



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Aplicación Android para la
detección de retinopatía
diabética**



Presentado por Miguel Fuente García
en Universidad de Burgos — 4 de julio de 2023
Tutor: D. Daniel Urda Muñoz y D. Nuño
Basurto Hornillos



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. Daniel Urda Muñoz y D. Nuño Basurto Hornillos, profesores del departamento de Digitalización, área de Ciencia de la Computación e Inteligencia Artificial,

Exponen:

Que el alumno D. Miguel Fuente García, con DNI 71313280D, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado “Aplicación Android para la detección de retinopatía diabética”.

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, 4 de julio de 2023

Vº. Bº. del Tutor:

D. Daniel Urda Muñoz

Vº. Bº. del co-tutor:

D. Nuño Basurto Hornillos

Resumen

La retinopatía diabética es una de las mayores causas de ceguera en los países desarrollados. Para diagnosticar el grado de la complicación, 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. Así mismo, la cita con el especialista tiene una demora que puede perjudicar la situación del paciente.

Por ello, se han desarrollado unos dispositivos, para las cámaras de los teléfonos 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 podría realizar la fotografía al paciente, lo que facilitaría un posible cribado, anterior a la creación de la cita con el especialista.

Este trabajo tiene como objetivo desarrollar una aplicación Android que facilite el trabajo del médico; permitiendo un primer diagnóstico del paciente mediante el envío de una imagen de la retina a modelos de aprendizaje computacional. De esta forma, los pacientes que tengan algún grado de retinopatía diabética, se les 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. This exam has a high cost for spanish public health, implying that the patient travels to a specialized 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 go to a hospital that has a retinograph and the family doctor could take the patient's photo, which would facilitate possible screening, prior to making an appointment with the specialist.

The objective of this work is to develop an Android app that help the doctor's work by allowing for an initial diagnosis of the patient through the submission of a retina image to computational learning models. This way, patients with any degree of diabetic retinopathy can undergo a comprehensive study.

As a result, the app RetinAI has been created.

Keywords

Health, diabetic retinopathy, neural networks, Android app.

Índice general

Índice general	iii
Índice de figuras	v
Índice de tablas	vi
Introducción	1
1.1. Estructura	3
Objetivos del proyecto	5
2.1. Objetivos generales	5
2.2. Objetivos técnicos	5
2.3. Objetivos personales	6
Conceptos teóricos	7
3.1. Conceptos de Redes Neuronales	7
3.2. Preprocesado	11
3.3. Transfer learning	14
3.4. Formato de la red neuronal	14
3.5. Desbalanceo de los datos	15
3.6. Guías de diseño Android	17
Técnicas y herramientas	19
4.1. Metodologías	19
4.2. Herramientas	20
4.3. Patrones de diseño	25
Aspectos relevantes del desarrollo del proyecto	27

5.1. Inicio	27
5.2. Metodologías	27
5.3. Formación	28
5.4. Conjunto de datos	29
5.5. Creación del modelo para detectar la calidad de imagen	29
5.6. Aplicación Android	33
Trabajos relacionados	39
6.1. Proyectos	39
6.2. Comparativa del proyecto	40
Conclusiones y Líneas de trabajo futuras	43
7.1. Conclusiones	43
7.2. Líneas de trabajo futuras	44
Bibliografía	47

Índice de figuras

3.1. Funcionamiento de una red neuronal	8
3.2. Funciones de activación	9
3.3. Ejemplo de una red neuronal	10
3.4. Bloque residual. Imagen extraída de [22].	11
3.5. Estructura básica de la red convolucional VGG16. Imagen extraída de [11]	12
3.6. Estructura básica de la red convolucional ResNet50v2. Imagen extraída de [29]	13
3.7. Matriz de confusión	16
3.8. Diferencias esquinas. Imagen obtenida de [25]	17
4.1. Modelo-Vista-Presentador	26
5.1. Imágenes con una calidad mala	30
5.2. Imágenes con una calidad buena	30
5.3. Comparación de sobreajuste e infrajuste. Imagen adaptada de [24]	31
5.4. Historial de la evolución de la “accuracy” de un modelo cuyo resultado de test era de 91 %	32
5.5. Comparación de los modelos destacables	33
5.6. Porcentaje de dispositivos según la API. Imagen obtenida de Android Studio.	34

Índice de tablas

6.1. Comparativa de las características de los proyectos.	40
---	----

Introducción

Durante los últimos años, se ha aumentado exponencialmente el uso de los dispositivos móviles, este aumento se debe principalmente a la comodidad que proporcionan 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 67,56 % de los usuarios prefieren Android y el 31.6 % iOS [32].

Los móviles, no solo han sufrido un desarrollo significativo en lo referente a 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. Una de estas funcionalidades es la integración de inteligencias artificiales que en los últimos años han reducido considerablemente los recursos y tiempo que necesitan para ejecutarse. Esta funcionalidad se ha convertido en una parte fundamental de la experiencia del usuario. Por ejemplo, aplicaciones de reconocimiento de voz como los asistentes virtuales (que utilizan algoritmos de procesamiento del lenguaje), también existen aplicaciones de reconocimiento facial para desbloquear el dispositivo (que se encuentra en el área de visión artificial) y el último ejemplo, son traducciones en tiempo real que facilitan la comunicación entre idiomas. Estos distintos usos que se dan a la inteligencia artificial están transformando la forma de interactuar con las aplicaciones móviles, brindando una experiencia más fluida y personalizada al usuario.

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. Como son aplicaciones del SACYL [3], de Adeslas [6], de la Organización Mundial de la salud [27] y de Google [23] entre otras.

Además de las aplicaciones mencionadas anteriormente, también se han desarrollado otras más específicas destinadas a ayudar a las personas con discapacidad visual, la cual afecta a más de 2.200 millones de personas en el mundo. Las principales causas de pérdida de visión son la degeneración macular relacionada con la edad, las cataratas, la retinopatía diabética, el glaucoma y errores de refracción no corregidos. Unos 1.000 millones de personas tienen un deterioro moderado o grave de la visión[4].

La retinopatía diabética [40] 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 [5]. En Estados Unidos, cada año la retinopatía diabética suma un 12 % de nuevos casos de ceguera [12]. Hay varios grados de la patología de la retinopatía diabética entre los que se encontrarían por orden de menor a mayor peligrosidad [26]:

- NPDR (Non-proliferative 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 oxígeno en la retina y en la formación de nuevos vasos;
- PDR(proliferative 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 pérdida de visión.

En el caso de la retinopatía diabética, también se han desarrollado aplicaciones móviles como Ret-iN CaM [2], la cual permite guardar imágenes

de las retinas de los pacientes, obteniendo un historial para realizar un estudio de la evolución de su patología. Esto es debido al desarrollo de una lente que permite obtener fotos de la retina desde el dispositivo móvil.

