Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

APP PARA LA LOCALIZACIÓN DE PERSONAS A PARTIR DEL RECONOCIMIENTO AUTOMÁTICO DE IMÁGENES

Autor: Carlos Tejeda Martínez

Tutor: Alfredo Gardel Vicente

Septiembre 2022

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

App para la localización de personas a partir del reconocimiento automático de imágenes

Autor: Carlos Tejeda Martínez

Tutor: Alfredo Gardel Vicente

Tribunal:

Presidente: María Soledad Escudero Hernanz

Vocal 1º: Juan Manuel Miguel Jiménez

Vocal 2º: Alfredo Gardel Vicente

Septiembre de 2022

Agradecimientos

Quiero agradecer en primer lugar a mi apoyo incondicional, mi familia, la que siempre ha estado apoyándome y ayudándome siempre que han podido independientemente de la dificultad en la que me encontrase.

También agradecer a los grandes compañeros que se han convertido en importantes amigos encontrados durante este trayecto, con los que he estado codo a codo sacando el trabajo adelante durante estos últimos años y disfrutando grandes momentos.

Por supuesto agradecer a mi tutor, Alfredo Gardel, quien con su atención y asesoramiento me ha ayudado en la elaboración del siguiente TFG.

Por último, agradecer a la institución del Museo Provincial de Guadalajara permitirme usar su espacio como entorno experimental del siguiente TFG, hecho fundamental en la consecución del mismo.

Resumen

Este TFG recoge el desarrollo de una solución Android para la localización de personas mediante el reconocimiento automático de imágenes por parte de un modelo de inteligencia artificial. Para obtenerse se combina el uso de una variedad de distintas herramientas, que aúnan los conceptos de inteligencia artificial y localización a lo largo del proyecto.

Los principales puntos del proyecto son la obtención de un clasificador de imágenes para un proceso de inferencia que debe ofrecer resultados que se puedan vincular a POIs para ofrecer una localización al usuario. Y como segundo punto el desarrollo de una aplicación móvil con S.O. Android capaz de ofrecer una interfaz sencilla e intuitiva al usuario, con capacidad de mostrar mapas y de alojar el modelo de inferencia.

Como fase final se realiza un experimento en un entorno real, enfrentando dos enfoques de diseñar la solución para concluir los resultados y alcance del TFG.

Palabras clave: Clasificador de imágenes, Android, TensorFlow, Mapbox

Summary

This TFG covers the development of an Android solution for the localisation of people through the automatic recognition of images by an artificial intelligence model. This is achieved by combining the use of a variety of different tools, which combine the concepts of artificial intelligence and localisation throughout the project.

The main points of the project are to obtain an image classifier for an inference process that should provide results that can be linked to POIs to provide a location to the user. The second point is the development of a mobile application with Android O.S. capable of offering a simple and intuitive interface to the user, with the capacity to display maps and to host the inference model.

As a final phase, an experiment is carried out in a real environment, confronting two approaches to design the solution to conclude the results and scope of the TFG.

Key words: Image classifier, Android, TensorFlow, Mapbox

ACRONIMOS

API (**A**pplication **P**rogramming **I**nterfaces): Software destinado al desarrollo de la programación que ofrece una biblioteca interponiendo una capa de abstracción entre el usuario y la misma.

AVD (**A**ndroid **V**irtual **D**evice): Herramienta del IDE Android Studio que replica un dispositivo móvil con el fin de efectuar las pruebas de desarrollo.

BLE (**B**luetooth **L**ow **E**nergy): Tecnología puntera en la localización en interiores basada en balizas, haciendo uso del protocolo Bluetooth y baterías de larga duración.

GPS (**G**lobal **P**osition **S**ystem): Sistema de posicionamiento global basado en la trilateración con los satélites que orbitan la Tierra.

IA (**I**nteligencia **A**rtificial): Rama de la ciencia de la computación que se encarga del desarrollo y refinamiento de algoritmos con el fin de dotar a las máquinas de capacidades similares a las de los seres humanos, como lo es la inteligencia.

IDE (**I**ntegrated **D**evelopment **E**nvironment): Plataforma software para el soporte de desarrollos software por parte de los programadores con herramientas y servicios integrados.

POI (**P**oint **O**f **I**nterest): Punto de interés para denotar una ubicación con algún significado.

QR (**Q**uick **R**esponse): Tipo de código para recopilar información o enlaces a mayores fuentes de información, de forma accesible mediante un escaneado del mismo código.

RFID (**R**adio **F**requency **ID**entification): Tipo de código que transmite información o enlaces a mayore fuentes de información, mediante la irradiación de onda de radio de corto alcance.

RGB (**R**ed **G**reen **B**lue): Codificación de color basada en los 3 colores primarios de los que recibe su nombre: rojo, verde y azul.

SDK (**S**oftware **D**evelopment **K**it): Conjunto de software con el fin de proporcionar herramientas para el desarrollo que ofrece la propia plataforma donde se está trabajando.

SO (**S**istema **O**perativo): Software que coordina los recursos hardware y da soporte a los programas para su funcionamiento.

TFG (**T**rabajo de **F**in de **G**rado): Proyecto para la finalización de un estudio universitario de grado.

Tabla de contenido

TABLA DE FIGURAS

1 INTRODUCCIÓN

Desde el inicio de los tiempos la localización ha sido un problema que ha preocupado a la especie humana. Ubicarnos en el entorno significa poder realizar tareas tan simples como trazar un camino para volver a casa cuando salimos a hacer recados; o tareas tan complejas como emprender una travesía oceánica como un buque mercante realiza en la actualidad.

Con el paso de los años el ser humano ha desarrollado diversas soluciones para orientarse y conocer la ubicación donde se encuentran una persona o un sistema de interés. Esto ha sido posible a través del uso de herramientas como los mapas, astrolabios, dispositivos de posicionamiento global basados en satélites como GPS…siendo esta última la opción más extendida en la actualidad gracias al conjunto de satélites que orbitan la Tierra. Ejemplo de ello son servicios de seguimiento, localizadores, dispositivos con aplicaciones que pueden interactuar con la información que suministra este servicio como lo es la famosa aplicación de Google, Google Maps.

Esta solución predominante en ocasiones no tiene el efecto esperado cuando los dispositivos que utilizan esta tecnología se encuentran en espacios cerrados, debido a que la composición de la estructura donde se encuentra el usuario impide la recepción de señal necesaria para determinar la localización. Ante esta situación hay que explorar alternativas para localizarse en entornos cerrados, debido a la naturaleza de dicha solución. En este Trabajo de Fin de Grado se desea desarrollar una aplicación que localice al usuario en un entorno cerrado en el cual no disponemos de acceso a la ubicación por GPS.

Algunas de estas alternativas para localización en interiores pasan por la localización mediante balizas activas. Estos dispositivos emisores requieren de la instalación de transmisores inalámbricos que emiten diferentes tipos de señal en un rango corto de distancias de alguna tecnología como puede ser WIFI, Bluetooth…con el fin de que el dispositivo pueda determinar su posición aproximada en función de las balizas y la intensidad de señal que llegan de estas. El sistema más extendido es el uso de balizas BLE, usadas en entornos cerrados como aeropuertos [1] o museos en el ámbito de la localización de personas, ya que presentan otros posibles usos, como su uso en inventarios de elementos…[2]. Aunque es una solución efectiva, esta implica una inversión en los dispositivos que realizan la función de baliza en el entorno cerrado. Además, en muchos casos se requiere de un dispositivo receptor no estándar. Por tanto, una solución que no conlleve tantos gastos se tendría que centrar

en depender más en el propio dispositivo que comunica la ubicación en la que se encuentra el usuario.

Siguiendo esta deducción se puede llegar a la conclusión que, sin depender de GPS, sin balizas y solo del propio dispositivo del usuario, los medios que nos restan para utilizar con posibilidades de uso en la localización es la cámara del dispositivo móvil, del que dispone la mayoría de los usuarios actualmente. Con esta capacidad la solución más enfocada a la localización por balizas es el reconocimiento por parte de la cámara de códigos como pueden ser los QR, mediante el cual se crea una asociación entre un código QR y una localización, que resuelve la aplicación destinada cuando escanea uno de estos. La facilidad de la generación y escaneado de los códigos lo ha convertido en una opción atractiva para indicar ubicaciones, siendo común en la actualidad múltiples plataformas para la conversión de coordenadas en códigos QR orientados a la aplicación de ubicación por excelencia de Google Maps [3]; aunque cualquier otra aplicación capaz de extraer la información del código obtendría igualmente la ubicación almacenada.

La solución de localización basada en códigos QR es una buena solución con un coste reducido, ya que solo se deben generar e imprimir los códigos que se destinan a los puntos de interés que sirven de información para la ubicación del usuario. Ahora bien, existen varias cuestiones que pueden ser problemáticas, como es la elaboración y mantenimiento de estas etiquetas QR que además se deben colocar en el entorno, ocupando espacio y con cierto efecto estético indeseado. Ante esto la solución que menos depende de indicadores artificiales externos sería la localización por reconocimiento de imágenes. Esta utilidad a diferencia de las anteriores alternativas únicamente depende del dispositivo con la aplicación instalada correspondiente y del propio entorno donde el usuario se quiere ubicar. No deja ser una recreación de la capacidad humana de localizarse mediante la interpretación del entorno, cuando éste lo conoce por un estudio previo o meramente por haber estado en un momento anterior. Siguiendo con este símil, se podría decir que la aplicación actúa como una entidad que previamente conoce el entorno y sabe ubicarse en éste, transmitiendo dicha información al usuario. En nuestro desarrollo para generar esta característica la se hará uso de la inteligencia artificial.

1.1 DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Por tanto, en el siguiente TFG se expondrá una solución a la localización de personas, especialmente en entornos donde la señal GPS es débil o inexistente, mediante una aplicación que mediante un modelo de inteligencia artificial pueda

interpretar imágenes que toma el usuario del entorno que abarca la aplicación, pudiendo solo con su interpretación devolver una ubicación aproximada en el entorno propuesto. Con el fin de ser alcanzable la herramienta a la gran mayoría, se propone una implementación para dispositivos móviles basados en sistema Android, sistema operativo que es de código libre y ampliamente usado.

Específicamente el problema directo a resolver utilizará de prueba real el Museo Provincial de Guadalajara[1], donde se quiere implementar una serie de rutas temáticas por el interior del edificio en función de las obras que se recorren, además de permitir al usuario localizarse dentro del Palacio del Infantado, sede del museo.

1.2 OBJETIVOS

Los objetivos que se buscan cumplir mediante la realización del siguiente TFG se enumeran en los siguientes puntos:

- **Documentación y estudio:**
  Estudio y comprensión de los fundamentos de la inteligencia artificial, para comprender la tecnología que se precisa usar para obtener un resultado funcional.
- **Creación de un Clasificador de Imágenes**:
  Crear un modelo de inferencia que cumpla la función de un clasificador de imágenes, para su aplicación como parte fundamental en la lógica de la aplicación mediante el uso de librerías de inteligencia artificial.
- **Uso de entornos virtuales y locales:**
  A partir del conocimiento del funcionamiento y características de entornos virtuales y locales que cumplirán la función de contenedores independientes, desarrollar la creación del modelo de inteligencia artificial, permitiendo una solución local.
- **Desarrollo de una aplicación Android:**
  Aprendizaje y desarrollo de una aplicación para un sistema operativo Android y planteamiento/generación de un proyecto desde cero donde alojar la inteligencia artificial. Se intentará cubrir el mayor abanico de versiones del sistema operativo, para ser utilizable por cualquier usuario.
- **Puesta en marcha:**

[1] **Museo Provincial de Guadalajara**. https://cultura.castillalamancha.es/museos/nuestros-museos/museo-de-guadalajara

Obtener una aplicación con una usabilidad y tasa de acierto lo más elevada posible, superior al 75%, a la hora de su uso en un escenario real.

1.3 ESTRUCTURA TFG

En este apartado se enumeran los puntos en los que se ha dividido este TFG, explicando de manera reducida el contenido y composición de cada capítulo.

En este primer capítulo se realiza una breve introducción al planteamiento del problema que se intenta resolver en el siguiente TFG, junto con los objetivos buscados a superar durante su realización para dar paso al cuerpo del TFG.

En el siguiente capítulo se revisa el marco teórico del TFG, realizando la exposición del estado del arte actual de la inteligencia artificial y la localización, siendo los campos tratados en la elaboración del TFG para dar una noción del punto de partida. Dentro de estas explicaciones se irá especificando los subcampos en función de la aplicación directa con el desarrollo del trabajo.

En el capítulo tercero se proporciona un resumen del conjunto de elementos que son recurridos para la consecución del clasificador de imágenes y la aplicación contenedora, explicando brevemente su uso e importancia en el desarrollo.

En el cuarto capítulo se ofrece una explicación detallada del desarrollo completo a través de fases de desarrollo específicas para cada parte que conforma la aplicación final.

Continuando con el quinto capítulo se presenta la puesta en práctica del desarrollo explicado creando la aplicación para el escenario de prueba y estudio de los resultados que esta ofrece mediante un set de pruebas en el entorno de aplicación.

Acabando con el sexto capítulo el TFG, se realiza una reflexión de conclusiones tras la terminación del proyecto, acompañada de un pequeño enfoque en escalabilidad de la aplicación o usos alternativos, con la exposición para terminar del presupuesto generado.

Por último, se incluye el listado de artículos, libros y demás fuentes de información utilizadas en la elaboración de la documentación.

2 MARCO TEORICO

En la elección del contenido de este apartado, se ha realizado un estudio de los campos básicos que se tratan en el desarrollo de la aplicación para exponer al lector de manera breve una pequeña introducción a estos para comprender mejor el desarrollo del TFG.

Debido a la naturaleza de la solución ofrecida, una aplicación que permite a las personas ubicarse a partir del diagnóstico de imágenes; los campos de conocimiento que se tratan en el capítulo son la inteligencia artificial y la localización. A la hora de tratar dichos campos, se iniciará por una descripción más general, avanzando a la explicación de detalles y subcampos que son los aplicados en el desarrollo de la aplicación.

2.1 INTELIGENCIA ARTIFICIAL

Desde la antigüedad los primeros grandes inventores siempre han mostrado un interés por la creación de sistemas que faciliten la vida a la humanidad simplificando nuestras tareas y permitiéndonos afrontar mayores retos. Algunos de estos avances han pasado por la implementación de autómatas, robots…hasta finalmente llegar a la época de la computación, donde verdaderamente se ha llegado al punto de tener una gran capacidad de realizar tareas de manera automática por parte de un ordenador o cualquier otro dispositivo con un sistema de cómputo integrado. En relación con este aumento de capacidad en los equipos computacionales, nace el interés por el desarrollo de la capacidad de realizar con éxito una tarea mediante el autoaprendizaje por parte de estos equipos, sin tener que seguir un juego fijo de reglas como cualquier programa.

Aunque los autores no se ponen de acuerdo en definir exactamente el punto de inicio de la Inteligencia Artificial, de forma mayoritaria se puede decir que es un término acuñado en la conferencia de Darmouth en 1956 por el informático John McArthy. Previamente ya se realizaron reflexiones y ensayos como puede ser la publicación por uno de los padres de la computación, Alan Turing, de la prueba con su mismo nombre destinada a reconocer la inteligencia de un sistema computacional. Esta se basa en un interlocutor realizando preguntas y si cataloga a quien responde como humano siendo en realidad una máquina, se considera inteligente; en 2014 se superó esta prueba por primera vez [4][5].

Esta rama de la ciencia de la computación tuvo un inicio lento debido a la limitación de capacidad de cómputo que presentaban los sistemas de la época para

resolver ciertos problemas. Pero con el aumento de esta capacidad, la modalidad fue atrayendo nuevos investigadores e inversores que empezaron a lograr los primeros hitos, evolucionando exponencialmente al estadio actual que está en constante progreso [6].

El éxito de este desarrollo también reside en su aplicación directa en un gran abanico de tareas, siendo un punto clave en la evolución de los campos en los que se aplica, logrando méritos como la detección de objetos en vídeo o imágenes, extracción de patrones, conducción autónoma, análisis de datos…

Debido a la naturaleza del problema que se pretende solucionar se clasifica en tres tipos comúnmente llamados inteligencia débil, general y fuerte. La primera consiste en la resolución de problemas bien definidos y acotados, es el tipo que menor complejidad presenta y reflejo de ello es la solución de inteligencia artificial que mayor desarrollo concentra, teniendo grandes resultados en sus campos de aplicación. La segunda presenta un aumento en la complejidad debido a que es una inteligencia que puede resolver cualquier problema intelectual y adaptarse al entorno, siendo la evolución de la inteligencia artificial débil. Por último y actualmente utópico nos encontramos la inteligencia fuerte que presenta la complejidad que aúna las características inteligencias anteriores, además del propósito de dotar al sistema de consciencia de sí misma [7].

Como anteriormente se ha mencionado en la clasificación de las inteligencias artificiales, la más extendida es la inteligencia débil, la cual será en la que nos centraremos en este TFG al ser un problema bien definido y bastante acotado. Entonces centrándonos en este tipo existen dos principales ramificaciones que son el desarrollo de sistemas expertos y el aprendizaje automático [8][9]. Se revisan ambos brevemente a continuación.

SISTEMAS EXPERTOS

Esta rama se basa en la creación de máquinas capaces de simular a un experto humano o conjunto de ellos en un área del conocimiento, para servir de asesor en cualquier circunstancia o de manera ininterrumpida, de manera acertada y con resultados sólidos acompañados del razonamiento seguido. Estos sistemas se suelen basar en conjuntos de reglas (lógica difusa), en conjuntos de casos (casos registrados) o en redes bayesianas (por probabilidades). Las únicas funciones de un experto humano que no podría cubrir (al menos hasta el momento) serían las propias características humanas de decisiones éticas, entender el contexto… [10][11]

APRENDIZAJE AUTOMÁTICO

Esta rama agrupa un conjunto de técnicas enfocadas a la mejora constante del desempeño de los modelos mediante la experiencia que los sistemas adquieren con su uso. Con ello, aunque el sistema en un primer momento sea poco preciso, con el tiempo puede ir refinando esta precisión hasta conseguir resultados sobresalientes. Dentro de estas técnicas la más destacada es el aprendizaje automático, que consigue su objetivo mediante el uso de redes neuronales, característica de la cual obtiene el nombre debido a la profundidad de capas que presenta una red neuronal. Tal es su desempeño que erróneamente la mayoría confunde el nombre de la rama que agrupa estas técnicas, por el nombre de este subconjunto [12].

Debido a este gran potencial y sus características, será la metodología del Aprendizaje Automático la que se adopte en este TFG para la realización del clasificador de imágenes necesario para que funcione la aplicación. Por ello se hará una mayor explicación a continuación.

2.1.1 APRENDIZAJE PROFUNDO (DEEP LEARNING)

El aprendizaje profundo (también conocido como Deep Learning) es el campo dentro del aprendizaje automático que más éxito ha recolectado en los últimos años debido a su desempeño y fiabilidad. Esto se debe al uso de redes neuronales como parte esencial en sus modelos, las cuales se pueden reajustar con cada interacción para poder ofrecer resultados más precisos en futuras interacciones.

Este enfoque ha permitido dar a conocer la inteligencia artificial al usuario promedio ya que se ha conseguido implantar como parte esencial o complemento en multitud de productos y servicios de una amplia gama de sectores. Ejemplo de ello son todas las aplicaciones de visión artificial, conducción autónoma, reconocimiento por voz, monitorización de redes…las cuales podemos ver allá a donde se vaya en la actualidad representadas por ejemplos como Alexa y los demás asistentes virtuales, la gama de automóviles Tesla, el autocompletado en navegadores...

Para entrenar el modelo de inferencia que se empleará, se heredan del aprendizaje automático los siguientes paradigmas para aplicar al proceso de aprendizaje del sistema [13][14]:

- APRENDIZAJE SUPERVISADO:
  Es la técnica predominante y se basa en suministrar al sistema con datos etiquetados, por tanto, el sistema se reajusta cuando obtiene una etiqueta por

entrada y otra tras el proceso de inferencia, dependiendo se si la predicción es la esperada o no, se adapta para minimizar la tasa de errores.
Es la técnica que emplearemos en el entrenamiento del clasificador de imágenes, ofreciendo al sistema imágenes agrupadas en carpetas con el nombre de las etiquetas de las ubicaciones a predecir.

- APRENDIZAJE NO SUPERVISADO:
  Se basa en suministrar al sistema con datos no etiquetados con el fin de que el propio modelo cree agrupaciones y resultados con el fin de observar nuevas metodologías de tratamiento y creación de conjuntos de datos afines.
- APRENDIZAJE SEMI-SUPERVISADO:
  Se basa en una fusión de los dos anteriores, utilizando un tipo de entrenamiento para tratar los datos crudos de entrada y con el segundo tipo entrenar como se obtienen los resultados de salida del modelo, tratando los resultados del primer entrenamiento.
- APRENDIZAJE POR REFUERZO:
  Se basa en la implementación de un sistema de recompensas el cual castiga al sistema cuando el resultado obtenido no es favorable y por el contrario recompensa a este cuando el resultado es bueno. En resumen, es un sistema de prueba y error, hasta perfeccionarlo completamente.
- APRENDIZAJE DE TRANSFERENCIA:
  Es un caso que se puede dar con el requisito imprescindible de haber aplicado alguno de los paradigmas anteriores. Esta técnica se basa en aplicar los parámetros aprendidos en otros modelos, permitiéndonos ahorrarnos tiempo y tener capacidad de replicar un buen entrenamiento en nuevos modelos con funciones similares.

El éxito de estos aprendizajes dependerá de la calidad y diversidad de datos que se suministran al sistema, en equilibrio con una selección exhaustiva de la arquitectura que más se ajuste al problema a resolver. La descompensación en alguna de ambas partes impedirá la obtención de un modelo funcional.

Algunos de los problemas más comunes que nos encontramos son el sobreajuste y el desajuste, el primero significa que con los casos de entrenamiento funcionará correctamente, pero a la hora de generalizar las conclusiones obtenidas fracasará. El segundo se puede dar debido a que el dataset no es suficiente para obtener patrones o que la red neuronal en la que se basa el sistema no es capaz de obtener patrones apropiados de los datos con los que trabaja.

A continuación, trataremos el campo de las redes neuronales para el enfoque del aprendizaje profundo, ya que la selección de esta es parte fundamental para el éxito futuro del sistema entrenado.

2.1.2 REDES NEURONALES

En el afán de crear sistemas inteligentes, los investigadores siempre han contemplado al cerebro humano como referencia en capacidad de razonamiento para intentar replicar su funcionamiento en sistemas informáticos. Entre los años 40 y 50 esta aproximación se siente palpable con los primeros modelos eléctricos de neuronas y ensayos académicos sobre el campo. No es sin embargo hasta el 1958 cuando Frank Rosenblatt crea la primera neurona artificial capaz de clasificar una entrada en dos posibles salidas, se la define el modelo como perceptrón teniendo la máquina el nombre de Mark I Perceptron. Posteriormente al año siguiente se consigue crear una aplicación real de red neuronal por Bernard Widrow y Marcian Hoff para la supresión del ruedo en comunicaciones, se llaman estas primeras redes neuronales ADALINE y MADALINE, iniciando el desarrollo de redes neuronales [15].

La base de toda red neuronal es la neurona. La neurona está conformada por una serie de entradas (denotadas como xi en la figura) cuyo conjunto conforma el estímulo que llega a la neurona. A cada entrada se la aplica un valor ajustable que se denomina peso (denotados como xi en la figura) y dota de un nuevo valor a la entrada original, la modificación de este factor es lo que permite reajustar el comportamiento de la neurona para solucionar el problema planteado. Por último, el conjunto de entradas multiplicadas por sus pesos correspondientes se suman paso previo a aplicarse una función de activación (una función escalón en la figura) en algunos casos se encuentra un elemento extra denominado bias (θ) que se incluye en el sumatorio [16]. La función de activación puede ser de diversos tipos como funciones *ReLu*, escalón, sigmoidales…dando un valor único de salida a la neurona. La primera neurona artificial se denominó perceptrón y era capaz de clasificar una entrada de valor continuo en dos etiquetas [17], presentaba una función escalón como en el esquema inferior.

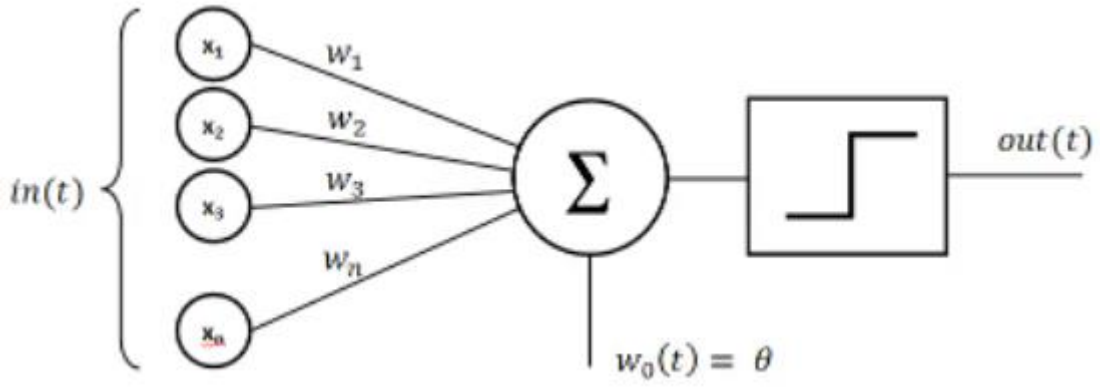


Ilustración 1-Perceptron, esquema básico de una neurona [16]

De acuerdo con la estructura neuronal del cerebro, estas se agrupan por capas que se van conectando las salidas de una a las entradas de la siguiente formando la red neuronal. Las redes neuronales se organizan en tres tipos de capas:

- **Capa de entrada**
  Esta es el primer conjunto de neuronas que recibe la información original que se suministra al sistema para realizar la predicción.
- **Capas ocultas**
  Es un conjunto o varios de neuronas interconectadas encargadas de realizar la inferencia del modelo con el fin de ofrecer un resultado a tratar por la última capa, la capa de salida. Su número define la característica de la red neuronal de profundidad.
- **Capa de salida**
  Esta es el último conjunto de neuronas que se encarga de aunar todos los resultados obtenidos previamente y ofrecer un resultado basado en las etiquetas del modelo.

En su conjunto forma la topología de la red definiendo características como la direccionalidad de las salidas de las capas o la interconectividad entre la salida de una capa y la entrada de la siguiente, de acuerdo con el fin que se busca obtener del modelo [18].

Con ello las capas y la red obtienen unas características por las que se pueden identificar, algunas de las más destacables son:

- **Grado conexión**

Característica relativa al porcentaje de conexiones que puede realizar con la siguiente capa y las que en realidad realiza. Evitar o ignorar conexiones en el entrenamiento genera un ruido para evitar el sobreajuste.

- **Anchura**
  Característica relativa al número de neuronas que conforman una capa.
- **Resolución**
  Característica relativa al número de entradas de información que posee una capa.
- **Profundidad**
  Característica relativa al número de capas ocultas que componen el modelo.
- **Direccionalidad**
  Característica relativa a la dirección de las conexiones entre capas, pudiendo ir hacía capas anteriores (recurrentes) o hacía la capa de salida (no recurrentes).

A continuación, la siguiente ilustración, se muestra un posible esquema de red neuronal artificial, donde se tiene un ejemplo de arquitectura básica, mostrando la capa de entrada (nodos amarillos), capas ocultas (nodos verdes) y finalmente la capa de salida (nodo rojo).

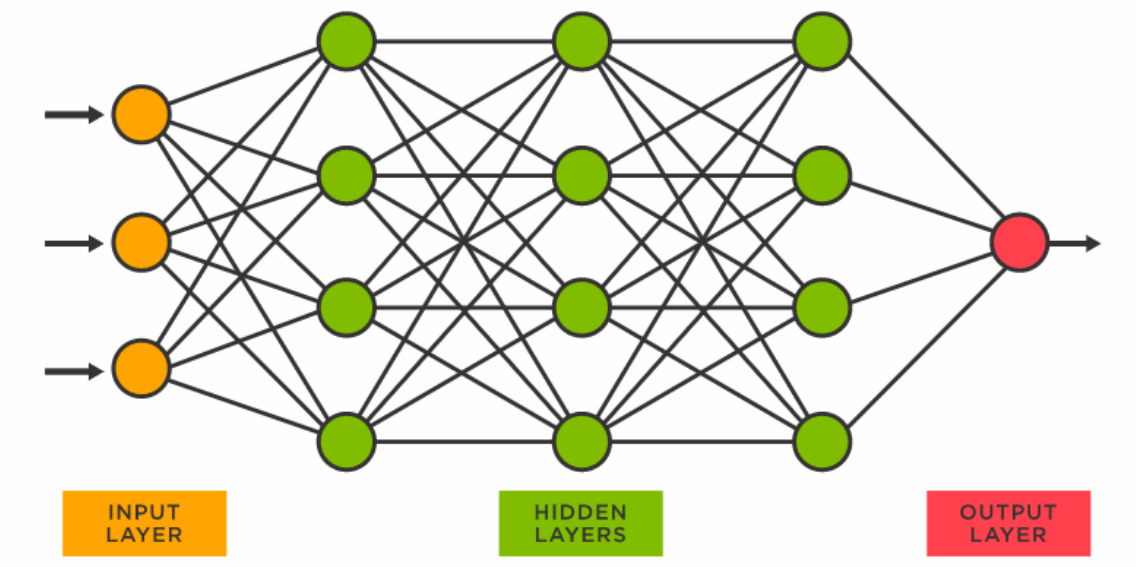


Ilustración 2-Esquema de red neuronal artificial [19]

Parte fundamental de una red neuronal es la asignación de pesos. Esta se basa en la aplicación de un algoritmo que partiendo de la función de coste reajusta los pesos del modelo, en otras palabras, es la encargada de que el modelo aprenda a realizar el trabajo que se le asigna. La función de coste se encarga de definir el error

entre el valor esperado y el obtenido al final del modelo, usualmente se utiliza el error cuadrático medio.

