UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

título del TFG

Presentado por Nombre del alumno
en Universidad de Burgos — 12 de junio de 2023
Tutor: nombre tutor

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Nombre del alumno, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 12 de junio de 2023

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

La retinopatía diabética es una de las mayores causas de ceguera en los países desarrollados. Para diagnosticar el grado de la anomalía, se realiza una imagen de la retina del paciente, captada por un retinógrafo. Esta prueba tiene un coste elevado para la sanidad pública, e implica a su vez que el paciente se desplace a un centro médico especializado.

Por ello, se han desarrollado unos dispositivos, para las cámaras de los móviles, que permiten obtener las fotografías de las retinas; de esta forma, el paciente no tendría que desplazarse a un hospital que disponga de un retinógrafo y el médico de familia le realizaría la foto al paciente.

En este trabajo, se propone facilitar la labor del médico, haciendo una aplicación Android a través de la cual se realizaría un primer diagnostico del paciente a partir de una imagen que se enviará esta foto a modelos de aprendizaje computacional existentes; haciendo que los pacientes que tengan algún grado de retinopatía diabética, se realice un estudio exhaustivo.

Como resultado, se ha creado la aplicación RetinAI.

Descriptores

Salud, retinopatía diabética, redes neuronales, aplicación Android.

Abstract

Diabetic retinopathy is a leading cause of blindness in developed countries. To determine the degree of abnormality, the retinographer takes a image of patient's retina, but this procedure is costly for spanish public health system, also requires the patient to travel to a medical center.

To address this, there are some lenses which have been developed for mobile phone cameras to take retina photos. This way, the patient would not have to travel to a hospital that has a retinographer, and the family doctor would take the patient's photo.

This work proposes to make the doctor's labor easier by creating an Android app that would make a preliminary diagnosis of the patient based on the photo taken, this photo will be sent to existing computational learning models. This will allow patients with any degree of diabetic retinopathy to undergo a comprehensive study.

As a result, the app RetinAI has been created.

Keywords

Health, diabetic retinopathy, neural networks, Android app.

Índice general

Índice de figuras

Índice de tablas

Introducción

Durante los últimos años, se ha aumentado exponencialmente el uso de los dispositivos móviles, este aumento se debe principalmente a la comodidad que proporciona respecto a otras tecnologías parecidas; siendo principalmente útiles para el uso de las redes sociales.

Este aumento, tiene como consecuencia el fomento del desarrollo de aplicaciones móviles, puesto que sin aplicaciones para estos dispositivos, no hubiesen tenido tanto auge. Estas aplicaciones están destinadas a dos sistemas operativos principalmente, que son Android e iOS; donde el 81,3 % de los usuarios prefieren Android y el 13.4 % iOS [1].

Los móviles, no solo se han desarrollado en software, sino en hardware también, donde se han aumentado las características de procesador, cámara, RAM,... Y con ello, los desarrolladores buscan crear aplicación que hagan uso de este hardware permitiendo muchas más funcionalidades; un uso que esta aumentando, es la integración de las inteligencias artificiales dentro de las propias aplicaciones.

Por otro lado, la salud y bienestar han sido temas importantes para el ser humano; y por lo tanto, también se han hecho aplicaciones móviles con este objetivo.

En el caso de la retinopatía diabética, se ha desarrollado una lente que permite obtener fotos desde el dispositivo móvil de la retina. Lo que ha llevado a aplicaciones móviles dedicadas a la retinopatía diabética, que permiten guardar imágenes de los pacientes, obteniendo un historial para realizar un estudio de la evolución de la patología en el paciente, como es el caso de ReTinCam.

La discapacidad visual afecta a más de 2200 millones de personas en el mundo, siendo las principales causas de pérdida de visión: degeneración

macular relacionada con la edad, cataratas, retinopatía diabética, glaucoma y errores de refracción no corregidos. Entre las personas con discapacidad visual, hay al menos 1000 millones de personas que tienen un deterioro moderado o grave de la visión[3].

En Estados Unidos, cada año la retinopatía diabética suma un 12 % de nuevos casos de ceguera [10].

La retinopatía diabética [32] es una complicación de la diabetes, que afecta al sistema ocular del ser humano. Es causada por la inflamación, escape o cierre de los vasos sanguíneos de la retina, pudiendo desarrollarse nuevos vasos sanguíneos a lo largo del tiempo. Esta anomalía, encabeza las causas de ceguera en los países desarrollados. Según la Organización Mundial de la Salud, hasta 1 millón de personas tienen ceguera debido a la diabetes [4]. Hay varios grados de la patología de la retinopatía diabética entre los que se encontrarían por orden de menor a mayor peligrosidad [22]:

- NPDR (Non-proloferative diabetic retinopathy), este grado se caracteriza por la ausencia de retinopatía;
- NPDR leve, donde los vasos sanguíneos empiezan a debilitarse, creando protuberancias llamadas micro-aneurismas;
- NPDR moderada, donde los vasos sanguíneos se siguen debilitando, se producen más hemorragias, pudiendo provocar visión borrosa;
- NPDR severa, en esta etapa, los vasos sanguíneos están dañados, causando falta de oxigeno en la retina y en la formación de nuevos vasos;
- PDR(proloferative diabetic retinopathy) o retinopatía diabética proliferativa, donde los vasos sanguíneos anormales que crecen en la retina y en el vítreo. Estos vasos pueden sangrar y provocar desprendimiento de retina, provocando la perdida de visión.

En la actualidad, cuando un paciente acude a su médico de familia por perdida de visión, el médico determinaría si hacerle una cita para que el especialista le haga una imagen de la retina con el retinógrafo, para descartar la posibilidad de retinopatía diabética.

Este proceso, conlleva elevados costes para la sanidad publica, ya que implica un gran número de pruebas, junto con el número de tramites administrativos, en algunos casos, solo para descartar las personas que no tienen

la enfermedad. Para el paciente supone un gran número de desplazamientos, que en algunos casos no se pueden permitir. Además de las horas de trabajo o lectivas que le supone al paciente.

Como solución a este problema, siguiendo con la tendencia de aplicar los avances informáticos a la medicina, se propone realizar una aplicación móvil, que permita al médico de familia hacer un estudio, haciendo una foto de la retina del paciente, utilizando una lente para el dispositivo móvil.

Además, para facilitar la labor del médico, esta aplicación móvil, proporciona redes neuronales convolucionales ya entrenadas, con las que se obtendría un primer análisis, enviando al especialista a aquellas personas que en cuyo resultado haya algún grado de retinopatía diabética.

De esta forma, al instalar la aplicación RetinAI creada con el objetivo de ser útil para el sistema sanitario, usando una interfaz sencilla para médicos sin experiencia previa en aplicaciones Android. Además, la aplicación facilita la creación de un informe médico ya que interpreta los resultados obtenidos de las imágenes, dando prioridad a aquellos casos que tengan más gravedad; y descartando aquellos casos que no tengan retinopatía diabética. Evitando una lentitud en el sistema sanitario debido a pacientes que no tienen esta complicación.

Permitiendo a los especialistas centrar su atención en aquellos pacientes que realmente tienen retinopatía diabética.

1.1. Estructura

La memoria se puede organizar en los siguientes apartados:

- **Introducción**: Apartado que dispone el tema y resume el contenido del trabajo.
- **Objetivos del proyecto**: Apartado donde se explican los objetivos que se buscan conseguir.
- **Conceptos teóricos**: Apartado del documento donde se exponen los conceptos claves del proyecto.
- **Técnicas y herramientas**: Apartado donde se explican las metodologías y herramientas utilizadas durante la realización del proyecto.
- **Aspectos relevantes**: Apartado donde se recogen los datos más importantes del desarrollo.

- **Trabajos relacionados**: Apartado donde se compara el trabajo realizado, con otros anteriores.
- **Conclusiones y lineas de trabajo futuras**: Apartado donde se expone de forma critica el trabajo realizado, indicando posibles mejoras a realizar en un futuro.