Actualmente cuando un paciente acude a su médico de familia por pérdida de visión, el médico le da cita para el especialista para que éste decida si se tiene que hacer una imagen del fondo de la retina con un retinógrafo, un dispositivo que permite obtener imágenes del fondo de la retina con una gran resolución, para descartar la posibilidad una retinopatía diabética. Una vez que el paciente acude a la consulta del especialista (lo que puede llevar varios meses de espera con las consiguientes complicaciones), se podrán descartar muchas personas que no tienen la enfermedad. Pero todas estas pruebas y los recursos invertidos suponen elevados costes para la sanidad pública y demoras en el tiempo de espera de los pacientes que realmente tengan un grado de retinopatía diabética.

Por estos motivos, si el médico de familia utiliza la lente comentada anteriormente, junto con el teléfono móvil, se podrían obtener las imágenes del fondo de la retina, con la suficiente calidad como para hacer un diagnóstico preliminar. De esta manera se podrían reducir también los costes económicos comentados anteriormente y sobre todo el tiempo de espera de los pacientes al reducir las consultas que se derivan al especialista.

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, a partir de una foto de la retina del paciente, utilizando una lente para el dispositivo móvil [31]. Además, para facilitar la labor del médico, esta aplicación móvil, proporciona redes neuronales convolucionales ya entrenadas, con las que el médico obtendrá un primer análisis del paciente, sabiendo cuándo la imagen tiene una calidad aceptable. De esta forma, se evitaría una lentitud en el sistema sanitario, enviando al especialista a aquellas personas que en cuyo resultado haya algún grado de retinopatía diabética.

1.1. Estructura

La memoria se puede organizar en los siguientes apartados:

- **Introducción:** Sección 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:** Parte del documento donde se exponen los conceptos claves del proyecto.
- **Técnicas y herramientas:** Lugar donde se explican las metodologías y herramientas utilizadas durante la realización del proyecto.
- **Aspectos relevantes:** Capítulo donde se recogen los datos más importantes del desarrollo.
- **Trabajos relacionados:** Comparación del trabajo realizado, con otros anteriores.
- **Conclusiones y líneas 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:** En este apartado donde se explica cómo se ha organizado el proyecto, y estudios sobre la rentabilidad tanto económica como legal tiene el proyecto.
- **Requisitos:** Parte del documento donde se explican los requisitos funcionales y no funcionales que tiene el proyecto.
- **Diseño:** Capítulo donde se describe la fase de diseño del proyecto.
- **Manual del programador:** Manual donde se recogen los aspectos que necesitaría un programador, tanto como para seguir el trabajo, como para poder entender el funcionamiento.
- **Manual del usuario:** Manual 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

- Proporcionar a los centros sanitarios una aplicación, que permita diferenciar cuándo un paciente posee o no retinopatía diabética.
- Agilizar el sistema sanitario, haciendo que los médicos de familia puedan dar un diagnóstico preliminar a partir de los datos proporcionados.
- Desarrollar un modelo de red neuronal que permita identificar la calidad de la imagen seleccionada, de forma que el médico sepa si la imagen tiene suficiente calidad, o tiene que repetir la imagen.

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.
- Usar la herramienta Gradle para la automatización de la construcción de software.
- Utilizar GitHub como herramienta para alojar proyectos.

- Utilizar Git como herramienta de control de versiones.
- Emplear 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.
- Crear pruebas, de forma que no se produzcan errores de ejecución.
- Convertir un modelo keras con formato “.h5” a un modelo tensorflow Lite con formato “.tflite”.
- Realizar una red neuronal convolucional utilizando un modelo keras ya entrenado.
- Utilizar 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, el enfoque principal se centra en las redes convolucionales, tanto en el preprocesado llevado, como la creación a través de transferencia de aprendizaje. Además, hay que tener en cuenta que se debe crear una aplicación Android, donde estas redes estén incluidas.

3.1. Conceptos de Redes Neuronales

En esta sección, se van a ver los conceptos más importantes de las redes neuronales convolucionales en el contexto de la inteligencia artificial.

La inteligencia artificial es una rama de la ciencia de la computación que trata de imitar la inteligencia humana para realizar tareas, buscando simular la capacidad de aprendizaje y razonamiento que tienen los seres humanos, entre otras cosas. Algunos de los objetivos de la inteligencia artificial son: representación de conocimientos, aprendizaje automático, procesamiento del lenguaje natural, inteligencia social e inteligencia general [17].

El aprendizaje automático, o *machine learning*, es una rama de la inteligencia artificial, cuyo objetivo es que las máquinas aprendan por sí mismas a partir de los datos proporcionados. Se considera aprendizaje a la adquisición de conocimientos a través del estudio, la experiencia o de ser enseñado; como esta definición no es útil para las máquinas, se define nombrando rendimiento en vez de conocimiento, así, se considera que el objetivo del aprendizaje es mejorar el rendimiento en futuras acciones [41]. El aprendizaje automático se apoya en clasificadores para generar un modelo. Entre los más comunes se encuentran: los árboles de decisión, las reglas de asociación, los algoritmos genéticos, las máquinas de soporte vectorial, *clustering*, las redes bayesianas y las redes neuronales artificiales. Según la naturaleza del

aprendizaje automático, se puede considerar uno de estos tipos: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje semisupervisado y aprendizaje por refuerzo.

El aprendizaje supervisado es un tipo de aprendizaje automático, en este se dispone de un conjunto de datos junto con el resultado que se espera obtener. Durante el entrenamiento, el modelo aprende a reconocer patrones y relaciones entre los datos de entrada y sus respectivas etiquetas, si el modelo está proporcionando resultados incorrectos, se pueden corregir de forma que se obtengan mejores resultados. Con el modelo ya entrenado, se pueden obtener predicciones de datos no vistos. Un problema que surge es el *overfitting*, el cual hace que el modelo aprenda los datos, dando los resultados correspondientes para cada instancia.

El aprendizaje supervisado se ha beneficiado enormemente de los avances en las redes neuronales artificiales (RNN). Estas redes son modelos computacionales con al menos una capa oculta [20], la cual consiste en una o más neuronas donde cada una calcula la suma de los valores de entrada, multiplicados por su peso correspondiente [19]. En la figura 3.1, se puede observar una red neuronal, donde cada círculo representa una neurona artificial, y las flechas la conexión entre las distintas neuronas.

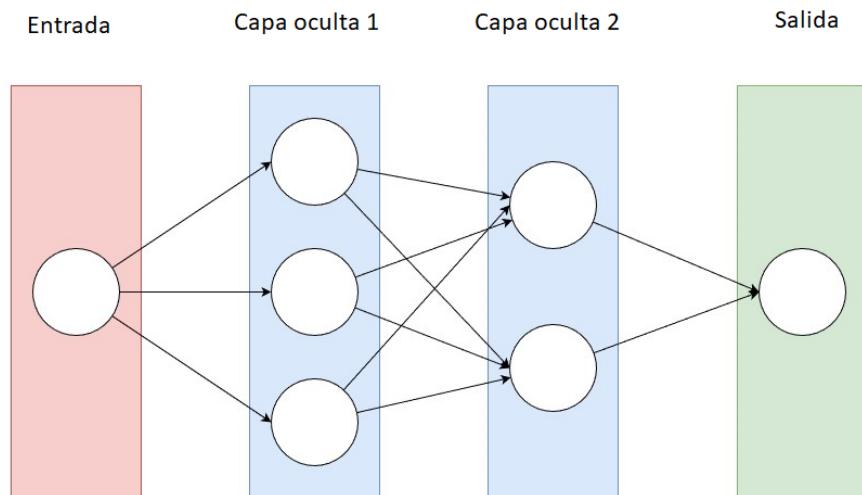


Figura 3.1: Funcionamiento de una red neuronal

El valor calculado se utiliza en una función de activación, que produce el valor de salida[16]. Entre las funciones de activación destacan las visibles en 3.2.

