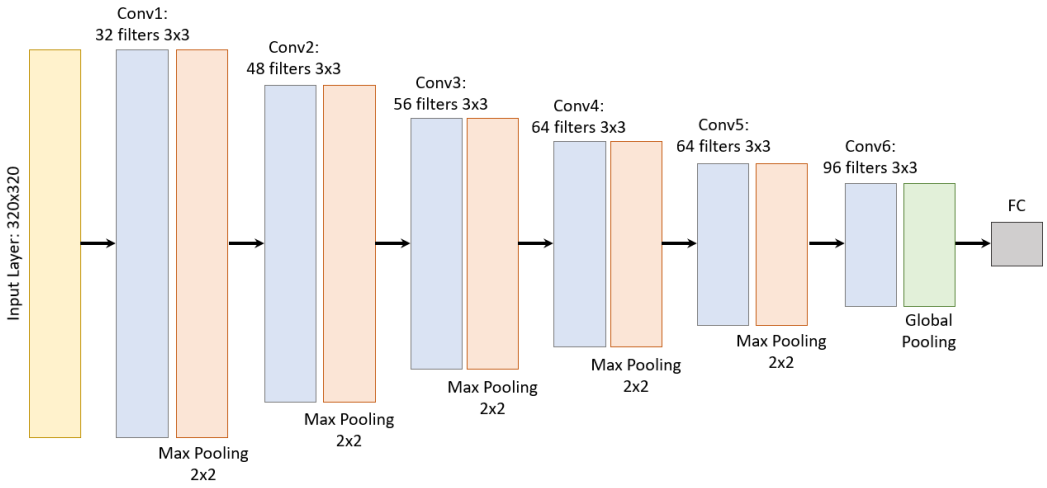

FIGURA 4.5: Representación de la arquitectura propuesta

4.3. Entrenamiento de modelos

En esta sección se resume todo lo relativo al entrenamiento de los tres modelos propuestos desde su configuración y ajuste de hiperparámetros, hasta la presentación de los resultados obtenidos y la valoración de los mismos.

4.3.1. Configuración del entrenamiento

En primer lugar, como ya se detalló en la Sección 3.2, se emplearán dos de los subconjuntos de la base de datos reservados inicialmente: el conjunto de entrenamiento y el conjunto de validación. Estos conjuntos serán adaptados según la capa de entrada necesaria en cada modelo, modificando su dimensión a la adecuada en cada caso (224x224x3 ó 320x320x3), y sus píxeles serán normalizados a la entrada para acelerar la convergencia del modelo. Estos datos para entrenamiento serán procesados en lotes de 32 imágenes. Además, para evitar sobreajuste sobre los datos de entrenamiento se realizan operaciones de aumento de datos sobre este subconjunto al inicio de cada época para que en cada fase del aprendizaje los datos tratados sean distintos y la red no aprenda unas imágenes concretas. Las operaciones propuestas tienen el fin de que las imágenes resultantes no queden en una posición antinatural para lo que pudiera ser una aplicación real sobre estos modelos y son combinaciones de las siguientes:

- Rotaciones aleatorias de la imagen de hasta 20°.
- Recortes aleatorios sobre los márgenes inferior, superior o laterales de las imágenes de hasta un 15 %.

- Acercamientos aleatorios de hasta un 10 % sobre la imagen.

Las decisiones fundamentales para definir la configuración del entrenamiento vienen dadas por la elección de la función de pérdida y del optimizador. La función de pérdida elegida es *Categorical Cross-Entropy*, comúnmente utilizada para clasificación multiclase, y que mide la desviación de la salida de la función *soft-max* de la última capa sobre la etiqueta real en formato *one-hot*. Como optimizador se escoge el algoritmo *Adam* [Kingma y Ba, 2014], que es un método estocástico de gradiente descendente basado en la estimación adaptativa de momentos de primer y segundo orden, al que se le asignan como parámetros: *learning-rate*=0.001, β_1=0.9 y β_2=0.99. La combinación de función de pérdida y optimizador permiten que se ejecute el algoritmo de *Backpropagation* que va ajustando los pesos de la red en cada fase del proceso de aprendizaje con el objetivo de que se minimice la función de pérdida.

Otra decisión importante es la elección del número de épocas de entrenamiento para asegurar que se llega a un punto en el que el ajuste de parámetros de la red es lo suficientemente bueno para clasificar imágenes de forma precisa. Debido a que el entrenamiento se va a realizar a través de las GPUs facilitadas por `Google Compute Engine` que presentan limitaciones de uso en su versión gratuita, no se fijará un número suficientemente alto de épocas de entrenamiento como premisa, si no que se irá decidiendo sobre la marcha según los resultados del entrenamiento. De esta forma lo que si se fija es que el número mínimo de épocas entrenadas será de 50, que el máximo será 150, y que se detendrá el entrenamiento en el caso de que se obtenga un máximo en el resultado de la validación sobre una época y éste no se supere en las 10 épocas consecuentes (siempre y cuando se haya alcanzado el número mínimo de épocas definido).

En este sentido, la métrica seleccionada para validar el resultado del entrenamiento sobre el subconjunto de datos reservado para este fin tras cada época de entrenamiento es el índice de *accuracy* o precisión, que se puede definir como el número de aciertos en la clasificación entre el número de predicciones totales realizadas, sin distinguir a que clase pertenece. La elección de esta métrica de carácter tan general tiene sentido ya que estamos considerando un conjunto de datos aproximadamente balanceado y un problema en el que se están considerando igual de importantes los aciertos/errores en la clasificación sobre cualquiera de las categorías.

4.3.2. Resultados del entrenamiento

De acorde a la decisión tomada respecto al número de épocas de entrenamiento en función de los resultados, se llega a un número distinto para cada modelo analizado. De entre dichas épocas de entrenamiento se guardarán los pesos asociados para cada modelo a la época con el mejor resultado sobre la métrica de validación considerada. El número de entrenamientos de cada modelo y el número de la época para la que se obtiene la máxima precisión en la clasificación sobre las imágenes del conjunto de validación se resumen en el Cuadro 4.1, junto con el valor máximo obtenido de dicha métrica.