Además, se adjunta como anexo los siguientes apartados:

- **Plan de proyecto**: Apartado donde se explica como se ha organizado el proyecto, y estudios sobre la rentabilidad tanto económica como legal tiene el proyecto.
- **Requisitos**: Apartado donde se explican los requisitos funcionales y no funcionales que tiene el proyecto.
- **Diseño**: Apartado donde se describe la fase de diseño del proyecto.
- **Manual del programador**: Apartado donde se recogen los aspectos que necesitaría un programador, tanto como para seguir el trabajo, como para poder entender el funcionamiento.
- **Manual del usuario**: Apartado donde se realiza una guía de usuario, con los pasos a seguir para un correcto funcionamiento de la aplicación.

Objetivos del proyecto

Los objetivos del proyecto se pueden dividir en 3 apartados, los cuales se verán a continuación.

2.1. Objetivos generales

- Desarrollar una aplicación Android, que permita realizar un estudio de la retinopatía diabética sobre los pacientes.
- Agilizar el sistema sanitario, haciendo que los médicos de familia puedan dar un diagnostico preliminar a partir de los datos proporcionados.

2.2. Objetivos técnicos

- Desarrollar una aplicación Android con soporte API 21, siendo compatible con dispositivos Android 5.0 (Lollipop) y superiores.
- Hacer uso de Material 3 como fuente para que la aplicación sea más accesible al usuario.
- Hacer uso de Gradle para la automatización de la construcción de software.
- Hacer uso de GitHub como herramienta para alojar proyectos.
- Hacer uso de Git como herramienta de control de versiones.
- Hacer uso de la metodología SCRUM haciendo uso a su vez de la herramienta ZenHub.

- Hacer uso de GitKraken para hacer uso de una interfaz que facilite las acciones con el repositorio en GitHub.
- Hacer uso de pruebas, automáticas y manuales de forma que no se produzcan errores de ejecución.
- Hacer uso de Python para convertir un modelo keras con formato ".h5" a un modelo tensorflow Lite con formato ".tflite".
- Hacer uso de keras, en Python, para realizar una red neuronal convolucional.
- Hacer uso de TensorFlow Lite, para poder proporcionar los resultados en una aplicación Android.

2.3. Objetivos personales

- Hacer uso de los conocimientos vistos durante la carrera.
- Aprender cómo desarrollar aplicaciones, en las que hay que implementar una red neuronal.
- Realizar una aplicación para dispositivos Android, que se pueda ejecutar en la mayoría de estos teléfonos, facilitando de esta forma la integración en el sistema sanitario.

Conceptos teóricos

En este proyecto, la mayor completitud del trabajo se encuentra en el preprocesado de las imágenes, el cual es necesario uno distinto según la red con la que se trabaje.

3.1. Conceptos de Redes Neuronales Convolucionales

En este apartado se pondrá en contexto sobre los conceptos más importantes a tener en cuenta sobre una red neuronal convolucional, partiendo desde la idea de la inteligencia artificial.

La inteligencia artificial es una tecnología que busca imitar la inteligencia humana, los objetivos de la inteligencia artificial son: representación de conocimientos, aprendizaje automático, procesamiento del lenguaje natural, inteligencia social e inteligencia general [15].

El aprendizaje automático, o *machine learning*, es una rama de la inteligencia artificial, cuyo objetivo es que las máquinas aprendan por si mismas. Se considera aprendizaje la acción de mejorar el rendimiento en futuras acciones[33]. Dependiendo de la naturaleza, se puede considerar aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo.

El aprendizaje automático, como resultado da un modelo, que esta basado en una de los siguientes clasificadores: Árboles de decisión, reglas de asociación, algoritmos genéticos, máquinas de soporte vectorial, *clustering*, redes bayesianas y redes neuronales artificiales.

Las redes neuronales artificiales (RNN) son modelos computacionales con al menos una capa oculta[18], la cual consiste en una o más neuronas

donde cada una calcula la suma de los valores de entrada, multiplicados por su peso correspondiente[17].

El valor calculado se utiliza en una función de activación, que produce el valor de salida[14]. Entre las funciones de activación destacan las visibles en 3.1.

- La función RELU (verde), la cual indica el máximo entre 0 y el valor calculado.
- La función lineal(rosa), la cual devuelve el mismo valor calculado.
- La función sigmoidea(cían), la cual devuelve un valor entre 0 y 1 según lo alejado que este del 0.
- La función escalón(morada), devuelve o 0 o 1 dependiendo del signo del valor calculado.

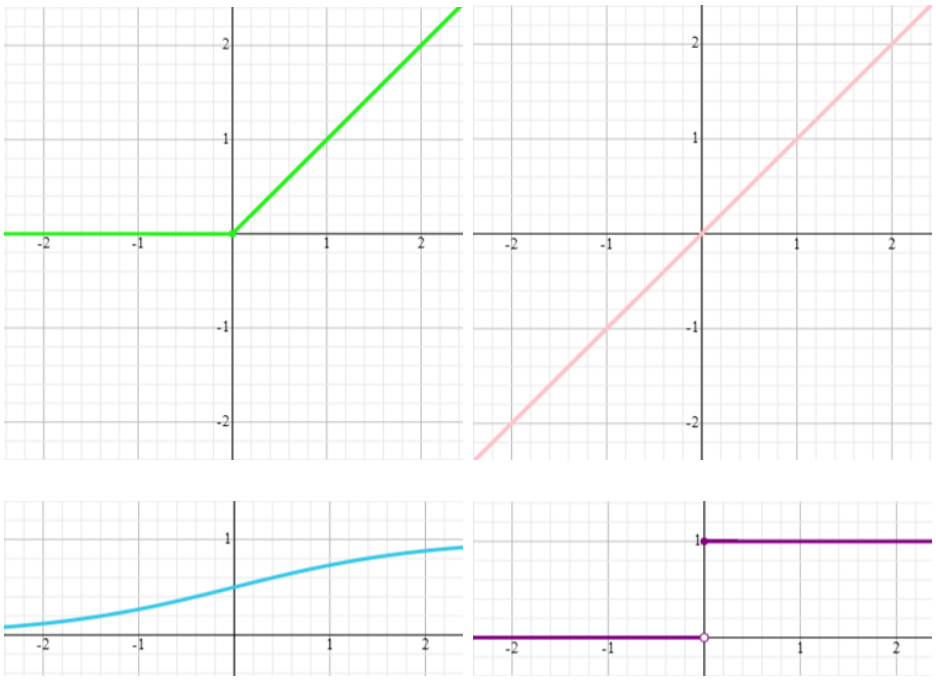

Figura 3.1: Funciones de activación

Dentro de las redes neuronales, se encuentran las redes neuronales convolucionales (CNN), es una red neuronal en la que hay al menos una capa

convolucional, consistiendo en la combinación de capas convolucionales, capas de agrupación y capas densas [13].

Una capa convolucional es un conjunto de neuronas artificiales que realizan varias series de operaciones convolucionales, actuando cada una sobre una submatriz de la matriz de entrada [11].

Una operación convolucional es aquella que a partir de una submatriz de la matriz de entrada, donde a veces es necesario realizar un filtrado, para ponderar los datos, calculando la suma de los valores de la submatriz, o en caso de que se haya realizado un filtro, del resultado de este, asignando el valor de la suma a una nueva matriz resultado con las dimensiones de la matriz de entrada [12].

La capa de agrupación es una capa de una red neuronal, es una capa que reduce el tamaño de la matriz de entrada, mediante operaciones como el máximo o la media de los valores de una submatriz[19].

La capa densa también llamada capa totalmente conectada, es una capa oculta, donde cada nodo esta conectado a todos los nodos de la siguiente capa oculta[16].

Una red neuronal residual es una red que permite el salto de capas intermedias. Para ello, se crea un bloque residual, lo que hace es al resultado de la ruta residual, sumar la entrada del bloque a la salida del bloque, como se puede ver en 3.2

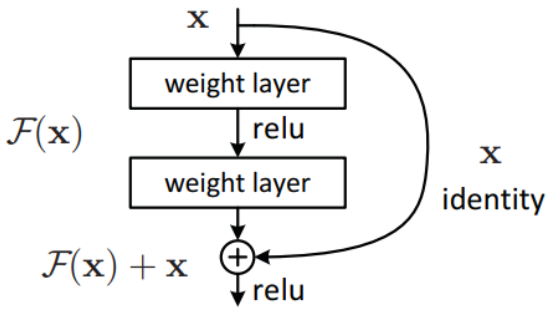

Figura 3.2: Bloque residual. Imagen extraída de [20]

3.2. Preprocesado

En el proyecto, se han realizado varios preprocesados, puesto que hay varias redes neuronales convolucionales.