- La función RELU (verde), la cual indica el máximo entre 0 y el valor calculado [3.2a](#).
- La función lineal(rosa), la cual devuelve el mismo valor calculado [3.2b](#).
- La función sigmoidea(cian), la cual devuelve un valor entre 0 y 1 según lo alejado que este del 0 [3.2c](#).
- La función escalón(morada), devuelve o 0 o 1 dependiendo del signo del valor calculado [3.2d](#).

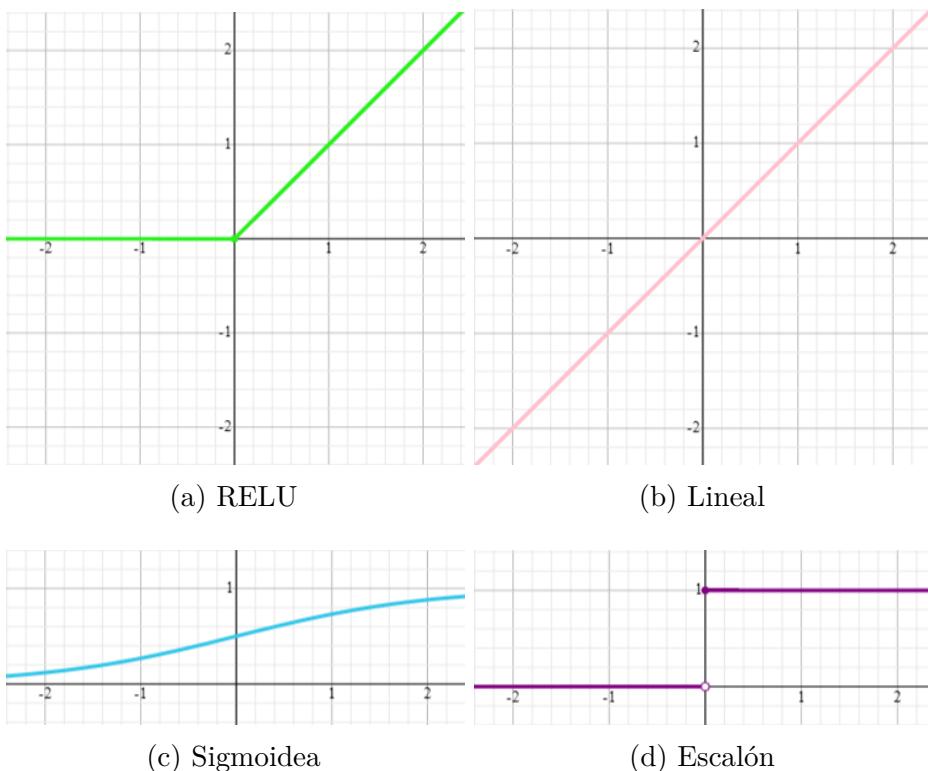


Figura 3.2: Funciones de activación

Para hacer más visual una red neuronal, se hará un ejemplo, suponemos una asignatura, donde un alumno realiza 2 exámenes, 2 trabajos prácticos y 1 trabajo final. La nota final de la asignatura se corresponde a un 50 % de los exámenes, donde el primer examen vale un 40 % y el segundo un 60 %, un 30 % a los trabajos prácticos, donde ambos tienen el mismo porcentaje, y el trabajo final que vale un 20 %.

El profesor de la asignatura quiere saber si un alumno ha aprobado o suspendido, con las siguientes notas: un 8 en el primer examen, un 1 en el segundo, en ambos trabajos prácticos un 5 y en el trabajo final un 6.

En la figura 3.3 se observa la exemplificación de la red, donde la capa roja se corresponde a la capa de entrada, la capa azul con la capa oculta de la red, y la capa verde con la capa de salida. En este caso, en la capa oculta se ha usado una función de activación RELU, obteniendo los valores de dentro de los círculos, y en la capa de salida, se utiliza una función escalón, con un bias de -5, en este caso, el alumno habría obtenido una nota de 4,6, como la suma de $-5 + 4,6$ es menor que 0, lo que significaría que el alumno ha suspendido.

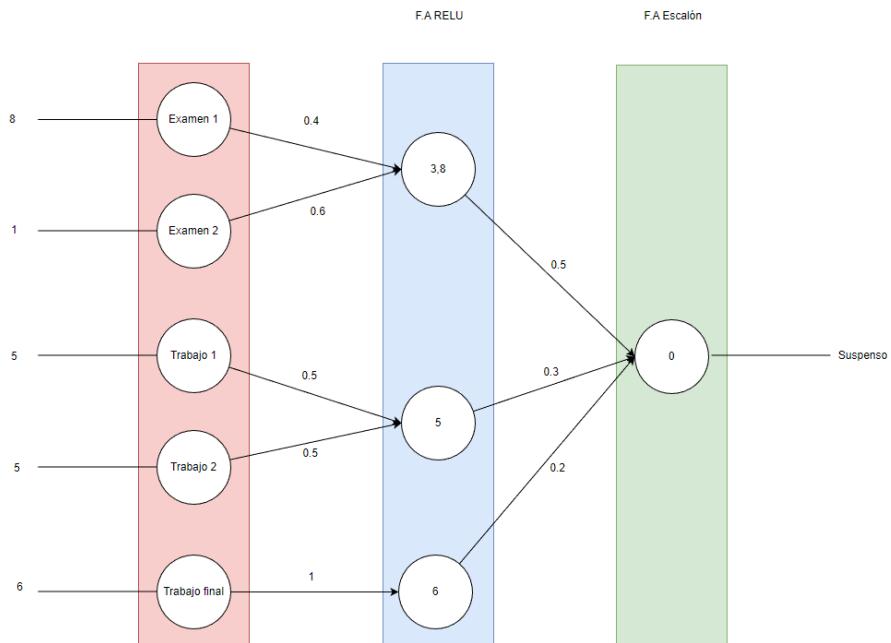


Figura 3.3: Ejemplo de una red neuronal

Dentro de las redes neuronales, se encuentran las redes neuronales convolucionales (CNN), estas redes son las que poseen al menos una capa convolucional, consistiendo en la combinación de capas convolucionales, capas de agrupación y capas densas [15].

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 [13].

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) se calcula la suma de los valores de esta submatriz asignando el resultado a la matriz de salida [14].

La capa de agrupación es una capa de una red neuronal que reduce el tamaño de la matriz de entrada, mediante operaciones como el máximo o la media de los valores de una submatriz [21].

La capa densa también llamada capa totalmente conectada, es una capa oculta, donde cada nodo está conectado a todos los nodos de la siguiente capa oculta [18].