En la asignación de pesos inicial se recomienda partir de valores no iguales a cero, hecho que puede conducir a una solución secundaría y menos efectiva. En cuanto al reajuste en el proceso de entrenamiento el algoritmo más extendido es el de *backpropagation* por descenso de gradiente, el cual reajusta los pesos una vez obtenida la predicción en el sentido de la capa de salida hacía la de entrada [20].

La importancia del algoritmo de asignación de pesos se refleja en el abandono hasta los años 80, debido a la imposibilidad de entrenar o converger en una solución óptima las topologías que se implementaban, también influido por el tiempo que tomaban los entrenamientos debido a la capacidad de cómputo del modelo. Finalmente, con el aumento de la capacidad de cálculo con el paso del tiempo y los primeros resultados prometedores tras décadas, este campo de la inteligencia artificial resucitó obteniendo éxitos sin precedentes hasta la actualidad, estando en pleno auge y encontrándose como solución en millones de aplicaciones [21].

2.1.3 REDES NEURONALES CONVOLUCIONALES

Las redes convolucionales son las redes usadas principalmente por excelencia en el procesamiento de imágenes, además de aplicaciones en el lenguaje natural o el reconocimiento de video. Esto se debe a su naturaleza enfocada a la extracción de características de los datos suministrados pasando por sus múltiples capas, las primeras conseguirán obtener características más simples y las últimas podrán obtener patrones más difíciles de ver a simple vista.

Esta obtención de características por parte del algoritmo se debe a la similitud de su estructura con la corteza visual primaria de un cerebro biológico, que al igual que la red, cuantas más capas atraviesa se consiguen diferentes tipos de características. Esto se consigue gracias a la operación que da nombre a este tipo de redes, la convolución. [22]

Debido a que su uso se concentra en el análisis de imágenes las unidades información de entrada al modelo serán píxeles. Los píxeles son la menor porción indivisible cuyo conjunto y variedad compone una imagen. Cada píxel tiene una serie de bits asignados, pudiendo adoptar valores comprendidos por la combinación de estos, representando cada valor un color distinto; normalmente se les asigna 8 bits pudiendo presentar valores entre el 0 y el 255 [23]. A su vez cada imagen tendrá una codificación que significará la existencia de una o más capas superpuestas, como es el caso de la famosa codificación RGB (3 mapas de píxeles superpuestos).

Como se mencionó al inicio, el algoritmo de la convolución es la principal característica de este tipo de redes neuronales. Esta operación se diferencia de la operación típica de toda neurona artificial de multiplicar las entradas por los pesos establecidos para posteriormente realizar una función de activación con el sumatorio de los valores obtenidos. Este algoritmo se basa en establecer los pesos en matrices que reciben el nombre de *kernels* que se desplazan como una ventana deslizante sobre la matriz de entrada, realizando una multiplicación escalar entre el *kernel* y el área delimitada por su ventana, guardando el resultado en una nueva matriz. Esta nueva matriz dependerá de las dimensiones de la matriz de entrada y el *kernel*, además del paso que se establezca para la ventana deslizante. Finalmente, a esta matriz resultado se la aplica como en las redes neuronales clásicas una función de activación, esta tiende a ser de tipo *ReLu*.

En la imagen inferior como la matriz resultante tiene la dimensión de 4x4 debido a que el *kernel* de 3x3 se aplica sobre la matriz de entrada 6x6 con una ventana deslizante con paso igual a 1 (avanza de píxel en píxel).

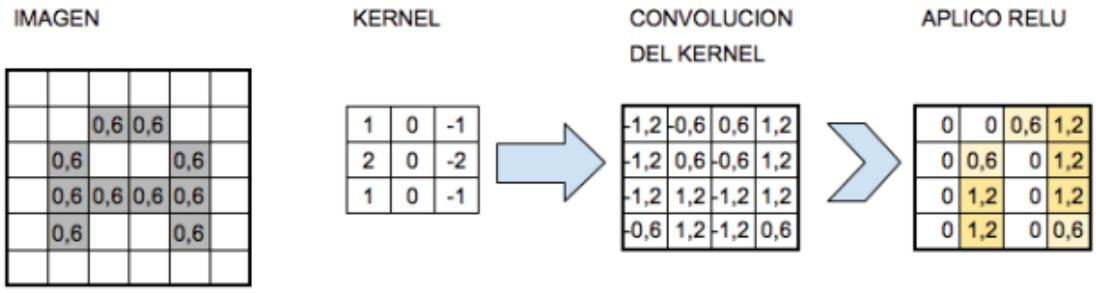


Ilustración 3-Ejemplo de convolución [22]

En la práctica no existe un único *kernel*, llamándose al conjunto de los distintos *kernel*, filtros, realizando todos ellos convoluciones sobre la matriz de entrada y generando nuevas matrices resultados. Debido al incremento de datos para la capa consecutiva se aplica una operación de reducción por matriz resultado para que el tamaño de datos que pasa a la siguiente capa sea asequible matemáticamente, realizando funciones de *Subsamplig* como puede ser *Max-Pooling*.

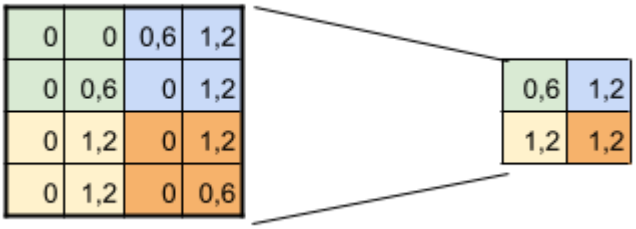


Ilustración 4-Ejemplo de Subsamplig sobre matriz resultante de convolución [22]

Comprendido el funcionamiento, las capas ocultas de la red convolucional consistirá en un ciclo continuo de las operaciones recién explicadas, siendo la salida de una capa la entrada de la siguiente, consiguiendo extraer diferentes y más costosas características con el paso continuo por capas. Debido a la complejidad del proceso, el suministro inicial de la imagen suele ir acompañado por una operación de preprocesamiento para la mejora del comportamiento del modelo.

Este preprocesamiento en primer lugar se basa en una regularización del rango de valores que presentan los píxeles, con el fin de que las operaciones matemáticas sean menos costosas. Esto se realiza mediante un reescalado de su valor, dividiendo cada píxel entre el rango completo de valores posibles que puede adoptar, reduciendo la escala entre 0 y 1. Como paso adicional y buena práctica, debido al desplazamiento de la ventana de los filtros, esta se centra más en el centro de la imagen que en el contorno, pudiendo perder información importante; por ello existe una técnica denominada *Padding* que consiste en añadir ceros alrededor del contorno para que no afecten a la convolución pero la ventana obtenga mayor información del contorno de la imagen [24].

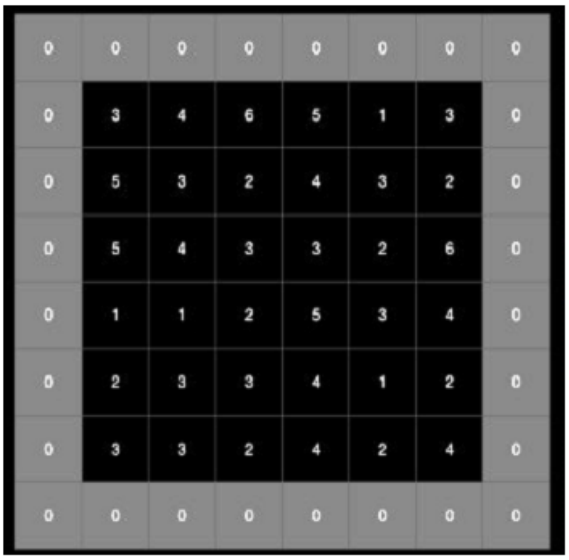


Ilustración 5-Técnica de Padding para preprocesamiento [24]

Finalmente, al acabar el conjunto de convoluciones llega el momento de extraer los resultados en función de las etiquetas que se buscan predecir. Por ello se abandonan las capas convolucionales para alimentar a un clasificador creado como una red neuronal clásica. Para realizar este traspaso entre red convolucional y red clasificadora es necesario aplanar la salida de la última capa convolucional, por último, se diseña una red clasificadora interconectada que, en su última capa, siendo la capa final de todo el modelo, aplica una función de activación *Softmax* para obtener el listado de probabilidades de las etiquetas posibles para predecir.

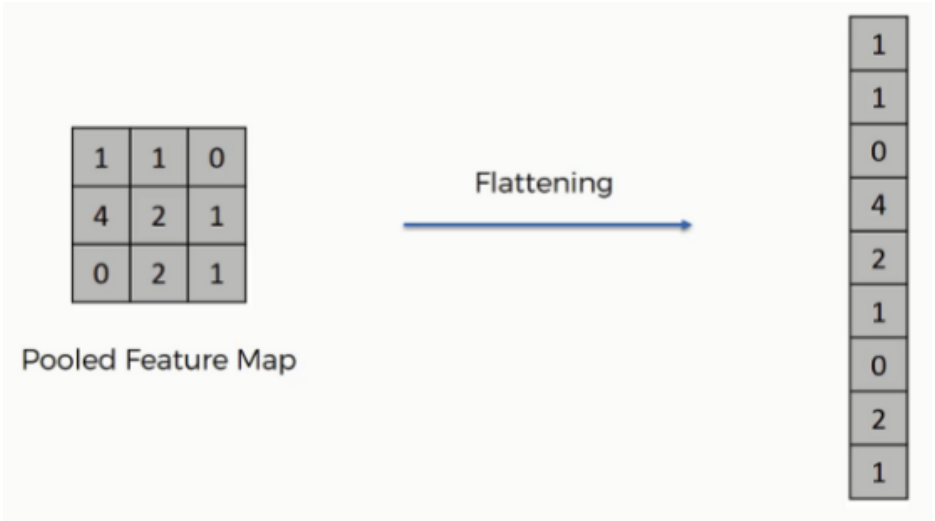


Ilustración 6 - Esquema planificación de salida de última capa oculta de tipo convolucional [24]

Como en las redes neuronales clásicas explicadas en el anterior apartado, al tratarse de entrenamiento supervisado, los pesos del modelo se recalcularán al final de cada iteración con el algoritmo de *Backpropagation* con descenso por gradiente,

modificando los valores de los *kernel* en las capas encargadas de la convolución y los pesos en las capas encargadas de la clasificación.

2.2 LOCALIZACIÓN

Una tarea básica del ser humano es conocer donde se ubica y conocer el entorno, para que en el día a día poder desplazarse a los sitios donde requiere ir. Aunque sea en la actualidad una tarea sencilla cuando pensamos en ubicarnos en nuestra localidad, el camino para ir al trabajo, para ir a hacer compras… cuando no se conoce como llegar a un destino, es donde cobra significado la necesidad de obtener la localización.

Esta respuesta se encontró en las épocas más tempranas mirando el cielo, usando las costelaciones como orientación en navegación, senderos… o en rústicos primarios mapas que reflejaban el inicio de las herramientas de orientación. Con el tiempo los medios para crear mapas o encontrarse orientado mirando las estrellas se fueron perfeccionado. Pero finalmente el mayor avance fue el uso del GPS cuando lo liberó el ejército de los Estados Unidos para el uso civil [25]. Esta tecnología se basa en el uso de la trilateración que consiste en que el dispositivo al recibir conexión con al menos 3 satélites, en función de la hora y localización cuando llega la conexión al satélite consigue obtener una ubicación aproximada de donde se encuentra el usuario, con las diferencias declaradas en estos datos [26].

A partir de este punto se ha utilizado esta tecnología como la principal herramienta de localización, implementándose en sistemas portátiles para permitir al usuario una localización de donde se encuentra siempre que tenga señal. Con todo ello en la actualidad el localizarse parece un problema del pasado, aunque siempre hay nuevas metodologías e investigaciones para encontrar métodos eficientes para suministrar localización a usuarios donde se encuentren.

Como se introdujo previamente la tecnología GPS ve su capacidad mermada en entornos interiores donde la señal satelital es débil o inexistente, es por ello que se han desarrollado múltiples técnicas de localización adaptadas a esta situación. Las principales técnicas aplicadas a la localización en interiores que han acumulado éxito en sus implementaciones son las siguientes [27]:

TRILATERACION

Es la técnica básica de localización utilizada por la localización GPS para conseguir un resultado. Consiste en determinar la ubicación mediante las distancias a

las que se encuentra el usuario respecto de las balizas que recibe señal. Se necesita la señal de 3 balizas mínimo.

TRIANGULACIÓN

Es una técnica extendida en la telefonía basada en el uso de la trigonometría para obtener la ubicación a partir de la distancia entre el receptor de la señal y las balizas, en conjunto con los ángulos que componen el triángulo que une las balizas y la entidad que se localiza. A diferencia de la trilateración depende de un elemento menos para su funcionamiento básico.

MULTILATERACIÓN

Es una técnica similar a la trilateración distinguiéndose en la forma que se calcula la distancia entre baliza y entidad, al calcularse por tiempo de recepción de las señales de la infraestructura de balizas, en vez de por su intensidad. Para su correcto funcionamiento requiere de mayor número de puntos de referencia que la trilateración.

ANÁLISIS DE LA ESCENA

Es una técnica más analítica ya que se determina un punto inicial respecto al que se determina la ubicación de la entidad respecto al punto de referencia y mediante el conocimiento del entorno.

PROXIMIDAD

Es una técnica que exige mayor infraestructura ya que se debe encontrar abundantemente en el entorno, al emitir señal de muy corto alcance (tecnología NFC…) o que sea necesario interactuar con las balizas (escaneados, códigos…) que permiten a la entidad localizarse.

Ilustración 7-Resumen de técnicas de localización en interiores [27]

Estas técnicas se han combinado con las tecnologías de vanguardia y se han confeccionado eficaces sistemas de localización en interiores con distintos enfoques como los que se explican a continuación.

2.2.1 SISTEMA DE LOCALIZACIÓN POR BALIZAS

El sistema de localización por balizas sería la aplicación a un entorno cerrado de la filosofía de la tecnología GPS, ya que las balizas se podrían comparar con la red de satélites, con los que se interactúa para que, a partir de la intensidad de señales, retardo de respuesta… se puede calcular la localización del usuario.

Dependiendo de la interacción con las balizas, hay dos tipos de sistemas, activos o pasivos [28].

El primero consiste en que el dispositivo de localización es el elemento que emite una señal hacía su alrededor, con el fin de detectar las balizas en estado de espera, las cuales emiten una señal de respuesta. Midiendo los tiempos entre emisión de la señal y respuesta a la misma por parte de la baliza, se concluye a la distancia que se encuentra el usuario de cada baliza y conociendo las ubicaciones de las balizas el programa del dispositivo permite calcular la localización aproximada donde se encuentra el portador del dispositivo.

Los sistemas pasivos por el contrario dotan a los dispositivos de localización de la capacidad de recepcionar señales entrantes. Debido a ello los roles se cambian en comparación con el caso activo, siendo las balizas las emisoras de una señal constante con la finalidad de que el programa pueda determinar la intensidad de la señal recepcionada, no únicamente una respuesta. Siguiendo el esquema anterior dependiendo de la intensidad se conoce a la distancia que el usuario se encuentra de la baliza y el programa conoce la localización de cada una, pudiendo triangular la ubicación del usuario.

Por factores de complejidad se ha implementado más el segundo enfoque debido a la sencillez de confeccionar balizas que emiten una señal constante y un dispositivo receptor, a un sistema de emisión y respuesta.

Como última característica variable de estos sistemas se trataría del tipo de protocolo de señal a utilizar, siendo los más utilizados el protocolo Wifi o el Bluetooth [29].

2.2.2 SISTEMA DE LOCALIZACIÓN POR ETIQUETAS

El sistema de localización por etiquetas supone un modelo más simple ya que cada localización a representar debe ir acompañada de una etiqueta que contendrá la información relativa a la ubicación. Estas etiquetas dependiendo de la tecnología implementada transmiten su información almacenada tras la interacción del usuario con ellas o entrando en un rango muy cercano a donde se encuentran.

Debido a esta sencilla capacidad de transmisión de información, entornos cerrados como museos se iniciaron en el uso de estos para asociar la información de las vitrinas, también almacenes con el fin de aplicar sistemas inteligentes de gestión de los mismos…Por tanto se crea un sistema de localización al correlacionar el código con el lugar que ocupa [30] [31].

La implantación de este tipo de sistema consistiría en la elaboración de etiquetas que almacenan un tipo de información, para que posteriormente una sencilla aplicación en un móvil o PDA pueda interpretarla en función de la naturaleza de la etiqueta. Principalmente hay dos tipos de etiquetas ampliamente usadas que se explicaran a continuación.

ETIQUETAS QR

Los códigos QR nacen como una evolución del código de barras clásico para mejorar la cantidad de información almacenada y la trazabilidad de producto en el sector logístico, su utilidad escaló rápidamente a distintas utilidades. La sencillez tras su escaneado de acceder a hipervínculos de sitos webs o la realización de acciones automáticas es la característica que popularizo y extendió su uso. En la actualidad se puede encontrar en distintos lugares como restaurantes, puntos de información… además de almacenes como inicialmente, especialmente tras la aparición de la pandemia del *SARS-CoV-2* [31].

El incremento de información almacenada en la etiqueta en comparación con su predecesora le permite optar a usarse como etiquetas para la localización, con la simple tarea de vincular información de la ubicación donde se situé la etiqueta. Este registro de la información de la ubicación donde se sitúa la etiqueta puede ser tan simple como el nombre que se le ha asignado a la misma o ser más complejo mediante la inserción de hipervínculos que mediante el navegador o una aplicación predeterminada muestra una interfaz con un mapa para indicar la ubicación actual al usuario.

Por tanto, un sistema de esta índole trataría de rellenar el entorno objetivo con etiquetas con códigos QR, registrando como información las coordenadas o similar con el fin de la localización, a la espera de que el usuario lo escanee y obtenga los resultados.

ETIQUETAS RFID

Las etiquetas RFID hacen uso de las señales de radiofrecuencia para permitir la comunicación entre el dispositivo lector y la misma, no teniendo que establecer

contacto visual ambas partes. Nace como una solución al proceso de identificación de productos, ya que previamente había que encontrar el producto y luego realizar la lectura de su código identificativo; mientras que este sistema suprime este proceso producto por producto, pudiendo identificar la totalidad de elementos del área de alcance de la señal emitida por el lector. Otra ventaja que presenta es el aumento de información que puede proporcionar la etiqueta [32].

Las etiquetas RFID están compuestas por un circuito integrado donde se almacena la información y una antena que emitirá esta al lector. El funcionamiento de este último elemento definirá el tipo de etiqueta que es, distinguiéndose entre etiqueta activa o pasiva dependiendo de la continuidad en la emisión de la señal o solo por respuesta a la señal del lector. En caso de ser activa debido al consumo se integra una batería en la etiqueta.

El funcionamiento se basa en obtención de la señal que emite la antena de la etiqueta con la información ligada a la misma por parte del lector. Esta se obtiene de manera directa en caso de uso de etiquetas activas y en caso de ser pasivas se debe enviar una señal previa desde el lector que activa la señal con la información de respuesta por parte de las antenas [33].

En la aplicación para la localización se trataría de utilizar la tecnología derivada de NFC basada en los mismos principios, pero en este caso sí que hay que tener la etiqueta al alcance para que acercando el lector obtener la información, al ser de menor alcance la señal (aproximadamente un rango de 20 cm) [34].

3 METODOLOGÍA Y HERRAMIENTAS EMPLEADAS

En el siguiente apartado se enumerarán las distintas herramientas que han sido requeridas para realizar las distintas partes del TFG, explicando la funcionalidad de cada una y su sentido y utilización a lo largo del desarrollo. Este listado lo conforman estructuras de datos, librerías de código libre, entornos de desarrollo y editores de distinta índole.

3.1 DATASET

En todo problema de entrenamiento de un modelo de inteligencia artificial se presenta la necesidad de alimentar al sistema para que aprenda mediante la extracción de patrones del conjunto suministrado como se explicó en el capítulo del aprendizaje profundo. Como se especificó en dicho apartado el tipo de datos suministrados dependen del problema al que se intenta dar solución, teniendo que ser

este conjunto una representación general de todas las posibilidades que se buscan manejar como posibles salidas del modelo de inferencia [35].

Para obtener un dataset hay que hacer un estudio previo de los resultados que se buscan obtener a la salida del modelo y en función de ello generar un dataset acorde a cubrir equitativamente con suficientes ejemplos cada etiqueta a predecir. Este conjunto de datos puede ser de diversos tipos como puede ser una tabla de datos de una base de datos, un conjunto de imágenes, archivos de audio, archivos de video…

La recolección de estos de manera correcta es una pieza clave en el entrenamiento ya que trabajar con un muestrario reducido conducirá al modelo a no llegar a ninguna conclusión, ya que no ha podido ajustarse correctamente el algoritmo de predicción. Por parte contrapuesta un conjunto de entrada que se centre en una o más etiquetas tampoco obtendrá un resultado óptimo ya que generalizará erróneamente los patrones de las etiquetas con mayor concentración respecto al resto del dataset, conllevando una predicción errónea o en muchos casos dando lugar al sobreajuste.

En el presente TFG se manejará un dataset comprendido por imágenes como tipo de dato de entrada al modelo, debido a la naturaleza del problema a resolver de crear un clasificador de imágenes. Por tanto, será preciso obtener un conjunto de imágenes de los lugares que se quieren reconocer, tomando las imágenes desde distintas perspectivas y ángulos, para ofrecer al entrenamiento un mayor número de perspectivas posibles del lugar para aumentar las probabilidades de mejores precisiones a la hora de predecir. Hecho importante antes de iniciar la recolección es determinar que imágenes son significativas y representativas de los lugares a reconocer posteriormente para no originar conjuntos etiquetados confusos. Ejemplo de ello sería formar un dataset con imágenes del suelo del entorno a registrar, cuando todos los puntos de interés del entorno tengan el mismo tipo de suelo, haciendo imposible la localización; por el contrario, habría que formarlo con elementos que no tiendan a repetirse de estos lugares a predecir [36].

Una herramienta adicional en este tipo de datasets de imágenes para incrementar el conjunto recogido de imágenes es la metodología de aumento de datos (*DataAugmentation*). Este método consiste en replicar datos del dataset original y aplicarles filtros o modificaciones, sin perder las características esenciales del dato [37]. Esta técnica es fácilmente aplicable en este caso debido a la multitud de filtros y

acciones que se pueden hacer sobre una imagen sin perder las características de esta.

3.2 LIBRERÍA OPEN CV

OpenCV [2] es una librería de código libre enfocada a problemas de visión artificial por computadora, nacida como un proyecto de investigación de la compañía Intel. Está disponible mediante sus APIs para principales lenguajes de programación como pueden ser Java, Python o C++.

El tratamiento y aplicación de los algoritmos de sus interfaces se basa en la operación con las imágenes, tratándolas como matrices y *arrays* de los pixeles que la forman. Con ello se consigue transformar la imagen a un formato manejable por un programa siendo el valor de un píxel dependiente del número de bits que se aplican para representar un píxel, rango escalable a la hora de simplificar las operaciones a realizar sobre estas matrices. Las escalas más relevantes son las de imágenes en blanco y negro ([0,1] 1 canal), imágenes en escala de grises ([0,255] 1 canal) e imágenes RGB ([0,255] 3 canales) [38].

Con los distintos métodos que nos ofrece la API se pueden aplicar diferentes filtros sobre las imágenes del dataset de manera sencilla pudiendo realizar la operación introducida en el apartado anterior del aumento del dataset, de cara a ofrecer al modelo a entrenar distintas imágenes a las originales para tener más variedad de vistas de la misma imagen, para la extracción de su patrón característico.

3.3 LIBRERÍA TENSOR FLOW

TensorFlow [3] es una herramienta de código abierto para la computación numérica, cualidad necesaria para el desarrollo de la inteligencia artificial, que ha conseguido por su excelencia y potencia de cómputo ser la plataforma más importante para el desarrollo de modelos basados en el aprendizaje profundo.

Este proyecto fue desarrollado por el equipo Google Brain, destinado al desarrollo de soluciones de inteligencia artificial para la compañía y para los usuarios de sus servicios; desde el 2011 con el fin de actualizar las librerías con las que se trabajaba por entonces. Como manera de obtener un desarrollo más rápido y extenso, en 2015 se tomó la decisión de ponerlo al alcance de cualquier usuario, acontecimiento que la convirtió en la gran plataforma actual, soportada por la gran comunidad de desarrolladores, manteniendo la naturaleza de código abierto aun

[2] **OpenCV**, librería de visión artificial. https://opencv.org/
[3] **TensorFlow**, librería de IA. https://www.tensorflow.org/

finalizada la primera versión. El hecho de congregar esta extensa comunidad detrás de la plataforma ha dotado a la herramienta de la característica de flexibilidad a la hora de desplegarse en múltiples tipos de dispositivos y tener un mayor alcance [39].

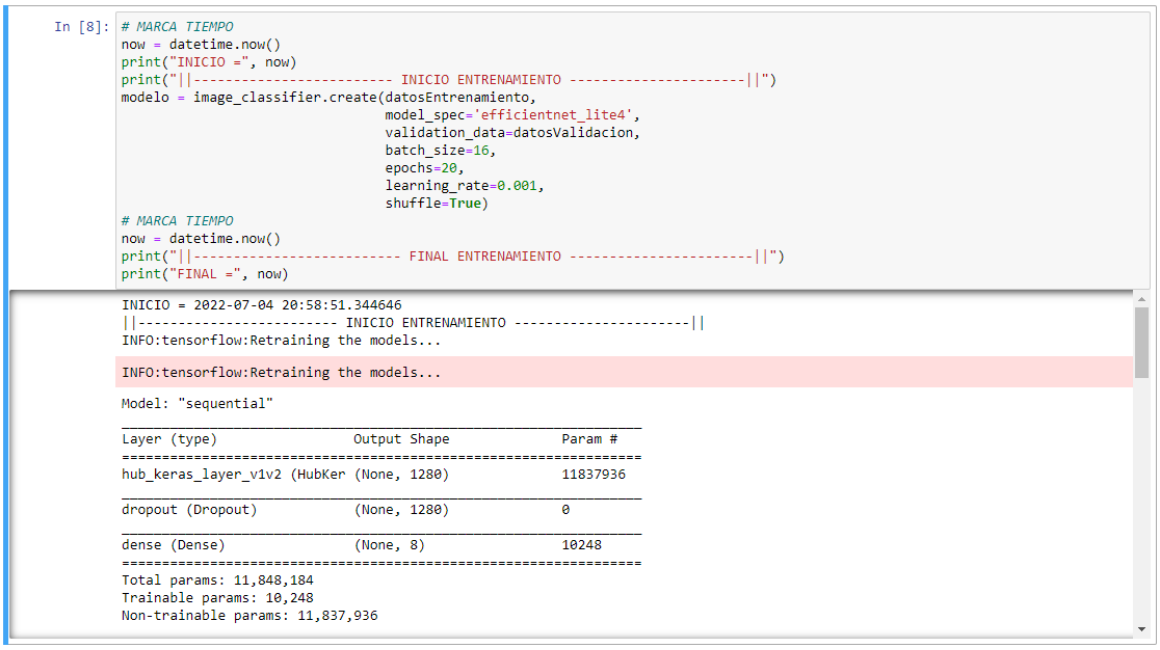
La escenificación de la productividad y fiabilidad de esta librería se ha demostrado por la confianza depositada por grandes gigantes tecnológicos y empresariales, como herramienta a la hora de crear sus propias aplicaciones internas o comerciales como son los casos de éxito en Airbus, Coca-Cola, GE Healthcare, Intel… [40]

La base del funcionamiento de esta librería se basa en los nodos y los tensores que los interconectan, formando un gráfico de flujo de datos. Los tensores son objetos matemáticos relativos a un campo vectorial y son invariantes a una modificación de la base [41], mientras que los nodos son operaciones matemáticas. A nivel de programación ofrecen un interfaz basado en Python para reducir la complejidad, pero internamente las operaciones se realizan basadas en C++ incrementando el rendimiento [42].

La demanda incremental de cómputo para la inteligencia artificial y el uso extendido de esta librería llevo al desarrollo de unidades de procesamiento enfocadas a la ejecución de esta librería denominadas TPU, que las siglas significan unidad de procesamiento de tensores, para un mayor potencial en el uso de esta librería accediendo a Google en la nube [42].

La principal competencia a esta librería son las alternativas de las librerías de PyTorch, CNTK o ApacheMXNet, pero aún sigue muy distante de arrebatarle el pódium a esta librería [42].