En las Figuras 4.6, 4.7 y 4.8 se puede ver la evolución de los valores de la función de pérdida (a la izquierda) y de la precisión (a la derecha) de tres modelos de redes neuronales presentados a lo largo de las diferentes épocas para las que ha sido entrenado cada modelo. Puede observarse que, en general, los valores de la función de

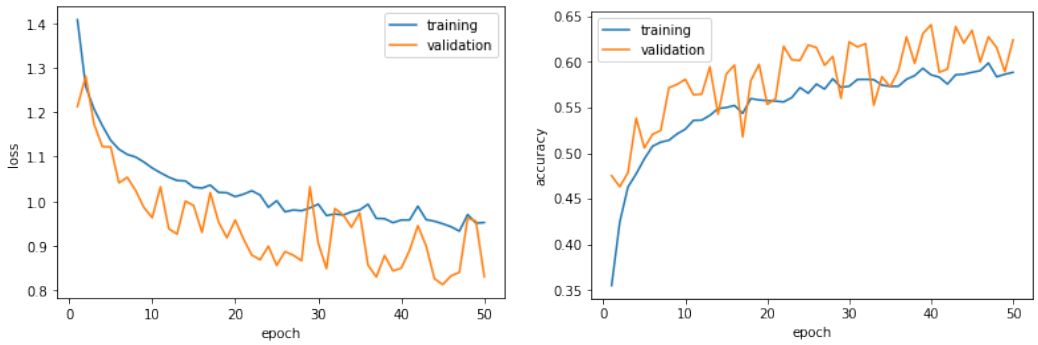

FIGURA 4.6: Evolución de los valores de la función de pérdida y *accuracy* durante el entrenamiento del modelo basado en VGG-16

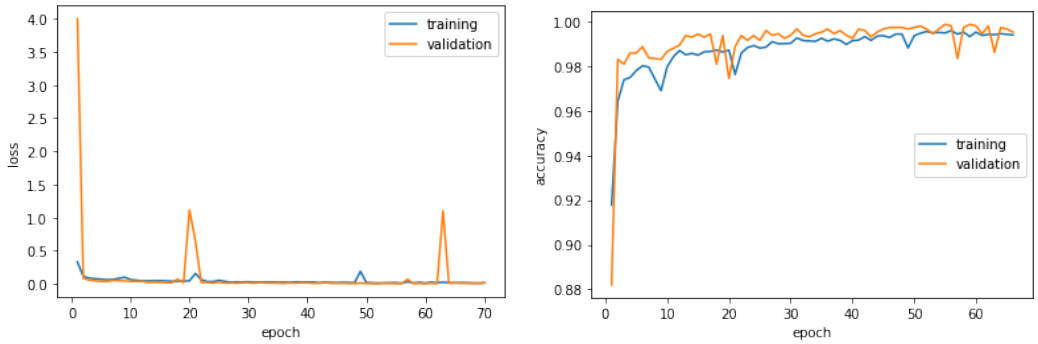

FIGURA 4.7: Evolución de los valores de la función de pérdida y *accuracy* durante el entrenamiento del modelo basado en Resnet-50

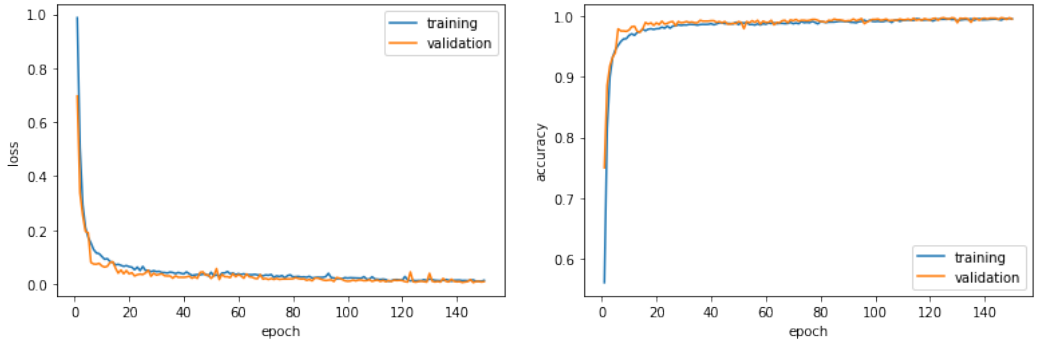

FIGURA 4.8: Evolución de los valores de la función de pérdida y *accuracy* durante el entrenamiento del modelo CNN propuesto

	Nº épocas entrenadas	**Época del mejor resultado**	*Accuracy*
VGG-16	50	40	64,07 %
Resnet-50	65	55	99,86 %
CNN Propuesta	150	146	99,79 %

CUADRO 4.1: Número de épocas entrenadas y mejor resultado para cada modelo sobre el conjunto de validación

pérdida sobre el conjunto de validación en los tres casos quedan por debajo de sus resultados sobre el conjunto de entrenamiento para cada época, aunque se aprecian algunas excepciones que se corrigen de forma inmediata en la siguiente época de entrenamiento. De forma análoga, las precisiones en validación son habitualmente mayores que sobre el conjunto de entrenamiento.

Este hecho es indicador de que los sistemas no están sobreajustándose a los datos de entrenamiento y que son capaces de clasificar imágenes nuevas del conjunto de validación de una forma más acertada a las que utilizan para ajustar sus pesos. Esto tiene sentido, ya que las imágenes de conjunto de entrenamiento tienen aplicadas transformaciones que pueden desvirtuar en algunos casos las imágenes, mientras que las de validación son más uniformes y es lógico que el modelo pueda predecir la clase a la que pertenecen con mayor precisión. En algunos casos vemos picos sobre las métricas de validación, en estos casos si puede decirse que en la época concreta los parámetros de la red se han sobreajustado a los datos de entrenamiento, pero son casos puntuales que quedan corregidos enseguida en el siguiente reajuste paramétrico realizado.

Sobre los resultados en el conjunto de validación se puede decir que el modelo basado en VGG-16 se ha estabilizado en valores lejanos a la precisión deseada (precisión máxima de un 64 %), mientras que los otros dos modelos sí que llegan a estabilizarse en valores mucho más altos (precisión cercana al 100 % en el mejor de los modelos), aunque el modelo de CNN propuesto necesita un número de épocas de entrenamiento mucho más alto que el ResNet-50 para alcanzar este resultado (también lógico porque este último no parte de ningún ajuste de parámetros en un entrenamiento previo). A este respecto se debe comentar que en el entrenamiento de la arquitectura VGG-16 se está limitando el entrenamiento a únicamente la última capa densa del sistema. Esta decisión ha estado motivada por las limitaciones de cómputo de los recursos gratuitos ofrecidos por `Google Cloud` y, sin duda, está afectado al rendimiento de la red.

Los pesos de los modelos para aquellas épocas en las que se alcanza la precisión máxima en validación, en cada caso, son guardados y serán utilizados para realizar pruebas sobre el tercer de los conjuntos reservados de la base de datos. Los resultados obtenidos sobre este conjunto de prueba se presentan en el apartado siguiente y servirán para obtener las conclusiones finales sobre el experimento.

Capítulo 5

Resultados

En este capítulo se comienzan introduciendo las métricas a considerar en la evaluación de los diferentes modelos, para posteriormente presentar los resultados de estas métricas sobre el conjunto de datos reservado para pruebas. Finalmente, se concluirá con el análisis de dichos resultados para tomar conclusiones sobre la elección del modelo que mejor actúa sobre el problema representado.

5.1. Métricas de evaluación

Con el fin de analizar mejor el comportamiento de los modelos sobre el conjunto de prueba y que sirva de ayuda para la elección del modelo que mejor resuelva el problema se hará un análisis más exhaustivo del resultado atendiendo a las distintas métricas que pueden obtenerse a partir de la matriz de confusión de cada clase. Para poder comprender lo que representan dichas métricas es importante introducir los siguientes conceptos:

- **Verdadero Positivo** (**TP**, por sus siglas en inglés), es el número de imágenes en las que la clase predicha en la clasificación se corresponde con la clase real de la imagen.
- **Verdadero Negativo** (**TN**), se refiere al número de imágenes que no son clasificadas dentro de una clase y que, efectivamente, no pertenecen a dicha clase.
- **Falso Positivo** (**FP**), hace correspondencia al número de imágenes que son clasificadas como pertenecientes a una clase siendo su clase real una distinta.
- **Falso Negativo** (**FN**), es el número de imágenes clasificadas como no pertenecientes a una clase diferente a la analizada y que, sin embargo, sí que se corresponden con la clase en cuestión.