Red neuronal convolucional VGG16, para la calidad de la imagen

En una red neuronal convolucional VGG16, el preprocesado necesario es que la imagen en vez de estar en formato RGB (Rojo verde y azul), tiene que ser formato BGR con los valores centralizados en 0. Además de este cambio, la imagen tiene que tener de 224 x 224 píxeles. [31] Para realizar este preprocesado en Python, se puede llamar a la función *tf.keras.applications.vgg16.preprocess_input*, para la aplicación de Android Studio, se debe realizar este preprocesamiento a mano. La estructura de la red convolucional VGG16, se puede observar en la figura 3.3, donde hay 13 capas convolucionales, y 4 de agrupación.

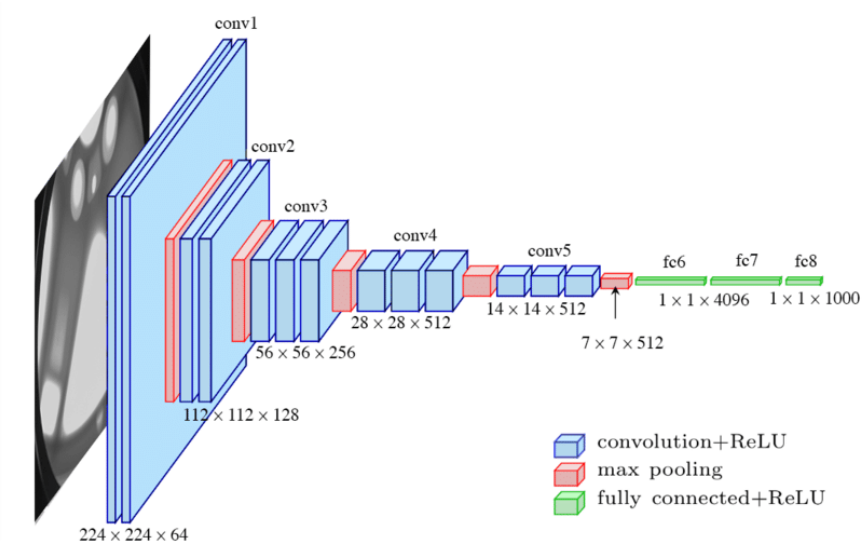

Figura 3.3: Estructura básica de la red convolucional VGG16. Imagen extraída de [9]

Red neuronal convolucional ResNet50V2, para la detección de retinopatía

La red ResNet50V2 es una red neuronal residual que, como se ha comentado anteriormente, permite saltarse capas, como se puede ver en la figura 3.2.

En la red neuronal convolucional ResNet50V2, el procesamiento es distinto al de VGG16, la gama de color es RGB, también se tiene que normalizar los datos, pero en este caso, el intervalo es [-1,1]. [30]

Esta red neuronal ya estaba entrenada, por tanto, no se tuvo que hacer ningún preprocesado en Python; pero al implementar la red en Android Studio, al igual que en la red VGG16, se debía realizar el preprocesamiento a mano.

Donde la formula para cambiar este valor vendría dada por:

$$ColorPreprocesado = (ColorSinPreprocesar - 0)/255{,}0 * 2 - 1$$

- Donde 0 representa el valor mínimo que puede tomar el color en concreto.
- Donde 255 representa el valor máximo que puede tomar el color en concreto.
- Y donde $*2 - 1$ es la operación para normalizar el valor en formato [-1, 1]

La estructura de la red convolucional ResNet50V2, se puede observar en la figura 3.4, en esta red neuronal convolucional,

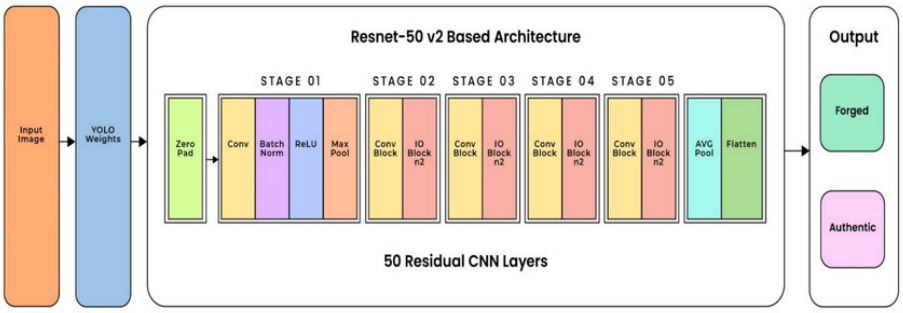

Figura 3.4: Estructura básica de la red convolucional ResNet50v2. Imagen extraída de [24]

3.3. Formato de la red neuronal

El formato de las redes neuronales convolucionales suele venir dado en formato ".h5" o keras, para facilitar la implementación en Android Studio, se

ha tenido que transformar este archivo en formato TensorFlow Lite, el cual requiere menos recursos, permitiendo su integración en dispositivos móviles.

Para realizar esta conversión, se ha utilizado un fichero Python, el cual se ha buscado en la documentación de TensorFlow Lite, y posteriormente, guardar el fichero. [27]

En el siguiente código, se puede observar el convertidor, donde file es el directorio donde se encontraría el archivo keras, nombre es el nombre de este fichero, y fileSalida es el directorio de salida.

```
model =
   tf.keras.models.load_model(file+nombre+'.h5')
converter =
   tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model= converter.convert()
nombreSalida = "calidad.tflite"
with open(fileSalida+nombre+'.tflite', 'wb') as
   f:
    f.write(tflite_model)
```

3.4. Desbalanceo de los datos

Para la minería de datos, el problema de que los datos estén desbalanceados causa que los modelos entrenados suelan producir resultados indebidos.

Un ejemplo del desbalanceo de datos, podría ser un modelo que entrena los números diciendo si son primos o no. En este caso, si se calcula el modelo por porcentaje de aciertos, en caso de devolver siempre que el número introducido no es primo, esta medida tiende a ser del 100 %. Por este motivo, se suelen usar otras métricas u otras formas de entrenar al modelo, para que se evite la disparidad de los datos.

La forma de solucionar este problema en este trabajo ha sido usando otras medidas como podrían ser precisión, recall y F1Score. En la figura 3.5 se puede observar la matriz de confusión de positivos y negativos. A partir de la cual, se explicaran las medidas.

La precisión es el numero de Verdaderos positivos entre la suma de verdaderos positivos y falsos positivos.

$$Precision = \frac{VerdaderosPositivos}{VerdaderosPositivos + FalsosPositivos}$$

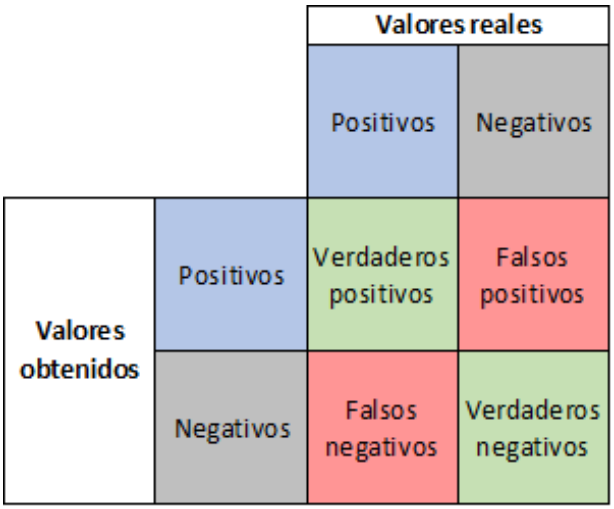

Figura 3.5: Matriz de confusión

Por otro lado, el recall, es el número de verdaderos positivos entre la suma de los verdaderos positivos y los falsos negativos.

$$Recall = \frac{VerdaderosPositivos}{VerdaderosPositivos + FalsosNegativos}$$

F1 Score es una medida obtenida de las dos anteriores, siendo la media armónica de estas.

$$F_1 = 2\frac{Precision * Recall}{Precision + Recall}$$

En el caso concreto, una imagen puede ser APTA o NO_APTA, encontrándose el desbalanceo de imágenes de 455 APTA y 103 NO_APTA. De esta forma, se ha considerado en la matriz 3.5, como "Positivos" a la clase NO_APTA y como "Negativos" a la clase APTA.