Ilustración 8 – Logo TensorFlow



```
In [8]: # MARCA TIEMPO
now = datetime.now()
print("INICIO =", now)
print("||------------------------- INICIO ENTRENAMIENTO ----------------------||")
modelo = image_classifier.create(datosEntrenamiento,
                                 model_spec='efficientnet_lite4',
                                 validation_data=datosValidacion,
                                 batch_size=16,
                                 epochs=20,
                                 learning_rate=0.001,
                                 shuffle=True)
# MARCA TIEMPO
now = datetime.now()
print("||------------------------- FINAL ENTRENAMIENTO ----------------------||")
print("FINAL =", now)

INICIO = 2022-07-04 20:58:51.344646
||------------------------- INICIO ENTRENAMIENTO ----------------------||
INFO:tensorflow:Retraining the models...

INFO:tensorflow:Retraining the models...

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hub_keras_layer_v1v2 (HubKer (None, 1280)              11837936
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 8)                 10248
=================================================================
Total params: 11,848,184
Trainable params: 10,248
Non-trainable params: 11,837,936
```

Ilustración 9-Inicio de ejecución de entrenamiento de un modelo preentrenado

3.3.1 LIBRERÍA TENSOR FLOW LITE

TensorFlow Lite [4] es una herramienta de código abierto al igual que la librería de la que desciende, con el fin de prestar los mismos servicios adaptándolos para dispositivos móviles y sistemas embebidos; debido a la reducción de capacidad de cómputo en comparación con un ordenador o mayores sistemas. Por tanto, se convierte en la librería principal para el desarrollo e implementación de inteligencia artificial, mayoritariamente aprendizaje profundo, en pequeños dispositivos. [43]

Algunas de las características que ofrece la librería de TensorFlow Lite son las siguientes:

- **Ligera**
  Debido a que son dispositivos reducidos, las unidades de almacenamiento instaladas no serán grandes, por lo que el tamaño de los modelos finales de inferencia será del menor tamaño posible.
- **Latencia reducida**
  Debido a ser terminales de uso, el tiempo de obtención de respuestas y resultados tendrá que ser lo más breve posible, por lo que se centrará en la elaboración de resultados de manera veloz.
- **Seguridad**

[4] **TensorFlow Lite**, librería de IA orientada a pequeños dispositivos.
https://www.tensorflow.org/lite?hl=es-419

Debido a ser terminales de los sistemas completos, será uno de los puntos más expuestos a riesgos externos al sistema, por lo que se tiene que reforzar esta característica.

- **Consumos eficientes**
  Debido a que son terminales, no siempre tienen una alimentación continua, por lo que mantener consumos bajos será importante para poder utilizar los modelos de manera más independiente a la alimentación posible.
- **Preentrenados**
  Debido a la naturaleza de los dispositivos reducidos, su capacidad de cómputo es reducida por lo que realizar un entrenamiento adecuado precisará de tiempos exageradamente largos, por lo que el entrenamiento no podrá realizarse en estos mismos dispositivos. Si acaso se puede abordar algún proceso de refinamiento posterior al entrenamiento previo.

Siguiendo el desarrollo de esta última característica enumerada hay que destacar la librería *Model Maker*. Esta librería es la encargada de permitir el aprendizaje por transferencia al usuario, reduciendo el tiempo de entrenamiento, además de obtener mejores resultados ya que seguramente el dataset usado del modelo preentrenado sea muy superior en cantidad y calidad al presentado para este segundo entrenamiento [44]. El abanico de modelos que permite esta librería cubre varios problemas del aprendizaje profundo como clasificadores de imágenes, problemas de lenguaje natural, detección de objetos… accediendo a estos mediante su plataforma dedicada a ello llamada *TensorFlow Hub*, donde se alojan múltiples modelos y variantes para aplicar a un mismo problema de inteligencia artificial [45].

Ilustración 10 - Logo TensorFlow Lite

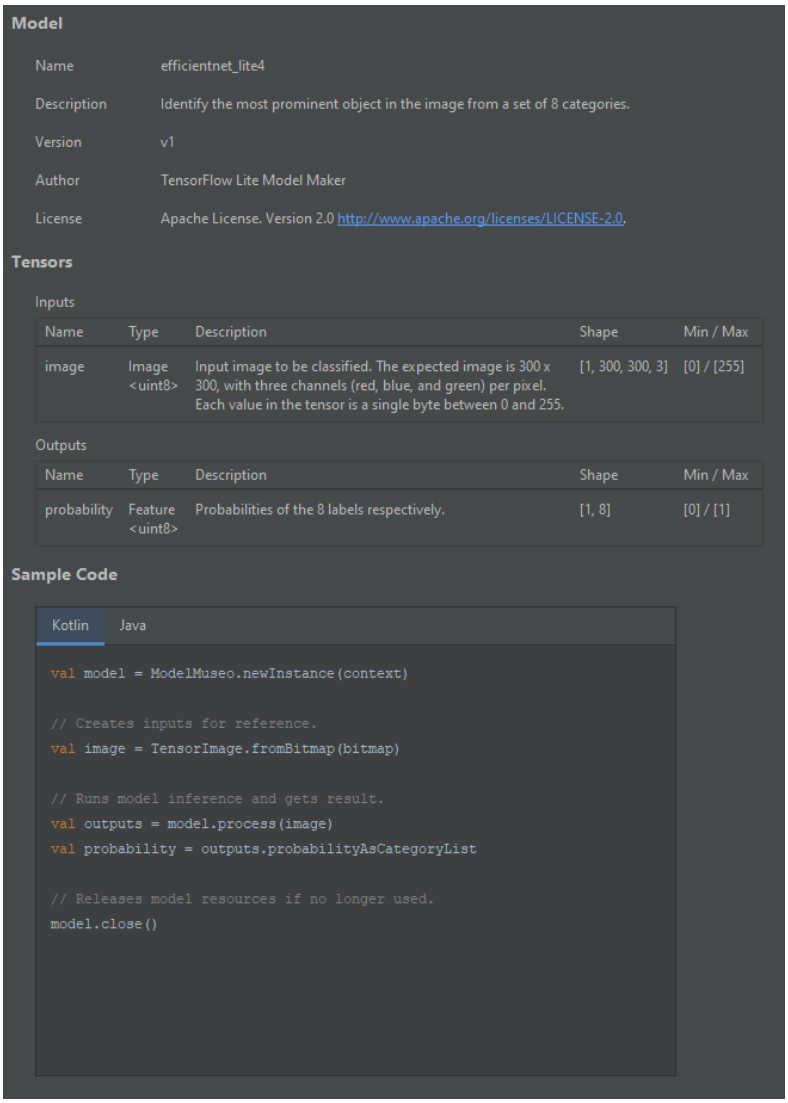


Ilustración 11-Modelo tflite integrado en aplicación Android

3.4 ENTORNO ANACONDA 3 y JUPYTER NOTEBOOK

Anaconda [5] es una distribución *Python* desarrollada por la compañía Anaconda Inc. con el enfoque de dotar a desarrolladores, especialmente científicos de datos, de una plataforma donde realizar investigaciones y desarrollos en los campos de la ciencia de datos e inteligencia artificial.

Entre sus características más destacadas encontramos que es multiplataforma por lo que las aplicaciones creadas en un sistema podríamos ejecutarlas en cualquier otro que tenga la distribución. Como parte importante cuenta con un gestor de paquetes (*Conda*), dependencias y entornos internos, para la descarga y administración de paquetes englobados en *Anaconda* o externos o trabajar en

[5] **Anaconda**, distribución Python para la ciencia de datos. https://www.anaconda.com/

entornos dependientes a la ejecución del sistema operativo, no interfiriendo con este [46].

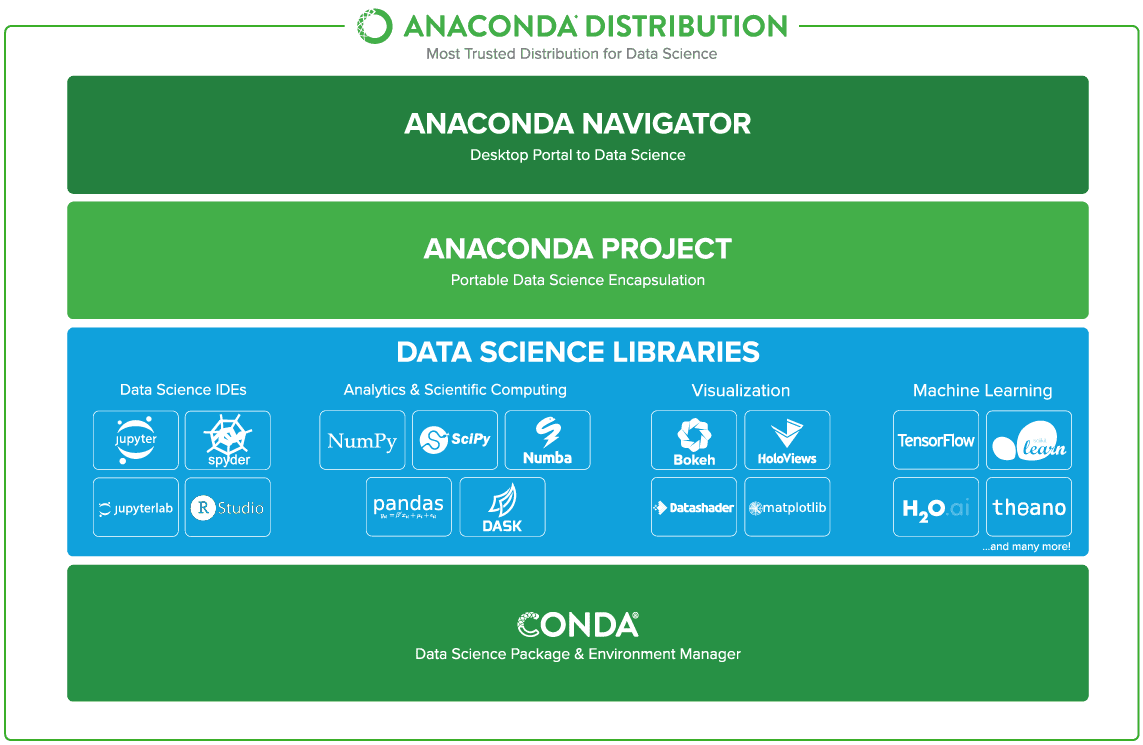


Ilustración 12-Estructura distribución Anaconda [46]

Debido a su enfoque hacía el trabajo con inteligencia artificial y la posibilidad de generar un entorno local independiente al sistema operativo, es la herramienta más propicia para realizar un entrenamiento en el propio equipo, pudiendo instalar fácilmente las dependencias necesarias para su funcionamiento mediante la propia distribución.

Jupyter Notebook[6] es una aplicación cliente-servidor puesta a disposición pública por la organización Proyecto Jupyter en 2015 destinada a la realización de cálculos, es por ello por lo que se ha convertido en una gran herramienta ligada a la ciencia de datos.

Esta aplicación se ejecuta en un navegador, permitiendo la realización de cuadernos en el que la ejecución se divide en celdas de operaciones y en celdas de resultado, permitiendo modularizar la ejecución de un script, teniendo un resultado visual por cada celda ejecutada siempre que así se disponga.

Aunque ampliamente se use para ejecución de código en lenguaje Python, la configuración del kernel que realiza las operaciones descritas en las celdas, puede ser

[6] **Jupyter Notebook**, plataforma computacional interactiva basada en web. https://jupyter.org/

configurado en múltiples diferentes lenguajes de programación, dotando a la herramienta de un mayor alcance [47].

Debido a las características mencionadas será la API que se utilizará en el entorno levantado localmente, para la ejecución de aumentos del dataset, entrenamiento del modelo y comprobación de resultados. La funcionalidad de la ejecución por celdas dotará de un mayor seguimiento al flujo del entrenamiento, pudiendo guardar posteriormente para futuras consultas el resultado de la ejecución completa.

Esta combinación de herramientas es la alternativa al entrenamiento del modelo de manera online en la nube, permitiendo un mayor control y seguimiento de la ejecución de todo el proceso, sin limitación de tiempo. Por contraparte en función del hardware a disposición los procesos se pueden alargar considerablemente en el tiempo o incluso ni iniciarse si no se tiene suficiente capacidad para abarcar las operaciones a realizarse.

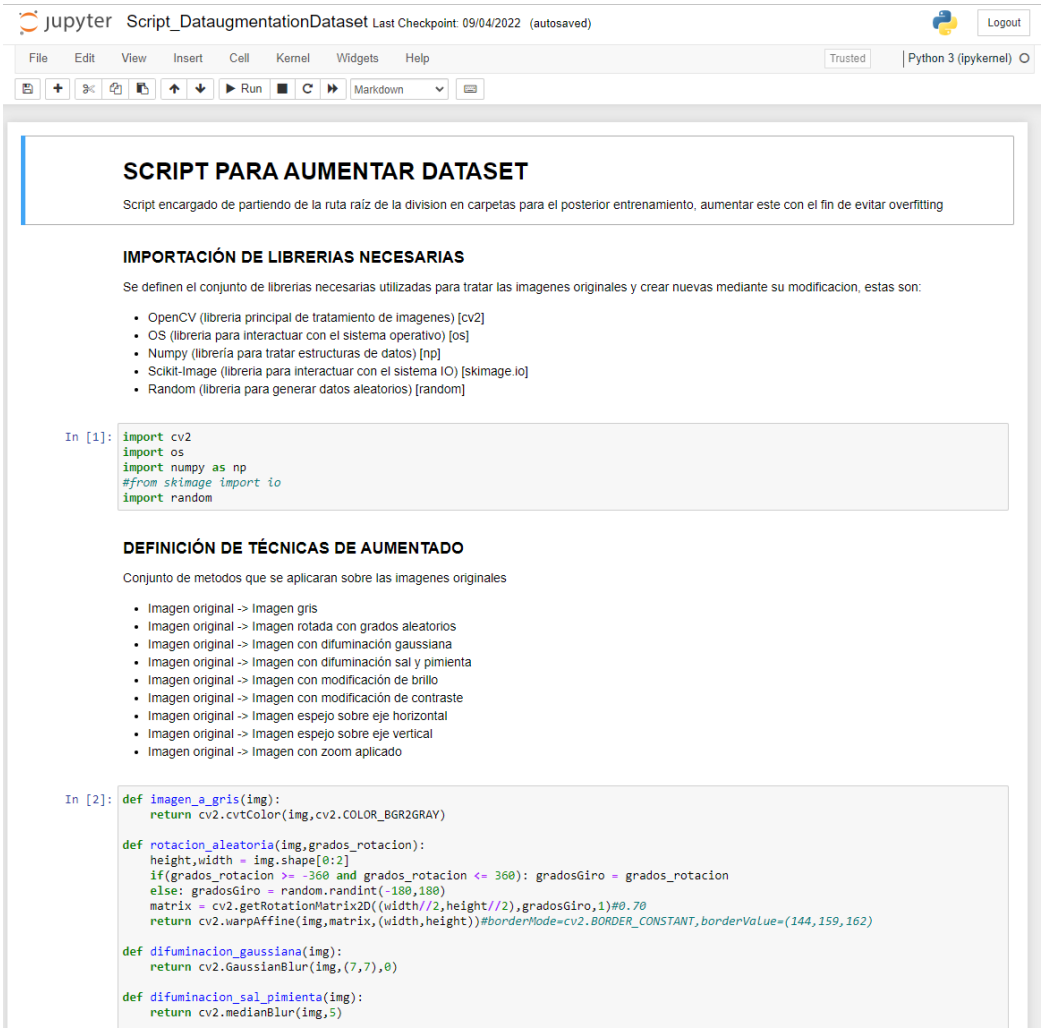


Ilustración 13-Interfaz Jupyter Notebook con cuaderno con resultados de ejecución guardados

Antes de entrar a explicar las etapas en las que se ha divido el proceso se explicará brevemente la creación del entorno local Anaconda en el que se ejecutan diversos scripts para la aumentación del dataset, entrenamiento del modelo…

3.4.1 CONFIGURACIÓN ENTORNO ANACONDA

Para desarrollar la etapa de diseño del dataset y entrenamiento del modelo al no tener un IDE concreto de desarrollo, como se tiene en la fase de programar la aplicación Android, se opta por crear un entorno local de trabajo donde programar y ejecutar los scripts necesarios.

La solución como se comenta en el anterior apartado de herramientas es la creación de un entorno local *Anaconda* donde ejecutar cuadernos *Python* mediante *Jupyter Notebook* para escribir y ejecutar nuestro código por celdas, una manera efectiva de dividir la ejecución del script e ir obteniendo *feedback*.

A la hora de crear el entorno se listan las librerías imprescindibles, no comunes (como pueden ser *numpy*, *matplotlib*…), necesarias para la realización de los scripts necesarios; estas son las siguientes:

Librería TensorFlow orientada a GPU [tensorflow-gpu]

Librería fundamental para el uso posterior de su librería descendiente TensorFlow Lite Model Maker para soportar y manejar el proceso de entrenamiento del modelo.

Herramientad CUDA [Cudatoolkit]

Herramienta adicional para el uso de la aceleración hardware de la unidad GPU que se dispone para reducir el tiempo de entrenamiento del modelo.

Herramienta Jupyter notebook [jupyter notebook]

Herramienta para la elaboración y ejecución de los scripts basado en un núcleo kernel.

Librería de TensorFlow Lite Model Maker [tflite-model-maker]

Librería para importar un modelo preentrenado de un clasificador de imágenes para reentrenarlo en el nuevo dataset que se utiliza.

Librería de OpenCV [scikit-image]

Librería para operar con las imágenes del dataset y poder elaborar el proceso de aumentación del dataset original.

Con el listado definido se procede a crear un entorno indicando por parámetros las librerías y herramientas que se precisan, encargándose el propio entorno de encontrar las versiones de librerías compatibles para crear un entorno con todo lo especificado. Tras determinar todas las sublibrerias necesarias y la compatibilidad entre ellas se inicia la preparación del entorno.

```
(base) PS C:\Users\BRKAO> conda create -n tfg_env tensorflow-gpu cudatoolkit jupyter notebook
```

Ilustración 14-Creación del entorno local Anaconda con las librerías y herramientas necesarias por parámetro

Una vez creado el entorno, para poder usarlo hay que activarlo y ya se opera dentro de este, pudiendo utilizar todos los recursos instalados con la creación del entorno.

```
(base) PS C:\Users\BRKAO> conda activate tfg_env
```

Ilustración 15-Activación del entorno local Anaconda

Dentro del entorno ya resta instalar mediante el uso de la herramienta *pip* tras actualizarla para asegurarse su mejor funcionamiento, la librería de TensorFlow Model Maker. Debido a una incompatibilidad que presenta con la versión instalada de TensorFlow al crear el entorno hay que reducir la versión de su librería *Keras*, parte del soporte de la inteligencia artificial de la API, a su versión 2.6.

```
(tfg_env) PS C:\Users\BRKAO> pip install --upgrade pip --user
```

Ilustración 16-Actualización de herramienta pip

```
(tfg_env) PS C:\Users\BRKAO> pip install tflite-model-maker --user
```

Ilustración 17-Instalación de la librería TensorFlow Lite Model Maker

```
(tfg_env) PS C:\Users\BRKAO> pip install keras==2.6.* --user
```

Ilustración 18-Reducción versión librería Keras por incompatibilidad

Finalmente, con la instalación de la librería de OpenCV para realizar el aumento del dataset, el entorno ya está completamente listo para soportar la ejecución de los scripts para crear el modelo de inferencia.

```
(tfg_env) PS C:\Users\BRKAO> pip install scikit-image --user
```

Ilustración 19-Instalación de librería de OpenCV

Para iniciar la escritura o la ejecución de estos cuadernos Python se ejecuta Jupyter Notebook, dándonos la interfaz perfecta de trabajo.

```
(tfg_env) PS C:\Users\BRKAO> jupyter notebook
```

Ilustración 20-Ejecución interfaz de trabajo Jupyter Notebook

3.5 IDE ANDROID STUDIO

Android Studio [7] es el IDE de desarrollo oficial del sistema operativo Android enfocada para sacar el mayor rendimiento al trabajo del programador y dotándola de múltiples herramientas de simulación, pruebas, *plugins*…Soporta los lenguajes de programación de Java, C++ y Kotlin.

El entorno ofrece una interfaz sencilla y funcional con diversas utilidades como la ayuda en tiempo real al programar mediante IntelliJ, plantillas para desarrollar soluciones Android para cualquier tipo de dispositivo, control de versiones, interfaz para el desarrollo del diseño visual…entre otras muchas características, convirtiéndolo en un gran entorno para el desarrollo de inicio a final de aplicaciones Android, teniendo al alcance todos los recursos y herramientas necesarias [48].

Otro destacado servicio del entorno es el AVD. Esta funcionalidad se encarga de simular un dispositivo móvil con las características que se preconfiguren, con el fin de probar el funcionamiento de la aplicación en desarrollo en un dispositivo final; por ello se permite al usuario refinar la ejecución, pudiendo observar mediante la simulación el comportamiento del programa en caso de no disponer de un dispositivo para la realización de pruebas. En caso de disponer de un dispositivo, otra función de gran interés relativa a la simulación es la de depuración por usb, permitiendo al igual que en la simulación acceder al registro de eventos y log, pudiendo revisar el buen funcionamiento, pero en este caso la ejecución se realiza en un dispositivo real obteniendo resultados más verídicos.

[7] **Android Studio**, IDE oficial de desarrollo Android. https://developer.android.com/studio

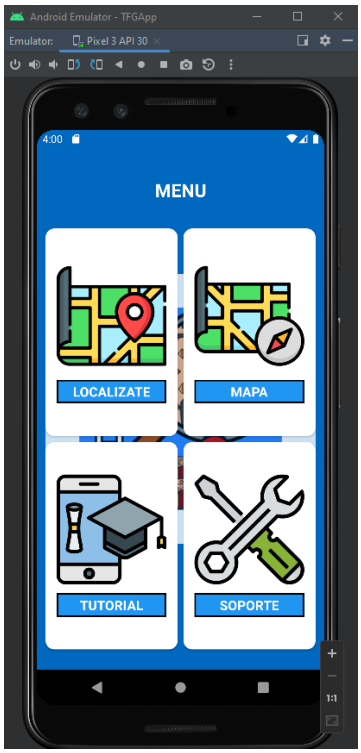


Ilustración 21-AVD Android Studio durante pruebas de aplicación

Esta herramienta es el segundo pilar del proyecto, tras la elección de herramientas para obtener el clasificador de imágenes, usándose para desarrollar en los aspectos lógicos y de diseño la aplicación móvil que alojará el modelo de inferencia, dando la posibilidad al usuario final de acceder a su funcionalidad de clasificación.

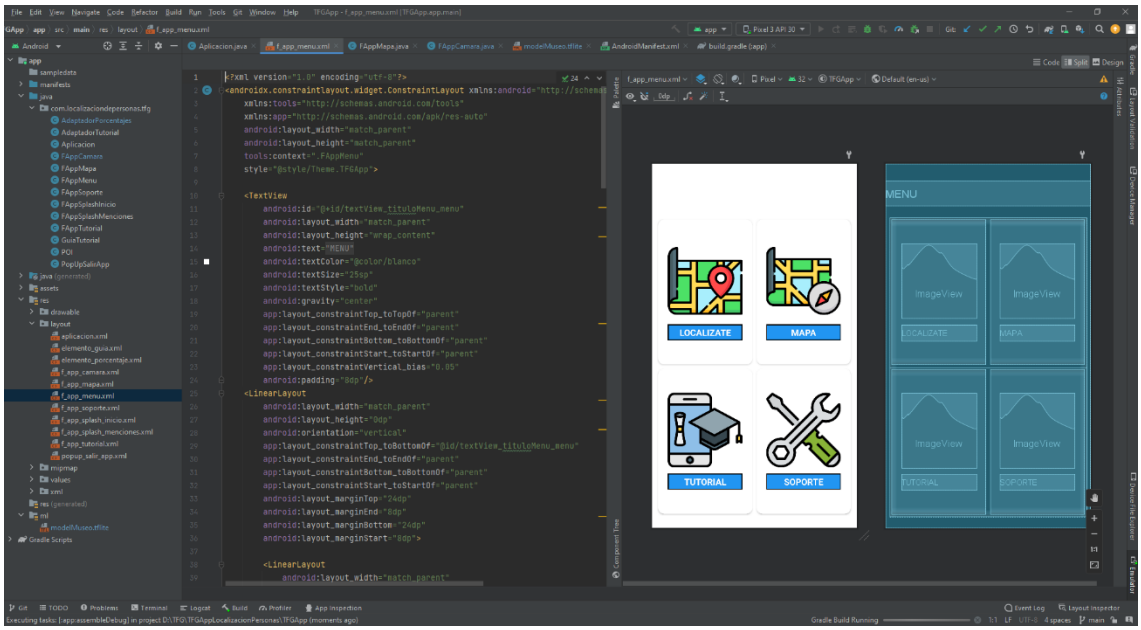


Ilustración 22-IDE Android Studio durante programación de la aplicación

3.5.1 MAPBOX MAPS SDK PARA ANDROID

Mapbox Maps SDK for Android [8] es una librería destinada al soporte y uso de mapas en dispositivos móviles, que permiten la aplicación de múltiples diseños y estilos para adaptarse de manera eficiente a la resolución del problema que cubren. Debido a su versatilidad es la alternativa líder en el mercado a Google Maps para el trabajo en aplicaciones móviles de mapas, para localización, navegación…[49]

Entre las características que presentan destaca la elevada posibilidad de personalización del mapa, permitiendo adaptarlo al máximo a la funcionalidad prevista. Otro gran detalle es que la funcionalidad esta embebida en la propia aplicación donde se implementa, no teniendo que contar con programas de terceros y abriendo la puerta a la ejecución offline. Parte de los elementos de diseño y puntos de interés (POI) que se quieren reflejar en los mapas pueden ser cargados mediante ficheros geojson de manera sencilla, pudiendo personalizar el resultado mostrado en el mapa, con información que el mapa cargado no muestra [50].

En este TFG es una parte fundamental para la visualización del resultado, debido a que se representa el mapa del entorno en el cual el usuario puede navegar libremente, informándose de los puntos de interés representados por iconos sobre el mapa y en caso de que se realice un proceso de inferencia, se mostrará al usuario la ubicación aproximada donde se encuentra.

La elección de esta herramienta aparte de las buenas prestaciones que aporta se apoya en los trabajos de investigación del proyecto “Sistema de guiado y localización en interiores mediante iluminación de edificios” (Ref. GUIA: SBPLY/19/180501/000049) del programa de investigación científica y proyectos de transferencia de tecnología de la Junta de Comunidades de Castilla La Mancha .

Este proyecto consiste en el desarrollo de una aplicación Android que usando el SDK de Mapbox muestra un mapa completo del Museo Provincial de Guadalajara, con el que el usuario puede explorar libremente.

Por tanto, la utilidad de manejo de mapas se basa en este proyecto (repositorio público “MuseoIndoorPosition”) [9], destacando los archivos *geojson* que recogen los trazos del mapa del museo, como se configura y maneja un mapa cargando su información desde archivos *geojson*; y el control de eventos sobre el mapa. Partiendo

[8] **Mapbox Maps SDK**, API de Mapbox para Android. https://docs.mapbox.com/android/maps/guides/
[9] **MuseoIndoorPosition**, Repositorio Github. https://github.com/dquezadag/MuseoIndoorPosition

de esta base se implementarán funciones y configuraciones adaptadas al problema a resolver.

Los nombres de usuario de dichos estudiantes son “dquezadag” y “darwinquezada”, presentando el proyecto.

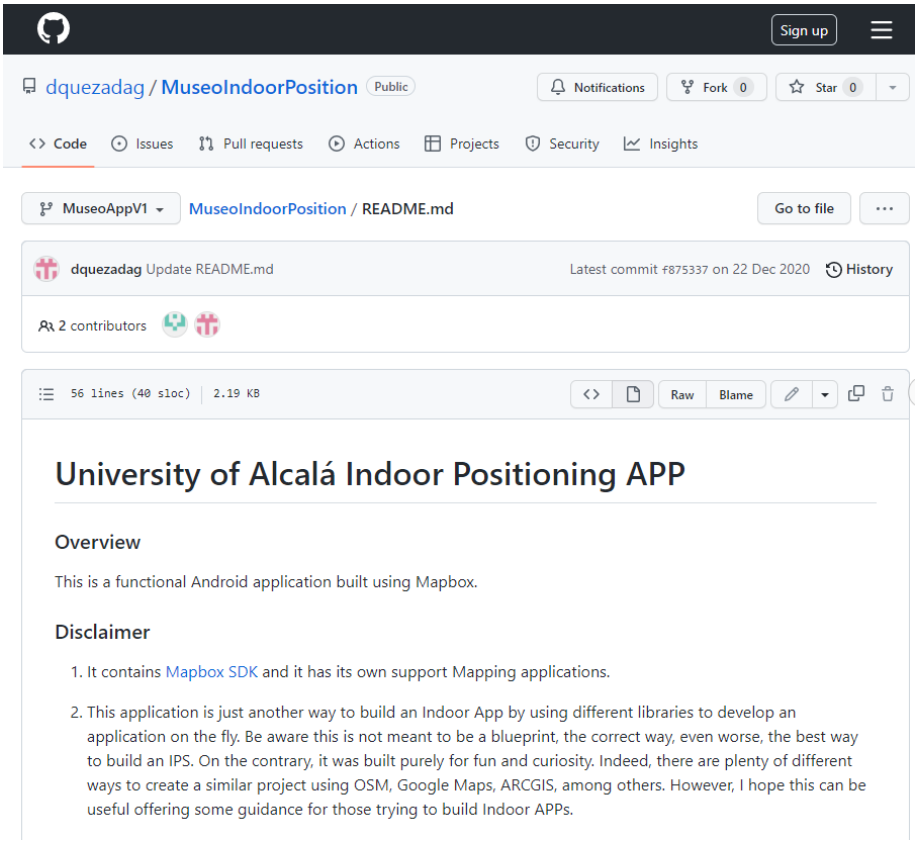


Ilustración 23-Repositorio público del proyecto

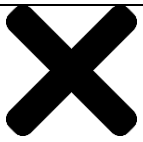
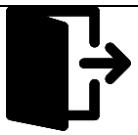
Ilustración 24 - Logo Mapbox

3.5.2 RECURSOS DISEÑO

A la hora de elaborar el diseño para que sea intuitivo y atractivo de cara al usuario de la aplicación, es necesario un trabajo extra a la hora de plantear las pantallas e iconos para que el usuario no tenga inconvenientes en su uso.