A partir de estos cuatro conceptos básicos de un problema de clasificación se definen para cada categoría del sistema planteado las siguientes métricas de evaluación:

- **Sensibilidad**, se puede definir como el número de imágenes que se clasifican correctamente dentro de una clase entre el total de imágenes del conjunto de datos pertenecientes a dicha categoría.

$$Sensibilidad = \frac{TP}{TP + FN} \tag{5.1}$$

- **Especificidad**, que hace referencia al número de imágenes que se clasifican correctamente como no pertenecientes a la clase analizada entre el total de imágenes del conjunto de datos que no pertenecen a esa clase.

$$Especificidad = \frac{TN}{TN + FP} \tag{5.2}$$

- **Exactitud**, que se corresponde con el número de imágenes clasificadas correctamente dentro de una clase entre el número total de predicciones realizadas sobre esa clase

$$Exactitud = \frac{TP}{TP + FP} \tag{5.3}$$

Por último, a parte de estas métricas específicas sobre cada clase, se analizará como resumen del rendimiento de cada modelo el *accuracy* global, como ya se había hecho sobre el conjunto de validación para seleccionar el mejor ajuste de pesos de cada modelo. Esta métrica puede entenderse como la media ponderada sobre el número de imágenes de cada clase de la sensibilidad o simplemente como el cociente entre el número total de predicciones correctas y el número total de imágenes del conjunto de datos.

5.2. Resultados

Se presentan a continuación los resultados de evaluar los modelos de redes presentados en la Sección 4.2 atendiendo a las métricas de evaluación que se acaban de describir en el apartado anterior. Para ello, se representará también la matriz de confusión, que muestra la distribución entre las clases predichas y las clases reales de cada una de las imágenes del *dataset* de prueba clasificadas y, que sirve de apoyo para el cálculo de dichas métricas.

VGG-16

Agrupando los resultados de la matriz de confusión (Figura 5.1) puede calcularse el número de TP, TN, FP y FN para cada una de las categorías y de esta forma evaluar las métricas propuestas. Para el caso de la red basada en VGG-16 quedan representadas en el Cuadro 5.1, y resultan en una precisión global del **64,58** % sobre el conjunto de datos de test.

Atendiendo a los valores obtenidos de estas métricas se puede ver que, tal y como se esperaba tras ver los valores máximos obtenidos sobre el conjunto de imágenes de validación, el resultado de esta arquitectura en la clasificación requerida por el problema no es lo suficientemente preciso.

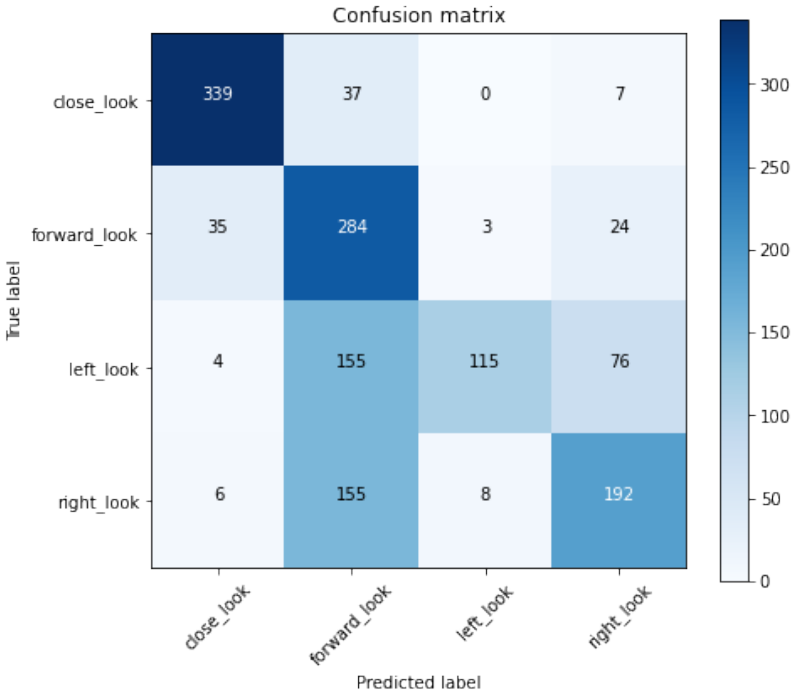

FIGURA 5.1: Matriz de confusión de la evaluación del modelo basado en la arquitectura VGG-16

	Especificidad	**Sensibilidad**	**Exactitud**
close_look	0,9574	0,8851	0,8828
forward_look	0,6828	0,8208	0,4501
left_look	0,9899	0,3286	0,9127
right_look	0,9008	0,5319	0,6421

CUADRO 5.1: Valores de las métricas de evaluación sobre cada clase del modelo basado en la arquitectura VGG-16

Este resultado que se ha estabilizado en el entrenamiento en valores de precisión bastante mejorables tiene que deberse a la decisión seguida de congelar todas las capas de la red neuronal excepto la última capa *Fully-Connected*. Esto puede suponer que la red no tenga la expresividad necesaria para adaptar sus pesos lo suficientemente bien como para caracterizar de una forma precisa las propiedades de las imágenes procesadas en el entrenamiento y, por tanto, no es capaz de clasificar nuevas imágenes con una precisión tan alta como se pretendería.

ResNet-50

En el caso de la red basada en la Resnet-50 se obtienen unos resultados prometedores quedando solamente clasificadas de forma errónea dos de las 1440 imágenes por las que está compuesto *dataset* de prueba, como queda plasmado en la Figura 5.2.

Se llega de esta forma a tener un *accuracy* global del **99,86** % en la evaluación de este modelo. El resto de las métricas quedan representadas en el Cuadro 5.2, donde también queda reflejado, a través de una sensibilidad del 100 %, que la totalidad de las imágenes procesadas que pertenecen categorías *close_look* y *forward_look* son

clasificadas con acierto.