La otra opción que no se ha utilizado, pero es importante destacarla, es el uso de oversampling o undersampling. Estas técnicas buscan igualar número de entidades en el conjunto de datos según la clase.

Mientras oversampling, busca añadir datos a partir del conjunto de datos, duplicando o modificando los datos entre distintas instancias. Esta opción no se ha ni planteado, puesto que si se duplican las imágenes NO_APTA, lo que va a producir es sobreajuste del modelo; y suponiendo que los datos

nuevos fuesen modificaciones de combinaciones de imágenes NO_APTA, se estaría introduciendo ruido, puesto que crearía imágenes no reales.

El undersampling al contrario que el oversampling, elimina datos del grupo mayoritario, hasta que hay un número similar de instancias, las instancias eliminadas se pueden utilizar para validación o para test. Al implementar este método, se pueden eliminar inicialmente instancias que clasifican muy bien el modelo, haciendo que dependa el código de la inicialización del conjunto de entrenamiento y del conjunto de test.

Técnicas y herramientas

4.1. Metodologías

SCRUM

SCRUM es una metodología ágil basada en una estrategia continua e incremental, cuyo objetivo es proporcionar un producto funcional al final de cada periodo de trabajo planeado (*sprint*), haciendo reuniones diarias y antes y después de cada *sprint* otras reuniones donde se explican los problemas que se han tenido y como se va a planear el siguiente.

El impedimento más notorio de esta metodología, es que hay un equipo de personas entre las que se encuentra el *product owner*, el *SCRUM master* y el equipo de desarrollo.

Además se recomienda que el equipo este formado por 3 a un máximo de 9 personas para un buen desarrollo, por tanto, en este trabajo ha sido complicado ir haciendo todas las acciones que se piden en la metodología SCRUM.

GitFlow

GitFlow es un flujo de trabajo, en el cual se ramifica el proyecto en *branches* donde cada una, contiene una parte del proyecto; de esta forma, se puede dejar una parte funcional sin modificar que sería la rama principal, y otras ramas, donde se van realizando los cambios, cuando en estas ramas, se termina la tarea que se esta realizando, haciendo un producto funcional, se realiza una operación de *pull request* para combinar ambas ramas. Esta operación es aceptada o denegada por el equipo de desarrollo que no haya participado en la realización de esta rama.

En el proyecto, se han creado tres *branches*, que son:

- *main*: es la rama principal, donde se alberga la versión estable del proyecto.
- boceto: es una rama secundaria, donde se realizan los cambios que se están realizando en la aplicación móvil.
- latex: es la rama secundaria donde se realizan los cambios que se están realizando en el documento LaTeX.

Método del pato de goma

El método del pato de goma, o *rubber duck debugging* [34], es un método informal para la revisión de código.

Este método es utilizado por muchos programadores y surgió porque normalmente los programadores han tenido experiencias donde han tenido que explicar el problema a otra persona que no entiende sobre programación, y mientras se esta explicando el código, encontrar posibles soluciones.

Por tanto, este método, consiste en vez de explicar el código a otra persona, en explicárselo a un pato de goma.

4.2. Herramientas

Repositorio

Entre las opciones para realizar el repositorio, se pensó en GitLab y en GitHub, tomando esta última por conocimiento de uso de esta.

GitHub es una plataforma web de alojamiento de repositorios que utiliza Git como sistema de control de versiones.

Por tanto, como GitHub utiliza Git de forma nativa, no fue necesario la elección de un sistema de control de versiones.

Como para el repositorio se necesitan archivos de más de 100 MB, se ha usado la extensión Git LFS, estos ficheros son los las redes neuronales, tanto formato keras, como formato TensorFlow Lite.

Gestión del proyecto

Entre las opciones para gestionar el proyecto, se pensó en poder realizarlo con GitHub Projects, con Jira, con ZenHub y Trello; tomando la opción de ZenHub, por ya estar familiarizado con la herramienta.

ZenHub es una herramienta de gestión de proyectos que se integra con GitHub. Proporciona una tabla Kanban, donde se ponen las actividades a realizar durante el *sprint*. Permite poner una prioridad a las tareas, siguiendo una estimación de póquer, y además, ofrece la posibilidad de ver gráficos *burndown* donde se representa las actividades que faltan por hacer en el *sprint*, también ofrece otros gráficos como *cumulative flow* y *velocity tracking*.

Guía de diseño

Para la realización de la aplicación, es necesario una guía de diseño que facilite al usuario la interacción con la aplicación. Por ello, se utilizo la ultima guía ofrecida por Google, llamada Material 3.

Material 3 es la última versión del sistema de diseño del *open source* de Google. En el documento, viene información de recomendación de componentes ante el mismo problema.

Herramienta de iconos

Para la adición de iconos en la aplicación, es necesario que las imágenes y o iconos sean *open source* para ello, se estuvieron mirando páginas que ofrecían estos iconos, entre las páginas, al final se seleccionaron pixabay para la obtención de imágenes, puesto que ofrece licencia de uso para proyectos comerciales y no comerciales.

Por otro lado, para los iconos se utilizó Google Fonts y la galería que ofrece Android por defecto, se utilizo Google Fonts, porque en la guía de desarrollo se recomendaba y a su vez, es de código abierto y gratuito.

Para el logotipo de la aplicación, se ha utilizado DALL-E , una inteligencia artificial que convierte texto a imágenes, tras varios intentos, se logró conseguir un logotipo bastante bueno, y con ello, se hicieron unos retoques con el uso del programa de edición de imágenes GIMP, para la eliminación de ruido y para proporcionarle una gamma de colores; como resultado se obtuvo el icono actual.

Entorno de desarrollo integrado (IDE)

Para el desarrollo de la aplicación móvil, se pensó en Eclipse, Android Studio y Unity.

Al final, se descarto Eclipse por no ofrecer una experiencia de desarrollo especifica para aplicaciones Android. Y también se descarto Unity, aunque Unity si que esta especializado en aplicaciones móviles, por otro lado, se especializa en el desarrollo de videojuegos y en aplicaciones con realidad virtual. Por lo tanto, se selecciono Android Studio, es el IDE oficial de Android y está desarrollado por Google, basado en IntelliJ IDEA. Ofrece un emulador donde se compila la aplicación y poder comprobar el correcto funcionamiento de esta. Además, para la compilación hace uso de Gradle, separando el código de la aplicación, de la compilación.

El lenguaje utilizado ha sido Java, aunque Android Studio ofrece la opción de Kotlin, con Java no hacia falta aprender un nuevo lenguaje, puesto que se ha cursado durante la carrera.

LaTeX

Para la realización del documento, se pensó utilizar MiKTeX, TeX Live u Overleaf.

Al final, se escogió Overleaf, puesto que las otras 2 herramientas eran locales, y Overleaf ofrece acceso desde la nube, además tiene integración con proyectos GitHub, facilitando la exportación al repositorio.

Comunicación

Para la comunicación con los tutores, se uso tanto email, como tutorías presenciales, de esta forma, las cuestiones y los avances realizados se hacen por email, y en caso de mostrar el funcionamiento de la aplicación o alguna duda más importante, se realiza presencialmente para un mejor entendimiento.

Librerías

Librerías Android Studio

Para la aplicación Android, se han usado las siguientes librerías:

AndroidX Appcompat

Es una librería estática de Android que al añadirla al proyecto permite el uso de funcionalidades no incluidas en el framework o utilizar APIs no disponibles para versiones anteriores. Tiene compatibilidad con versiones de Android con una API 14 o mayor.

Material

Esta librería estática de Android permite implementar las especificaciones incluidas en Material Design. Tiene compatibilidad con versiones de Android con una API 14 o mayor.

ConstraintLayout

Esta librería de Android permite al desarrollador una forma flexible y adaptable de poner los *Widgets*.

JUnit

Es un framework para Java, que permite realizar pruebas unitarias.

AndroidX Espresso

Es un framework de pruebas de interfaz de usuario (UI) para las aplicaciones Android.