Una red neuronal residual es aquella que permite el salto de capas intermedias. Estas conexiones permiten que la red aprenda las características residuales, es decir, las diferencias entre las entradas y las salidas esperadas. Para ello, se crea un bloque residual, que en lugar de transmitir la salida de una capa directamente a la siguiente, se introduce una conexión residual que suma la salida de una capa a la entrada de otra capa posterior como se puede ver en 3.4.

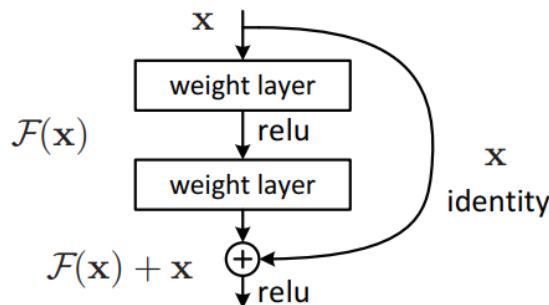


Figura 3.4: Bloque residual. Imagen extraída de [22].

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), está en un formato BGR(Azul, verde y rojo), donde la diferencia entre estos formatos es el orden en el que se encuentran los colores, y en el preprocesado, también es necesario que los valores finales estén centralizados en 0. Además de este cambio, la imagen debe tener de 224 x 224 píxeles, redimensionando las imágenes que se introduzcan a la red. [39] 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 ha implementado este preprocesamiento. La estructura de la red convolucional VGG16, se puede observar en la figura 3.5, donde hay 13 capas convolucionales, y 4 de agrupación.

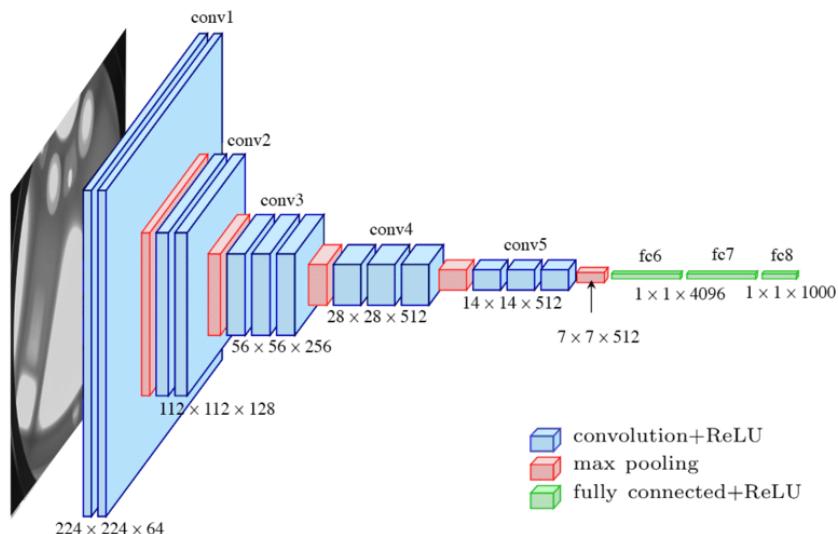


Figura 3.5: Estructura básica de la red convolucional VGG16. Imagen extraída de [11]

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 3.4.

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] [38].

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

Donde la formula para cambiar este valor vendría dada por:

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

- 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 3.6, en esta red neuronal convolucional, hay 50 capas repartidas entre 5 bloques. El primer bloque no es convolucional, pero los otros 4 sí que contienen capas convolucionales de tamaños variados, incluyendo convoluciones de 1x1, 3x3 y 1x1.

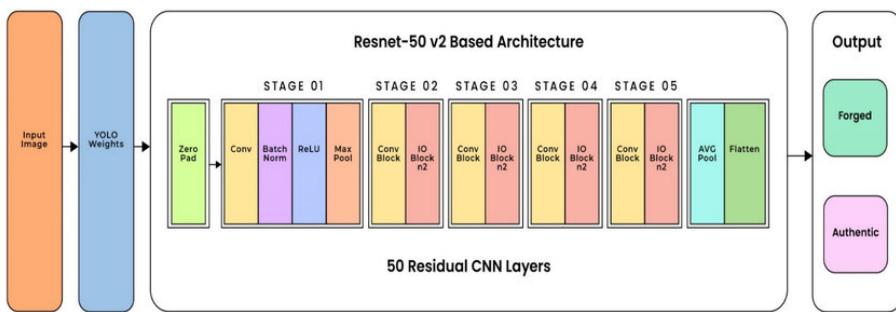


Figura 3.6: Estructura básica de la red convolucional ResNet50v2. Imagen extraída de [29]

3.3. Transfer learning

El aprendizaje transferido o “Transfer learning” es una técnica del *Machine learning*, la cual consiste en reutilizar el conocimiento ya aprendido de un modelo, en otra actividad que tenga un objetivo similar.

En el ámbito que nos concierne, se ha utilizado el modelo VGG-16 que ha sido entrenado con el dataset de ImageNet “Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)”. El objetivo de este modelo es clasificar imágenes entre 1.000 categorías distintas, aprendiendo las características visuales de cada una.

Para transferir el modelo, es necesario congelar las capas, para que no se entrenen. Con esto realizado, solo falta obtener el resultado, que puede ser binario o multiclase dependiendo del problema.

Además, se ha agregado una capa de agrupación para hacer uso de las capas densas. La primera capa densa consta de 64 neuronas con una función de activación RELU. Por otro lado, la segunda capa densa actúa como la capa de salida, y su propósito es evaluar la calidad de la imagen.

El código de creación del modelo sería el siguiente:

```
vgg16 = VGG16(include_top=False, weights='imagenet',
               input_shape=(224, 224, 3))
for layer in vgg16.layers:
    layer.trainable = False
model = Sequential()
model.add(vgg16)
model.add(tensorflow.keras.layers.Flatten())
model.add(tensorflow.keras.layers.Dense(64,
                                         activation='relu'))
model.add(tensorflow.keras.layers.Dense(num_classes,
                                         activation='sigmoid'))
```

3.4. 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 [35].