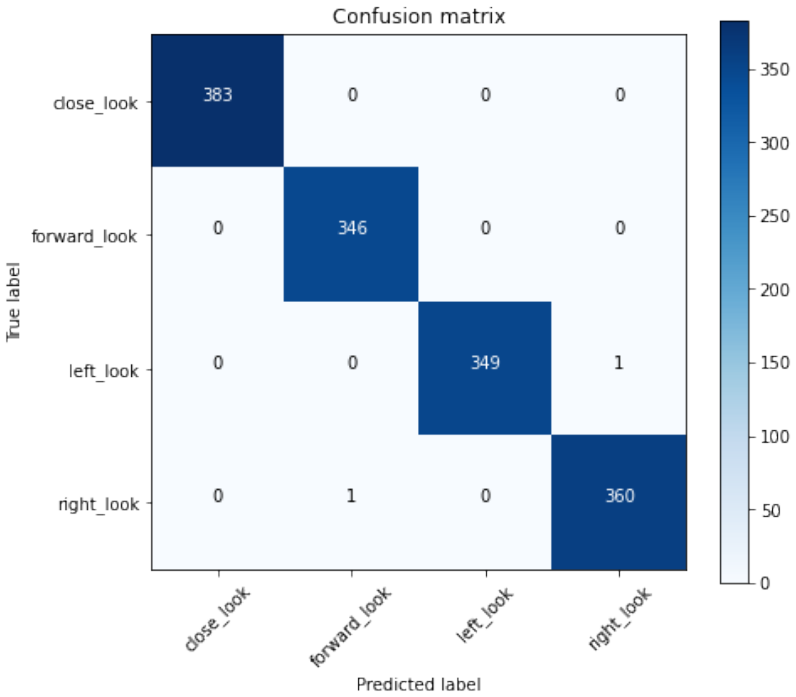

FIGURA 5.2: Matriz de confusión de la evaluación del modelo basado en la arquitectura ResNet-50

	Especificidad	**Sensibilidad**	**Exactitud**
close_look	1,0000	1,0000	1,0000
forward_look	0,9991	1,0000	0,9971
left_look	1,0000	0,9971	1,0000
right_look	0,9991	0,9972	0,9972

CUADRO 5.2: Valores de las métricas de evaluación sobre cada clase del modelo basado en la arquitectura ResNet-50

CNN Propuesta

Por último, en el caso de la CNN propuesta se obtiene de forma curiosa un resultado idéntico al evaluar la base de datos al que se obtenía con la arquitectura basada en ResNet-50. Hecho que queda representado en la Figura 5.3, que tiene una estructura idéntica a la que se obtenía para el modelo anterior. Por lo que de nuevo se obtiene una precisión global de **99,86** % en este caso.

Las métricas de evaluación representadas en el Cuadro 5.3 son por tanto exactamente iguales a las del modelo anterior. Lo que pone de manifiesto que para un conjunto de imágenes como el estudiado en este experimento una arquitectura sencilla, como la propuesta, puede predecir la clase de las imágenes con la misma precisión que una red más compleja.

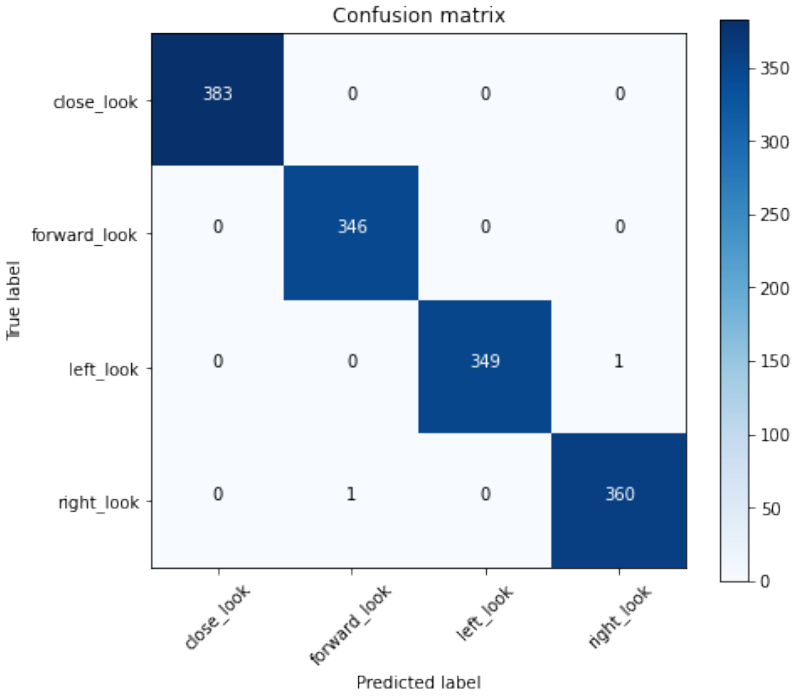

FIGURA 5.3: Matriz de confusión de la evaluación del modelo basado en la arquitectura CNN propuesta

	Especificidad	**Sensibilidad**	**Exactitud**
close_look	1,00000	1,0000	1,0000
forward_look	0,9991	1,0000	0,9971
left_look	1,0000	0,9971	1,0000
right_look	0,9991	0,9972	0,9972

CUADRO 5.3: Valores de las métricas de evaluación sobre cada clase del modelo basado en la arquitectura CNN propuesta

5.3. Análisis y discusión

Teniendo en cuenta los resultados presentados en el apartado anterior, puede realizarse un análisis comparativo entre los tres modelos estudiados. Para ello, se atiende primero como medida más representativa de las predicciones a la comparación entre la precisión global de los modelos. Los valores de esta métrica para cada red entrenada se resumen en el Cuadro 5.4.

	VGG-16	**Resnet-50**	**CNN Propuesta**
***Accuracy* global**	64,58 %	99,86 %	99,86 %

CUADRO 5.4: Comparativa de la precisión global de cada modelo sobre el conjunto de prueba

Atendiendo a los resultados obtenidos de esta métrica tiene sentido que se descarte como mejor solución el modelo basado en la arquitectura VGG-16 ya que sus resultados quedan muy alejados de las *accuracys* tan altas que se obtienen sobre el conjunto de prueba por los dos modelos restantes. Como ya se ha comentado, este mal resultado queda condicionado por un entrenamiento que se ha estabilizado con unos pesos no lo suficientemente buenos por la baja expresividad que se le ha

dejado a la red tomando como base ejemplos de problemas similares tratados en la literatura. Esto se manifiesta al comparar el número de parámetros entrenables que resulta en menos de 20.000, pertenecientes únicamente a la capa densa de salida del modelo, como se ve reflejado en la comparación con número de parámetros de los otros modelos (en el Cuadro 5.5), que son del orden de 10 veces mayor en la CNN propuesta, y de 100 veces en la arquitectura basada en Resnet-50.

	VGG-16	**Resnet-50**	**CNN Propuesta**
Parámetros totales	134.276.932	23.597.572	164.044
Parámetros entrenados	16.388	16.119.812	164.044

CUADRO 5.5: Comparativa del número de parámetros totales y entrenados sobre cada modelo estudiado

Se podría mejorar el resultado del modelo VGG-16 a través de la realización de un nuevo entrenamiento disminuyendo el número de capas congeladas, idealmente no dejando ninguna. Sin embargo, se tiene que tener en cuenta que el incremento del número de parámetros de entrenamiento también aumenta el tiempo de compilación y de entrenamiento del modelo llegándose a un punto inviable desde el punto de vista computacional para poder hacerlo gratuitamente con los servidores de `Google Cloud`, que tienen límites de tiempo de entreno y capacidad. Por lo tanto, este análisis no se llegará a implementar en el ámbito de este escrito y se propondrá como trabajo futuro. Por los motivos mencionados, la solución a través del modelo basado en la arquitectura VGG-16 se descarta.