TensorFlow Lite

Es una librería de Android que permite desplegar modelos de aprendizaje automático en los dispositivos móviles.

SQLite

Es una librería de Android que permite la integración de bases de datos en dispositivos móviles.

Librerías de Python

Para la creación del modelo se han usado las siguientes librerías:

time

Es un módulo de la librería estandar de Python que permite trabajar con tiempos.

numpy

Es una librería de Python que permite manejar *arrays*, álgebra lineal, transformaciones de fourier y matrices.

pandas

Es una librería de Python que permite analizar, limpiar, explorar y manipular el conjuntos de datos.

Matplotlib

Es una libreria de python para ver visualmente gráficas de nivel bajo.

sklearn.metrics.confusion_matrix

Es una función de la librería de scikit-learn que permite trabajar con matrices de confusión.

tensorflow.keras

Es un framework de aprendizaje automatico, de este se utilizan varias clases:

- preprocessing.image.ImageDataGenerator: es una clase de Python que permite diversas transformaciones y aumentos de datos, como rotación, cambio de tamaño, recorte, cambio de brillo, entre otros.
- tensorflow.keras.applications.VGG16: es una clase de Python que proporciona la implementación de un modelo VGG16.
- tensorflow.keras.applications.VGG16.preprocess_input: es una función de la clase VGG16 que proporciona el preprocesado de la imagen en un modelo VGG16.
- tensorflow.keras.models.Sequential: es una clase de Python que permite agrupar un conjunto de capas, para convertirlas en un modelo.
- tensorflow.keras.layers: es un módulo que contiene información sobre las capas.
- tensorflow.keras.optimizers.Adam: es una clase que implementa el algoritmo Adam. El cual es un algoritmo de descenso de gradiente estocástico. Siendo un optimizador eficiente con consumo pequeño de recursos.
- tf.keras.metrics.Precision: es una clase que permite calcular la precisión de las predicciones con respecto a las etiquetas.
- tf.keras.metrics.Recall: es una clase que permite calcular el "recall" de las predicciones con respecto a las etiquetas.

sklearn.model_selection.test_train_split

Es una función de la librería de scikit-learn que permite dividir el conjunto de datos en conjunto de entrenamiento y conjunto de pruebas.

os

Es una librería de Python permite trabajar con funcionalidades del sistema operativo; a su vez se ha usado el módulo path de esta librería, el cual permite obtener la ruta desde donde se esta ejecutando el archivo.

4.3. Patrones de diseño

Modelo-Vista-Presentador

Es una derivación del patrón modelo-vista-controlador, la diferencia es que se cambia el controlador por un presentador, el cual sirve como intermediario entre el modelo y la interfaz.

- El modelo, define los datos que se utilizan en la interfaz.
- El presentador, funciona como intermediario, recuperando los datos del modelo, y cambiando la vista de la interfaz.
- La vista, es la interfaz de usuario.

Puesto que Android Studio, ya proporciona separaciones para hacer uso de un modelo-vista-presentador, su implementación ha sido sencilla, donde las *activities* son las diferentes interfaces, los archivos java del directorio "com.example.retinopatia" son los presentadores de las vistas, y en el directorio DataBase se encuentra el modelo de datos.

En la figura 4.1, se puede observar su funcionamiento.

Patrón singleton

Este patrón sirve para restringir la creación de objetos. En el caso utilizado, ha sido para la base de datos, de esta forma, no se creaba de cero cada vez que se iniciaba la aplicación.

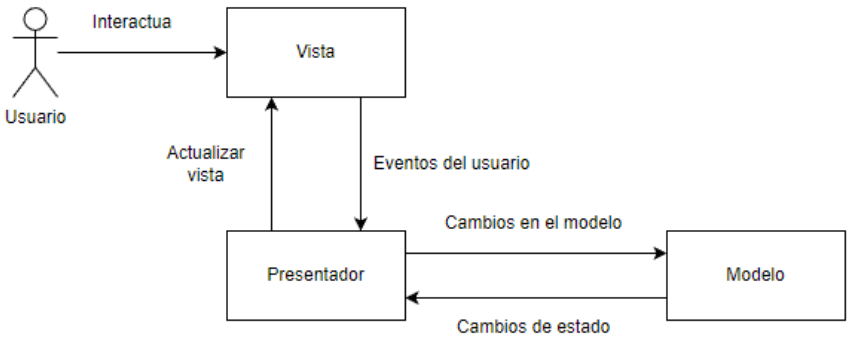

Figura 4.1: Modelo-Vista-Presentador

Aspectos relevantes del desarrollo del proyecto

En este apartado se recogen los aspectos más importantes. Explicando las decisiones tomadas del proyecto, y las consecuencias que suponen, comentando los errores y como se solucionaron.

5.1. Inicio

Una vez se me explicó la idea que se buscaba con este proyecto, me gustó la idea de poder hacer una aplicación que pudiera servir al sistema sanitario publico.

En este inicio, empecé a imaginarme como podría ser la aplicación haciendo bocetos mentales de sus distintas pantallas, como era muy efímero, se hizo una búsqueda de aplicaciones similares, para comprobar alguna interfaz a añadir, donde se encontró la aplicación RetinCam. Con una idea más formada, se hicieron los bocetos iniciales de la aplicación.

5.2. Formación

Para el proyecto no se necesito una formación especifica, puesto que para la aplicación de Android Studio se usó el lenguaje Java, visto durante la carrera, además también se hizo una aplicación de Android Studio en la asignatura “Interacción hombre-maquina” y para la creación de la red neuronal se usó el lenguaje Python, que también se ha visto durante la carrera.

A su vez, ha sido necesario el uso de tutoriales, y de guías de Android Studio y de keras para implementar correctamente el código.

Cámara en Android Studio

Una de las implementaciones en las que se tuvo que formar fue hacer uso de la cámara en una aplicación, guardando posteriormente la imagen obtenida.

Para ello, primero era inicializar la cámara, que se obtuvo una referencia de la propia Android Hacer foto con la cámara [5].

Al implementar inicializar cámara surgió un problema con los permisos de la aplicación, y no se abría la cámara, por lo que se buscó como implementar que se pidieran los permisos de la cámara al usuario, que se obtuvo una referencia de la propia Android Obtener permisos [7].

Una vez abierta la cámara, se tenia que guardar el resultado, puesto que se hacia la foto, pero no se mostraba al usuario, ni se guardaba de forma interna. Llegando otra vez a la pagina de Android, donde en un ejemplo se podía ver como implementarlo. Guardar resultado de la actividad [6].

Con la imagen obtenida de forma local, solo había que mostrarla al usuario.

Galería en Android Studio

Al igual que en la cámara, no se tenían los conocimientos para implementar este requisito.

Por ello, se utilizó el mismo enlace que en la cámara para obtener los resultados.Guardar resultado de la actividad[6].

Para llamar a esta actividad, se usó la respuesta de un foro de stackoverflow [23], que permitió llamar de forma correcta a esta actividad.

Como en el caso de la cámara, se pensó que también sería necesario el uso del permiso para leer almacenamiento externo. Surgiendo un error de ejecución. Se pensó que era un error de programación hasta que se encontró el siguiente enlace Version Android 11[8]. En este enlace, se comenta como a partir de la API 30, el permiso para leer almacenamiento externo no es necesario.

Una vez implementado este apartado, se uso correctamente la aplicación.

TensorFlow en Android Studio

Para implementar TensorFlow en Android Studio, se uso su versión para móviles TensorFlow Lite.

Como no se conocía como implementar un modelo, se fue a la guía de TensorFlow Lite, donde recomendaban el uso de un Interprete. Siguiendo el siguiente enlace como guía Guía de TensorFlow Lite [29].

Modelo en python

Para crear un modelo keras en python, se usó la guía que viene en TensorFlow para clasificación de imágenes[28].

Posteriormente, se escogió el modelo VGG16 por su conocimiento en el preprocesado, como modelo; por lo que se cambiaron las capas por una llamada al modelo VGG16.

Con el modelo ya creado, se calculo la "accuracy" entre un conjunto de modelos, escogiendo el máximo valor, en un conjunto de entrenamiento.