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)
```

3.5. 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 3.7 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.

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

		Valores reales	
		Positivos	Negativos
Valores obtenidos	Positivos	Verdaderos positivos	Falsos positivos
	Negativos	Falsos negativos	Verdaderos negativos

Figura 3.7: 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.

$$\text{Recall} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}}$$

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

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

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 3.7, 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.

3.6. Guías de diseño Android

Para realizar la aplicación se siguió la guía de diseño de Google Material3.

En los botones, se recomienda la forma circular en las esquinas, de forma que sea más visible para los ojos. En la figura 3.8 se puede ver la comparación entre las esquinas con pico y redondeadas, en esta explica que las formas con esquinas hacen que el enfoque esté fuera del rectángulo, mientras que en las redondeadas consigue que el enfoque esté dentro.

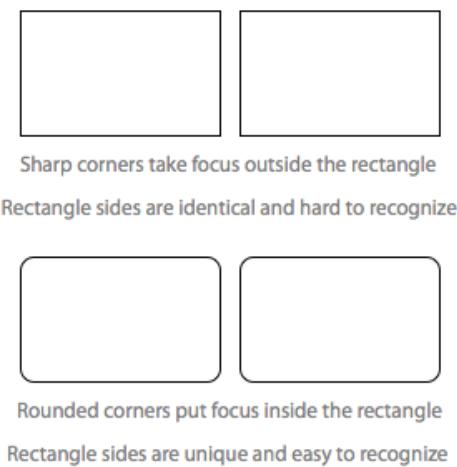


Figura 3.8: Diferencias esquinas. Imagen obtenida de [25]

Para los colores de la aplicación, como es una aplicación orientada en el bienestar y salud de las personas, estos colores tenían que ser claros con tonos verdes y azules, colores usados también en el diseño del logo de la aplicación, por otro lado, hay personas que prefieren un fondo oscuro a uno claro, por lo que se incluyó la opción de modo oscuro en la aplicación.

La gamma de color usada fue: en modo claro, gris claro para el fondo #E0E0E0 y los botones azules claros #90CAF9, y para el modo oscuro, gris oscuro para el fondo #353740 y los botones verdes claros. Para el color del texto, varía entre blanco y negro según el fondo. Buscando en todo momento contraste entre los objetos y el fondo de la aplicación.

Durante la ejecución de la aplicación, es preferible que los distintos objetos estén deshabilitados, a invisibles, de esta forma, el usuario entiende que si interactúa con la aplicación, ese objeto se habilitará. Como es el caso de la realización de la foto, que hasta que el usuario no haya seleccionado una imagen no puede pasar a la siguiente pantalla. Esto es contraproducente en algunos casos, puesto que los mensajes de error es mejor mostrarlos solo cuando el error ocurre; como es el caso de iniciar sesión incorrectamente o el caso de que la calidad de la imagen sea baja.

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 está 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* [42], 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 está 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 ícono actual.

Entorno de desarrollo integrado (IDE)

Para el desarrollo de la aplicación móvil, se pensó en [Eclipse](#), [Android Studio](#) y [Unity](#).

Al final, se descartó Eclipse por no ofrecer una experiencia de desarrollo específica 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 estándar 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 librería 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 automático, 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.train_test_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 está 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.

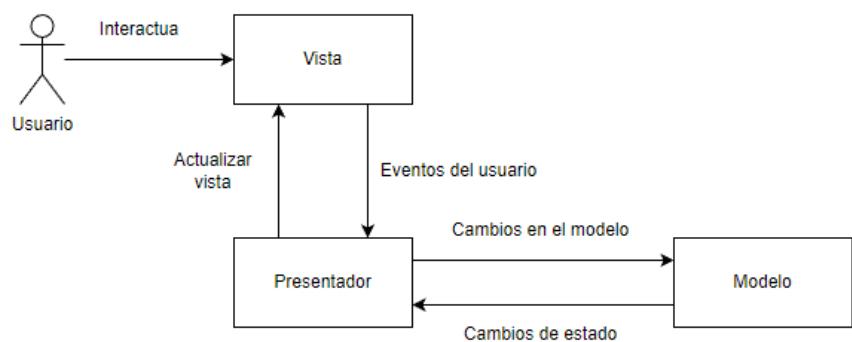


Figura 4.1: Modelo-Vista-Presentador

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 público.

En este inicio, empecé a imaginarme como podría ser la aplicación haciendo bocetos mentales de las interfaces, como era muy efímero, se hizo una búsqueda de aplicaciones similares, donde se encontró la aplicación Ret-iN CaM. Con una idea más formada, se hicieron los bocetos iniciales de la aplicación.

5.2. 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 Kanban, tanto físicamente como con la herramienta ZenHub. Aunque con el paso de los sprints, se dejó 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.

5.3. Formación

Para el proyecto no fue necesario el aprendizaje de un nuevo lenguaje, puesto que para la aplicación de Android Studio se usó el lenguaje Java, visto durante la carrera y para la creación de la red neuronal se usó el lenguaje Python, que también se ha visto.

Pero, sí que ha sido necesario el uso de tutoriales, guías tanto de Android Studio y como keras y tensorflow para implementar correctamente el código.

Para la utilización de la red VGG-16, se leyó el artículo: *Very Deep Convolutional Networks for Large-Scale Image Recognition*(Karen Simonyan y Andrew Zisserman) [30].

Además han surgido otros errores, tanto de programación, como de incompatibilidad entre otras cosas; que fueron solucionados por la comunidad de stackoverflow y de YouTube.

5.4. Conjunto de datos

El conjunto de datos utilizado para la creación del modelo, es un conjunto de datos real, lo que significa que las imágenes no se han creado ni artificialmente; normalmente esta idea se asocia a ensayos clínicos aleatorios, donde primero se desidentifica la imagen obtenida del paciente.

Existen varios problemas debido al conjunto de datos real, entre ellos, se han observado el desbalanceo de datos, que es debido a que hay una disparidad en la cantidad de imágenes de un tipo y de otro, esto puede producir que el modelo cree patrones donde no los hay. Otro problema, es la calidad de la imagen, debido a que estas imágenes no tienen porque realizarse en el mismo hospital, puede haber variaciones en el procedimiento del retinógrafo.