Para tomar la mejor solución entre los dos modelos restantes se presenta también (en el Cuadro 5.6) la comparativa del tiempo medio que toma el modelo en la predicción de imágenes ya que este será un factor clave a la hora de poder darle una aplicación en tiempo real a los modelos. Como esto depende del tamaño del lote procesado, se tomará el tiempo medio que tarda en predecir de forma individual e independiente un grupo de 10 imágenes, ya que de dar una aplicación real a las redes entrenadas esta sería la forma en la que procesará los datos.

	VGG-16	**Resnet-50**	**CNN Propuesta**
Tiempo de inferencia medio	18,7856 ms	22,4430 ms	16,9781 ms

CUADRO 5.6: Comparativa del tiempo medio de inferencia por imagen sobre el conjunto de prueba

Como puede comprobarse el tiempo de predicción depende de complejidad (profundidad) de la red neuronal por lo que con el modelo propuesto se obtienen predicciones en un tiempo menor que para la arquitectura basada en Resnet-50. Prestando atención a este hecho, así como, al número de parámetros usados en el entrenamiento y al coste que esto supone, se puede concluir como una buena decisión tomar como solución final la arquitectura de CNN propuesta que, a pesar de su sencillez, presenta unos resultados de precisión sobre el conjunto de imágenes del experimento tan bueno como el de la arquitectura basada en Resnet-50.

Sin embargo, como el objetivo secundario de este trabajo es, partiendo de la experimentación sobre la base de datos elegida, el poder extender la solución a un

problema real de reconocimiento de imágenes que pueda clasificar en tiempo real imágenes tomadas por una cámara web; se decide realizar una segunda validación para poder analizar los resultados que presentan los dos modelos que han sido capaces de presentar buenos resultados sobre las imágenes del conjunto del experimento sobre imágenes a priori similares tomadas a partir de una cámara web sobre el autor de este trabajo.

Este nuevo *dataset* que se va a analizar no resulta tan representativo como el original, al disponer solamente 48 imágenes, divididas en 12 para cada una de las cuatro clases representadas en el estudio; pero servirá para dar una idea de cuál será la precisión de los modelos sobre un entorno real alejado del experimento inicial. Una muestra de este conjunto, en el que se alternan imágenes con y sin lentes, y en diferentes condiciones de iluminación, se representa en la Figura 5.4.

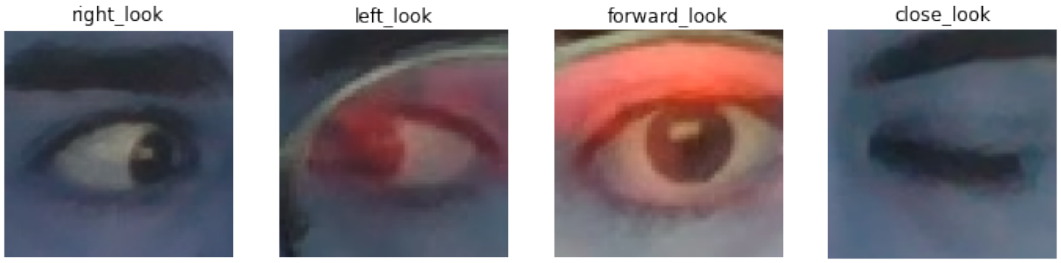

FIGURA 5.4: Muestra de una imagen por cada clase del nuevo conjunto de datos diseñado para la segunda validación

Se presentan, análogamente a como se hizo en la Sección 5.2, los resultados sobre este nuevo conjunto de imágenes de prueba diseñado para poder establecer unas comparaciones y conclusiones finales sobre la utilización de los modelos.

Resnet-50

Como queda demostrado, en la representación de la matriz de confusión para las predicciones sobre este segundo conjunto de imágenes de la Figura 5.5 y en las métricas que se derivan de ella en el Cuadro 5.7, se obtienen unos resultados que quedan bastante alejados de los obtenidos sobre las imágenes reservadas para pruebas del *dataset* original, reduciéndose la precisión global desde un 99,86 % hasta un **70,83** % en el nuevo conjunto de datos.

Esto puede deberse a que, aunque a priori las imágenes representen lo mismo y de una forma similar, se están realizando sobre un sujeto distinto y en unas condiciones distintas, por lo que las características de las imágenes pueden terminar difiriendo en una medida importante a las del conjunto de datos inicial del experimento. De esta forma, se origina un resultado que no es tan preciso y que podría mejorarse extendiendo el entrenamiento a imágenes como las ahora propuestas.

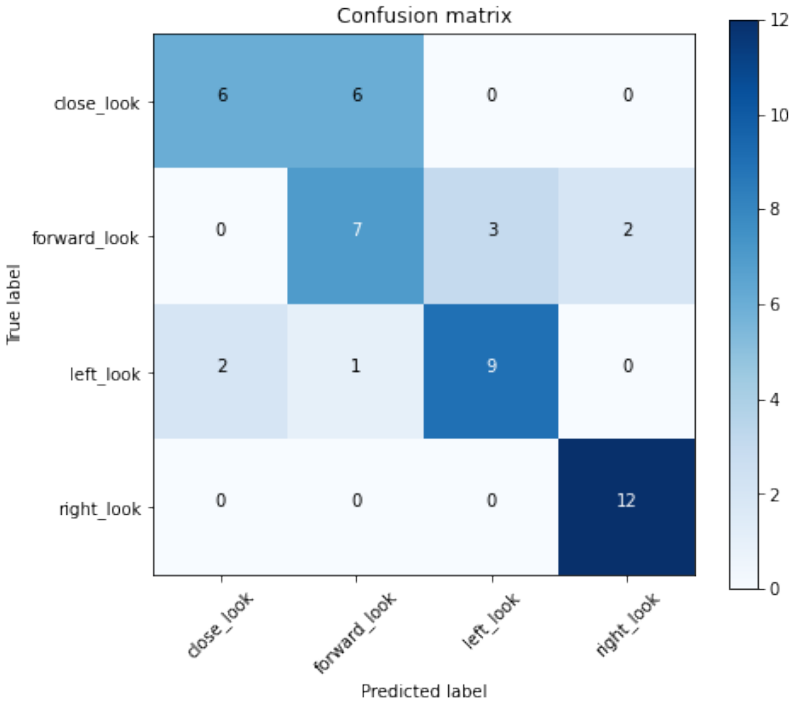

FIGURA 5.5: Matriz de confusión de la evaluación del modelo basado en la arquitectura ResNet-50 sobre el nuevo conjunto de imágenes

	Especificidad	**Sensibilidad**	**Exactitud**
close_look	0,9444	0,5000	0,7500
forward_look	0,8055	0,5833	0,5000
left_look	0,9167	0,7500	0,7500
right_look	0,9444	1,0000	0,8571

CUADRO 5.7: Valores de las métricas de evaluación sobre cada clase del modelo basado en la arquitectura ResNet-50 sobre el nuevo conjunto de imágenes

CNN Propuesta

En este caso se vuelve a poner de manifiesto, como se refleja en la correspondiente matriz de confusión de la Figura 5.6 y en sus métricas asociadas en el Cuadro 5.8, que la utilización de un conjunto de imágenes nuevo y con distintas características al de entrenamiento para hacer una segunda validación afecta muy negativamente a la precisión del modelo en las predicciones.