Aunque en asignaturas de la carrera, se había visto el desbalanceo de datos en la vida cotidiana, al inicio se consideró que no era un desbalanceo tan exagerado.

Este pensamiento se descartó puesto que podría dar problemas a la hora de usar la aplicación. Usando la precisión y "recall" como medidas.

Otras búsquedas

Además de las búsquedas mencionadas anteriormente, cabe destacar otros errores, tanto de programación, como de incompatibilidad entre otras cosas; que fueron solucionados por la comunidad de stackoverflow y de YouTube.

5.3. Metodologías

Como ya se ha comentado anteriormente, se decidió utilizar una metodología SCRUM, no siguiéndose al completo, puesto que los equipos de desarrollo en esta metodología están compuesto de 3 a 9 personas, a su vez, hay reuniones que no se han podido realizar, entre otras cosas. Pero, con esta metodología se ha buscado que el proyecto tuviese una metodología ágil.

Un fallo cometido con respecto esta metodología ha sido que los martes cambiaba de sprint a las 8 de la mañana, y la revisión para finalizar el sprint, se realizaba los martes por la mañana, haciendo que algunas veces las tareas cambiasen de sprint cuando no debían, y por tanto, la reunión para comenzar el sprint se realizaba con el sprint comenzado.

Las características ágiles de este proyecto son:

- Los sprints planeados han tenido una duración de 2 semanas, entregando un incremento al final de cada uno.
- Se han realizado reuniones, tanto para finalizar el sprint, como para el inicio del siguiente, teniendo en consideración el fallo comentado anteriormente.
- Las tareas que se planeaban para un sprint, se estimaban y priorizaban en un tablero canvas, tanto físicamente como con la herramienta ZenHub. Aunque con el paso de los sprints, se dejo de hacer físicamente.
- Para comprobar el progreso del proyecto, se ha utilizado los gráficos burndown, que aunque los ofrece ZenHub, se han realizado a mano, por el fallo comentado.

Al principio del proyecto se planeaba utilizar otras metodologías como *Test-Driven-Development* o como *Data-driven testing*, junto con pruebas automáticas, vistas durante el año académico en la asignatura Validación y pruebas, para comprobar como las implementaciones que se iban realizando en el proyecto no se veían afectadas entre ellas y que se implementaban correctamente.

Pero al implementar las distintas interfaces, se decidió hacer pruebas manuales, las cuales proporcionan una mayor comprensión de la interacción que tiene el usuario con la aplicación, buscando siempre que el usuario entienda el sistema.

5.4. Aplicación Android

Para trabajar en el proyecto, se pensó en utilizar Android Studio o UNITY, la opción seleccionada fue Android Studio, porque ya se tenían nociones previas.

Con el IDE seleccionado, se realizó el boceto al igual que se tenía realizado a mano, seleccionando como API para trabajar la versión 21, que como

informa Android Studio al crear el proyecto, se puede ejecutar en el 99,5 % de dispositivos 5.1.

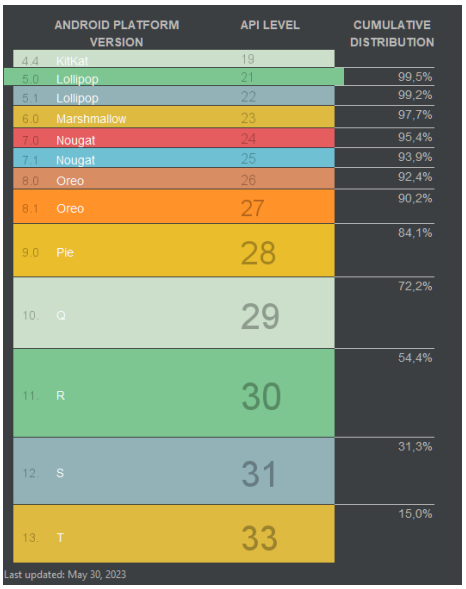

Figura 5.1: Porcentaje de dispositivos según la API. Imagen obtenida de Android Studio.

Después de terminar el boceto(Issue #7), se modifico la interfaz, puesto que en una reunión se propuso añadir y eliminar funcionalidades a la aplicación, en esta reunión se planteó añadir un modo oscuro, añadir la posibilidad de entrar como invitado, añadir una red neuronal que determine la calidad de la imagen, y en caso de ser mala, se repita la imagen, y una opción para cambiar el orden de los ojos, permitiendo que el médico elija su propia preferencia. Estas funcionalidades han sido introducidas en Issues posteriores.

Durante la implementación se encontraron varios bugs, que en algunos casos se encontraban durante la programación de esas partes del código, y en otros casos, se encontraban durante la ejecución de la aplicación de forma normal. De esta forma, en algunos casos se programaron Issues orientadas a la eliminación de estos errores, y en otros casos se hizo sobre la propia Issue que se estaba introduciendo.

Entre los bugs encontrados, cabe destacar los siguientes:

- Bug con el modo oscuro entre actividades, al introducir la variable que indicaba que el usuario había activado el modo oscuro, al pasar

la variable de una actividad a otra, en algunos lugares estaba mal instanciada y producía errores de ejecución.

- Bug con el modo oscuro de texto e imágenes no legibles, cuando se implemento el modo oscuro, algunas vistas de la interfaz no se cambiaron de forma correcta, de forma que el color del fondo no se distinguía de la vista.

- Bug con el botón que cambia el orden de los ojos, durante la implementación de esa parte, solo se probó si el botón cambiaba el orden de forma correcta, y así era, pero cada vez que se entraba a esa ventana estaba siempre en la misma posición.

- Bug con LocalDateTime, al principio para introducir en la base de datos la fecha del informe introducido o la fecha de nacimiento de los médicos y los pacientes, se utilizaba la clase LocalDateTime, aunque Android Studio avisaba de que era recomendable no usar esa clase, se utilizó, el problema surgió cuando se probó con un Android con versión API inferior, donde ni inicializaba la aplicación.

- Bug con la base de datos, la primera base de datos, era un conjunto de clases java donde con un constructor se inicializaban los usuarios, pacientes, informes,... El problema surgió cuando cada vez que se inicializaba la aplicación se usaba el constructor. Se implementó el patrón singleton y seguía ocurriendo el mismo problema, por ese motivo, se decidió utilizar SQLite para la base de datos.

- Bug con la petición de permisos, en la forma que se pedian los permisos, creaba infinitas comprobaciones de los permisos de cámara, haciendo que si no se contestaba durante un tiempo, la aplicación “crasheará”. Se solucionó eliminando el bucle y pidiendo un sólo permiso a la vez.

5.5. Implementación de las redes neuronales

Al iniciar el proyecto no se tenían conocimientos de TensorFlow, por lo que se tuvo que buscar información sobre como se podría integrar una red neuronal en una aplicación Android. Además las redes neuronales se recomendaban implementar en TensorFlow Lite, por lo que era necesario una conversión del modelo. Por lo que se obtuvo formación de las siguientes páginas:

- Cargar un modelo TensorFlow Lite en Android [26]

- Convertir modelo keras en modelo TensorFlow js [25]. Con esta idea, se usó la idea para convertir en formato TensorFlow Lite.
- Preprocesamiento modelo VGG16 [31]
- Preprocesamiento modelo ResNet50V2 [30]

Con la información obtenida de los enlaces anteriores, se hizo un modelo básico que calculaba la operación XOR en la Issue #21; para comprobar su correcto funcionamiento.

Posteriormente, se recomendó el uso de la red VGG-16 ya entrenada para comprobar los resultados con imágenes, para ello, se uso la red publicada por Lorenzo Baraldi [2]. Para introducir esta red neuronal se necesita el preprocesamiento de los datos y para su ejecución se necesita saber el numero de salidas de la red; esta red esta destinada a el conjunto de datos "imagenet" [35] lo que significa que busca clasificar entre 1000 categorías distintas. Dando a cada una un porcentaje de acierto, y utilizando como predicción la categoría con mayor porcentaje.

Se cometió un error al implementar esta red, pero, no se detectó este error, hasta que se implementó la red neuronal para detectar si la imagen es apta o no. La issue relacionada a esta actividad, es Issue #24.