Para ello existen repositorios de iconos gratuitos para añadir a las aplicaciones sin ningún problema adicional, permitiendo su uso en proyectos con la correspondiente atribución en el desglose de recursos empleados. En este caso se ha usado la página web de *Flaticon* [10], dedicada para la descarga de iconos vectoriales y stickers para los proyectos en los formatos que se requieran, teniendo una gran variedad de diseños.

Como requisito para su uso libre se declara que hay que realizar mención de los artistas de los iconos que se usen, por lo que se presenta un breve listado enumerando los iconos descargados de FlatIcon y el nombre de usuario de los autores.

IMAGEN	AUTOR	IMAGEN	AUTOR
	Freepik		Pixel perfect
	Freepik		Pixel perfect
	Freepik		Smashicons
	Freepik		Tanah Basah
	justicon		Tanah Basah
	monkik		Uniconlabs

[10] **FlatIcon**, página web de iconos gratuitos. https://www.flaticon.es/

En complemento con los iconos obtenidos del sitio web, se ha usado editores de código libre como lo es *GIMP 2* [11] o *InkScape* [12], para la creación de ciertos recursos, la redimensión de recursos de terceros y últimos retoques de diseño que se pueden aportar de manera externa a la aplicación, mediante técnicas de aplicación de capas, guías, plantillas...

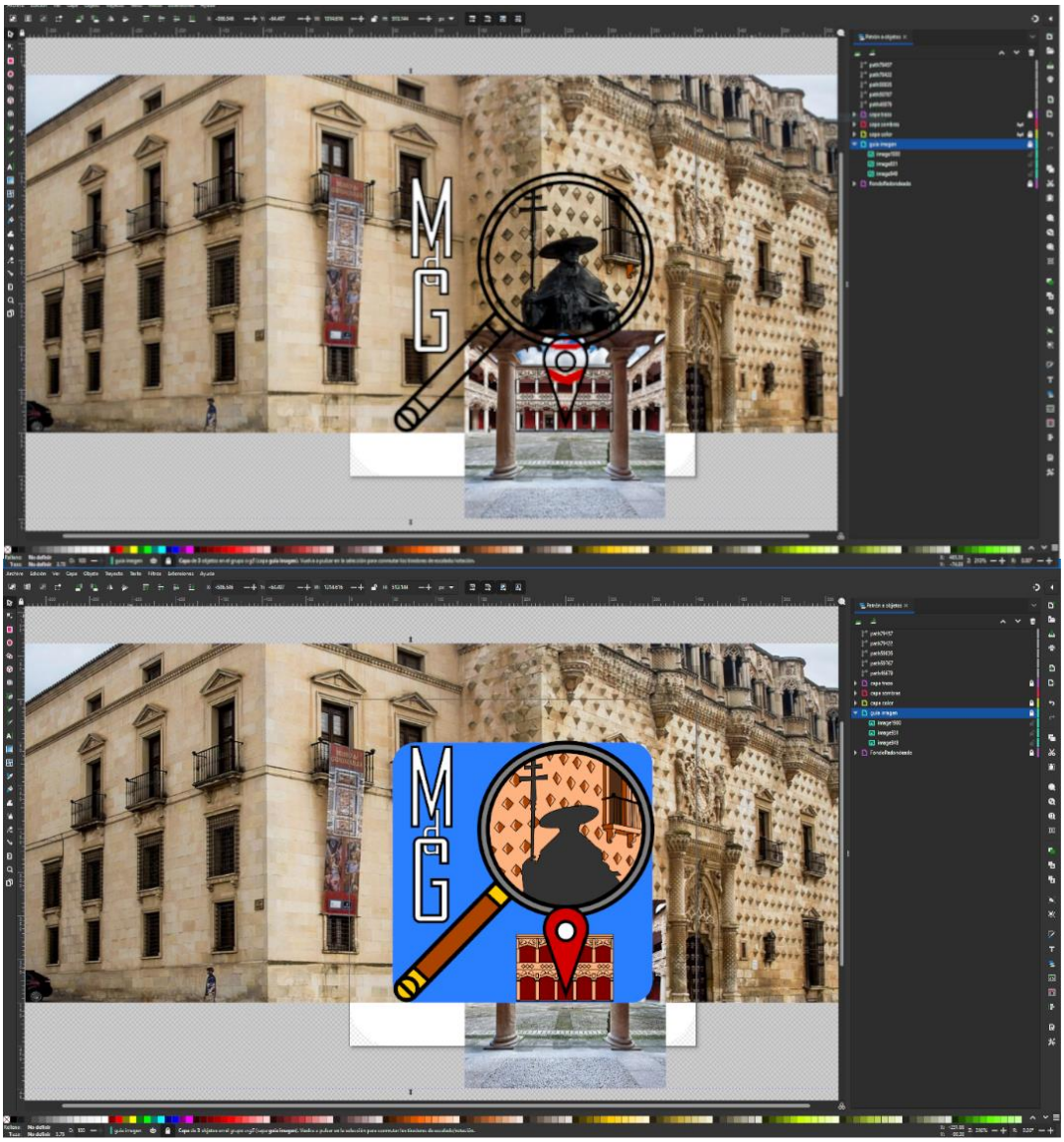


Ilustración 25-Creación de logo de la aplicación con el editor InkScape

[11] **Gimp 2**, editor de imágenes y retoques. http://www.gimp.org.es/
[12] **InkScape**, editor de vectores gráficos. https://inkscape.org/es/

3.6 RELACIÓN Y USO DE LAS HERRAMIENTAS NECESARIAS

Para entender el uso y aplicación de cada una de las herramientas a continuación se realiza un mapa para relacionarlas con su uso en el TFG. Este se dividirá en dos partes, basándonos en las principales etapas de desarrollo realizadas, la creación del modelo de inteligencia artificial y la creación de la aplicación final.

Para la creación del clasificador de imágenes siempre se trabajará en el entorno Anaconda, configurado como previamente se ha explicado, mediante la interfaz que nos ofrecerán los cuadernos de Jupyter Notebook. La creación del modelo de inferencia se divide en dos pasos, realizando un script independiente por cada uno. Primero se ejecutará el script dedicado a la aumentación del dataset, cuyo resultado utilizará el script consecutivo para el entrenamiento y exportación del modelo.

Siguiendo el encapsulamiento de herramientas debido a su uso, este resulta en una organización representada en el siguiente esquema, para la elaboración del clasificador de imágenes.

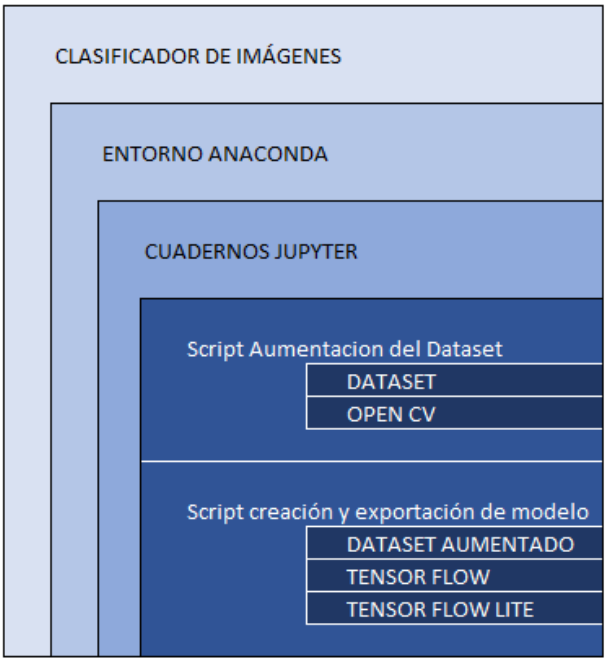


Ilustración 26-Esquema de relación y uso de las herramientas para la creación del clasificador de imágenes

Continuado con la segunda etapa, la aplicación se implementará dentro del entorno de desarrollo de Android Studio, segmentando su elaboración en la parte funcional de la misma (Lógica) y en la parte visual e interactiva de cara al usuario

(Diseño). Como parte destacable en la realización de la lógica es el uso de la librería de Mapbox para recoger la interacción del usuario sobre el mapa y la librería integrada en el propio entorno para el soporte e integración de los modelos de inteligencia artificial. Por último, para el diseño de las pantallas se hace uso de editores para crear y retocar imágenes en combinación de iconos descargados del sitio web Flaticon; también hay que hacer mención del elemento visual donde se cargan los mapas de la librería externa Mapbox.

El esquema resultante de la relación de las herramientas mencionadas sería el representado en la siguiente ilustración.

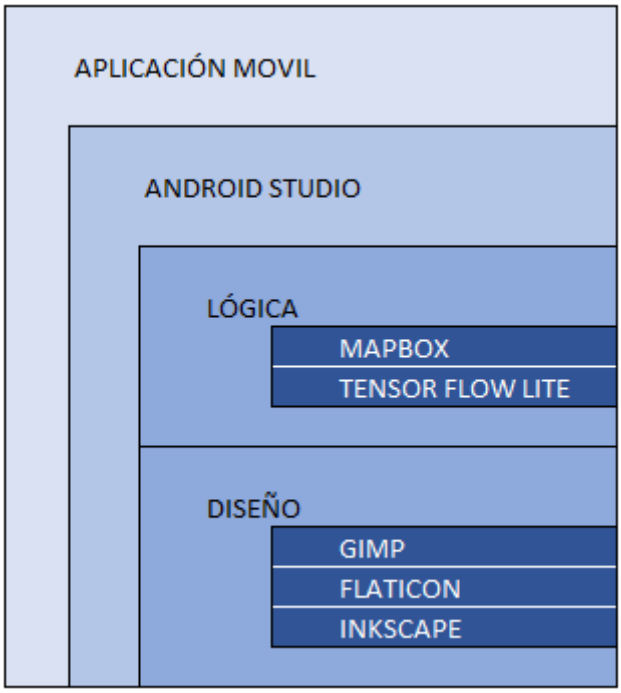


Ilustración 27-Esquema de relación y uso de las herramientas para la creación de la aplicación móvil

4 DESARROLLO

El desarrollo del TFG se ha realizado en diversas fases las cuales recogen la generación de las partes necesarias para definir el flujo de la aplicación y conseguir que esta sea funcional. Estas tareas serán explicadas en el orden cronológico del desarrollo, siendo la manera óptima para alcanzar el objetivo final. El resumen de las fases se muestra visualmente en la siguiente gráfica, con una estimación de la complejidad e importancia de cada apartado a desarrollar.

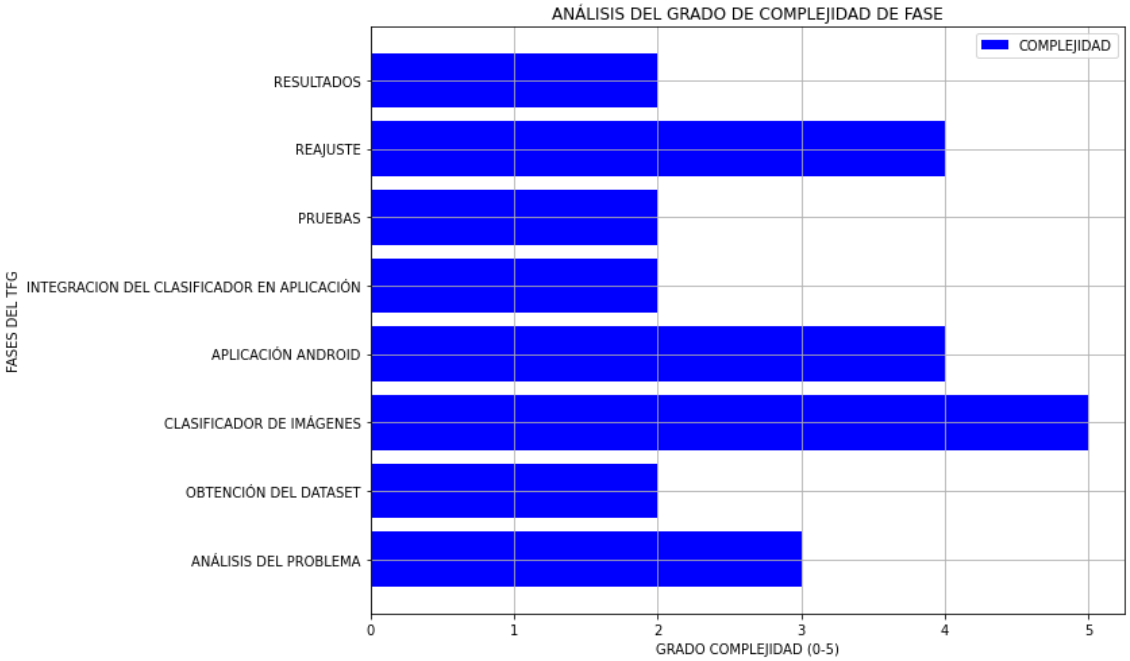


Ilustración 28-Análisis del grado de complejidad de cada fase

Como se resalta en la figura se ha catalogado cada fase dentro de una escala del 0 al 5, siendo el 5 la mayor complejidad. Como fases más críticas y complejas del desarrollo se encuentran las encargadas de crear el modelo de inferencia y la aplicación contenedora de esta ya que concentran el peso principal del desarrollo. Complementando estas fases también se encuentra en la misma escala el reajuste tras la primera prueba experimental, ya que tras las observaciones que se realicen el reajuste de modelo y aplicación supone el final del proyecto.

Con un grado menor de complejidad, pero no menos importantes se encuentran el resto de las tareas intermedias e iniciales como el análisis del problema. Como fase más destacable dentro de este grupo está el análisis del problema, hecho fundamental para establecer una hoja de guía y bases sólidas para la elaboración de la solución propuesta en el TFG.

El análisis del problema se considera explicado en los apartados anteriores partiendo del marco teórico necesario para su elaboración y búsqueda de las herramientas para conseguirlo. Por consiguiente, se continua la explicación a partir de la fase de la obtención del dataset.

4.1 OBTENCIÓN DEL DATASET

Como se ha explicado anteriormente entre el marco teórico de la inteligencia artificial y la herramienta del dataset, la creación de este es el primer paso que dar en la creación de una solución basada en un modelo de inteligencia artificial. Debido a la

necesidad de que el modelo sea un clasificador de imágenes para interpretar el entorno que rodea al usuario, la fuente de información suministrada serán imágenes del entorno.

La primera tarea antes de recoger las imágenes del entorno sería la planificación e identificación de los lugares de interés que pasaran a denominarse puntos de interés (POI) con la finalidad de limitar la recolección. Esta planificación se realiza mediante la observación del entorno, delimitando las áreas que cada POI comprende y buscando los elementos más característicos de cada una de estas áreas, de los cuales obtendremos las imágenes para el entrenamiento.

En el caso de experimentación del Museo de Guadalajara como áreas divisibles para la localización y asignación de POIs se escogen las distintas salas que conforman el museo, siendo un total de 7 salas. A la hora de poder identificar estas salas de tomará como referencia las vitrinas/obras que se encuentran en cada una de las salas.

Ilustración 29-Ejemplo de una vitrina del Museo Provincial de Guadalajara

Con estas premisas se recorre cada vitrina/obra del museo obteniendo varias imágenes, desde distintos ángulos y perspectivas para cubrir de una forma más adecuada una representación lo más completa de cada vitrina/obra; anotando la vitrina/obra fotografiada con un identificador numérico incremental y la sala en la que se encuentra. Tras finalizar la recolección los resultados son los siguientes:

SALAS	N.º IMÁGENES	N.º VITRINAS/OBRAS
Vestíbulo II	283	29
Colección Permanente I	111	18

Colección Permanente II	48	6
Colección Permanente III	111	18
Colección Permanente IV	41	7
Escipión	78	16
Ola	45	6
Atlanta	19	8
TOTAL	736	108

Tras tener las imágenes restaría organizarlas por directorios cuyo título representará la etiqueta con la que se quiere identificar cuando el modelo interprete que la imagen suministrada corresponde al conjunto de imágenes agrupado.

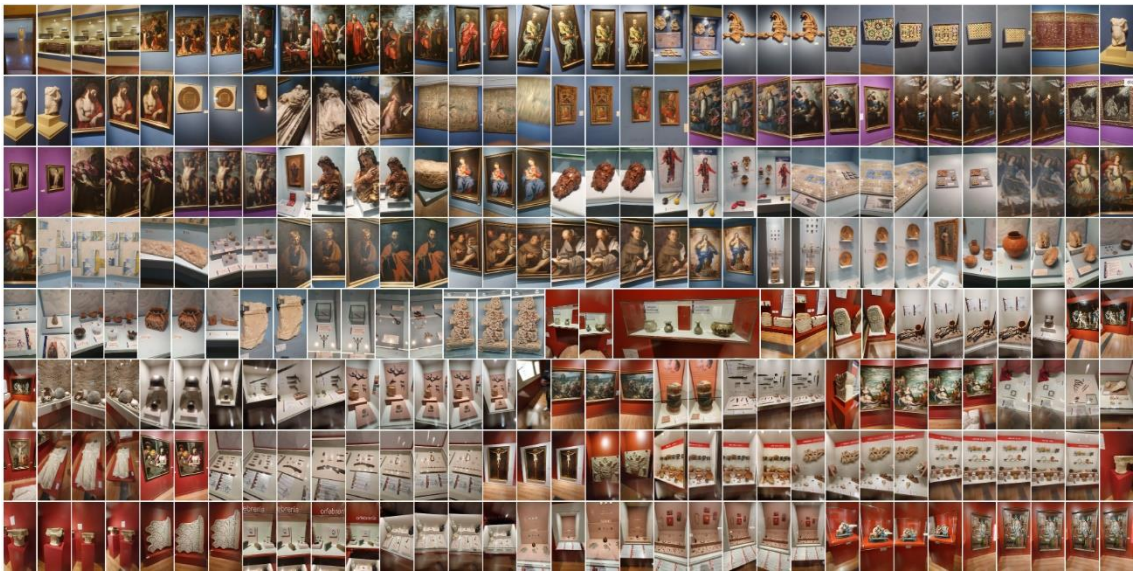


Ilustración 30-Muestra de diferentes imágenes capturadas de las obras del Museo Provinicial de Guadalajara

Llegado a este punto se plantean dos enfoques posibles de clasificación, hecho que se refleja en la estructuración del dataset que se le suministra en la etapa de entrenamiento. De ambos enfoques se busca interpretar la sala en la que nos encontramos, pero se puede llegar a esta conclusión de maneras distintas expuestas a continuación:

ENFOQUE DE CLASIFICADOR POR SALAS

El enfoque de clasificador por salas propone una solución más sencilla ya que la salida directa del clasificador será la probabilidad de en qué sala se

encuentra el usuario. Esto se traduce en una estructura del dataset dividido en tantos directorios como salas, 7 en el caso de experimentación, donde en cada uno se agrupan todas las imágenes relativas a las distintas vitrinas/obras que se encuentran en dicha sala.

Ilustración 31-Muestra de organización del dataset para el enfoque del clasificador por salas

Por tanto, esto es una agrupación de datos sencilla, pero se produce una generalización debido a que la inteligencia artificial buscará un patrón común en todas las imágenes del directorio para identificar la etiqueta. Esto en el caso de uso de un museo que en una misma sala existe una gran variedad de vitrinas/obras puede ser una causa de perdida de precisión a la hora de ajustar el modelo en el entrenamiento. Por otro lado, si hay espacios muy distintos donde los elementos son pequeños, pero de cierto parecido, puede ser útil.

ENFOQUE DE CLASIFICADOR POR VITRINAS

El enfoque de clasificador por vitrinas propone una solución más compleja ya que la salida directa del clasificador será la probabilidad de en qué vitrina/obra se encuentra el usuario, por lo que hay que establecer una relación entre vitrina/obra y la sala en la que se encuentra. Esto se traduce en una estructura del dataset dividido en tantos directorios como vitrinas/obras, 108 en el caso de experimentación, donde en cada uno se agrupan todas las imágenes relativas a la vitrina/obra en cuestión. Finalmente, en el nombre del directorio se creará indicando el número asignado a la vitrina, junto con la sala a la que pertenece.

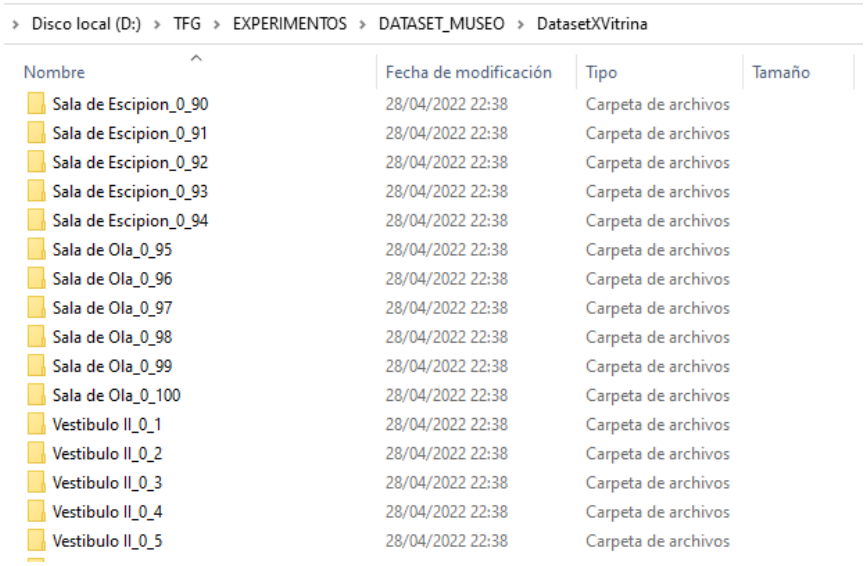


Ilustración 32-Muestra de organización del dataset para el enfoque del clasificador por vitrinas

Por tanto, esto es una mayor división de datos en conjuntos más pequeños, consiguiendo una solución más específica ya que la inteligencia artificial no debe generalizar entre distintos tipos de vitrinas/obras, sino que es una única, teniendo un patrón característico propio.

Debido a los dos enfoques se decide realizar dos clasificadores para posteriormente en el apartado de experimentación enfrentarlos para comprobar que enfoque presenta mejores resultados, por tanto, se diseñan dos dataset para cada modelo.

Una vez con los dataset creados, debido a la cantidad limitada de imágenes que se pueden recoger del museo se opta por realizar un proceso de aumento del dataset, con el fin de que el entrenamiento conste de más imágenes para perfeccionar la futura predicción. Para ello se realiza un script en el entorno local Anaconda que se ejecutará con cada dataset, creando nuevas imágenes a partir de las originales tras aplicarlas un filtro. Tras la ejecución del script cada uno de los dataset multiplicará su número de imágenes por 20 pasando a tener 9300 imágenes en el dataset que se le suministra al proceso de entrenamiento.

El script consta de una primera parte donde se definen los métodos que aplicaran un tipo de filtro sobre la imagen original, ya que algunos métodos de la librería de OpenCV requieren más de una línea de código para obtener una nueva imagen transformada.

```python
# Metodos para crear nuevas imagenes a partir de la original
def imagen_a_gris(img):
    return cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

def rotacion_aleatoria(img,grados_rotacion):
    height,width = img.shape[0:2]
    if(grados_rotacion >= -360 and grados_rotacion <= 360): gradosGiro = grados_rotacion
    else: gradosGiro = random.randint(-180,180)
    matrix = cv2.getRotationMatrix2D((width//2,height//2),gradosGiro,1)#0.70
    return cv2.warpAffine(img,matrix,(width,height))

def difuminacion_gaussiana(img):
    return cv2.GaussianBlur(img,(25,25),0)

def difuminacion_sal_pimienta(img):
    return cv2.medianBlur(img,25)

def brillo(img):
    brilloAleatorio = random.randint(-127,127)
    return cv2.addWeighted(img,1,np.zeros(img.shape,img.dtype),1,brilloAleatorio)

def contraste(img):
    contrasteAleatorio = round(random.uniform(1.0,3.0),1)
    return cv2.addWeighted(img,contrasteAleatorio,np.zeros(img.shape,img.dtype),0,0)

def espejo_horizontal(img):
    return cv2.flip(img,0)

def espejo_vertical(img):
    return cv2.flip(img,1)

def zoom(img):
    # Cortamos imagen
    height,width = img.shape[0:2]
    startRow = int(height*.15)
    startCol = int(width*.15)
    endRow = int(height*.85)
    endCol = int(width*.85)
    # Reescalamos
    return img[startRow:endRow,startCol:endCol]
```

Ilustración 33-Muestra de uso de la librería OpenCV para implementar métodos para aumentar el dataset de imágenes

Una vez que hemos definido los métodos que devolverán una nueva imagen al dataset derivada de la original (9 tipos de filtros implementados), debido a la estructura del dataset se realiza un conjunto de bucles anidados para recorrer cada carpeta y cada imagen suministrando como punto de partida la carpeta raíz de donde descuelgan las carpetas con las imágenes de las salas o vitrinas. Cuando se llega a una imagen, esta se transforma a una matriz para poder operar sobre ella con los métodos declarados y se la aplica un conjunto de modificaciones haciendo uso de estas funciones con distintos parámetros. El resultado de cada modificación es una nueva imagen que se almacena en el mismo directorio que la imagen original para no tener que reubicarla posteriormente. Al aplicarse a cada imagen original un total de 19 operaciones se consigue el efecto de aumento de 20 veces mayor al tener la imagen original y sus nuevas derivadas.

```
# Imagen a gris
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_gris.jpg"),imagen_a_gris(img))
# Rotacion aleatoria
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_rotacion1.jpg"),rotacion_aleatoria(img,15))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_rotacion2.jpg"),rotacion_aleatoria(img,-15))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_rotacion3.jpg"),rotacion_aleatoria(img,30))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_rotacion4.jpg"),rotacion_aleatoria(img,-35))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_rotacion5.jpg"),rotacion_aleatoria(img,45))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_rotacion6.jpg"),rotacion_aleatoria(img,-45))
# Difuminacion
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_difGauss.jpg"),difuminacion_gaussiana(img))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_difSyP.jpg"),difuminacion_sal_pimienta(img))
# Brillo
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_brillo1.jpg"),brillo(img))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_brillo2.jpg"),brillo(img))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_brillo3.jpg"),brillo(img))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_brillo4.jpg"),brillo(img))
# Contraste
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_contraste1.jpg"),contraste(img))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_contraste2.jpg"),contraste(img))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_contraste3.jpg"),contraste(img))
# Espejo
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_espejoH.jpg"),espejo_horizontal(img))
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_espejoV.jpg"),espejo_vertical(img))
# Zoom
cv2.imwrite(os.path.join(ruta_dataset,carpetaClase,nombreReformateo[0]+"_zoom.jpg"),zoom(img))
```

Ilustración 34-Muestra de uso de métodos implementados para el aumento del dataset, aplicando diversos efectos

Llegados a este punto los dataset se dan por preparados y se inicia la siguiente etapa del entrenamiento de los clasificadores. Paso a previo a la siguiente etapa se realiza un breve resumen de cada filtro implementado para el aumento del dataset y la imagen derivada que genera [51].

Imagen a escala de grises (1 Aplicación)

Con el método definido de `imagen_a_gris` se obtiene una imagen nueva en escala de grises derivada de la original.

Esta operación se realiza mediante el método `cvtColor` de la librería la cual se encarga de realizar transformaciones de color sobre una matriz de entrada. Usando el parámetro `COLOR_BGR2GRAY` realiza la transformación del formato RGB de la imagen original a escala de grises.

Ilustración 35-Ejemplo de modificación por escala de grises para la aumentación del dataset

Rotación aleatoria con eje en el centro de la imagen (6 Aplicaciones)

Con el método definido de `rotación_aleatoria` se obtiene una imagen rotada respecto a su eje central y respecto a su posición original.

El método implementado permite definir cuantos grados rotar y en que sentido, dependiendo de si la cantidad es negativa (giro en sentido antihorario) o positiva (giro en sentido horario), la imagen original. Para evitar errores se comprueba que la cantidad facilitada esta comprendida en el rango entre -360º y 360º, en caso de ser superior o inferior se opta por asignar una rotación aleatoria en el nuevo rango entre -180º y 180º.

Con los grados de giro definidos se crea una primera matriz contenedora para la imagen original rotada a partir de suministrarle al método de la librería `getRotationMatrix2D` las coordenadas del centro de la matriz original, los grados de rotación y la relación a la hora de realizar el giro; en este caso se selecciona una relación lineal. Con la matriz contenedora resultante se pasa a utilizar una segunda función de la librería, la función `wrapAffine`, que nos devolverá la imagen rotada mediante la relación entre la matriz rotada y la original. Para obtener la imagen derivada se la pasa por parámetros la imagen original, el contenedor de la imagen rotada y finalmente se define el tamaño de la imagen resultado, siendo la dimensión de la imagen original.

En la práctica los giros planificados para el aumento del dataset respecto al eje central son de 15º, -15º, 30º, -30º, 45º y -45º. Estos giros se han seleccionado por las posibles inclinaciones más probables que puede realizar un usuario a la hora de realizar una fotografía.

Ilustración 36-Ejemplo de modificación por rotación respecto al eje central (30º) para la aumentación del dataset

Difuminación gaussiana de la imagen (1 Aplicación)

Con el método definido de `difuminacion_gaussiana` se obtiene una imagen con efecto borroso en comparación con la imagen original.

Esta operación se realiza haciendo uso del método `GaussianBlur` de la librería basado en la aplicación de un filtro gaussiano sobre la matriz original. A esta función se la facilita la imagen original, se especifica el tamaño del kernel para realizar las convoluciones y un valor para la derivación en el kernel gaussiano.