En la figura 5.1 se pueden observar imágenes del dataset con una calidad mala, ya sea porque se ha realizado mal la retinografía, o porque el retinógrafo fuese defectuoso.

Por otro lado, en la figura 5.2 se pueden observar imágenes de buena calidad. Donde en la imagen de la izquierda, el paciente tiene un grado alto de retinopatía diabética y la imagen de la derecha no tiene ningún grado.

5.5. Creación del modelo para detectar la calidad de imagen

Para crear un modelo keras en python, se usó la guía que viene en TensorFlow para [clasificación de imágenes](#)[36] utilizando un modelo VGG-16.

El conjunto de datos se divide en 2 partes, el conjunto de imágenes, y un csv con el nombre de la imagen y la calidad que tiene esta. La calidad

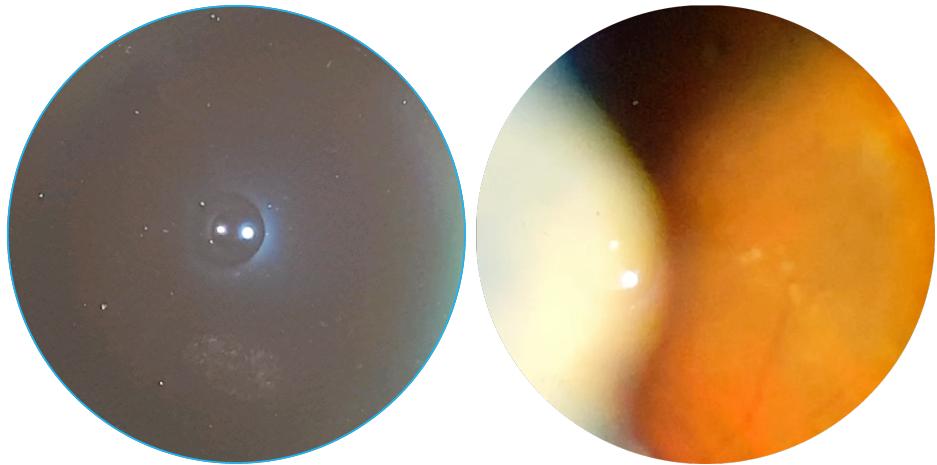


Figura 5.1: Imágenes con una calidad mala



Figura 5.2: Imágenes con una calidad buena

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 desbalanceados.

Por ello se usan las medidas precisión y recall. Para comprobar que modelo es mejor, se ha representado la matriz de confusión y se ha calculado 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 en la creación del modelo es la tasa de aprendizaje. 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 5.3, se puede observar el sobreajuste y el infrajuste, que son los resultados posibles al escoger una tasa de aprendizaje alta o baja.

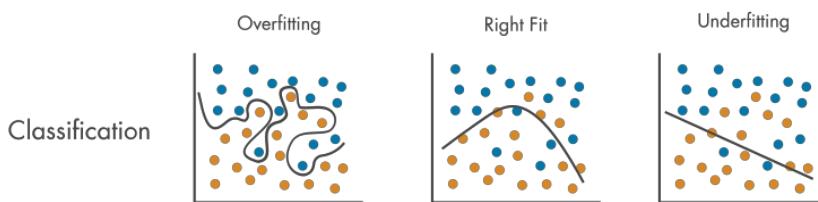


Figura 5.3: Comparación de sobreajuste e infrajuste. Imagen adaptada de [24]

Además, en un principio, se quería clasificación binaria con clasificación multiclase. Cuando se compararon estos valores, se utilizaba la medida “accuracy”, que como ya se ha explicado no es recomendable en clasificación binaria. Durante la integración de la matriz de confusión en la clasificación multiclase, no se logró una buena visualización, y por ello no se hizo esta comparación con F1-Score.

En la gráfica 5.4, se observa el historial del mejor caso observado para la clasificación binaria.

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 5.5, 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:

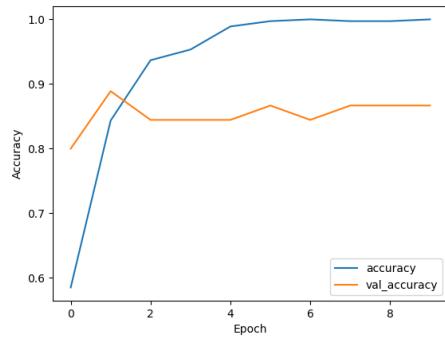


Figura 5.4: Historial de la evolución de la “accuracy” de un modelo cuyo resultado de test era de 91 %

$$Precision = \frac{11}{11 + 2} = 0,85$$

$$Recall = \frac{11}{11 + 0} = 1$$

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

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.

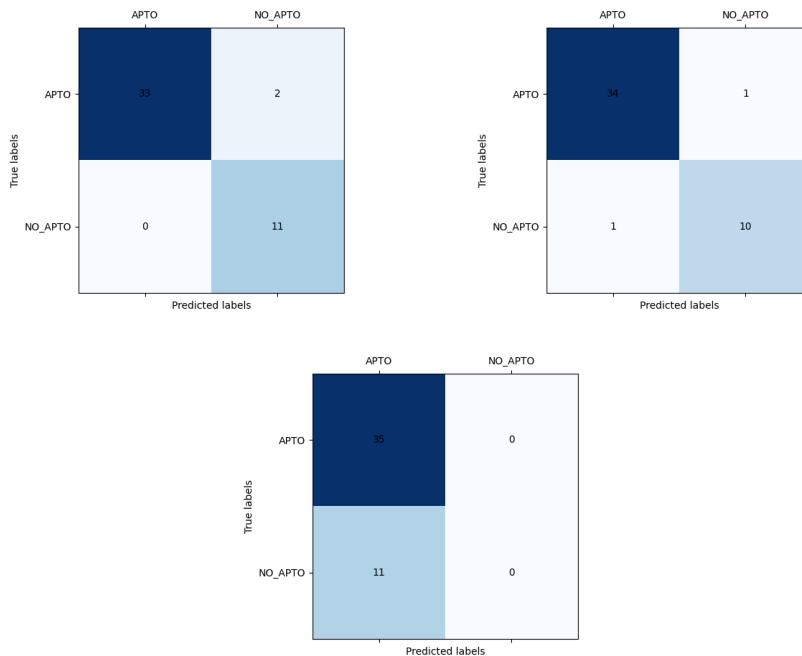


Figura 5.5: Comparación de los modelos destacables

5.6. 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 tenía que crear un proyecto, donde se seleccionó como API la versión 21, que como informa Android Studio, se puede ejecutar en el 99,5 % de dispositivos [5.6](#).

Después de terminar el boceto igual al realizado manualmente, ([Issue #7](#)), se realizaron cambios en la interfaz, añadiendo un modo oscuro, la posibilidad de entrar como invitado, 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

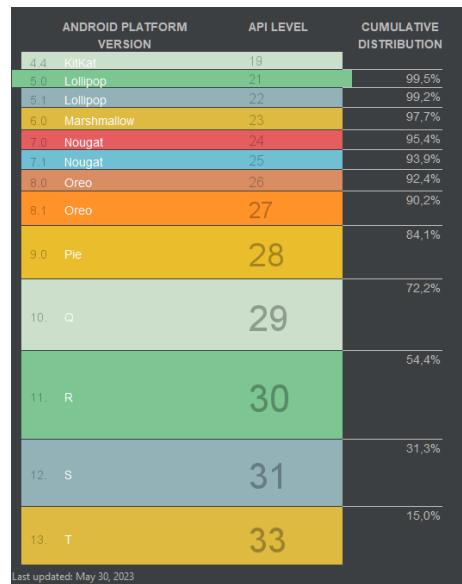


Figura 5.6: Porcentaje de dispositivos según la API. Imagen obtenida de Android Studio.

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 añadió 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 pedían 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.

Cámara en Android Studio

La primera implementación en la que se necesito una guía, fue en el uso de la cámara en una aplicación, guardando posteriormente la imagen obtenida.

Para ello, primero se necesitaba inicializar la cámara, que se obtuvo una referencia de la propia página de Android [7].

Una vez implementado, surgió un problema debido a los permisos de la aplicación, puesto que no se abría la cámara. Buscando como implementar la petición de los permisos de la cámara durante la ejecución; esta referencia también se obtuvo de la propia Android [9].

Ya abierta la cámara, se tenía que guardar la imagen realizada, puesto que se hacia la foto, pero no se mostraba al usuario ni se guardaba de forma interna. Llegando otra vez a la página de Android, donde en un ejemplo se podía ver como se implementaba [8].

Con la imagen obtenida de forma local, solo había que mostrarla al usuario tarea que ya se sabía realizar.

Galería en Android Studio

Al igual que en la cámara, no se tenían los conocimientos para implementar este requisito, pero como la cámara se implementó anteriormente, se utilizó el mismo enlace para obtener la imagen seleccionada. [8].

Para llamar a esta actividad, se usó la respuesta de un foro de [stackoverflow](#) [28], 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 que a partir de la API 30, el permiso para leer almacenamiento externo no es necesario [10].

Implementación de las redes neuronales usando TensorFlow Lite

Para integrar las redes neuronales se recomiendan implementar en formato TensorFlow Lite, por lo que era necesario una conversión del modelo, puesto que inicialmente tenían formato keras. Por lo que para integrar las redes neuronales se buscó información de las siguientes páginas:

- Cargar un modelo TensorFlow Lite en Android [34]
- Convertir modelo keras en modelo TensorFlow js [33]. Con esta idea, se usó la idea para convertir en formato TensorFlow Lite.
- Crear objeto Interprete [37].
- Preprocesamiento modelo VGG16 [39]
- Preprocesamiento modelo ResNet50V2 [38]

Con la información obtenida de los enlaces anteriores, se observó que para ejecutar un modelo era necesario abrir un Interprete con el cual se ejecuta el modelo, que es necesario hacer un conjunto de transformaciones .

Esta transformación de una cadena de texto a byteBuffer se hace de esta forma:

```
AssetFileDescriptor fileDescriptor =
    context.getAssets().openFd(fileName);
FileInputStream inputStream = new
    FileInputStream(fileDescriptor.getFileDescriptor());
```

```
FileChannel fileChannel = inputStream.getChannel();
long startOffset = fileDescriptor.getStartOffset();
long declaredLength = fileDescriptor.getDeclaredLength();
MappedByteBuffer mappedByteBuffer =
    fileChannel.map(FileChannel.MapMode.READ_ONLY,
    startOffset, declaredLength);
```

Con el modelo ya cargado, se necesita crear el *input*, donde se encuentra la imagen preprocessada y el *output*, que es una variable donde se guardan los resultados de la ejecución del modelo. El interprete se ejecuta con el comando run, junto con los atributos mencionados.

Para probar si el interprete funcionaba, se hizo un modelo básico que calculaba la operación XOR en la [Issue #21](#).

Posteriormente, se usó la red publicada por Lorenzo Baraldi [1] para probar los resultados con imágenes. Para introducir esta red neuronal se necesita el preprocessamiento de los datos y para su ejecución se necesita saber el número de salidas de la red; esta red está destinada a el conjunto de datos “Imagenet” [43] lo que significa que busca clasificar entre 1000 categorías distintas. La issue relacionada a esta actividad, es [Issue #24](#). Además, en esa tarea, se pensó que algunas redes neuronales tardan más de lo esperado en ejecutarse, y para que el usuario no tenga que esperar a que se termine de ejecutar, se implementó un hilo con las actividades de la red neuronal, permitiendo que se ejecute en segundo plano y se almacene 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. Añadiendo el preprocessamiento correspondiente. Esta Issue se corresponde con [Issue #41](#). Sufriendo pequeños cambios en futuras releases.

Los resultados de esta red pueden ser:

- 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.

Por último, se añadió la red de calidad, la cual es una red convolucional VGG-16. Añadiendo su preprocesamiento correspondiente que ya se había realizado anteriormente. Esta Issue se corresponde con [Issue #42](#). Sufriendo pequeños cambios en futuras releases.

Los resultados de esta red pueden ser:

- 0 indica que la foto tiene una calidad mala.
- 1 indica que la foto tiene una calidad buena.

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 sacó 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

D-EYE 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
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 está destinada a médicos de la sanidad pública, 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 diagnóstico 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

En este apartado, se desarrolla las conclusiones obtenidas del proyecto realizado y los futuros usos o cambios a añadir al proyecto.

7.1. Conclusiones

A continuación se muestra una lista de las conclusiones que se pueden obtener de este trabajo:

- Se han cumplido muchos de los objetivos planteados al inicio del documento, a excepción del objetivo de agilizar el sistema sanitario, puesto que no se ha distribuido la aplicación, y en el caso de haberlo hecho, y que el sistema sanitario español quiera esta aplicación, la agilización no se podría percibir hasta pasado un tiempo.
- El uso de Android Studio ha aportado facilidades en el desarrollo de la aplicación, donde la documentación ha sido un factor clave. Por otro lado, el intento de que la aplicación pueda ejecutarse en el máximo número de dispositivos, ha conseguido que la aplicación no esté optimizada, con herramientas específicas de ultimas versiones.
- Muchos de los conceptos vistos en el proyecto, se han visto durante la carrera, profundizando conceptos como las redes neuronales, la interacción con la aplicación, bases de datos,...

- Antes de la realización de este proyecto, no se miraba la documentación de las herramientas utilizadas, y en la mayoría de casos, ayudan a resolver las dudas que se tienen.
- La utilización de una metodología ágil en este proyecto no se ha notado, puesto que no hay cliente y no se cambian los requisitos a lo largo del tiempo, ocurre lo mismo con el SCRUM master, como no hay un equipo como tal, siendo todos los actores la misma persona, todas las reuniones se han omitido y además, la estimación de las tareas es complicada, porque no se tiene un conocimiento previo de las tareas, y porque es un proyecto individual, es decir, con más personas se podría estimar por media. Por tanto, no creo que sea recomendable esta metodología para este tipo de proyectos individuales.

Finalmente, se ha conseguido una aplicación funcional que permite a los médicos, o al personal identificado, obtener resultados sobre la imagen capturada. Además, se ha añadido el modelo que detecta si una imagen tiene una calidad correcta, para que de esta forma, no se generen expedientes con una calidad mala, donde el resultado puede no ser el real.

En este caso, el usuario no tiene que ser experto en aplicaciones Android, ni en redes neuronales; donde se deja elegir al médico la red neuronal, se debería explicar en qué ocasiones es mejor cada una.

7.2. Líneas de trabajo futuras

El origen de la aplicación surge de la utilización de ésta en el sistema sanitario público, por lo que se propone utilizar esta aplicación como base sobre la que trabajar.

Base de datos

En primer lugar, será necesario cambiar la base de datos. Esto es así, porque SQLite permite crear bases de datos locales, y en caso de que se quiera acceder a un expediente creado desde otro dispositivo móvil, no existiría. Por este motivo, se debería conectar la aplicación a una base de datos online.

Redes neuronales convolucionales

Otro posible cambio, son las redes neuronales, en caso de obtener nuevas redes neuronales más precisas, o que consuman menos recursos, se pueden: o añadir (cambiando la vista de la interfaz, añadiendo una nueva opción para el modelo) o también se podría cambiar una de las actuales (borrando el modelo anterior, y añadiendo el nuevo).

Además, en un futuro, se aumentará el conjunto de datos, permitiendo a las redes neuronales

Añadir versiones Android

Actualmente, la aplicación se puede ejecutar desde dispositivos Android con una API mayor a 21, se podría aumentar el número de dispositivos disponibles, y por otro lado, es posible que en estos dispositivos, al tener distinta resolución que los dispositivos actuales, sea necesario ajustar el contexto xml dinámicamente.

Aumentar el número de sistemas operativos

El proyecto ha sido creado para Android. Hay que aumentar el número de dispositivos, para que los usuarios no se vean obligados a utilizar un teléfono Android. Siendo la opción más recomendable añadir una versión para iOS.

Distribución

Actualmente, no se ha implementado una opción para distribuir la aplicación y que el usuario pueda descargarla.

Se recomienda el uso de Play Store, puesto que es la distribuidora oficial de Android y la más usada por los usuarios de Android.

Uso de la aplicación por el paciente