Además, en la tarea anterior, se pensó que debido a que algunas redes neuronales tardan más de lo esperado en ejecutarse, para que el usuario no tuviese que estar esperando a que se terminase de ejecutar. Así, se implemento un hilo con las actividades de la red neuronal, permitiendo que al pulsar el botón de evaluar la imagen, se ejecute en segundo plano, y cuando termina se almacena el resultado en la base de datos.

Finalmente, se añadió la red entrenada para la clasificación de la retinopatía diabética, la cual es una red convolucional ResNet50v2. Cambiando a su vez, el preprocesamiento de la imagen, ya que es distinto, como se ha comentado anteriormente.

Esta Issue se corresponde con Issue #41. Sufriendo pequeños cambios en futuras releases. Se modificó, de forma que las 1000 categorías fueran 5, donde los valores significaban:

- 0 indica que el paciente tiene NPDR.
- 1 indica que el paciente tiene NPDR leve.
- 2 indica que el paciente tiene NPDR moderada.

- 3 indica que el paciente tiene NPDR severa.
- 4 indica que el paciente tiene PDR.

5.6. Creación del modelo para detectar la calidad de imagen

Para la creación del modelo, se utilizó en modelo VGG-16 puesto que ya se había utilizado anteriormente para practicar.

Partiendo de esta idea, el conjunto de datos se divide en 2 partes, el conjunto de imágenes, y un csv donde se indica el nombre de la imagen y la calidad que tiene esta. La calidad esta compuesto de números del 1 al 5, y se considera una calidad aceptable cuando el valor es 4 o 5, y no es aceptable cuando es menor que 4. Para definir el conjunto de entrenamiento, de test y de validación se dividió el conjunto de datos de forma 80 % para entrenar, 10 % de test y el otro 10 % validación.

Como métricas utilizadas, se empezó utilizando la *accuracy*, la cual mide el número de aciertos totales entre el número total de ejemplares. Como se ha comentado en el apartado de Conceptos teóricos, esta medida no es recomendable para datos desbalanceado.

Por ello se usan las medidas precisión y recall, con las que posteriormente se obtiene el F1-Score. Para comprobar que modelo es mejor, se ha representado la matriz de confusión para ver estas medidas de forma visual y posteriormente, calculando el f1-score, el cual se ha introducido en el nombre del modelo creado.

Como el modelo creado puede variar según el conjunto de entrenamiento escogido, se ha ejecutado varias veces para obtener el mejor resultado posible; otro valor a tener en cuenta a la hora de la creación es la tasa de aprendizaje del modelo. Una tasa de aprendizaje baja suele hacer que el modelo no se ajuste lo suficiente al resultado esperado; y una tasa de aprendizaje demasiado alta, puede producir que el modelo aprenda los datos. En la imagen 5.2, se puede observar el sobreajuste y el infrajuste, que son los resultados posibles al escoger una tasa de aprendizaje alta o baja.

Además, en un principio, se quería comparar sobre si clasificación binaria, era más eficiente que clasificación multiclase. Cuando se compararon estos valores, se utilizaba la medida “accuracy”, que como ya se ha explicado no es recomendable su técnica en clasificación binaria. Aún así, se obtuvo

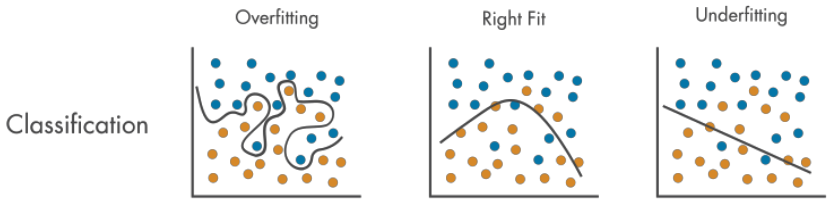

Figura 5.2: Comparación de sobreajuste e infrajuste. Imagen adaptada de [21]

resultados mejores en clasificación binaria que multiclase. Por ello, se decidió usar estos modelos.

En la gráfica 5.3, se observa el historial del mejor caso observado para la clasificación binaria.

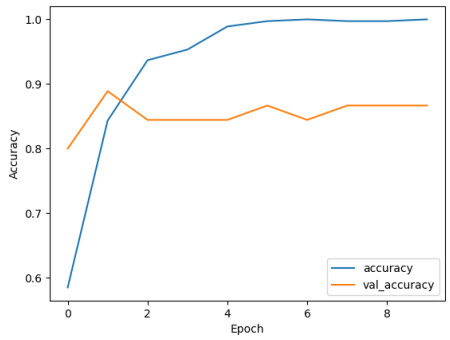

Figura 5.3: Historial de la evolución de la "accuracy" de un modelo cuyo resultado de test era de 91 %

Para obtener los nuevos modelos, con f1-score se volvió a ejecutar el código, pero, puesto que se observo que la clasificación binaria proporcionaba mejores resultados, se decidió comparar solo entre esta clasificación.

Al observar las matrices de confusión obtenidas, había 2 resultados mejores que los demás, y en varios casos, se obtuvo el resultado mencionado anteriormente, donde el modelo indicaba que todas las imágenes eran validas, pero en este caso, no se calculaba la "accuracy".

En la figura 5.4, se puede observar como y la matriz inferior, hace referencia al caso comentado, y las 2 matrices de confusión de la parte superior, hacen referencia a 2 modelos que clasifican de forma correcta casi todos los datos de test.

Para la matriz de la izquierda:

$$Precision = \frac{11}{11+2} = 0{,}85$$
$$Recall = \frac{11}{11+0} = 1$$
$$F_1 = 2\frac{0{,}85 * 1}{0{,}85+1} = 0{,}9189$$

Para la matriz de la derecha:

$$Precision = \frac{10}{10+1} = 0{,}909$$
$$Recall = \frac{10}{10+1} = 0{,}909$$
$$F_1 = 2\frac{0{,}909 * 0{,}909}{0{,}909+0{,}909} = 0{,}909$$

Para la matriz de abajo:

$$Precision = \frac{0}{0+0} = NaN$$
$$Recall = \frac{0}{0+11} = 0$$
$$F_1 = 2\frac{NaN * 0{,}909}{NaN+0{,}909} = NaN$$

Entre las 2 opciones de la figura, aunque ambas han tenido la misma cantidad de aciertos, según la formula del f1-score, el modelo de la izquierda es mejor, puesto que acierta correctamente en todos los que predice como APTO, cosa que el modelo de la derecha no hace.

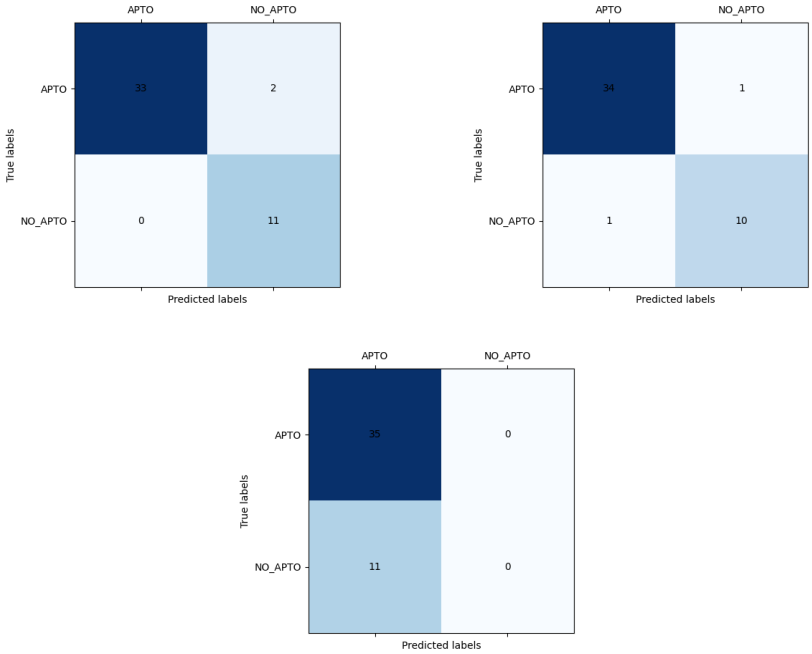

Figura 5.4: Comparación de los modelos destacables

Trabajos relacionados

6.1. Proyectos

Ret-iN CaM

Ret-iN CaM es la aplicación que se ha tomado de referencia para hacer el proyecto. Es una aplicación iOS que posteriormente saco una versión para Android. Es una aplicación que permite realizar imágenes y vídeos de la retina con una gran resolución, proporcionando los informes del paciente, los cuales se pueden exportar para ser compartidos con otros especialistas. A su vez, tiene una interfaz simple e intuitiva, lo que facilita a los usuarios interactuar fácilmente con ella. Principal motivo por el que se ha escogido esta aplicación.