Ilustración 37-Ejemplo de modificación por difuminación gaussiana para el aumento del dataset

Difuminación sal y pimienta de la imagen (1 Aplicación)

Con el método definido de `difuminación_sal_pimienta` se obtiene una imagen con efecto borroso, distinto efecto borroso que la anterior difuminación explicada, respecto a la imagen original.

Esta operación se realiza mediante el método `medianBlur` de la librería, mediante la aplicación de un filtro que asigna a la celda que se encuentra en mitad del kernel el valor de la operación de la mediana resultante con las celdas restantes dentro del rango del filtro. Al método se le suministra por parámetros la imagen original y el tamaño con el que se formará el kernel.

Ilustración 38-Ejemplo de modificación por difuminación de sal y pimienta para el aumento del dataset

Modificación aleatoria del brillo de la imagen (4 Aplicaciones)

Con el método definido de `brillo` se obtiene una imagen con un brillo distinto al que tiene la imagen original.

Para obtener el efecto se obtiene un valor aleatorio comprendido entre -127 y 127, para modificar el brillo a más oscuro (valores negativos) o más luminoso (valores positivos). Con el valor aleatorio obtenido se hace uso del método `addWeighted` de la librería para lograr esta modificación de brillo a toda la imagen únicamente se modifica el valor de gamma de la imagen, siendo el quinto parámetro.

En la práctica al poder variar la iluminación del elemento a reconocer o el ajuste de luminosidad de la cámara se aplican 4 veces la función para obtener distintos brillos de la imagen

Ilustración 39-Ejemplo de modificación de brillo para la aumentación del dataset

Modificación aleatoria del contraste de la imagen (3 Aplicaciones)

Con el método definido de `contraste` se obtiene una imagen con un contraste modificado al de la imagen original.

Para variar el contraste de la imagen se obtiene un valor aleatorio comprendido ente 1 y 3, para variar la intensidad del contraste a aplicar. Con la intensidad definida se utiliza el método `addWeighted` de la librería para asignar la nueva intensidad de contraste modificando el valor de Alpha de la imagen original, pasando esta mediante el segundo parámetro.

En la práctica al poder variar la iluminación del elemento a reconocer o el ajuste de luminosidad de la cámara se aplican 3 veces la función para obtener distintos brillos de la imagen.

Ilustración 40-Ejemplo de modificación de contraste para la aumentación del dataset

Espejo sobre eje horizontal de la imagen (1 Aplicación)

Con el método definido `espejo_horizontal` se obtiene una imagen con efecto espejo respecto su eje horizontal de la imagen original.

La aplicación del efecto espejo se realiza con el método `flip` de la librería, el cual reordena en orden inverso las filas de la matriz original obteniendo este efecto. Simplemente por parámetros hay que indicar la imagen original y el valor 0 para identificar el efecto espejo horizontal.

Ilustración 41-Ejemplo de modificación por efecto espejo sobre eje horizontal para la aumentación del dataset

Espejo sobre eje vertical de la imagen (1 Aplicación)

Con el método definido `espejo_vertical` se obtiene una imagen con efecto espejo respecto su eje vertical de la imagen original.

La aplicación del efecto espejo se realiza con el método `flip` de la librería, el cual reordena en orden inverso las columnas de la matriz original obteniendo este efecto. Simplemente por parámetros hay que indicar la imagen original y el valor 1 para identificar el efecto espejo vertical.

Ilustración 42-Ejemplo de modificación por efecto espejo sobre eje vertical para la aumentación del dataset

Zoom al centro de la imagen (1 Aplicación)

Con el método definido de `zoom` se obtiene una imagen que es el resultado de aplicar efecto zoom al centro de la imagen original.

Este es el único método que no precisa del uso de funciones de la librería de *OpenCV*, ya que se basa en el uso de las propiedades de la imagen trabajada como matriz. Para obtener la nueva imagen en un primer lugar se obtiene la dimensión de la matriz, con la información del número de filas y columnas se definen las filas y columnas que delimitarán la nueva imagen aplicando una proporción respecto a la dimensión total. Finalmente se devuelve la matriz resultado al indicar que devuelva la matriz original pero únicamente en los intervalos especificados por las columnas y filas calculadas.

Ilustración 43-Ejemplo de modificación por efecto zoom sobre el eje central para el aumento del dataset

4.2 CREACIÓN DEL CLASIFICADOR DE IMÁGENES

Con el dataset confeccionado para realizar el entrenamiento, es el momento de seleccionar la arquitectura del modelo paso previo a seleccionar como se entrenará en dicha arquitectura.

En la documentación de la librería de TensorFlow Lite Model Maker para comprender su funcionamiento y poder cargar y adaptar a nuestro problema un modelo preentrenado, se pueden observar que tiene una serie de arquitecturas adaptadas para su uso en terminales móviles, no haciendo necesario usar su característica de buscar un modelo adaptado en el propio repositorio de TensorFlow llamado *TFHub* [52].

Cuando se examinan los modelos preentrenados que ofrece la propia librería son los siguientes: *EfficientNetLite0*, *EfficientNetLite1*, *EfficientNetLite2*, *EfficientNetLite3*, *EfficientNetLite4*, *MobileNetV2Spec*, *Resnet50*.

Examinando la comparativa entre los modelos presentados en los propios documentos de Google cuando registran el éxito de sus arquitecturas *EfficientNet*, observamos que de los modelos que nos ofrecen el que tiene un mejor desempeño es el EffificientNetLite4 poseedor de una arquitectura *EfficientNet B4* [53].

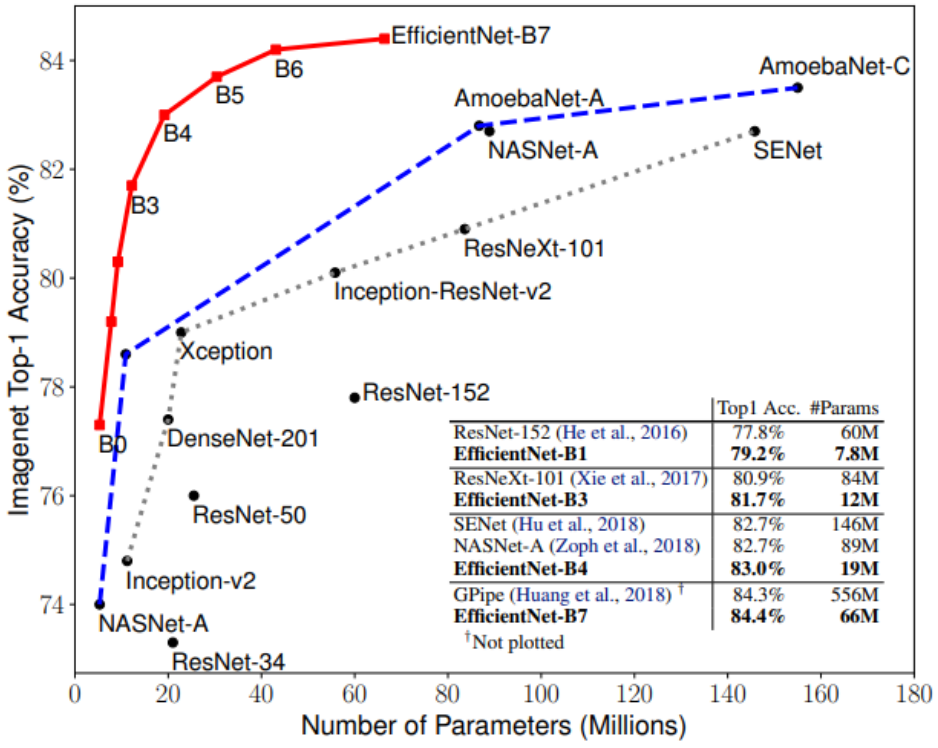


Ilustración 44-Gráfica de comparativa entre modelos de clasificación de imágenes [53]

La elección del modelo a utilizar se inicia con la creación del cuaderno Python donde ejecutaremos el entrenamiento utilizando esta librería dentro del entorno local definido previamente.

En primer lugar, hay que realizar una importación de las librerías que requiere el código que se va a ejecutar. A continuación, como se trabaja en local, hay que suministrar la ruta al directorio raíz del dataset del que se cuelgan los subdirectorios que actúan de etiquetas de predicción.

Llegados a este punto ya se empieza a utilizar la librería para realizar la división del dataset antes del entrenamiento mediante la clase `DataLoader`, que es la encargada de obtener las imágenes a partir de la ruta facilitada y posteriormente dividirlas en los conjuntos de entrenamiento (80% del dataset), test (10% del dataset) y validación (10% del dataset).

Con los datos segmentados correctamente resta definir los hiperparámetros de entrenamiento que utilizará la interfaz `image_classifier` para realizar el reentrenamiento del modelo cargado con el dataset de nuestro problema a resolver. En esta fase para llegar a los hiperparámetros más favorables se realiza un proceso de prueba y error, cambiando la configuración dependiendo de los resultados que se generan.

Debido a los dos datasets mencionados en el anterior subapartado por cada enfoque, las operaciones mencionadas se realizan por cada conjunto, ajustando hiperparámetros distintos por cada modelo. Los procesos de entrenamiento resultantes serían los siguientes:

ENFOQUE DE CLASIFICADOR POR SALAS

Tras el proceso de ajuste de los hiperparámetros la configuración más favorable ha sido la siguiente, en azul se indican los modificados:

HIPERPARÁMETRO	NOMBRE API	VALOR
Modelo	`model_spec`	'efficient_lite4'
Conjunto validación	`validation_data`	Conjunto de validación
Tamaño batch	`batch_size`	32
Épocas	`epochs`	20
Pasos por época	`steps_per_epoch`	-
Entrenar modelo completo	`train_whole_model`	-
Tasa de dropout	`dropout_rate`	0.2
Tasa aprendizaje	`learning_rate`	0.004
Momentum	`momentum`	-

Datos desordenados	`shuffle`	No
Aumentación del dataset	`use_augmentation`	No
Usar librería Hub	`use_hub_library`	Si
Pasos calentamiento	`warmups_steps`	-
Ruta modelo temporal	`model_dir`	tempfile.mkdtemp()
Iniciar entrenamiento	`do_train`	Si

Con los hiperparámetros modificados de los que vienen por defecto con los que se denotan una mayor variación de resultados, se llama al método `create()` el cual dará inicio al reentrenamiento siempre que el hiperparámetro `do_train` no se modifique a `False`.

Tras 3 horas y 15 minutos de entrenamiento se finaliza el proceso ofreciendo unos resultados de precisión media del 95.58%, que al convertirse en formato tflite se reduce al 94.44%, y una perdida media del 0.6632. La imagen a continuación resume los resultados del entrenamiento y muestra gráficamente como el entrenamiento es correcto al reducirse la perdida de manera continuada tras cada iteración y el aumento de precisión al mismo tiempo sin grandes alteraciones. Además, se puede observar la evidencia de que no se produce sobreajuste al mantenerse los valores de perdida de la validación inferiores a los del entrenamiento, sin crecidas del primer conjunto sobre el segundo, evidenciando el buen comportamiento del entrenamiento. Aunque los resultados de precisión son muy buenos los valores de pérdida para la precisión obtenida podrían ser mejores, pudiendo encontrarnos con errores ligeramente destacables a la hora de la predicción, esto puede deberse a la generalización de todas las vitrinas/obras asignándolas la etiqueta de su sala.

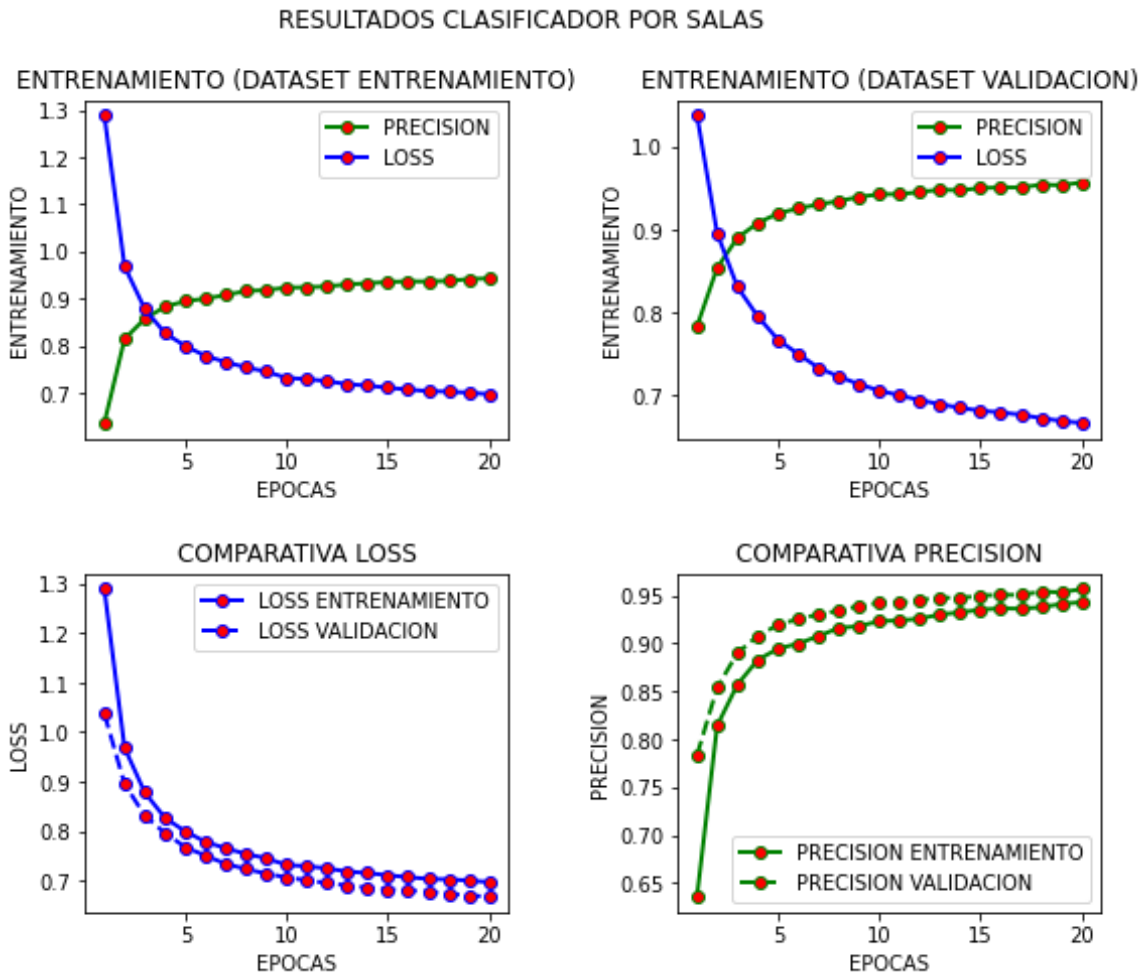


Ilustración 45-Resumen resultados entrenamiento enfoque clasificador de salas

ENFOQUE DE CLASIFICADOR POR VITRINAS

Tras el proceso de ajuste de los hiperparámetros la configuración más favorable ha sido la siguiente, en azul se indican los modificados:

HIPERPARÁMETRO	NOMBRE API	VALOR
Modelo	`model_spec`	'efficient_lite4'
Conjunto validación	`validation_data`	Conjunto de validación
Tamaño batch	`batch_size`	16
Épocas	`epochs`	20
Pasos por época	`steps_per_epoch`	-
Entrenar modelo completo	`train_whole_model`	-
Tasa de dropout	`dropout_rate`	0.2
Tasa aprendizaje	`learning_rate`	0.001
Momentum	`momentum`	-

Datos desordenados	`shuffle`	No
Aumentación del dataset	`use_augmentation`	No
Usar librería Hub	`use_hub_library`	Si
Pasos calentamiento	`warmups_steps`	-
Ruta modelo temporal	`model_dir`	tempfile.mkdtemp()
Iniciar entrenamiento	`do_train`	Si

Con los hiperparámetros modificados de los que vienen por defecto con los que se denotan una mayor variación de resultados, se llama al método `create()` el cual dará inicio al reentrenamiento siempre que el hiperparámetro `do_train` no se modifique a `False`.

Tras 3 horas y 28 minutos de entrenamiento se finaliza el proceso ofreciendo unos resultados de precisión media del 99.45%, que al convertirse en formato tflite se reduce al 99.32%, y una perdida media del 0.9840. La imagen a continuación resume los resultados del entrenamiento y muestra gráficamente como el entrenamiento es correcto, aunque se denota un inicio de sobreajuste al final del entrenamiento, al iniciarse una leve superación de los valores de perdida en validación a los de entrenamiento; al ser ligero se opta por cortar el entrenamiento en 20 épocas a favor de que se reduzca la pérdida lo máximo posible al ser más elevada. Al igual que en el caso anterior los valores de precisión son demasiado buenos en comparación con los valores de pérdida, que en este modelo son superiores al anterior probablemente debido al aumento de etiquetas que conlleva un conjunto de imágenes por etiqueta inferior al modelo anterior que todas las imágenes se distribuían entre 8 etiquetas y no 108 como en este. Con ello significa que en caso de una predicción errónea sea más evidente el error que en el modelo anterior, pero al estar las etiquetas mejor definidas con imágenes de una única vitrina/obra la predicción tenderá al ser más precisa, pudiendo aprovecharse ligeramente del inicio del sobreajuste si el dataset recoge un amplio abanico de perspectivas posibles de fotografías de la vitrina/obra.

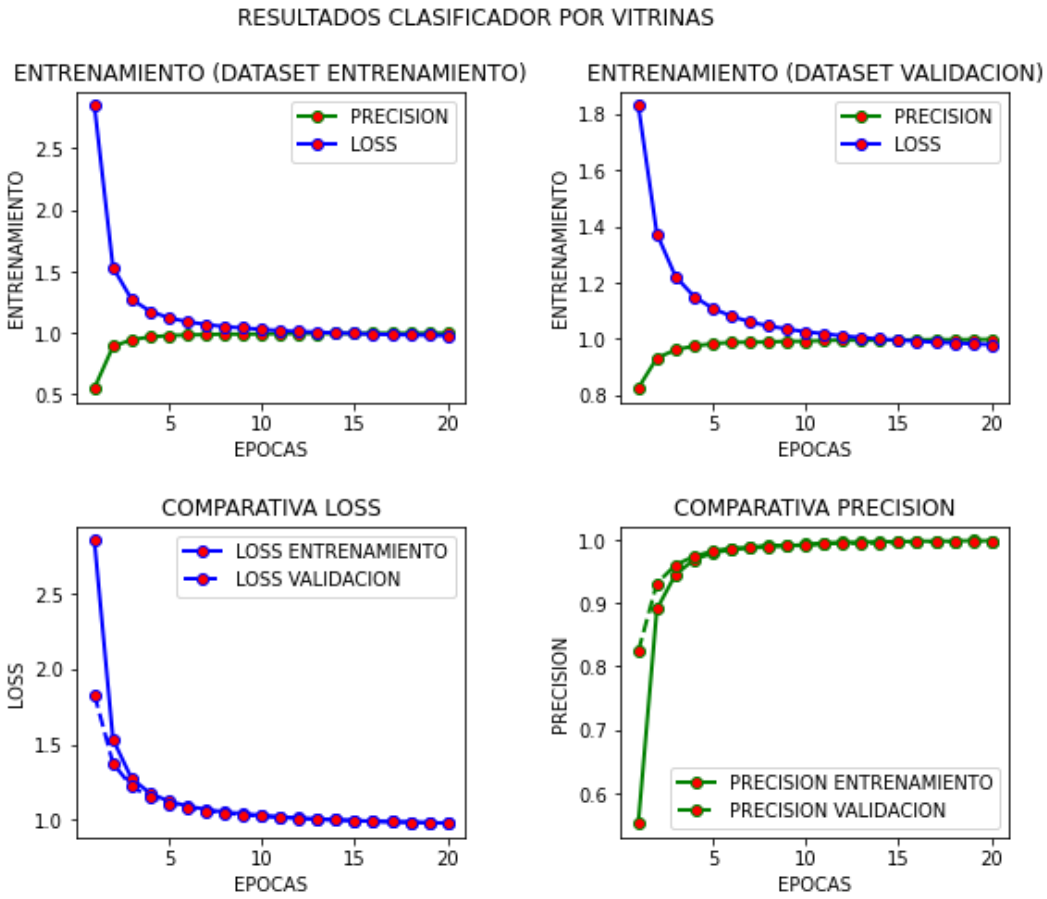


Ilustración 46-Resumen resultados entrenamiento enfoque clasificador por vitrinas

Tras finalizar cada uno de los entrenamientos por cada modelo obteniendo resultados favorables, es momento de exportar los modelos al formato que necesitamos para su ejecución en los terminales Android, formateando el modelo obtenido a TensorFlow Lite con extensión identificativa de ".tflite".

Cuando se completa el proceso de exportación ya se tienen ambos clasificadores listos para su integración en la aplicación Android.

4.3 APLICACIÓN ANDROID

Con el clasificador creado o concurrentemente realizando su debido entrenamiento se puede comenzar la planificación y elaboración del contenedor donde instanciaremos el modelo de inferencia para poder utilizarlo en cualquier punto del entorno donde trabajamos. Por tanto, antes de abrir el IDE y empezar a programar, hay que establecer un flujo de funcionamiento de cara al usuario, manejable e intuitivo para que el uso de la aplicación sea sencilla y funcional. Por ello se parte del siguiente esquema para iniciar el desarrollo del proyecto Android.

Parte de la planificación previa al inicio de la programación pasa por la identificación de flujos de funcionamiento de manera intuitiva de cara al uso futuro por el usuario. Para ello hay que identificar las posibles acciones y consecuencias que conlleva cada elección o interacción por parte de la persona, con ello se plasma en el siguiente diagrama de casos de uso que refleja la totalidad del funcionamiento a desarrollar.

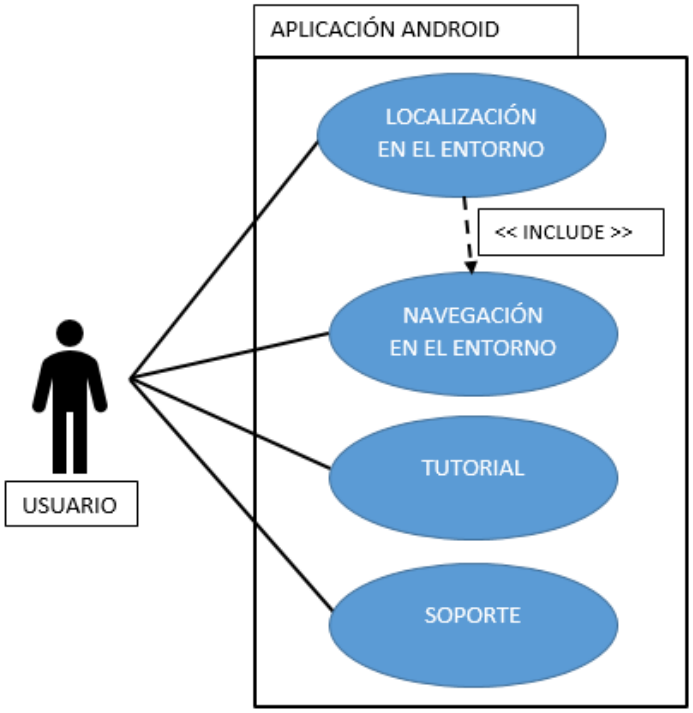


Ilustración 47-Esquema de uso aplicación Android

Con los casos de usos definidos, como paso último para iniciar la programación se desarrolla el diagrama de flujo de la aplicación, uniendo los conceptos extraídos en la elaboración del diagrama de casos de uso, identificando acciones y condiciones que definirán el comportamiento de la aplicación. Hay que realizar un flujo coherente e intuitivo para que la aplicación desarrollada consiga estas características. El diagrama a continuación representa el conjunto de flujos, representándose en un recuadro bordeado verde las pantallas, en forma de óvalo azul las acciones y operaciones que se realizan y finalmente en forma de rombo amarillo los condicionales que se presentan.

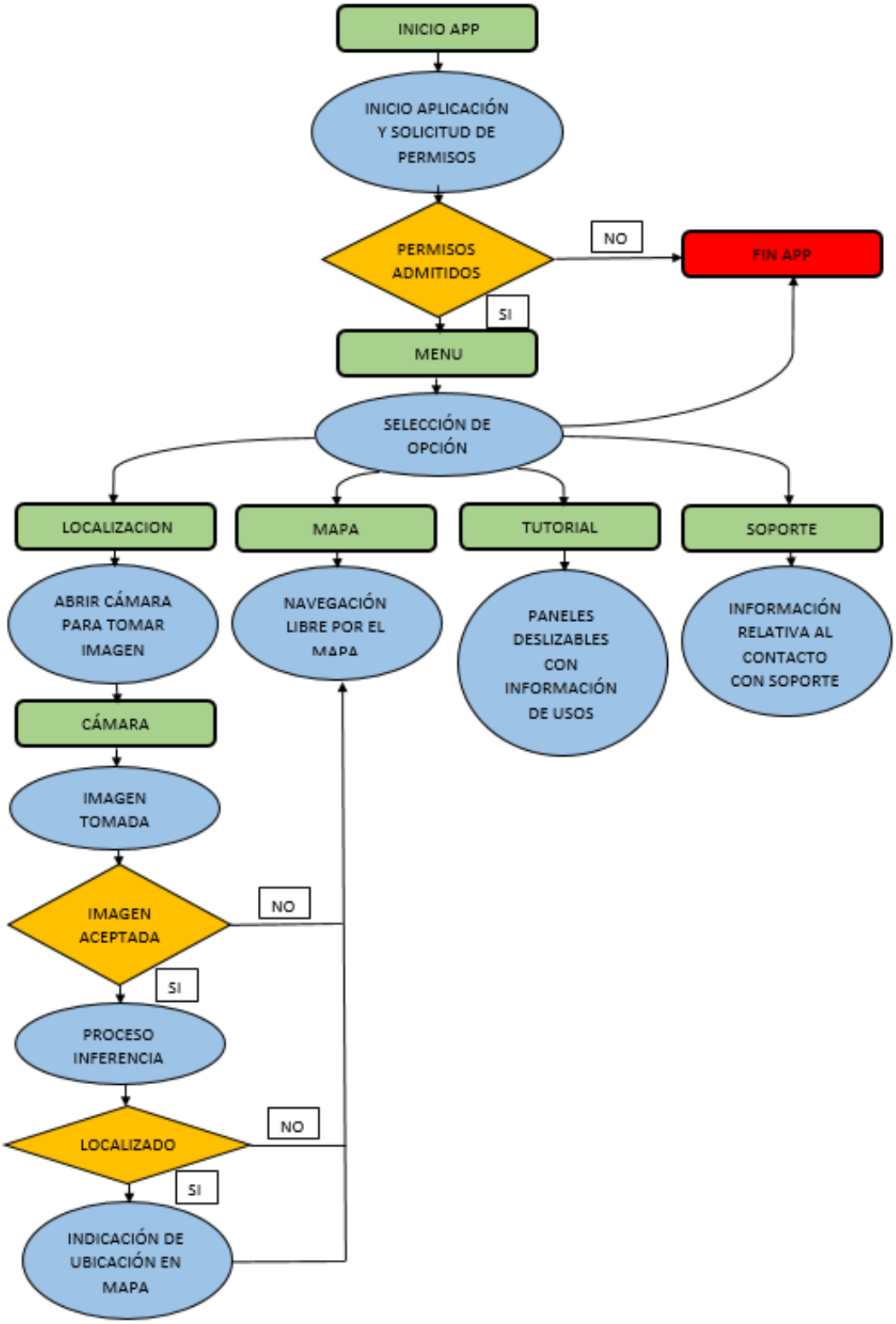


Ilustración 48-Flujo de uso de la aplicación Android

Con el problema a desarrollar planteado se inicia la creación de las actividades y los fragmentos que estas manejaran para replicar todo el funcionamiento decidido. Durante la creación de la parte lógica también se la suma el diseño de las pantallas buscando ser intuitivas y cómodas de usar de cara al usuario.

Para desarrollar una aplicación ligera y fluida se ha apostado por una estructura con una actividad única con un conjunto de fragmentos que representarán cada una de las pantallas que se necesitan durante el desarrollo para implementar la idea propuesta. La actividad hereda las características de la clase `AppCompatActivity`, mientras que los fragmentos heredan de la clase `Fragment`.

Esta actividad será la que contiene todo el peso de la aplicación ya que se comporta como manejadora de todos los eventos que ocurren mediante la instanciación y transición de los fragmentos que se relacionan con cada opción. La implementación de cada opción como fragmentos y delegación del control en esta actividad contenedora se debe a la optimización del funcionamiento. Esta optimización se basa en desarrollar las opciones y pantallas como fragmentos en vez de actividades, haciendo la aplicación más ligera y ahorrando el tiempo de creación y transición entre actividades, teniendo todas como nexo la actividad del menú. Por tanto, a nivel de diseño la actividad solo está formada por el fondo de pantalla y el contendor donde se cargan los diseños de cada fragmento.