Para este modelo, dicha reducción de la precisión es mucho más grave que para el modelo basado en ResNet-50, pasándose desde un 99,86 % de *accuracy* sobre el conjunto de test original hasta tan solo un **41,67** % en el nuevo *dataset*. Lo que indica que se realizan más predicciones erróneas que correctas sobre estas nuevas imágenes, por lo que, a pesar de los buenos resultados sobre la base de datos inicial, este modelo no será adecuado para ser extendido a cualquier tipo de imágenes sobre las que no haya entrenado.

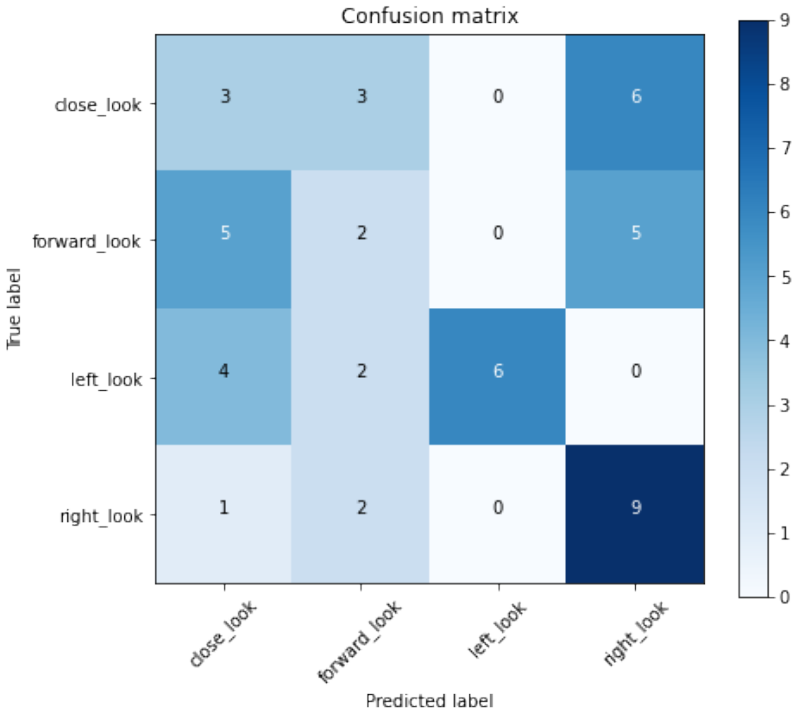

FIGURA 5.6: Matriz de confusión de la evaluación del modelo basado en la arquitectura CNN Propuesta sobre el nuevo conjunto de imágenes

	Especificidad	**Sensibilidad**	**Exactitud**
close_look	0,7222	0,2500	0,2308
forward_look	0,8056	0,1667	0,2222
left_look	1,0000	0,5000	1,0000
right_look	0,6944	0,7500	0,4500

CUADRO 5.8: Valores de las métricas de evaluación sobre cada clase del modelo basado en la arquitectura CNN Propuesta sobre el nuevo conjunto de imágenes

Capítulo 6

Conclusiones

6.1. Conclusiones y valoración

Este trabajo partía con el objetivo principal de llegar a una red neuronal que pudiera clasificar imágenes de ojos humanos en cuatro categorías (ojo cerrado, mirada al frente, mirada izquierda y mirada derecha) de forma consistente. Para ello, se ha utilizado un conjunto de 14.384 imágenes que ha sido dividido en tres subconjuntos: entrenamiento (80 %, 11505 imágenes), validación (10 %, 1439 imágenes) y test (10 %, 1440 imágenes); para proceder al entrenamiento, selección y evaluación del modelo, respectivamente. A este respecto se han implementado arquitecturas basadas en VGG-16, Resnet-50 y una arquitectura más reducida. En los casos de las redes del estado del arte VGG-16 y Resnet-50, los entrenamientos de los modelos han partido de pesos previamente ajustados para problemas de clasificación más complejos (siguiendo el concepto de *Transfer learning*).

Del mismo modo, se planteaba como objetivo secundario probar a extender el uso de la arquitectura que presente unos mejores resultados de precisión/rendimiento sobre el problema inicial a un conjunto de datos cualquiera de similares características para darle una aplicación de uso real al sistema creado y poder predecir en tiempo real imágenes nuevas generadas por una cámara web. Para ello se realizaría una segunda validación sobre un nuevo conjunto de imágenes más reducido diseñado con este fin.

De los resultados presentados en el Capítulo 5, obtenidos en la persecución de los objetivos citados, puede concluirse lo siguiente:

- En primer lugar, sobre los resultados de evaluar los mejores modelos obtenidos para cada una de las arquitecturas propuestas sobre el conjunto de prueba reservado, se tiene que el modelo VGG-16 no ha conseguido llegar a unos valores de precisión en la predicción suficientemente altos como para ser considerado como una buena solución del problema planteado. La explicación sobre este hecho es que el entrenamiento realizado ha tenido lugar aprovechando todo su conocimiento previo sobre ImageNet, congelando todas las capas de entrenamiento menos la de salida. Esto puede suponer una primera buena aproximación, pero para llegar a resultados altos es altamente recomendable entrenar la red desde un punto más cercano al inicial para no limitar tanto su expresividad y que pueda ajustar sus parámetros correctamente al conjunto de datos propuesto. Sin embargo, dado el gran número de parámetros que maneja esta arquitectura, realizar esto supone un problema computacional teniendo en cuenta los recursos utilizados y, por tanto, el reentrenamiento se ha cancelado en el alcance de este proyecto.

- Por otro lado, fruto de este mismo experimento sobre el conjunto de prueba inicial se tiene que tanto el modelo basado en la arquitectura ResNet-50, como el propuesto, han logrado resultados excelentes, llegando a alcanzar una precisión del 99,86 % ambos. Si se comparan estos dos modelos podría concluirse que tanto por su sencillez (número de parámetros entrenables mucho menor), como por el tiempo de inferencia medio que necesita la red para procesar y predecir una nueva imagen, el modelo de CNN propuesto puede representar la mejor solución al problema dado, aunque haya necesitado de más épocas de entrenamiento para estabilizarse, lo que es lógico ya que en este caso no se usa *Transfer Learning* y los parámetros tienen que ser ajustados desde cero, ya que el modelo no dispone de referencias previas sobre ningún otro *dataset*. De esta forma quedaría también demostrado que para un conjunto de datos de las características del estudiado, una arquitectura más sencilla podría dar resultados tan buenos como una de gran profundidad y complejidad.