D-EYE 2.0

Es un proyecto que permite la toma de imágenes y vídeos de alta calidad; permite a los médicos ver el nervio óptico sin necesidad de dilatar las pupilas; permite a los médicos ver si el paciente tiene trastornos neurológicos relacionados con el ojo.

6.2. Comparativa del proyecto

Características	RetinAI	Ret-iN CaM	D-EYE 2.0
Aplicación Android	X	X	
Aplicación iOS		X	X
Creación de usuarios			X
Cambio de modo oscuro y claro	X		
Permite iniciar sesión como invitado	X		
Permite guardar la sesión		X	X
Permite elegir el paciente	X	X	X
Ver historial del paciente	X	X	X
Crear nuevos informes	X	X	X
Permite diferenciar entre ojos	X	X	X
Permite hacer imágenes	X	X	X
Permite hacer vídeos		X	X
Escoger imágenes desde la galería	X		
Red neuronal para los resultados	X		
Versión gratuita	X	X	X

Tabla 6.1: Comparativa de las características de los proyectos.

De esta forma, se puede ver las ventajas que ofrece el proyecto.

- Actualmente, hay más móviles con sistema operativo Android que con iOS, por tanto, se ha realizado la aplicación en un sistema Android por este motivo.
- No se permite la creación de usuarios, puesto que como la aplicación esta destinada a médicos de la sanidad publica, la entidad encargada les proporcionará las cuentas para la aplicación.
- La aplicación tiene la opción de cambiar entre modo oscuro y modo claro, de esta forma, permite al usuario adaptarla a su preferencia.
- Al iniciar sesión como usuario, los médicos podrán tener un diagnostico rápido de un paciente, sin que se almacene el informe en la base de datos.
- A la hora de seleccionar pacientes, se ha considerado la protección de datos de los pacientes y para que el médico seleccione a uno, tendrá que poner el DNI.

- Como es posible que se analice una foto tomada desde otro dispositivo. Se ha considerado esta idea mostrando en el explorador de archivos las imágenes.
- Ofrece una red neuronal ya entrenada, la cual determina el grado de retinopatía diabética que tiene el paciente. Característica en la que se basa la aplicación.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] Fattoh Al-Qershi, Muhammad Al-Qurishi, Sk Md Mizanur Rahman, and Atif Al-Amri. Android vs. ios: The security battle. In *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, pages 1–8, 2014. doi: 10.1109/WCCAIS.2014.6916629.

[2] Lorenzo Baraldi. Vgg-16 pre-trained model for keras, 2016. URL https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3. [Online; Accedido 9-junio-2023].

[3] Organización Mundial de la Salud. Ceguera y discapacidad visual, 2022. URL https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment. [Online; Accedido 10-febrero-2023].

[4] Organización Mundial de la Salud. Diabetes, 2022. URL https://www.who.int/es/news-room/fact-sheets/detail/diabetes. [Online; Accedido 18-febrero-2023].

[5] Android Developers. Cómo usar la cámara - android developers, 2023. URL https://developer.android.com/training/camera/camera-intents. [Online; Accedido 12-junio-2023].

[6] Android Developers. Cómo obtener un resultado de una actividad - android developers, 2023. URL https://developer.android.com/training/basics/intents/result?hl=es-419. [Online; Accedido 12-junio-2023].

[7] Android Developers. Cómo solicitar permisos - android developers, 2023. URL https://developer.android.com/training/

permissions/requesting?hl=es-419. [Online; Accedido 12-junio-2023].

[8] Android Developers. Política de almacenamiento de android 11 y privacidad - android developers, 2023. URL https://developer.android.com/about/versions/11/privacy/storage?hl=es-419. [Online; Accedido 12-junio-2023].

[9] Max Ferguson, Ronay Ak, Yung-Tsun Tina Lee, and Kincho H. Law. Automatic localization of casting defects with convolutional neural networks. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1726–1735, 2017. doi: 10.1109/BigData.2017.8258115.

[10] Centers for Disease Control and Prevention (CDC). Blindness caused by diabetes–massachusetts, 1987-1994. *MMWR Morb Mortal Wkly Rep.*, 1996.

[11] Google. Convolutional layer, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#convolutional_layer. [Online; Accedido 24-mayo-2023].

[12] Google. Convolutional operation, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#convolutional_operation. [Online; Accedido 24-mayo-2023].

[13] Google. Convolutional neural network, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#convolutional-neural-network. [Online; Accedido 24-mayo-2023].

[14] Google. activation function, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#activation_function. [Online; Accedido 12-junio-2023].

[15] Google. artificial intelligence, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#artificial-intelligence. [Online; Accedido 12-junio-2023].

[16] Google. fully connected layer, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#fully_connected_layer. [Online; Accedido 24-mayo-2023].

[17] Google. hidden layer, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#hidden_layer. [Online; Accedido 12-junio-2023].

[18] Google. neural network, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#neural-network. [Online; Accedido 12-junio-2023].

[19] Google. Pooling, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#pooling. [Online; Accedido 24-mayo-2023].

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[21] MathWorks. Understanding overfitting in machine learning. URL https://www.mathworks.com/discovery/overfitting.html. [Online; Accedido 10-junio-2023].

[22] Cecelia Koetting OD, FAAO. The four stages of diabetic retinopathy. *Modern Optometry*, 2019.

[23] Stack Overflow. Get image from the gallery and show in imageview, 2016. URL https://stackoverflow.com/questions/38352148/get-image-from-the-gallery-and-show-in-imageview. [Online; Accedido 12-junio-2023].

[24] Emad Qazi, Tanveer Zia, and Abdulrazaq Almorjan. Deep learning-based digital image forgery detection system. *Applied Sciences*, 12:2851, 03 2022. doi: 10.3390/app12062851.

[25] TensorFlow. Tensorflow.js: Importa un modelo keras a tensorflow.js, 2022. URL https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=es-419. [Online; Accedido 9-junio-2023].

[26] TensorFlow. Tensorflow lite: Carga un modelo keras en android, 2022. URL https://www.tensorflow.org/lite/android/quickstart?hl=es-419#load_a_keras_model_in_android. [Online; Accedido 9-junio-2023].

[27] TensorFlow. Tfliteconverter tensorflow lite api, 2022. URL https://www.tensorflow.org/lite/api_docs/python/tf/lite/TFLiteConverter. [Online; Accedido 27-mayo-2023].

[28] TensorFlow. Tutorial de clasificación de imágenes con tensorflow - tensorflow, 2023. URL https://www.tensorflow.org/tutorials/images/classification?hl=es-419. [Online; Accedido 12-junio-2023].

[29] TensorFlow. Guía de inferencia de tensorflow lite - tensorflow, 2023. URL https://www.tensorflow.org/lite/guide/inference?hl=es-419. [Online; Accedido 12-junio-2023].

[30] TensorFlow. Resnet50v2, 2023. URL https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet_v2/ResNet50V2. [Online; Accedido 27-mayo-2023].

[31] TensorFlow. Vgg16, 2023. URL https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16. [Online; Accedido 27-mayo-2023].

[32] David Turbert. La enfermedad ocular diabética, 2022. URL https://www.aao.org/salud-ocular/enfermedades/la-enfermedad-ocular-diabetica. [Online; Accedido 20-mayo-2023].

[33] Wikipedia. Machine learning, 2023. URL https://en.wikipedia.org/wiki/Machine_learning. [Online; Accedido 12-junio-2023].

[34] Wikipedia. Rubber duck debugging, 2023. URL https://en.wikipedia.org/wiki/Rubber_duck_debugging. [Online; Accedido 10-junio-2023].

[35] yrevar. imagenet 1000 class idx to human readable labels, 2019. URL https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a. [Online; Accedido 9-junio-2023].