Por tanto, el elemento de diseño vinculado con la única actividad se compone de elementos comunes para toda la aplicación, siendo estos el fondo establecido de pantalla durante la ejecución y el contendor donde se cargarán los fragmentos. Este contenedor se denomina `FrameLayout` y con el manejador de fragmentos, `SupportFragmentManager`, se modifica el contenido de este dependiendo del elemento de diseño relacionado con la instancia de fragmento que se dispone en uso. Estos se instancian mediante una llamada a un método por cada uno que le devuelve su instancia configurada con sus interfaces para comunicarse con la actividad, que sería la clase contenedora. Después con una serie de variables booleanas se tiene control de los fragmentos en uso para poder reaccionar adecuadamente a cualquier evento percibido por la aplicación, siempre habiendo una única variable con valor verdadero, significando que es el fragmento en funcionamiento. A continuación, se ilustra el ejemplo de método que instancia el fragmento que recoge la lógica del menú y su configuración del interfaz para comunicarse con la actividad contenedora, además del ejemplo de transiciones entre fragmentos, en este caso invocados por una selección de una opción.

```
// METODO PARA INSTANCIAR FRAGMENT MENU
private FAppMenu instanciaFAppMenu() {
    return FAppMenu.newInstance(new FAppMenu.OnEventos_FAppMenu() {
        @Override
        public void seleccionLocalizate() {
            // CAMBIO DE FRAGMENT: MENU --> MAPA (CON LOCALIZACION ACTIVA)
            fragmentMapa = instanciaFAppMapa( localizacionActiva: true);
            getSupportFragmentManager().beginTransaction().replace(R.id.frameLayout_contenedorFragment_aplicacion, fragmentMapa).commit();
            fMenuActivo = false;
            fMapaActivo = true;
        }

        @Override
        public void seleccionMapa() {
            // CAMBIO DE FRAGMENT: MENU --> MAPA (CON LOCALIZACION DESHABILITADA)
            fragmentMapa = instanciaFAppMapa( localizacionActiva: false);
            getSupportFragmentManager().beginTransaction().replace(R.id.frameLayout_contenedorFragment_aplicacion, fragmentMapa).commit();
            fMenuActivo = false;
            fMapaActivo = true;
        }

        @Override
        public void seleccionTutorial() {
            // CAMBIO DE FRAGMENT: MENU --> TUTORIAL
            fragmentTutorial = instanciaFAppTutorial();
            getSupportFragmentManager().beginTransaction().replace(R.id.frameLayout_contenedorFragment_aplicacion, fragmentTutorial).commit();
            fMenuActivo = false;
            fTutorialActivo = true;
        }

        @Override
        public void seleccionSoporte() {
            // CAMBIO DE FRAGMENT: MENU --> SOPORTE
            fragmentSoporte = instanciaFAppSoporte();
            getSupportFragmentManager().beginTransaction().replace(R.id.frameLayout_contenedorFragment_aplicacion, fragmentSoporte).commit();
            fMenuActivo = false;
            fSoporteActivo = true;
        }
    });
}
```

Ilustración 49-Ejemplo de método para instanciar fragmento con la configuración de su interfaz

4.3.1 INTRODUCCIÓN SPLASH

Cuando se inicia la aplicación al usuario se le recibe con una introducción con una pareja de fragmentos de tipo *Splash* que mostrará información característica de la aplicación y se realizará una gestión de los permisos necesarios para la ejecución.

DISEÑO

Debido a la naturaleza *Splash* de las pantallas, caracterizadas por mostrar iconografía relacionada con la aplicación que se está iniciando además de su título y otros datos informativos; se definen dos pantallas para definir toda la información.

La primera de ellas introduce la propia aplicación mostrando el logo asociado a esta y el título con el que se identifica. Para la segunda se muestra las principales partes participantes de la aplicación como son la Escuela Politécnica de la UAH (promotora del TFG), el Museo Provincial de Guadajara (escenario de pruebas), el IDE Android Studio (plataforma de desarrollo), el SDK de Mapbox (API de mapas), la librería TensorFlow (librería de IA) y la web FlatIcon (web de iconos de uso libre).

Ilustración 50-Introducción Splash con partes participantes

FUNCIONAMIENTO

A parte de la introducción a la aplicación e información de las partes participantes, esta actividad se utiliza como filtro para el manejo de los permisos necesarios para que la aplicación funcione.

Los permisos en las aplicaciones Android es una medida de seguridad del sistema operativo que bloquea el acceso a recursos o acciones del dispositivo a la aplicación instalada para que no tenga acceso total al dispositivo sin conocimiento del usuario. Estos permisos controlan una funcionalidad o servicio que Android permite usar, por tanto, para poder utilizarlos es necesario declarar que permisos se van a utilizar en el código y solicitar al usuario que los habilite cuando utilice la aplicación. Siempre se deben solicitar los permisos necesarios para el funcionamiento sin exigir acceso a más de los que sean estrictamente requeridos, en el código se declaran en el manifiesto de la aplicación.

En el caso de esta aplicación se necesita acceso a la cámara del dispositivo para realizar las fotografías del entorno donde nos encontramos y posteriormente guardarlas temporalmente para alimentar al modelo de inferencia y reconocer el lugar. Por ello se solicitan al usuario permitir los permisos de acceso a cámara y de acceso al almacenamiento para el uso de la aplicación.

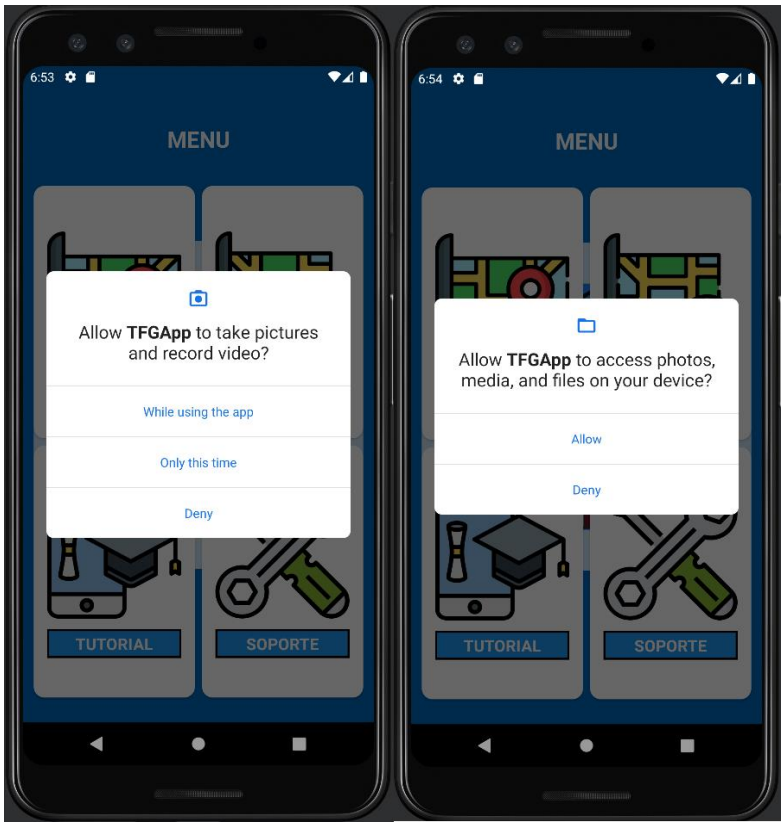


Ilustración 51-Solicitud de permisos al usuario de la aplicación Android

PROGRAMACIÓN

Con cada arranque de la aplicación un método consulta el estado de los permisos, por lo que en caso de que no estén garantizados se inicia la solicitud de permisos mediante el uso de métodos propios de la actividad.

Paso previo a realizar la solicitud de permisos es imprescindible declarar estos en el fichero de manifiesto que registra la estructura del programa, es importante definir los estrictamente necesarios al uso que tenga la aplicación.

```
<!-- Permiso de acceso a cámara-->
<uses-permission android:name="android.permission.CAMERA" />
<!-- Permiso de escritura y lectura en el almacenamiento -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Ilustración 52-Declaración de permisos necesarios en el manifiesto de la aplicación Android

Tras la solicitud de permisos se sobrescribe otro método propio de la actividad que se mantiene en espera a la decisión del usuario respecto a los permisos solicitados, con el fin de manejar las posibles decisiones. En caso de que no se garanticen, se entiende que se piden para que funcione correctamente la aplicación,

por lo que se cierra esta ya que no se puede asegurar el correcto funcionamiento. En caso contrario se cede el control al fragmento de menú.

Independientemente del estado, solicitud o respuesta de permisos los fragmentos actúan de manera *Splash*, mostrándose cada uno durante un breve intervalo de tiempo definido y siendo reemplazados por el fragmento consecutivo. Esto se lleva a cabo mediante instancias de la clase `Handler`, ya que con el método `postDelayed` se puede implementar una ejecución de código al cabo de una cantidad de milisegundos pasados también por parámetros al método, tiempo que empieza su cómputo tras ejecutarse el método. Por tanto, dentro de la ejecución de código después del intervalo de tiempo definido, se modifica el fragmento cargando, dando la sensación de transición con el paso del tiempo de los fragmentos *Splash*.

```
// Configuramos la transicion a la vista de menu al cabo de 4 segundos
Handler splashMeciones = new Handler();
splashMeciones.postDelayed(new Runnable() {
    @Override
    public void run() {
        // Instanciamos menu
        fragmentMenu = instanciaFAppMenu();
        getSupportFragmentManager().beginTransaction().replace(R.id.frameLayout_contenedorFragment_aplicacion,fragmentMenu).commit();
        fMenuActivo = true;
        fSplashMencionesActivo = false;
    }
}, delayMillis: 4000);
```

Ilustración 53-Ejemplo de método utilizado para la transición de las pantallas Splash

4.3.2 MENÚ

Con los permisos garantizados para la aplicación, se permite el uso del fragmento de menú que se carga tras el tiempo prefijado para los fragmentos Splash, mostrando las posibles funciones que se pueden realizar con la aplicación.

DISEÑO

El diseño se escenifica en una cuadrícula para presentar las posibles funciones mediante vistas conformadas por una pareja de icono con texto, para facilitar mediante una imagen y el título de la opción una idea intuitiva de su utilidad. Las opciones como se pueden observar son los cuatro casos de uso especificados en la planificación.

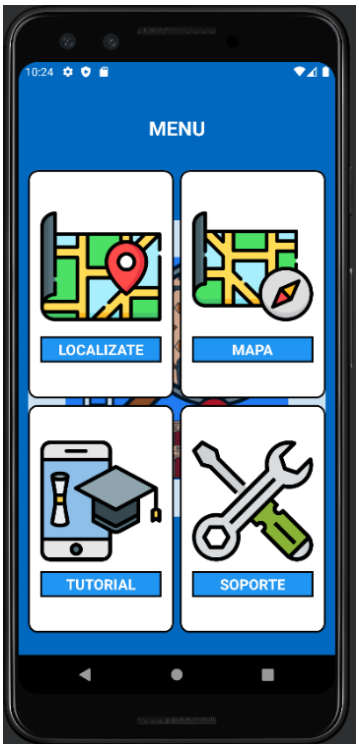


Ilustración 54-Fragment menú de la aplicación Android

FUNCIONAMIENTO

Con la interacción del usuario sobre cualquier opción inicializa la funcionalidad.

PROGRAMACIÓN

Con la pulsación del usuario sobre una de las vistas de las funcionalidades de la aplicación, se inicia la transición del fragmento de menú al vinculado con la opción seleccionada desencadenando la creación de la vista del fragmento y las acciones programadas en su inicio. Esto se hace mediante el interfaz que se implementa cuando se instancia el fragmento del menú en la actividad contenedora, donde se debe realizar la modificación del fragmento cargado en el contenedor.

4.3.3 LOCALIZACIÓN EN EL ENTORNO

Representado por la opción con título “LOCALIZATE” y un icono de un mapa con un icono de localización, es la opción que permite al usuario utilizar la función principal de la aplicación para localizarse en el entorno mediante una fotografía tomada por el mismo dispositivo.

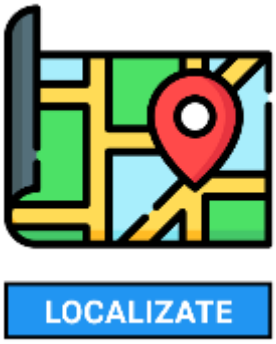


Ilustración 55-Opción para localizarse

DISEÑO

El diseño se centra en la vista de la API de Mapbox, que permite cargar el mapa del entorno y reflejar el conjunto de capas y acciones que se pueden realizar mediante la interacción con esta; ocupando la totalidad de la pantalla. Superpuesto sobre esta vista se presentan los botones de acciones de la aplicación y mapa, además de un panel informativo relativo a los POIs.

En la parte inferior de la pantalla se presenta el botón con icono de cámara, el cual es encargado de reiniciar el proceso de localización, desplegando acto seguido a su acción la cámara para tomar una fotografía del entorno. Acompañandolo a su izquierda se encuentra un botón con la finalidad de mostrar u ocultar el panel informativo de los POIs, para poder consultar la información del *POI* seleccionado o localizado, o en caso contrario ocultarla para navegar más cómodamente por el mapa.

El panel informativo de POIs recoge de manera breve la información relativa al POI seleccionado en libre navegación o por resultado de localización; indicando título, coordenadas y descripción asociada al POI.

FUNCIONAMIENTO

Al seleccionar la opción se abre la aplicación de cámara por defecto del terminal para que el usuario fotografié la vitrina/obra donde se encuentra. Cuando realiza la fotografía, saldrá un menú para tomar otra fotografía, rechazar o aceptar la fotografía tomada. En el primer caso se volverá al inicio del funcionamiento, en el segundo se abrirá el mapa para la navegación libre y por último se iniciará el proceso de inferencia para predecir la localización.

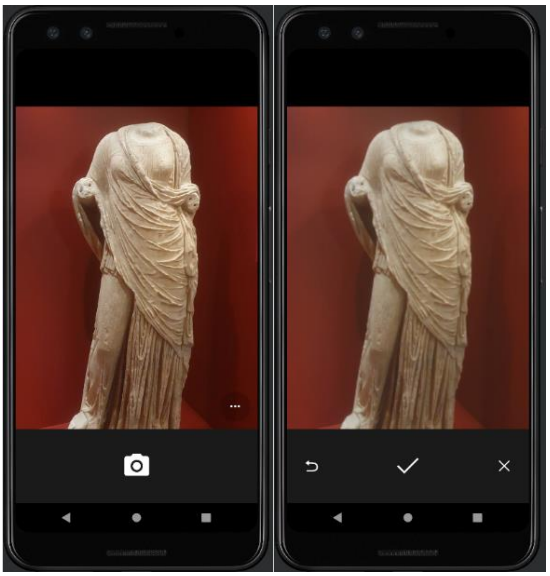


Ilustración 56-Captura de imagen por parte de cámara llamada desde la aplicación Android

Dando el último de los casos, ya que la navegación libre se explicará en el siguiente apartado, se muestra la vista del mapa con una vista indicativa de que la aplicación está realizando operaciones, por lo que el usuario tiene que esperar. Finalmente, cuando desaparece significa que el clasificador ha llegado a un resultado, en caso de que no se produzca ninguna acción a continuación significará que el resultado no es concluyente, por lo que no se indica ninguna localización. En caso contrario cuando se obtiene un resultado, se reflejará la localización predicha por el modelo en el mapa, superponiendo el icono de localización sobre el icono del POI mientras se centra la vista sobre el mismo y se despliega el panel informativo indicando la información de la localización.

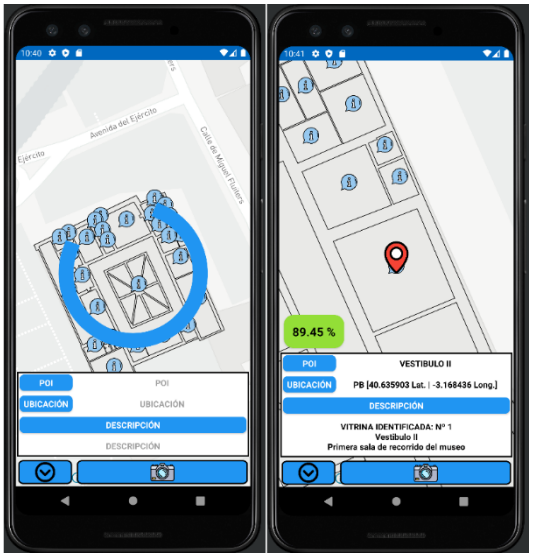


Ilustración 57-Procesamiento de imagen por el clasificador de imágenes integrado en la aplicación Android

Posteriormente a la localización se podrá seguir usando el mapa como posteriormente se explicará, mediante la libre navegación.

PROGRAMACIÓN

Cuando se reemplaza el fragmento de menú por el de mapa, se indica que la localización está habilitada por lo que se inicia la llamada a la aplicación por defecto de cámara del terminal Android mediante un método que queda en espera de que la cámara finalice su operación.

Una vez la actividad de cámara finaliza, esta devuelve un resultado al método que ha quedado en espera y en caso de que se acepte la fotografía tomada, se obtiene la imagen capturada guardándola temporalmente en almacenamiento interno del dispositivo. Con esta imagen se transforma a un objeto bitmap para poder suministrarla como entrada al modelo de inferencia.

Para cargar el modelo entrenado previamente en el entorno local gracias a Android Studio se puede integrar en el proyecto de una manera sencilla. Para ello se añade como si se añadiese cualquier otro tipo de elemento como las actividades, fragmentos y *layouts* que forman la aplicación. Una vez seleccionada la opción solo resta seleccionar la ruta donde se encuentra el clasificador con extensión ".tflite".

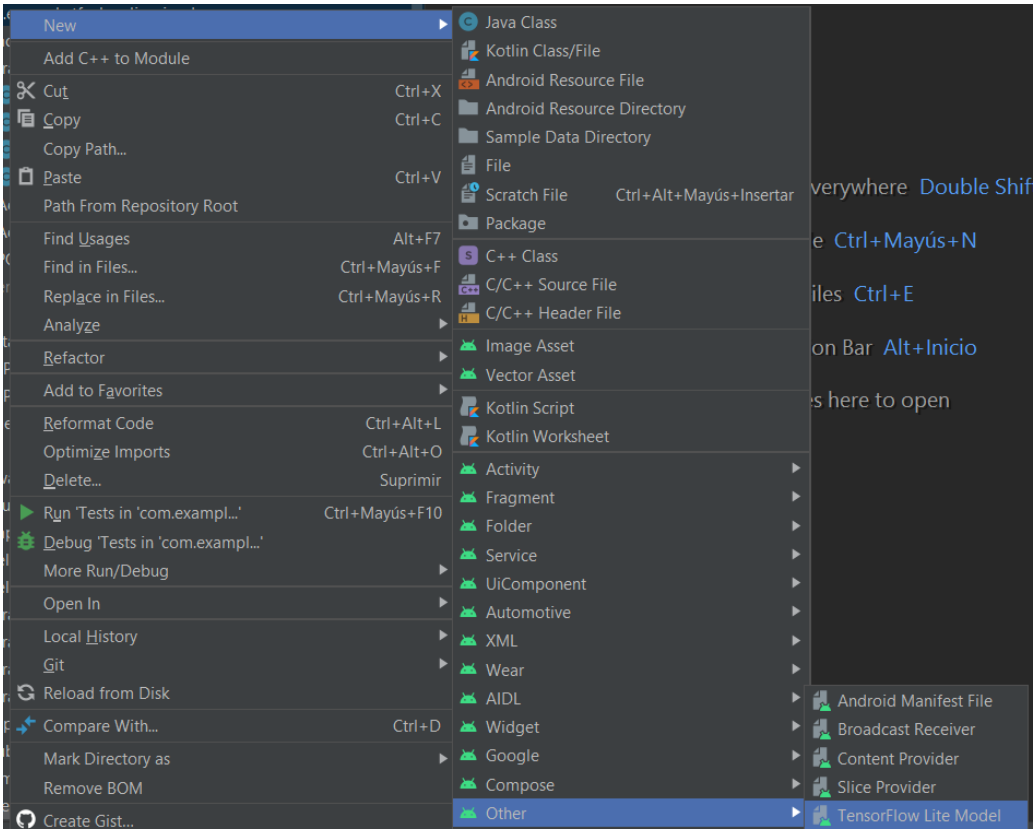


Ilustración 58-Importación de modelo TensorFlow Lite en aplicación Android

Cuando se carga el modelo en el proyecto y se abre el fichero del modelo importado, el propio entorno nos ofrece un resumen del clasificador y nos despliega una breve guía de uso del modelo dentro de nuestro código, la guía se presenta en lenguaje Kotlin y en Java. Debido a la sencillez de los pasos a seguir, usar un modelo de inteligencia artificial está al alcance de cualquier programador.

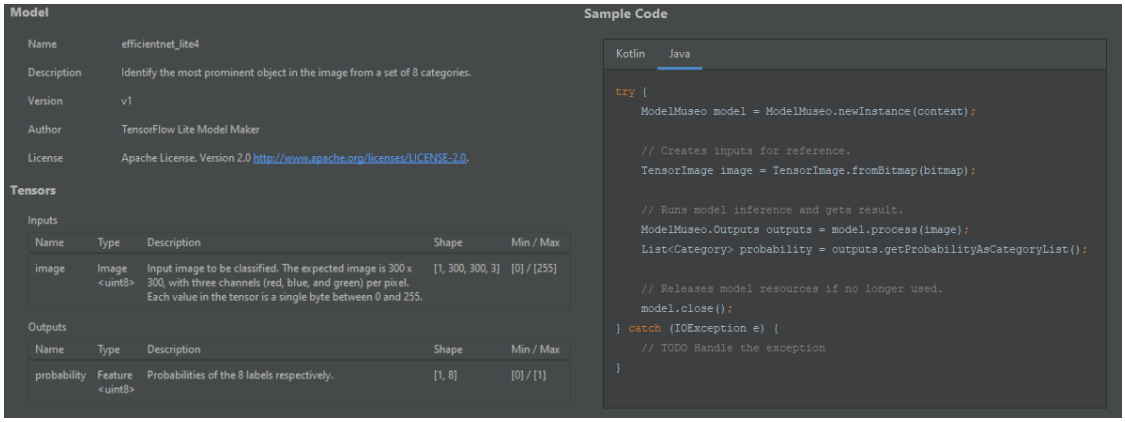


Ilustración 59-Resumen importación clasificador por salas en formato TensorFlow Lite

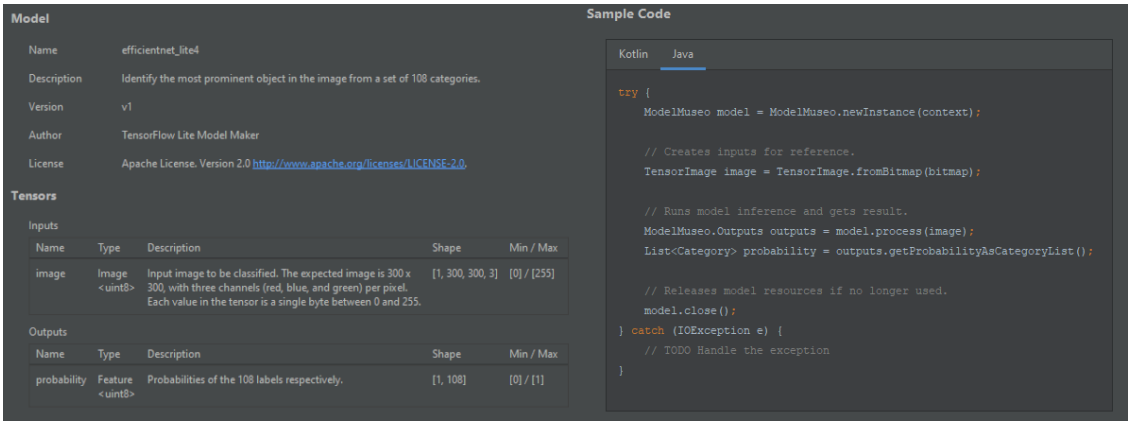


Ilustración 60-Resumen importación clasificador por vitrinas en formato TensorFlow Lite

Por tanto, con el modelo listo para ser integrado por código, paso previo a utilizarlo hay que redimensionar el bitmap al tamaño de entrada de datos del clasificador, con un tratamiento previo para orientar correctamente la imagen y evitando perder calidad en el reescalado. Con la imagen lista redimensionada se la pasa por parámetro al método encargado de la predicción, el cual tras finalizar su ejecución devolverá el listado de probabilidades por etiqueta predichas.

```
// METODO PARA MANEJO DE IMAGEN TEMPORAL
private void manejoRespuestaCamaraRutaGuardadoTemporal(){
    // Obtenemos imagen original
    imagenEntrada = BitmapFactory.decodeFile(rutaFotoTemporal);
    try {
        ExifInterface ei = new ExifInterface(rutaFotoTemporal);
        int orientation = ei.getAttributeInt(ExifInterface.TAG_ORIENTATION, ExifInterface.ORIENTATION_UNDEFINED);
        switch(orientation) {
            case ExifInterface.ORIENTATION_ROTATE_90:
                imagenEntrada = rotateImage(imagenEntrada,  angle: 90);
                break;
            case ExifInterface.ORIENTATION_ROTATE_180:
                imagenEntrada = rotateImage(imagenEntrada,  angle: 180);
                break;
            case ExifInterface.ORIENTATION_ROTATE_270:
                imagenEntrada = rotateImage(imagenEntrada,  angle: 270);
                break;
            case ExifInterface.ORIENTATION_NORMAL:
            default:
                imagenEntrada = imagenEntrada;
        }
    }catch (IOException ioe){
        Log.e( tag: "FAppMapa.manejadorActivityResultCamera.onActivityResult", msg: "ERROR: "+ioe.getMessage());
    }
    // Reescalamos entrada a IA
    imagenEntrada = reescaladoBitmapConCalidad(imagenEntrada,dimensionImagenEntrada,dimensionImagenEntrada);
    // Eliminamos imagen temporal
    new File(rutaFotoTemporal).delete();
    // Lanzamos tarea de ubicacion
    new TareaLocalizacionIA(imagenEntrada).execute();
    procesoInferenciaActivo = true;
    gestionVistas();
}
```

Ilustración 61-Ejemplo de tratamiento de la imagen original

Con el listado de probabilidades se procesa para ordenar las probabilidades en orden de mayor a menor, tras ello se extrae la etiqueta con probabilidad más elevada y si supera la probabilidad definida de corte se la considera como válida. En caso de considerarse válida se inicia el proceso de identificar la localización, superponiendo el icono de localización sobre el POI correspondiente mientras se centra este en el centro de la pantalla y se completa su información en el panel informativo. Junto a la información relativa al POI se añade la información de la predicción como la probabilidad con la que se ha determinado la ubicación devuelta. En caso del clasificador por vitrinas también se añaden en el panel informativo el número de vitrina del dataset identificada en la imagen.

Debido a que el proceso de inferencia toma su tiempo para interpretar la imagen de entrada, todo el proceso se inserta en una tarea asíncrona con el fin de que la aplicación no detecte la demora de resultados por procesamiento como que la aplicación se ha colgado.

4.3.4 NAVEGACIÓN EN EL ENTORNO

Representado por la opción con título “MAPA” y un icono de un mapa, es la opción que permite al usuario explorar y navegar libremente por el entorno registrado, consultando los POIs y la composición de localizaciones del entorno.

Ilustración 62-Opción de mapa

DISEÑO

El diseño se centra en la vista de la API de Mapbox, que permite cargar el mapa del entorno y reflejar el conjunto de capas y acciones que se pueden realizar mediante la interacción con esta; ocupando la totalidad de la pantalla. Superpuesto sobre esta vista se presentan los botones de acciones de la aplicación y mapa, además de un panel informativo relativo a los POIs.

En el lateral derecho se presentan los botones cuya acción cambia la planta del entorno que se representa, indicando de verde la que se encuentra representada por el momento.

En la parte inferior de la pantalla se presenta el botón con icono de cámara, el cual es encargado de reiniciar el proceso de localización, desplegando acto seguido a su acción la cámara para tomar una fotografía del entorno. Acompañandolo a su izquierda se encuentra un botón con la finalidad de mostrar u ocultar el panel informativo de los POIs, para poder consultar la información del POI seleccionado o localizado, o en caso contrario ocultarla para navegar más cómodamente por el mapa.

El panel informativo de POIs recoge de manera breve la información relativa al POI seleccionado en libre navegación o por resultado de localización; indicando título, coordenadas y descripción asociada al POI.

FUNCIONAMIENTO

Se carga el mapa del piso asignado por defecto con todos los POIs indicados y ninguno seleccionado, en espera que el usuario pulse sobre alguno para consultarlo.

El usuario explora el mapa deslizando el mapa con el dedo y usando dos dedos puede variar la inclinación de la cámara o variar el *zoom*. En caso de que se salga de los límites fijados alrededor del entorno de la aplicación, se notifica de este hecho ocultando los botones del diseño.

Haciendo clic sobre cualquier POI se realiza la selección de este, indicándose con el icono de localización superpuesto al del propio POI y se rellena el panel informativo.

PROGRAMACIÓN

Tras reemplazar el fragmento de menú por el de mapa, se inicia la configuración del mapa mediante la API de Mapbox, definiendo el contenido del mapa y su respuesta a eventos; además de la configuración de los botones que acompañan la vista del mapa.

Iniciando la configuración se obtiene para el manejo de información una estructura de mapa de pares clave-valor para recoger todos los POIs que se representarán en la vista. Posteriormente mientras se carga el estilo del mapa que se muestra, se configura la limitación de espacio de uso del mapa junto cómo reaccionar a los cambios y finalmente se definen las distintas capas que conforman el mapa.