La aplicación ha sido desarrollada con el objetivo de ser usada por los médicos. En un futuro, se podría dar soporte también a los pacientes, donde podrían ver los informes, o incluso en caso de tener el hardware hacer la propia toma proporcionándole una guía de cómo hacerlo correctamente.

Idiomas

Como el proyecto surgió con la idea de ser utilizado en el servicio sanitario público español, el único idioma implementado ha sido el castellano; en un futuro se podrían llegar a añadir las lenguas oficiales como catalán, gallego y euskera.

En caso de querer internacionalizar la aplicación, añadir en cualquier caso el inglés, y posteriormente, los idiomas oficiales de los países donde se vaya a usar la aplicación.

Actualizaciones

Como la aplicación trata sobre la salud de las personas, un tema que pertenece al grupo de categorías especiales de datos personales según el Reglamento General de Protección de Datos; es importante que todas las actualizaciones comprueben que no se pueden filtrar los datos; y en caso de que pueda ser así, sacar una nueva actualización rápidamente.

Tests unitarios

Como durante el proyecto se valoraba más la interacción del usuario con la aplicación, se realizaron test manuales. De esta forma, se podrían implementar tests automáticos para comprobar el correcto funcionamiento.

Bibliografía

- [1] Lorenzo Baraldi. Vgg-16 pre-trained model for keras, 2016. URL <https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3>. [Online; Accedido 9-junio-2023].
- [2] S.L. Canela Tech. Ret in cam - app store, 2023. URL <https://apps.apple.com/us/app/ret-in-cam/id1509765945>. [Online; Accedido 26-junio-2023].
- [3] Junta de Castilla y León. Cita previa - google play, 2023. URL <https://play.google.com/store/apps/details?id=com.citaprevia.citaprevia>. [Online; Accedido 26-junio-2023].
- [4] 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].
- [5] Organización Mundial de la Salud. Diabetes, 2022. URL <https://www.who.int/es/news-room/fact-sheets/detail/diabetes>. [Online; Accedido 18-febrero-2023].
- [6] SegurCaixa Adeslas SA de Seguros y Reaseguros. Adeslas salud y bienestar, 2023. URL <https://play.google.com/store/apps/details?id=com.qaracter.asybapp>. [Online; Accedido 26-junio-2023].
- [7] Android Developers. Cómo usar la cámara - android developers, 2023. URL <https://developer.android.com/training/camera/camera-intents>. [Online; Accedido 12-junio-2023].

- [8] 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].
- [9] Android Developers. Cómo solicitar permisos - android developers, 2023. URL <https://developer.android.com/training/permissions/requesting?hl=es-419>. [Online; Accedido 12-junio-2023].
- [10] 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].
- [11] 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.
- [12] Centers for Disease Control and Prevention (CDC). Blindness caused by diabetes-massachusetts, 1987-1994. *MMWR Morb Mortal Wkly Rep.*, 1996.
- [13] Google. Convolutional layer, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#convolutional_layer. [Online; Accedido 24-mayo-2023].
- [14] Google. Convolutional operation, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#convolutional_operation. [Online; Accedido 24-mayo-2023].
- [15] Google. Convolutional neural network, 2023. URL <https://developers.google.com/machine-learning/glossary?hl=es-419#convolutional-neural-network>. [Online; Accedido 24-mayo-2023].
- [16] Google. activation function, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#activation_function. [Online; Accedido 12-junio-2023].
- [17] Google. artificial intelligence, 2023. URL <https://developers.google.com/machine-learning/glossary?hl=es-419#artificial-intelligence>. [Online; Accedido 12-junio-2023].

- [18] Google. fully connected layer, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#fully_connected_layer. [Online; Accedido 24-mayo-2023].
- [19] Google. hidden layer, 2023. URL https://developers.google.com/machine-learning/glossary?hl=es-419#hidden_layer. [Online; Accedido 12-junio-2023].
- [20] Google. neural network, 2023. URL <https://developers.google.com/machine-learning/glossary?hl=es-419#neural-network>. [Online; Accedido 12-junio-2023].
- [21] Google. Pooling, 2023. URL <https://developers.google.com/machine-learning/glossary?hl=es-419#pooling>. [Online; Accedido 24-mayo-2023].
- [22] 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>.
- [23] Google LLC. Google fit - google play, 2023. URL <https://play.google.com/store/apps/details?id=com.google.android.apps.fitness>. [Online; Accedido 26-junio-2023].
- [24] MathWorks. Understanding overfitting in machine learning, 2023. URL <https://www.mathworks.com/discovery/overfitting.html>. [Online; Accedido 10-junio-2023].
- [25] UX Movement. Why rounded corners are easier on the eyes, 2011. URL <https://uxmovement.com/thinking/why-rounded-corners-are-easier-on-the-eyes/>. [Online; Accedido 30-junio-2023].
- [26] Cecelia Koetting OD, FAAO. The four stages of diabetic retinopathy. *Modern Optometry*, 2019.
- [27] World Health Organization. Who info app - google play, 2023. URL <https://play.google.com/store/apps/details?id=org.who.infoapp>. [Online; Accedido 26-junio-2023].
- [28] 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].

- [29] 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.
- [30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.
- [31] D-Eye Srl. D-eye care features, 2023. URL https://d-eyecare.com/en_ES/product#features. [Online; Accedido 26-junio-2023].
- [32] StatCounter Global Stats. Global mobile os market share, 2023. URL <https://gs.statcounter.com/os-market-share/mobile/worldwide>. [Online; Accedido 26-junio-2023].
- [33] 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].
- [34] 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].
- [35] TensorFlow. TfLiteconverter tensorflow lite api, 2022. URL https://www.tensorflow.org/lite/api_docs/python/tf/lite/TFLiteConverter. [Online; Accedido 27-mayo-2023].
- [36] 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].
- [37] 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].
- [38] TensorFlow. Resnet50v2, 2023. URL https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet_v2/ResNet50V2. [Online; Accedido 27-mayo-2023].
- [39] TensorFlow. Vgg16, 2023. URL https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16. [Online; Accedido 27-mayo-2023].

- [40] David Turbert. La enfermedad ocular diabética, 2022. URL <https://www.aao.org/salud-ocular/enfermedades/la-enfermedad-ocular-diabetica>. [Online; Accedido 20-mayo-2023].
- [41] Wikipedia. Machine learning, 2023. URL https://en.wikipedia.org/wiki/Machine_learning. [Online; Accedido 12-junio-2023].
- [42] Wikipedia. Rubber duck debugging, 2023. URL https://en.wikipedia.org/wiki/Rubber_duck_debugging. [Online; Accedido 10-junio-2023].
- [43] yrevar. imagenet 1000 class idx to human readable labels, 2019. URL <https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a>. [Online; Accedido 9-junio-2023].