- En un segundo análisis de resultados sobre un pequeño conjunto de imágenes de prueba distinto al original y diseñado a través de imágenes tomadas por *webcam*, en un intento de poder extender el conocimiento de las redes entrenadas sobre cualquier tipo de imágenes similares, se concluye que ninguno de los dos modelos anteriormente aceptados predicen con valores de precisión cercanos a los obtenidos sobre el *dataset* original. La precisión obtenida sobre este conjunto de datos es de un 71 % en el caso del modelo basado en ResNet-50, y de tan solo un 42 % para la arquitectura propuesta. Estos resultados nos indican que aunque las imágenes nuevas parezcan similares a las del *dataset* usado originalmente para el experimento, su características pueden diferir en mayor medida de lo esperado, y que por tanto, para obtener unos resultados más precisos se necesitaría obtener un nuevo conjunto de datos lo suficientemente grande para extender el entrenamiento y realizar el ajuste sobre las imágenes obtenidas por medio de una cámara web concreta para poder usar los modelos de forma óptima en la aplicación a un problema real de *Eye-Tracking*.

- Por último, se tiene de los resultados del punto anterior que, si en cualquier caso se quisiera utilizar el entrenamiento realizado para extenderlo a una aplicación del mundo real, a pesar de las advertencias de tener que reentrenar las redes para un caso de uso concreto para obtener resultados óptimos, el mejor modelo a utilizar será el basado en la arquitectura ResNet-50. Este modelo parece entender mejor el concepto de las imágenes y aunque tenga cierto sobreajuste al conjunto de imágenes inicial del experimento, tiene una capacidad de clasificación moderada sobre nuevas imágenes distintas aplicadas al mundo real. Como muestra de ello, se representa como cierre de este trabajo una pequeña muestra de lo que sería una aplicación de la tecnología desarrollada.

6.2. Muestra de una aplicación real

Para mostrar la extensión de la mejor red obtenida (arquitectura basada en ResNet-50) a unos datos distintos al conjunto inicial que puedan tomarse directamente en tiempo real se toma como ayuda una librería de visión por computadora de `Python` llamada `OpenCV` [Bradski, 2000] [Pulli et al., 2012] que permite la detección de objetos en tiempo real [Lienhart y Maydt, 2002], entre los cuales encontramos un apartado especializado en cara y ojos [Kasinski y Schmidt, 2010]. El aprovechamiento de estos

desarrollos permite implementar una aplicación básica con pocas líneas de código.

Empleando el apartado de detección de ojos de la librería y realizando una serie de modificaciones sobre el código para capturar como una imagen el contenido de cada detección durante un vídeo en *streaming*, y transformando el tamaño de estas capturas para que tengan las mismas dimensiones de entrada que la arquitectura de red entrenada, se logra pasarle en tiempo real imágenes del estado de ambos ojos y que el modelo sea capaz de predecir este estado. Finalmente, sobre cada ojo se pinta por pantalla la resolución de esta predicción junto con un recuadro indicador del contenido del vídeo que se está capturando en cada momento.

Aprovechando estas transformaciones se han alternado las etiquetas de mirada a la izquierda y a la derecha para de esta forma cambiar el sistema de referencia y que la predicción sea realmente el estado de la mirada del usuario sobre el que se realiza la experimentación. Se representa en las siguientes imágenes resultados de las cuatro clasificaciones sobre el autor del trabajo. El código utilizado para conseguir dichos resultados se puede consultar en el Apéndice A.

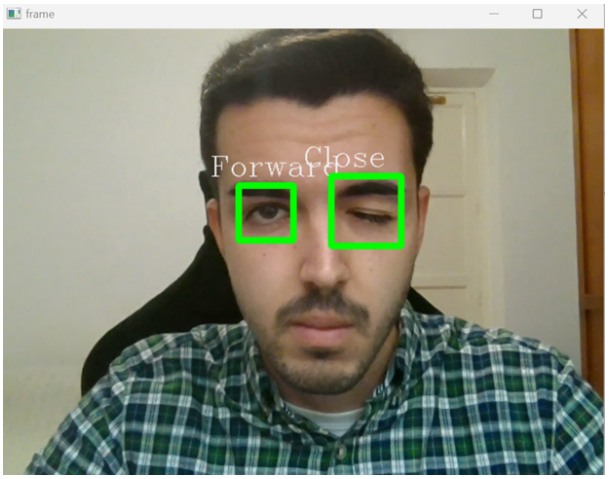

FIGURA 6.1: Ejemplo de uso de la aplicación planteada en la predicción del ojo derecho mirando al frente y el izquierdo cerrado

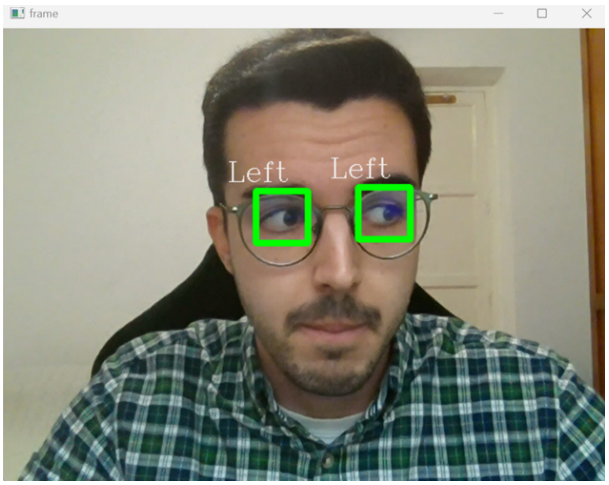

FIGURA 6.2: Ejemplo de uso de la aplicación planteada en la predicción de ambos ojos mirando hacia la izquierda

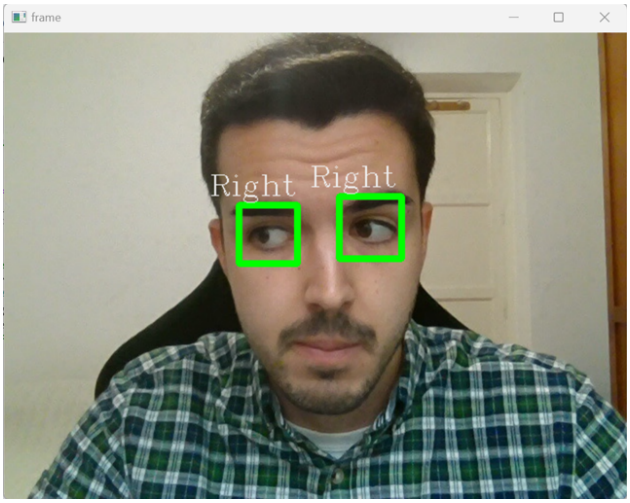

FIGURA 6.3: Ejemplo de uso de la aplicación planteada en la predicción de ambos ojos mirando hacia la derecha

6.3. Trabajos futuros

A continuación, se detallan una serie de puntos que sería interesante investigar en una hipotética ampliación de la experimentación realizada en este trabajo:

- En primer lugar, sería interesante conseguir los recursos computacionales necesarios para poder entrenar el modelo VGG-16 desde un punto más cercano al inicial para comprobar si su rendimiento con un reentrenamiento en el que no se limite tanto la expresividad de la red pudiera derivar en resultados tan buenos como los presentados por los otros dos modelos propuestos sobre el conjunto de datos inicial.
- En segundo lugar, también considerando la disponibilidad de mayores recursos, sería interesante poder alargar los entrenamientos durante un número mayor de épocas para comprobar si sus resultados mejoran los presentados que han estado basados en el momento en el que se estabilizan las métricas de validación durante el entrenamiento por la limitación que se ha tenido en el entrenamiento por su implementación en las GPUs de `Google Cloud` que podrían verse subsanadas con la compra de un entorno PRO. Además, esto serviría para poder establecer comparaciones más sólidas basadas en un número de épocas fijas para todos los modelos.
- Como siguiente paso, de cara a la extensión del problema a un entorno real sería de interés poder disponer de un conjunto de imágenes mucho más grande o mucho más diverso para que el entrenamiento sobre este pueda aplicarse a cualquier problema similar, sin verse condicionado por las características concretas de un conjunto de imágenes.
- Ante la posible dificultad de lo propuesto en el punto anterior y en el sentido de aplicar la tecnología a un entorno real, crear un conjunto de datos formado por imágenes tomadas desde un mismo dispositivo sobre un grupo de distintas personas que sea lo suficientemente grande y representativo para entrenar, validar y probar los modelos. De esta forma podría comprobarse si ajustando los modelos a unas imágenes de un entorno determinado podrían obtenerse

resultados de predicción tan buenos como los obtenidos sobre el *dataset* del experimento inicial y de esa forma poder demostrar si pudiera conseguirse una aplicación con resultados óptimos sobre este mismo dispositivo u otros con idénticas características.

- Por último, y en función de los resultados de la experimentación del punto previo, si no se consiguen resultados los suficientemente buenos como para poder plantear una aplicación con carácter general, ni siquiera sobre un mismo dispositivo, podría simplificarse aún más el problema y probarse para un dispositivo concreto y un usuario sobre el cual predecir concreto. Si los resultados en este caso fueran positivos se podría establecer una aplicación que primero tomara algunas imágenes sobre el usuario y en el dispositivo concretos, que realice un pequeño entrenamiento sobre estas, partiendo de los pesos previamente ajustados en el experimento de carácter más general, y que pueda utilizarse con una precisión óptima sobre casos acotados de esta forma.

Apéndice A

Código de la aplicación desarrollada

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

import os
data_folder = 'C:/Users/anton/Desktop/TFM/Trabajo/dataset/Eye dataset'
classes = os.listdir(data_folder)
classes.sort()
classes = ['Close', 'Forward', 'Right', 'Left']

model = tf.keras.models.load_model('model_resnet_v2_dropout.h5')
model.summary()

import cv2

cap = cv2.VideoCapture(0)
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + '
    haarcascade_eye.xml')
font = cv2.FONT_HERSHEY_COMPLEX

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    eyes = eye_cascade.detectMultiScale(gray, 1.3, 5)
    for (ex, ey, ew, eh) in eyes:
        cv2.rectangle(frame, (ex, ey), (ex + ew, ey + eh), (0, 255, 0),
     5)
        eye = frame[ey:ey+eh, ex:ex+ew, :]
        eye = cv2.resize(eye, (224,224), interpolation = cv2.INTER_AREA
    )
        eye = eye.reshape(1,224,224,3)
        eye = eye.astype('float32') / 255
        pred = model.predict(eye)
        n = classes[np.argmax(pred)]
        cv2.putText(frame, n , (ex-30,ey-10), font, 1, (240,240,240))
        print(n)

    cv2.imshow('frame', frame)

    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Bibliografía

Ansari, Mohd Faizan, Pawel Kasprowski y Marcin Obetkal (2021). «Gaze Tracking Using an Unmodified Web Camera and Convolutional Neural Network». En: *Applied Sciences* 11.19, pág. 9068.

Bradski, Gary (2000). «The openCV library.» En: *Dr. Dobb's Journal: Software Tools for the Professional Programmer* 25.11, págs. 120-123.

Chinsatit, Warapon y Takeshi Saitoh (2017). «CNN-based pupil center detection for wearable gaze estimation system». En: *Applied Computational Intelligence and Soft Computing* 2017.

Dahmani, Mahmoud et al. (2020). «An intelligent and low-cost eye-tracking system for motorized wheelchair control». En: *Sensors* 20.14, pág. 3936.

Deng, Jia et al. (2009). «Imagenet: A large-scale hierarchical image database». En: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, págs. 248-255.

He, Kaiming et al. (2016). «Deep residual learning for image recognition». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*, págs. 770-778.

Huang, Gao et al. (2017). «Densely connected convolutional networks». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*, págs. 4700-4708.

Kasinski, Andrzej y Adam Schmidt (2010). «The architecture and performance of the face and eyes detection system based on the Haar cascade classifiers». En: *Pattern Analysis and Applications* 13, págs. 197-211.

Kingma, Diederik P y Jimmy Ba (2014). «Adam: A method for stochastic optimization». En: *arXiv preprint arXiv:1412.6980*.

Krizhevsky, Alex, Ilya Sutskever y Geoffrey E. Hinton (2017). «ImageNet Classification with Deep Convolutional Neural Networks». En: *Commun. ACM* 60.6, 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

Lienhart, Rainer y Jochen Maydt (2002). «An extended set of haar-like features for rapid object detection». En: *Proceedings. international conference on image processing*. Vol. 1. IEEE, págs. I-I.

Lin, Min, Qiang Chen y Shuicheng Yan (2013). «Network in network». En: *arXiv preprint arXiv:1312.4400*.

Marin-Santos, Diego et al. (2022). «Automatic detection of crohn disease in wireless capsule endoscopic images using a deep convolutional neural network». En: *Applied Intelligence*, págs. 1-15.

Matsuyama, Eri (ene. de 2020). «A Deep Learning Interpretable Model for Novel Coronavirus Disease (COVID-19) Screening with Chest CT Images». En: *Journal of Biomedical Science and Engineering* 13, págs. 140-152. DOI: 10.4236/jbise.2020.137014.

Pulli, Kari et al. (2012). «Real-time computer vision with OpenCV». En: *Communications of the ACM* 55.6, págs. 61-69.

Russakovsky, Olga et al. (2015). «Imagenet large scale visual recognition challenge». En: *International journal of computer vision* 115.3, págs. 211-252.

Shah, Kayvan (2020). *Eye-dataset*. DOI: 10.34740/KAGGLE/DSV/1093317. URL: https://www.kaggle.com/dsv/1093317.

Simonyan, Karen y Andrew Zisserman (2014). «Very deep convolutional networks for large-scale image recognition». En: *arXiv preprint arXiv:1409.1556*.

Sugata, T y C Yang (nov. de 2017). «Leaf App: Leaf recognition with deep convolutional neural networks». En: *IOP Conference Series: Materials Science and Engineering* 273, pág. 012004. DOI: 10.1088/1757-899X/273/1/012004.

Szegedy, Christian et al. (2015). «Going deeper with convolutions». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*, págs. 1-9.

Takeyas, Bruno López (2007). «Introducción a la inteligencia artificial». En: *Instituto Tecnológico de Nuevo Laredo. Web del autor: http://www. itnuevolaredo. edu. mx/takeyas*.