```
// METODO PARA CARGAR FICHEROS GEOJSON POIs DESDE ASSETS
private ArrayList<POI> cargaPOIs(String nombreGeojson){
    // Preparamos variable de salida
    ArrayList<POI> listadoPOIs = new ArrayList<>();
    // Leemos archivo
    String[] contenidoGeojson = cargarJSONDesdeAssets(nombreGeojson).split( regex: "\\r\\n");
    for(int i=0; i<contenidoGeojson.length; i++){
        if(contenidoGeojson[i].contains("properties")){
            // Obtencion de la informacion conociendo la estructura del geojson
            String nombre = contenidoGeojson[i].split( regex: "\"name\":")[1].split( regex: ",")[0].replaceAll( regex: "\"", replacement: "");
            String descripcion = contenidoGeojson[i].split( regex: "\"description\":")[1].split( regex: ",")[0].replaceAll( regex: "\"", replacement: "");
            String tipo = contenidoGeojson[i].split( regex: "\"type\":")[2].split( regex: ",")[0].replaceAll( regex: "\"", replacement: "");
            String piso = contenidoGeojson[i].split( regex: "\"floor\":")[1].split( regex: "\\}")[0].replaceAll( regex: "\"", replacement: "");
            Point punto = Point.fromJson(contenidoGeojson[i].split( regex: "\"geometry\":")[1].split( regex: "\\}")[0] + "}");
            // Introduccion del punto
            listadoPOIs.add(new POI(nombre,descripcion,tipo,Integer.parseInt(piso),new LatLng(punto.latitude(),punto.longitude())));
        }
    }
    return listadoPOIs;
}
```

Ilustración 63-Ejemplo de obtención de la información relativa a los POI del mapa

La configuración de los limites se realiza estableciendo un cuadrado de coordenadas, determinando que, si el centro del mapa mostrado se encuentra dentro de este con un *zoom* que permite la visualización del entorno, se considera dentro de

límites, pudiendo ejercer el normal funcionamiento explicado anteriormente. Por el contrario, si el centro se encuentra fuera o el *zoom* es demasiado reducido, impidiendo la normal visualización, se ocultan todos los botones de acciones e información, para indicar al usuario que se ha salido del área de uso de la aplicación.

La inserción de las distintas capas es el hecho clave que permitirá que el mapa muestre toda la información necesaria con detalle. Para ello se establece la capa base sobre el mapa que consiste en la que se definen los contornos de las salas. Sobre esta se superponen la capa destinada a representar todos los POIs con su icono característico y sobre esta se añade una última capa para mostrar los resultados de localización o la selección de un POI mediante su icono característico distinguible al de los POIs.

```
// Configuracion manejador de simbolos
manejadorSimbolos = new SymbolManager(mapView_mapa,mapboxMap,mapboxStyle, belowLayerId: null,new GeoJsonOptions().withTolerance(0.4f));
manejadorSimbolos.setIconAllowOverlap(true);
manejadorSimbolos.setTextAllowOverlap(true);
// Configuracion capas
capaTrazosPlanta = new GeoJsonSource(JSON_PLANTA,cargarJSONDesdeAssets( nombreGeoJson: "PlantaBaja.geojson"));
mapboxStyle.addSource(capaTrazosPlanta);
cargarCapaEstructuras();
// Configuracion POIs
capaPuntosPOI = new GeoJsonSource(JSON_POIs, FeatureCollection.fromFeatures(listadoPOIsPB));
mapboxStyle.addSource(capaPuntosPOI);
capaLocalizacionPOI = new GeoJsonSource(JSON_LOCALIZACION, FeatureCollection.fromFeatures(new ArrayList<>()));
mapboxStyle.addSource(capaLocalizacionPOI);
cargarCapaPOIs();
```

Ilustración 64-Ejemplo de configuración inicial de capas, manejadores y POIs en el mapa

Como última fase en la configuración del mapa mediante la API se define un disparador para actuar cuando el usuario pulsa sobre algún POI para indicar su selección. Este método de escucha propio de la librería Mapbox, obtiene la característica cargada previamente en el mapa que se encuentra bajo la pulsación realizada por el usuario, con lo que se puede determinar el POI seleccionado, buscando mediante el identificador asignado a la característica.

Finalmente se configuran las opciones de los botones anexos. La interacción con el botón de cámara se ejecuta el método que llamará a la actividad de la cámara en espera de resultado, para manejarlo y mostrarlo en el mapa si es correcto; proceso explicado en la opción anterior. Por último, el botón vinculado al panel informativo definirá visible o invisible este mismo para mejorar la vista al mapa.

4.3.5 TUTORIAL

Representado por la opción con título “TUTORIAL” y un icono de un teléfono con un birrete haciendo referencia al concepto de graduarse, curso… Es la opción que permite al usuario que le muestra un resumen visual de la mecánica y flujo de

funcionamiento de la aplicación en caso de desconocer el funcionamiento de la aplicación.

Ilustración 65-Opción de tutorial

DISEÑO

Para formar un tutorial ordenado y fácil de comprender por el usuario, se realiza una lista deslizable de vistas con el formato de imagen y breve texto explicativo. Al ser un listado lineal el usuario no puede saltarse pasos, por lo que se explica en orden las acciones que se deben realizar para poder obtener resultados. Tras explicar los pasos básicos a seguir para utilizar la aplicación, se realiza una breve guía de simbología y botones que puede encontrarse durante el uso de la explicación.

Ilustración 66-Ejemplo de diseño de tutorial con varias vistas

FUNCIONAMIENTO

El funcionamiento al tener un orden lineal las pantallas explicativas consisten en desplazarse a través de ellas arrastrando el dedo de un lateral, al contrario, navegando hacía la vista anterior o la siguiente.

PROGRAMACIÓN

La programación consiste en reemplazar el fragmento de menú con el de tutorial y quedarse en espera de que el usuario quiera retroceder al menú de nuevo. A nivel de las vistas del tutorial se realiza un adaptador específico para poder cargar la vista personalizada de cada pantalla del tutorial y suministrando los pares de imágenes más texto explicativo; la transición entre vistas deslizando el dedo de un lateral al contrario se maneja mediante el adaptador. Como manera auxiliar de ubicar al usuario en el proceso del tutorial, se incluye un método que se dispara con la transición de cada vista del tutorial, con el fin de que se remarque de un color distinto el punto que coincide con el número de diapositiva en la que se encuentra.

4.3.6 SOPORTE

Representado por la opción con título “SOPORTE” y un icono de herramientas haciendo referencia al concepto de ayuda, arreglo…, es la opción que muestra al usuario la información de contacto con la entidad que gestiona la aplicación en caso de duda o problema que se le presente al usuario. Es una opción más orientada a la recolección de incidencias más que de dudas de uso que se podrían resolver revisando la opción de tutorial.

Ilustración 67-Opción de soporte

DISEÑO

A nivel de diseño se reduce a un cuadro informativo donde se indica la manera de contactar con el soporte y la dirección o teléfono de contacto.

PROGRAMACIÓN

A nivel de programación es la opción más sencilla ya que se reduce a reemplazar el fragmento de menú con el de soporte, únicamente teniendo que estar pendiente de que el usuario desee retroceder al menú.

5 RESULTADOS

Como último paso para comprobar el resultado del proceso que describe el TFG se ha probado la aplicación de forma experimental. Como se detalló al inicio de la explicación del desarrollo, para elaborar el trabajo se ha contado con la colaboración del Museo Provincial de Guadalajara, pudiendo utilizar sus obras y su entorno para reflejar la aplicación de la solución de localización de personas a partir de imágenes.

A continuación, se detallará el circuito de experimentación planteado y los resultados que se han obtenido por cada enfoque planteado de clasificador.

5.1 CIRCUITO EXPERIMENTACIÓN

Para comprobar el desempeño de la aplicación se define un circuito a lo largo de las salas del museo, recogiendo una muestra representativa del 25% de las vitrinas/obras que se encuentran por sala. Esto significa que si en la sala en cuestión consta de 8 vitrinas/obras se seleccionarán 2 de ellas aleatoriamente para identificar la sala mediante la interpretación de sus imágenes tomadas.

Con el fin de que los resultados sean veraces y las vitrinas/obras no sean escogidas sabiendo que el modelo se desenvuelve mejor en su clasificación, se crea un script que se encarga de enumerar el listado de vitrinas/obras por sala del museo que se utilizarán en el experimento. Tras su ejecución el resumen de las vitrinas/obras incluidas en el circuito se refleja en la siguiente tabla:

SALA	N.º VITRINAS/OBRAS SELECCIONADAS
Vestíbulo II	8
Colección Permanente I	5
Colección Permanente II	2
Colección Permanente III	5
Colección Permanente IV	2
Escipión	4
Ola	2
Atlanta	2
RESUMEN	30

Con las vitrinas/obras del circuito definidas se procede a realizar un circuito por cada versión de la aplicación preparada con cada uno de los distintos dos clasificadores entrenados. El resultado de cada circuito se recoge en los apartados a continuación.

Durante cada circuito se tomas tres imágenes por vitrina/obra seleccionada intentando recrear un posible comportamiento de un usuario, por lo que se decide tomar una imagen en diagonal desde la izquierda de la vitrina/obra, una segunda imagen desde el centro y una imagen final diagonal desde la derecha. Tras finalizar cada circuito se obtienen un total de 90 predicciones de las imágenes tomadas para analizar los resultados.

A la hora de determinar la validez se han comprobado las predicciones a nivel individual por imagen tomada y a nivel de vitrina/obra para comprobar la predicción media juntando los 3 resultados de las distintas perspectivas [Ilustración 68]. Debido a que solo se estudia la primera predicción se trabaja con los términos absolutos de acierto y error, omitiendo las opciones de falso negativo o falso positivo, dependiendo si la predicción ha sido correcta o no.

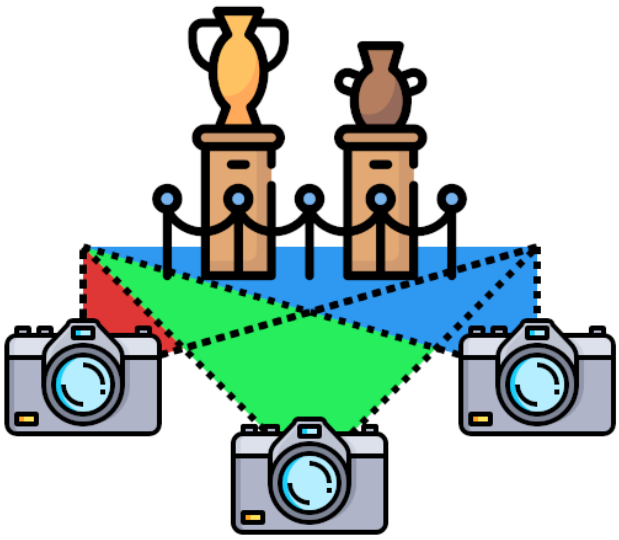


Ilustración 68-Esquema con la toma de 3 imágenes desde diferentes perspectivas de cada obra

Cuando se comprueban los resultados a nivel de vitrina/obra se establece una media entre las probabilidades con acierto en la predicción, contando como cero probabilidades de error de predicción. Finalmente, en el cómputo final de casos correctos e incorrectos se determina como error de predicción media cuando la media

entre las probabilidades predichas erróneamente supere a la media de las predichas correctamente, en este caso la probabilidad media de predicción errónea se realiza entre los resultados de predicción errónea.

5.1.1 RESULTADOS ENFOQUE CLASIFICADOR POR SALAS

Finalizado el circuito de experimentación con la aplicación del clasificador de imágenes por salas integrado, se obtiene el siguiente listado de probabilidades obtenidas al primer intento de fotografía que se realiza sobre cada obra/vitrina en las distintas perspectivas explicadas en el apartado anterior.

Analizando estos resultados de manera individual por cada imagen capturada, se obtiene una tasa de 8 errores de las 90 fotografías tomadas en total, traduciéndose en un porcentaje de acierto del 91.11% en la práctica, no muy distante del porcentaje de precisión obtenido durante el entrenamiento del 95.44% del modelo exportado en formato tflite.

Deteniéndose en los errores de predicción recogidos se comprueba que los errores se concentran en una de las salas con menor cantidad de imágenes para el entrenamiento y en la predicción de cuadros. El primer problema se podría reducir con una obtención más efectiva de imágenes de las vitrinas/obras de la sala para su entrenamiento, dentro de lo posible, porque también coincide en ser una de las salas más reducidas, por lo que la obtención de imágenes está limitada a su número de vitrinas/obras e intentando no repetir imágenes desde perspectivas muy similares para evitar un posible sobreajuste. El segundo problema por lo contrario nace debido al enfoque generalista del modelo al asignar las distintas vitrinas/obras bajo la etiqueta de la sala donde se encuentran, no teniendo un patrón de predicción común. Entonces en este caso es probable que las obras/vitrinas de estilo lienzo o cuadro se identifiquen antes con una sala que la mayoría de los elementos sean del mismo tipo, que con la sala correcta.

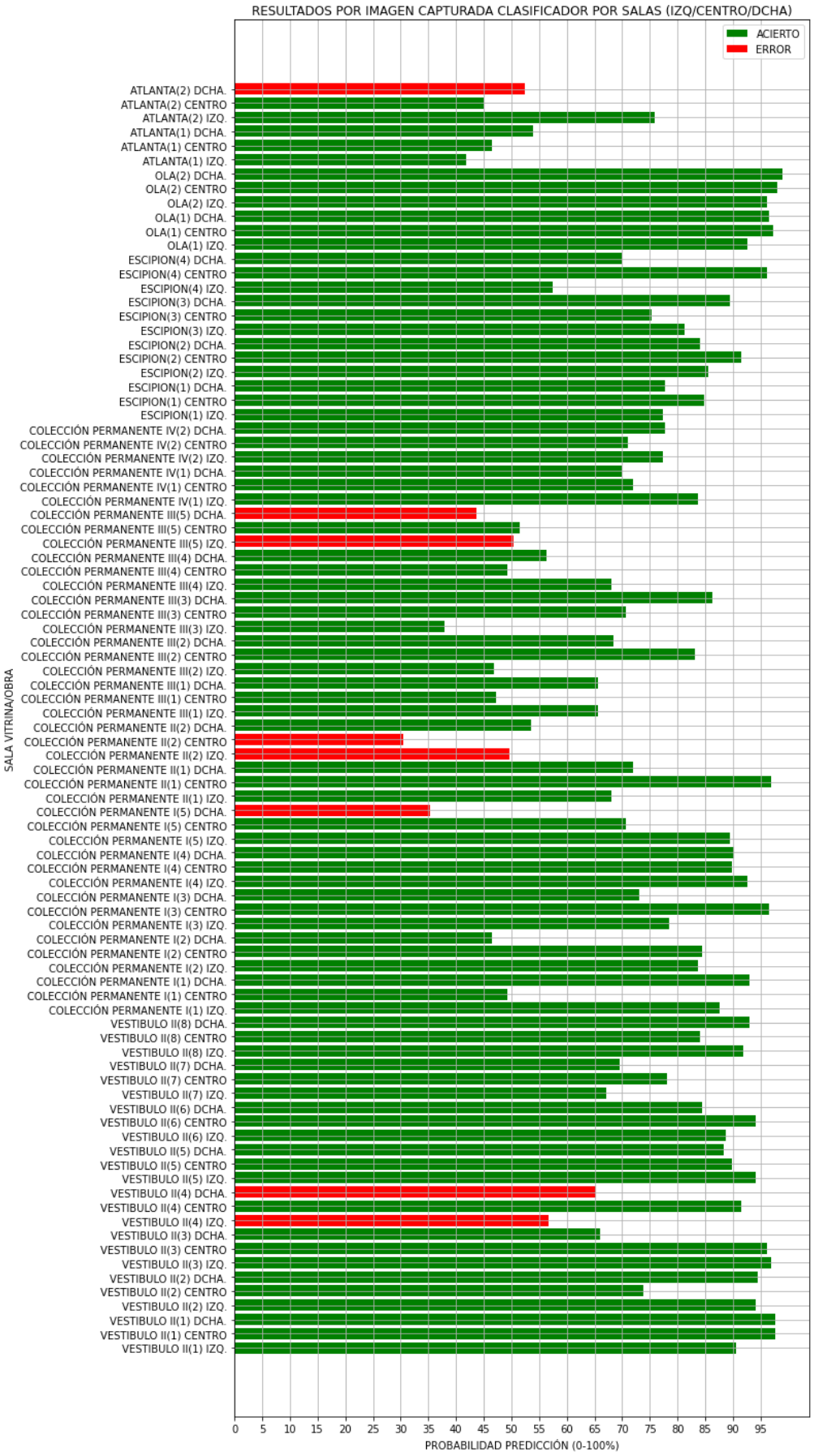


Ilustración 69-Resumen resultados predicciones por imagen de clasificador por salas

Evaluando los mismos resultados a nivel de vitrina/obra fotografiada se siguen observando resultados positivos con una tasa de 4 predicciones medias erróneas frente a las 30 vitrinas/obras incluidas en el circuito, traduciéndose en una tasa de precisión del 86.66% algo más reducida que en el análisis individual al concentrarse los errores de predicción encontrados en las mismas obras/vitrinas y no obtener mejores probabilidades de precisión desde otras perspectivas, pero aun así siguen siendo buenos resultados.

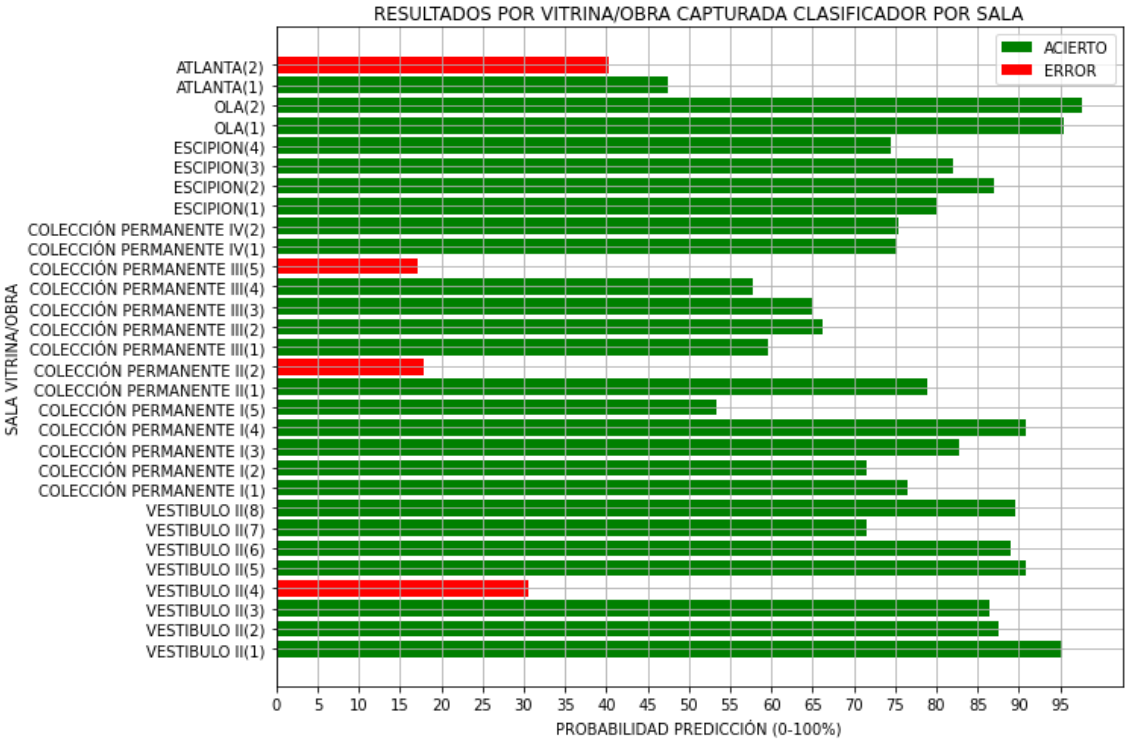


Ilustración 70-Resumen resultados predicciones por vitrina/obra de clasificador por salas

Resumiendo los resultados vistos desde ambas perspectivas se obtienen resultados muy positivos al superarse la tasa propuesta de una precisión superior al 75% con predicciones sólidas al presentarse una probabilidad media en las predicciones correctas de un 78.02% desde el punto de vista individual por fotografía y un 77.96% desde el punto de vista en conjunto por vitrina/obra. Como contraparte se puede presenciar también una probabilidad media en las predicciones erroneas del 47.97% desde el punto de vista individual por fotografía, reduciendose al 26.43% desde el punto de vista en conjunto por vitrina/obra; lo que significa una probabilidad ligeramente destacable ya que una predicción erronea no debería acumular demasiada probabilidad. Esto último puede darse debido a la capa clasificadora del propio modelo que al repartir la probabilidad del 100% entre todas sus etiquetas, al tener únicamente 8 etiquetas en caso de ser una predicción erronea puede presentar

estas probabilidades ya que de base en caso de repartir equittivamente esta, cada una tendría una probabilidad base del 12.5%.

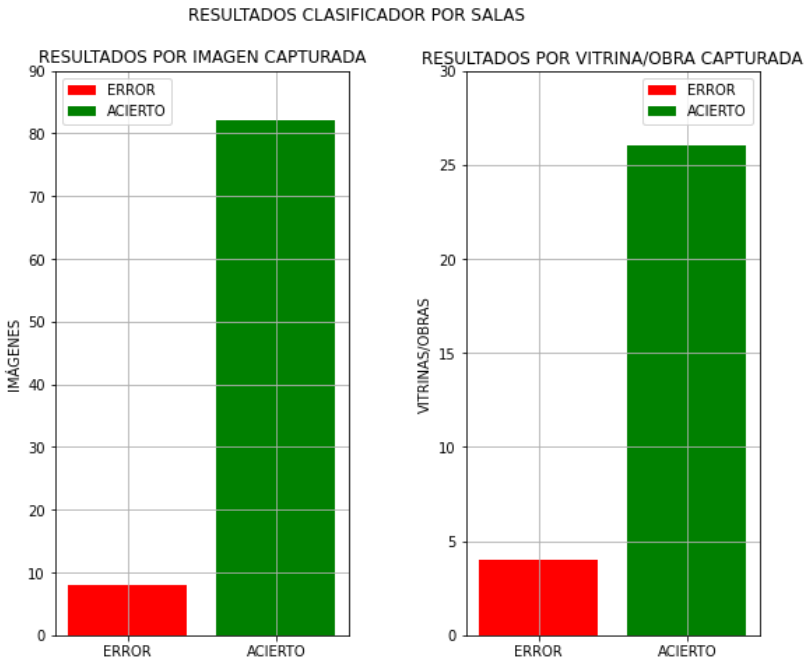


Ilustración 71-Resumen resultados de predicción clasificador por salas

5.1.2 RESULTADOS ENFOQUE CLASIFICADOR POR VITRINAS

Repitiendo el circuito de experimentación con la aplicación del clasificador de imágenes por vitrinas integrado, se obtiene el siguiente listado de probabilidades a partir de localizarnos con un único intento de fotografía por cada perspectiva planteada para cada vitrina/obra.

Incluso con la simple observación de la ilustración 72 se puede comprobar que los resultados estudiados de manera individual por cada fotografía realizada arroja una tasa de 1 error de las 90 fotografías efectuadas, traduciéndose en el porcentaje de acierto del 98.88%, casi idéntico al porcentaje de precisión que se obtuvo en la etapa de entrenamiento del 99.32% cuando se exporto a formato “tflite”. Debido a ello presenta en este primer análisis un comportamiento excepcional en la práctica, prediciendo correctamente a falta de 1 caso la totalidad de los casos de la experimentación.

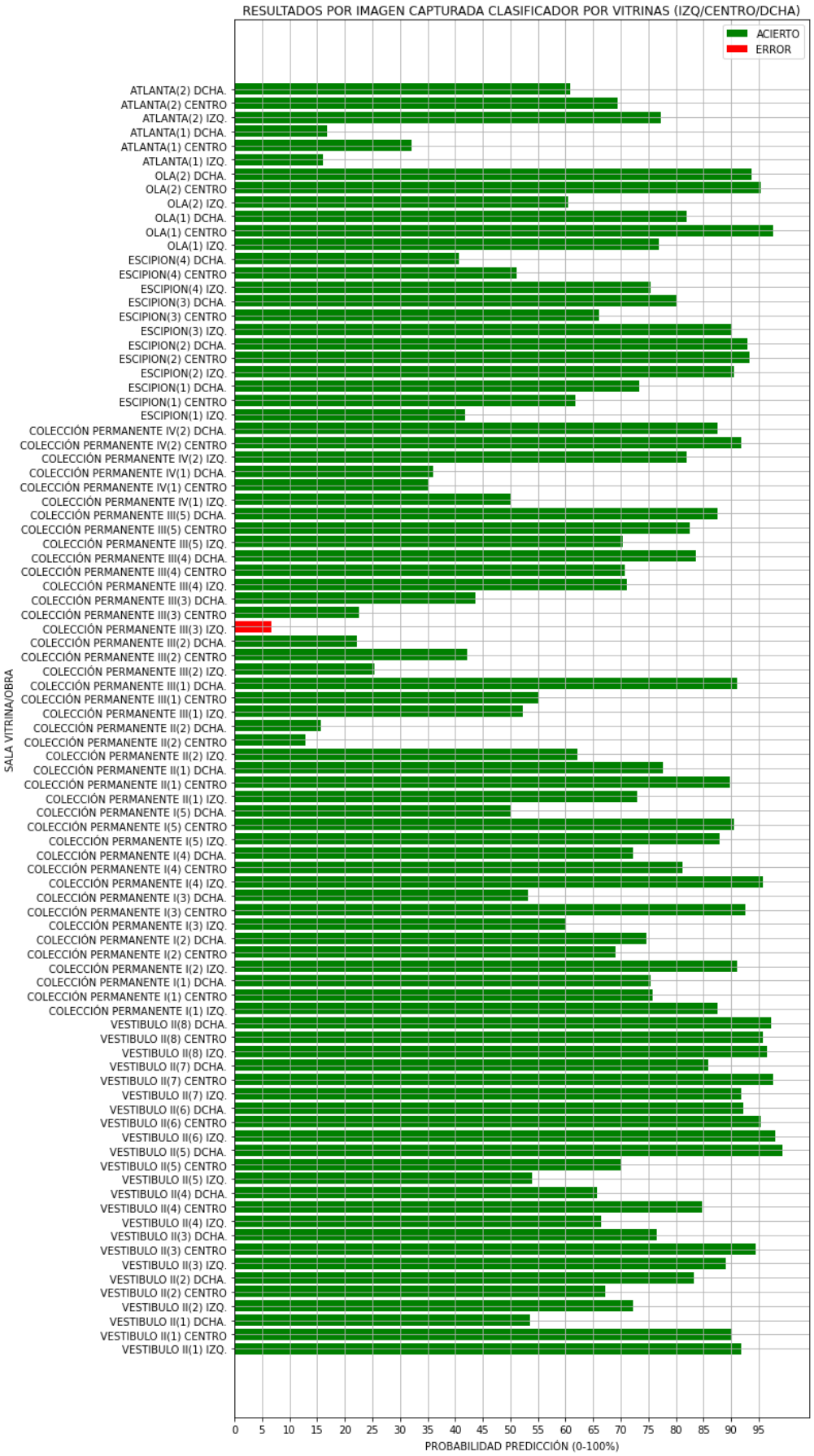


Ilustración 72-Resumen resultados predicciones por imagen de clasificador por vitrinas

Siguiendo la evaluación con la agrupación de las probabilidades a nivel de vitrina/obra fotografiada se obtiene el mejor resultado posible ya que no se presentan errores significando que la tasa de precisión es del 100%, teniendo un comportamiento experimental excelente. La mejora respecto al error encontrado en el análisis individual por fotografía se debe a que el modelo es capaz de reconocer la vitrina/obra con mayor probabilidad que la obtenida en la predicción errónea, compensando el error de predicción.

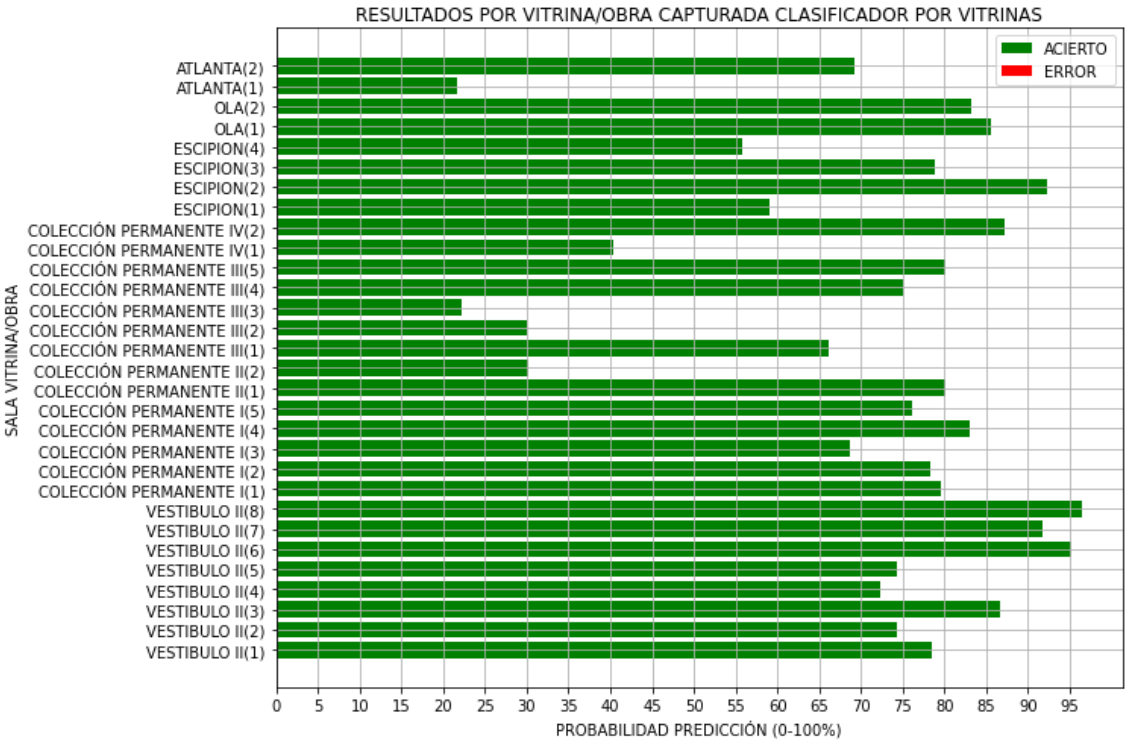


Ilustración 73-Resumen resultados predicciones por vitrina/obra de clasificador por vitrinas

Agrupando los análisis efectuados con ambas perspectivas se concluye que el enfoque y el modelo que lo implementa son de muy buena calidad y muy eficaz en el problema experimental aplicado. Esta conclusión se respalda en la superación de la propuesta de presentar una precisión del 75% llegando a la excelencia en el estudio por vitrina/obra. Además de en la tasa de precisión también se observan los buenos resultados obtenidos al tener una media de probabilidad por predicción acertada del 71.19% en el análisis por fotografía tomada y del 70.40% en el análisis por vitrina/obra. A la hora de examinar la probabilidad media por predicción errónea se obtiene un 6.64%, resultado muy positivo para que en caso de la predicción sea errónea por nivel de probabilidad no se consideré verdadera, esto también se debe al funcionamiento de la capa clasificadora del modelo de inferencia cuya probabilidad repartida

equitativamente entre las etiquetas es del 0.92% de base por etiqueta, repartiéndose demasiado la probabilidad en caso de no tener una predicción clara.

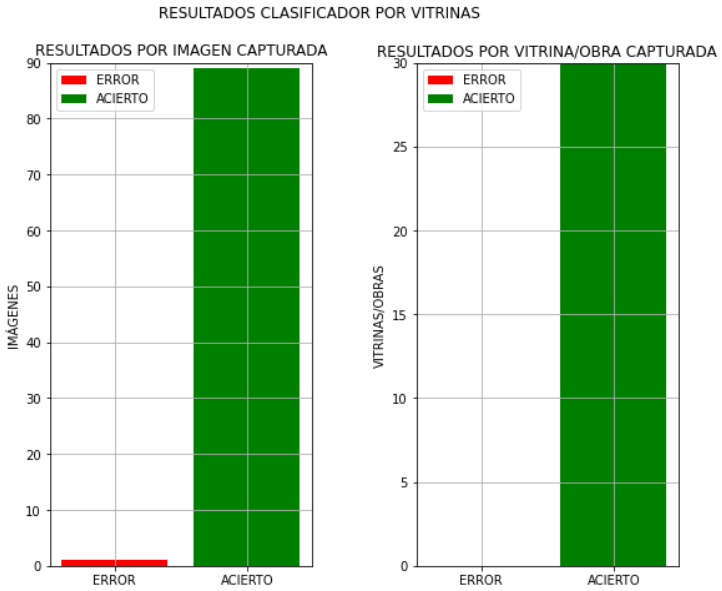


Ilustración 74-Resumen resultados de predicción clasificador por vitrina

5.1.3 COMPARATIVA RESULTADOS DE AMBOS MODELOS

Como se ha expuesto anteriormente, los resultados obtenidos en ambos circuitos de experimentación han superado las expectativas iniciales. Existen por tanto dos soluciones viables para la resolución del problema de la localización de personas en entornos cerrados. Es por ello que se desea conocer cuál es la opción que presenta mejores resultados. A continuación, se compararán sus principales atributos.

Como primera medida para comparar ambas opciones se puede recurrir a medir la precisión obtenida por ambos modelos, a partir de la división de predicciones correctas entre las predicciones realizadas. Estas se evaluarán por fotografía tomada (naranja) y por vitrina evaluada (azul), por cada clasificador. Los resultados se muestran en la ilustración 75.

Los resultados en precisión son bastantes altos (> 86.66%) en el estudio por vitrina/obra de la experimentación con el clasificador por salas, pero valorando este atributo destaca el modelo con el clasificador por vitrinas presentando mejores tasas e incluso presentando en el estudio por vitrina/obra una tasa cercana al de 100% de precisión.

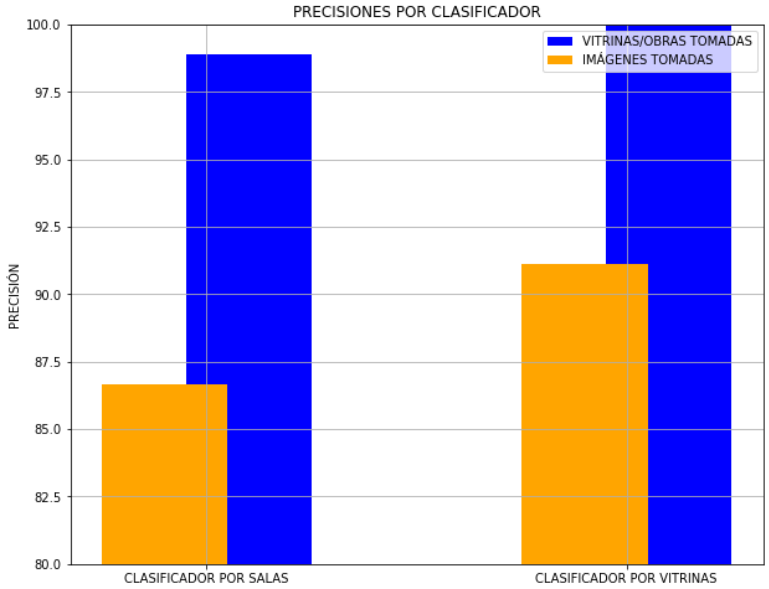


Ilustración 75-Comparativa precisión de clasificadores

Como segunda medida para finalizar la comparativa entre los resultados de los clasificadores, se opta por calcular la probabilidad media que se obtiene por predicción cuando es errónea o cuando es correcta. Para obtener este dato se estudian los resultados por enfoque, por tipo de agrupamiento y por tipo de resultado de predicción. Esto significa seleccionar todas las predicciones con mismo resultado, correcto o erróneo, realizar un sumatorio de las probabilidades obtenidas por cada predicción y finalmente dividir este valor entre el número de predicciones sumadas. Por lo que las barras de la ilustración 76 representan la probabilidad media que una predicción errónea puede presentar (rojo) y la probabilidad media que una correcta puede presentar (verde).

Los resultados de probabilidad media obtenidos cuando la predicción es correcta demuestran predicciones sólidas y fiables al presentar un valor elevado de probabilidad como resultado (> 70.4 %). Estos valores ayudan al usuario a identificar la predicción como válida cuando los resultados superan o son próximos a este umbral, además de demostrar el buen funcionamiento del clasificador cuyas predicciones acumulan la probabilidad sobre la etiqueta correcta, sin presentar segundas opciones incorrectas que compitan con el primer resultado en probabilidad. En este caso el modelo clasificador por salas presenta los resultados más fiables llevándose poca diferencia con el modelo alternativo.

A diferencia de las probabilidades medias obtenidas cuando la predicción es correcta, cuando es incorrecta las probabilidades medias obtenidas por modelo son más dispares. Esta variación se refleja en el salto de probabilidad media obtenida de

hasta un 40 % entre ambos clasificadores. La evidente disparidad se debe al enfoque generalista que sigue el clasificador por salas; el modelo al generalizar todas las obras/vitrinas de distintos tipos de una misma sala puede no encontrar un patrón común entre ellas que se asocie a la sala. La dificultad de ajustar un patrón común da un mayor margen a las predicciones erróneas, ya que el clasificador no es capaz de dictaminar con seguridad una etiqueta donde acumular una mayoría de probabilidad y debido a la capa softmax del clasificador, esta se reparte entre las restantes opciones. Por ello tipos de obras/vitrinas que se repitan en diversas salas ocasionarán que la probabilidad se reparta entre las distintas salas, ofreciendo una predicción con un valor promedio de probabilidad (~40-50 %) pudiendo estar la etiqueta correcta como segunda o tercera etiqueta más probable. Por el contrario, el clasificador por vitrinas al ser una solución más específica durante el entrenamiento obtiene patrones más ajustados a las obras/vitrinas, que le permiten realizar predicciones más sólidas al acumular una mayoría de probabilidad sobre una etiqueta, que tiende a ser la correcta. Por este motivo, el clasificador por vitrinas es una mejor opción ya que cuando se obtiene una predicción errónea tiene un valor de probabilidad muy reducido, entendiendo rápidamente que la predicción sea seguramente errónea, como los resultados demuestran.

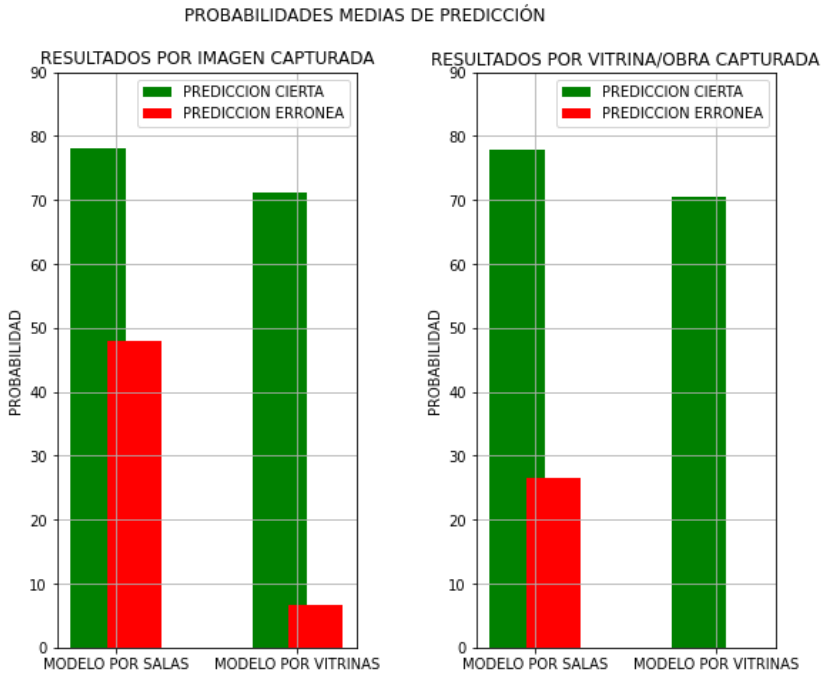


Ilustración 76-Comparativa de probabilidades medias de predicción exitosa y errónea

Tras esta reflexión sobre los atributos resultantes del comportamiento de los modelos en los circuitos experimentales, se concluye que, aunque ambas soluciones son lo suficientemente buenas, el enfoque más específico del clasificador por vitrinas es el más efectivo y fiable para la aplicación en los problemas de localización en interiores.

Finalmente, se puede obtener la aplicación y su desarrollo completo accediendo al repositorio público en Github con el título de "LocalizacionDePersonasAPartirDelReconocimientoAutDeImagenes" [13]; ofreciendo acceso libre a cualquier usuario con interés en el desarrollo de una solución de la misma naturaleza, teniendo posibilidad de replicar el trabajo realizado.

6 CONCLUSIONES Y TRABAJOS FUTUROS

Finalmente, tras la realización del TFG es hora de realizar una pequeña reflexión respecto los objetivos planteados en el inicio del mismo y valorar posibles usos de la solución desarrollada.

Con la creación del clasificador de imágenes se ha conseguido comprender el funcionamiento de la inteligencia artificial, específicamente las redes convolucionales, y su implementación mediante el uso de las múltiples herramientas y librerías orientadas a la ciencia de datos y la propia inteligencia artificial, fáciles y sencillas de utilizar por cualquier usuario con una base de conocimientos.

La integración del modelo en una aplicación Android demuestra la adaptación que permite este tipo de tecnología, convirtiéndose en más accesible de cara al usuario promedio mediante implementaciones en aplicaciones móviles como la desarrollada.

Con la puesta en marcha de la aplicación experimental en el Museo Provincial de Guadalajara y examinando los resultados, deja al descubierto su buen desempeño en el problema de la localización, con resultados que superan a los planteados en un primer momento al inicio del TFG.

Por todo ello se concluye que el desarrollo es exitoso, pudiéndose aplicar las metodologías seguidas a problemas de índole similar o incluso planteándose futuras mejoras al desarrollo implementado.

[13] LocalizacionDePersonasAPartirDelReconocimientoAutDeImagenes, repositorio Github. https://github.com/carlosbrkao/LocalizacionDePersonasAPartirDelReconocimientoAutDeImagenes

6.1 POSIBLES CAMPOS DE APLICACIÓN

Como se ha podido comprobar a lo largo del desarrollo del TFG los campos de aplicación de un clasificador de imágenes orientado a la localización de personas depende de los siguientes pilares.

El primer pilar es que para el clasificador funcione es necesario que el conjunto de etiquetas a clasificar pueda ser distinguible ya que en caso de ser etiquetas con un complejo grado de predicción, hasta para un experto humano, puede ser un objetivo inalcanzable. Un ejemplo de ello sería predecir cuantos días lleva una fruta en descomposición, ya que los cambios evidentes se podrían clasificar, pero detectar modificaciones por días sería complicado por los pequeños cambios que presentaría.

El segundo pilar es la importancia en la relación entre las etiquetas predichas y la ubicación que se la vincula, evento fundamental para traducir la salida de un modelo a una localización real. Dentro de esta vinculación entra el aspecto de la selección de elementos a predecir de interés y distinguibles por la inteligencia artificial, ya que todos los elementos que presente un entorno pueden no ser válidos para ubicarse.

Por tanto, los campos de aplicación en los que se podría utilizar una aplicación de estas características serían en entornos donde se cumplan estos dos pilares y la señal GPS fuese débil, teniendo que recurrir a estas alternativas. Como se ha demostrado en la experimentación, sería útil para museos, edificios históricos…

6.2 TRABAJO FUTURO: MEJORAS Y ESCALABILIDAD

Los campos de empleo de la aplicación como se ha explicado son múltiples para el estado actual implementado en este TFG, pero es un desarrollo que está abierto a mejorar para poder ofrecer resultados más fiables y precisos, como cualquier sistema informático.

A continuación, se exponen tres ideas de mejora que podrían desarrollarse para un sistema más perfeccionado, con cambios asequibles partiendo del proyecto actual.

Modelo de inferencia doble para comprobar resultados por comparación con las salidas de ambos modelos.

A la hora de mejorar el sistema, al igual que se asigna una probabilidad de corte que decide si la predicción se puede considerar verdadera o errónea, una manera de sustituir este método de verificación sería este modelo doble.

Consistiría en pasar la misma imagen capturada por ambos modelos y considerar que la predicción es cierta siempre que se obtenga el mismo resultado en ambos, y por el contrario ser erróneo en caso de que no coincidan en su predicción.

Un ejemplo de ello podía ser aunar ambos clasificadores probados en una misma aplicación y pasar la imagen tomada por ambos, comparando los resultados que ofrecen sobre la misma fotografía.

Modelo de inferencia alojado en un servidor al que se le envían solicitudes con las imágenes a tratar de predecir

Esta implementación destaca por el intercambio entre perdida de latencia, ya que se tienen que comunicar los dispositivos con un servidor, por ganancia en modelos más pesados. Esto se traduce en modelos destinados a tener mayor capacidad de manejar mayor número de pesos, pudiendo optar a mayores precisiones que modelos más pequeños como los implementados en una aplicación móvil. A su vez también se pueden obtener resultados de manera más rápida al tener mayores capacidades de cómputo.

Modelo de inferencia alojado en un servidor con supervisión de un agente

Esta opción saca partido de la capacidad superior de un modelo más elaborado soportado en un servidor que recibe las imágenes obtenidas por los dispositivos de los usuarios y añade la opción de introducir un agente que se encarga de seguir con el entrenamiento constante del modelo, mediante la supervisión de las predicciones que realiza e indicando al modelo si ha sido correcta o en caso de ser incorrecta indicar que etiqueta tenía que haber predicho, continuando el entrenamiento mientras que está en uso.

7 BIBLIOGRAFÍA

[1] Pavithra Babu, “10 Airports using beacons to take Passenger experience to the next level” (10 de octubre de 2021), Beaconstac Blog

https://blog.beaconstac.com/2016/03/10-airports-using-beacons-to-take-passenger-experience-to-the-next-level/

[2] Accent systems, “Beacons: the cornerstone of indoor positioning”, Accent systems

https://accent-systems.com/beacons-the-cornerstone-of-indoor-positioning/

[3] QRcode Chimp, “Generador de código QR para Google Maps”, QRcode Chimp

https://es.qrcodechimp.com/qr-code-generator-for-googleMaps#:~:text=%C2%BFQu%C3%A9%20es%20un%20c%C3%B3digo%20QR,y%20escribir%20la%20direcci%C3%B3n%20manualmente.

[4] National Geographic España, “Breve historia visual de la inteligencia artificial” (2 de diciembre de 2020), National Geographic España

https://www.nationalgeographic.com.es/ciencia/breve-historia-visual-inteligencia-artificial_14419

[5] Diego de la Torre, “¿Cómo nació la Inteligencia Artificial? (2018), ThinkBig

https://blogthinkbig.com/historia-como-nacio-inteligencia-artificial

[6] Rockwell Anyoha, “The History of Artificial Intelligence” (28 de agosto de 2017), SITN Harvard University

https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/

[7] Javier Pastor, “Que es la inteligencia artificial” (20 de octubre de 2018), Xataka

https://www.xataka.com/robotica-e-ia/que-inteligencia-artificial

[8] Juan Antonio Pascual Estapé, “Inteligencia artifinal: qué es, cómo funciona y para qué se utiliza en la actualidad” (24 de agosto de 2019), Computer Hoy

https://computerhoy.com/reportajes/tecnologia/inteligencia-artificial-469917

[9] Azure, “¿Qué es la inteligencia artificial?, Azure

https://azure.microsoft.com/es-es/overview/what-is-artificial-intelligence/#types

[10] Usuarios EcuRed , “Sistemas Expertos” (13 de marzo de 2018), EcuRed

https://www.ecured.cu/Sistemas_expertos

[11] Nicoletta Boldrini, “Sistemas Expertos: qué son, su clasificación, cómo funcionan y para qué se utilizan” (19 de noviembre de 2021), Innovación Digital 360

https://www.innovaciondigital360.com/i-a/sistemas-expertos-que-son-su-clasificacion-como-funcionan-y-para-que-se-utilizan/

[12] Microsoft, “Aprendizaje profundo frente a aprendizaje automático en Azure Machine Learning” (5 de enero de 2022), Microsoft

https://docs.microsoft.com/es-es/azure/machine-learning/concept-deep-learning-vs-machine-learning

[13] Brett Grossfeld, “Aprendizaje profundo y aprendizaje automático: una forma sencilla de entender la diferencia” (23 de enero de 2020), Blog de Zendesk

https://www.zendesk.es/blog/machine-learning-and-deep-learning/

[14] José Antonio Sanchez, “¿Cómo aprenden las máquinas? Machine Learning y sus diferentes tipos” (31 de agosto de 2020), Datos.gob.es

https://datos.gob.es/es/blog/como-aprenden-las-maquinas-machine-learning-y-sus-diferentes-tipos

[15] Jaspreet, “A Concise History of Neuronal Networks” (14 de agosto de 2016), Towards Data Science

https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec

[16] Mike A., “Perceptrón ¿qué es y para qué sirve?” (7 de marzo de 2022), DataScientest

https://datascientest.com/es/perceptron-que-es-y-para-que-sirve#:~:text=Un%20perceptr%C3%B3n%20es%20una%20neurona%20artificial%2C%20y%2C%20por%20tanto%2C,aprendizaje%20supervisado%20de%20clasificadores%20binarios.

[17] Kate Strachnyi, “Brief History of Neural Networks” (23 de enero de 2019), Medium

https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec

[18] Pedro Larrañaga, Iñaki Inza y Abdelmalik Moujahid, “Tema 8. Redes Neuronales”, Dpto. de Ciencias de la Computación e Inteligencia Artificial de la Universidad del País Vasco-Euskal Herriko Unibertsitatea

http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf

[19] TIBCO, “What is a Neuronal Network?”, TIBCO

https://www.tibco.com/reference-center/what-is-a-neural-network

[20] José David Villanueva García, “Redes neuronales desde cero (II): algo de matemáticas” (18 de septiembre de 2020), IArtificial.net

https://www.iartificial.net/redes-neuronales-desde-cero-ii-algo-de-matematicas/

[21] José David Villanueva García, “Redes neuronales desde cero(I) – Introducción” (23 de octubre de 2020), IArtificial.net

https://www.iartificial.net/redes-neuronales-desde-cero-i-introduccion/

[22] Juan Ignacio Bagnato, “¿Cómo funionan las Convolutional Neural Networks? Visión por Ordenador” (29 de noviembre de 2018), AprendeMachineLearning.com

https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

[23] Francisco Barber, "¿Qué es un píxel y cuál es su función en fotografía digital?" (2 de noviembre de 2016), CocoSchool

https://www.cocoschool.com/pixel-funcion-fotografia-digital/#:~:text=Un%20p%C3%ADxel%20es%20el%20punto,im%C3%A1genes%20en%20millones%20de%20p%C3%ADxeles.

[24]Sarahí Silva y Estefanía Freire, "Intro a las redes neuronales convolucionales" (23 de noviembre de 2019), BootCamp AI

https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8

[25] Aarthi Ravikumar, "Historia de los satélites GPS y el seguimiento por GPS para uso comercial" (23 de junio de 2020), Geotab.com

https://www.geotab.com/es/blog/historia-de-los-satelites-gps/

[26] Gabri, "¿Cómo funcionan los dispositivos GPS? Trilateración vs Triangulación" (30 de mayo de 2018), Acolita.com

https://acolita.com/como-funcionan-los-dispositivos-gps-trilateracion-vs-triangulacion/

[27] Atria Innovation, "Sistemas de localización en interiores" (4 de agosto de 2020), Atria Innovation

https://www.atriainnovation.com/sistemas-localizacion-interiores/

[28] Rf-star, "La tecnología de baliza facilita el posicionamiento en interiores" (29 de octubre de 2021), Rf-star

https://es.rfstariot.com/beacon-technology-makes-indoor-positioning-easier_n111

[29] SITUM, "Sistemas de localización en interiores" (24 de febrero de 2022), SITUM

https://situm.com/es/blog/posicionamiento-en-interiores/sistemas-de-localizacion-en-interiores/

[30] Mecalux Esmena, "Códigos QR en logística: velocidad y flexibilidad" (7 de julio de 2020), Mecalux Esmena

https://www.mecalux.es/blog/qr-logistica

[31] Apoorva Hegde, "QR Codes for Museums: Improve Visitor Experience" (15 de junio de 2022), Beaconstac

https://blog.beaconstac.com/2021/03/qr-codes-for-museums/

[32] Kimadi, "RFID-Tecnología de identificación por radiofrecuencia", Kimadi

https://www.kimaldi.com/rfid_tecnologia_de_identificacion_por_radiofrecuencia/

[33] Tecnipesa, "Qué es y cómo funciona la tecnología RFID" (24 de marzo de 2021), Tecnipesa

https://www.tecnipesa.com/blog/69-tecnologia-rfid-que-ventajas-tiene

[34] Javier Penalva, "NFC: qué es y para que sirve en este 2022" (30 de junio de 2022), Xakata

https://www.xataka.com/moviles/nfc-que-es-y-para-que-sirve

[35] Alexandre Gonfalonieri, “How to build a dataset for your Machine Learning Project” (14 de febrero de 2019), towardsdatascience.com

https://towardsdatascience.com/how-to-build-a-data-set-for-your-machine-learning-project-5b3b871881ac

[36] SelectStar, “Creating the Best Quality Image Dataset” (30 de mayo de 2020), SelectStar

https://selectstar-ai.medium.com/creating-the-best-quality-image-dataset-720f612944ed

[37] Aysengul Takimoglu, “What is Data Augmentation? Techniques & Examples in 2022” (11 de febrero de 2022), AIMultiple

https://research.aimultiple.com/data-augmentation/

[38] Rafael Marín, “¿Qué es OpenCV? Instalación en Python y ejemplos básicos” (12 de febrero de 2020), RevistaDigital INESEM

https://revistadigital.inesem.es/informatica-y-tics/opencv/

[39] Javier Buhigas, “Todo lo que necesitas saber sobre TensorFlow, la plataforma para Inteligencia Artificial de Google” (14 de febrero de 2018), Puentes Digitales

https://puentesdigitales.com/2018/02/14/todo-lo-que-necesitas-saber-sobre-tensorflow-la-plataforma-para-inteligencia-artificial-de-google/

[40] TensorFlow, “Casos de éxito”, TensorFlow

https://tensorflow.org/about/case-studies?hl=es-419

[41] Fisicalandia, “Tensor”, Fisicalandia

https://fisicalandia.com/matematicas/tensor

[42] Ligdi González, “¿Qué es TensorFlow?¿Cómo funciona?” (1 de junio de 2021), AprendeIA

https://aprendeia.com/que-es-tensorflow-como-funciona/

[43] Renu Khandelwal, “A Basic Introduction to TensorFlow Lite” (15 de junio de 2020), Towards data science

https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292

[44] TensorFlow, “Creador de modelos TensorFlow Lite”, TensorFlow

https://www.tensorflow.org/lite/guide/model_maker

[45] TensorFlow, “TensorFlow Hub”, TensorFlow

https://www.tensorflow.org/hub?hl=es-419

[46] Luigys toro, “Anaconda Distribution: La Suite más completa para la Ciencia de datos con Python”, DesdeLinux

https://blog.desdelinux.net/ciencia-de-datos-con-python/

[47] IONOS, “Jupyter Notebook: documentos web para análisis de datos, código en vivo y mucho más” (28 de febrero de 2019), IONOS Digital Guide

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/jupyter-notebook/

[48] Android developers, “Introducción a Android Studio” (17 de mayo de 2021), Android developers

https://developer.android.com/studio/intro?hl=es-419#:~:text=Un%20sistema%20de%20compilaci%C3%B3n%20flexible,app%20en%20ejecuci%C3%B3n%20sin%20reiniciarla

[49] Anastasia Osypenko, “Mapbox vs Google Maps: Choosing a Map for Your App”, Madappgang

https://madappgang.com/blog/mapbox-vs-google-maps-choosing-a-map-for-your-app/

[50] Pablo Guardiola, “Mapbox: el SDK de mapas abierto” (24 de octubre de 2017), Genbeta

https://www.genbeta.com/desarrollo/mapbox-el-sdk-de-mapas-abierto

[51] Mokhtar Ebrahim,“Tutorial de procesar imágenes en Python (usando OpenCV)” (5 de marzo de 2019), Likegeeks

https://likegeeks.com/es/procesar-de-imagenes-en-python/#Detectar_los_bordes

[52] TensorFlow, “Creador de modelos TensorFlow Lite”, TensorFlow

https://www.tensorflow.org/lite/guide/model_maker

[53] Mingxing Tan, “EffificientNet: Improving Accuracy and Efficiency through AutoML and ModelScaling” (29 de mayo de 2019), Google AI Blog

https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

ANEXO I – PLIEGO CONDICIONES

En el siguiente anexo se listan el conjunto de equipos utilizados a lo largo del desarrollo, tanto para el entrenamiento del clasificador de imágenes y desarrollo de la aplicación; como para la instalación y puesta en marcha de la aplicación en un dispositivo móvil.

A I.1 Máquina para el entrenamiento de la IA y el desarrollo Android

Ordenador de sobremesa con prestaciones suficientes para soportar un proceso de entrenamiento en tiempos aceptables para la generación del clasificador de imágenes y el desarrollo de la aplicación Android.

- Procesador: Intel ® Core ™ i7-10700F CPU @ 2.90GHz
- Memoria: 16 GB RAM DDR4 (2 Slots x 8 GB RAM DDR4)
- Gráfica: NVIDIA GeForce RTX 3060 Ti 16 GB~
- Sistema operativo: Windows 10 Home 64bits (10.0, compilación 19043)

A I.3 Dispositivo portable de pruebas

Terminal móvil de carácter Smartphone para la instalación de la aplicación Android final y prueba experimental del TFG.

- Procesador: Qualcomm ® Snapdragon ™ 845
- Coprocesador: Pixel Visual Core
- Memoria: 4 GB RAM LPDDR4
- Gráfica: Adreno 630
- Sistema operativo: Android 12
- Cámara trasera: 12.2 MP

ANEXO II – PRESUPUESTO

Como segundo anexo del TFG se expone la información de tiempo y coste invertido a lo largo del desarrollo. Por lo que se expone una relación entre las tareas enumeradas y tiempo invertido debido a su complicación, y los costes relacionados a la realización de las tareas.

A II.1 DURACIÓN DEL PROYECTO

El TFG se ha fragmentado en fases con el fin de distribuir el trabajo por objetivos de un mismo tipo. Para reflejar el tiempo dedicado a cada fase, significando un mayor tiempo una etapa más compleja que las demás; se opta por una estimación de tiempo empleando la medida de semanas.

Tras realizar la división de fases obtenemos un total de 9 etapas cuya carga de trabajo se puede distribuir en un tiempo aproximado de 20 semanas, dando por concluido el proyecto al final de estas.

Como se puede observar hay etapas que se pueden realizar concurrentemente al no depender una de otra en su totalidad [Ilustración 77]. Por ejemplo, se puede obtener el dataset cuando el análisis del problema está casi finalizado al tener claro el tipo del dataset a recolectar en ese momento del análisis. Además, esta acción puede retroalimentar a la primera tarea ya que se puede obtener información del entorno de aplicación a la hora de realizar el dataset.

El resumen de las fases se muestra visualmente en el siguiente cronograma, mostrando el orden y tiempo en semanas dedicado por tarea. En total son 20 semanas, una media de trabajo de 25h semanales, obteniendo una dedicación cercana a las 500h. Siendo un reflejo de la experiencia de realización del TFG.

FASES	SEMANAS																			
	1	2	3	4	4	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Analisis del problema																				
Obtención del dataset																				
Selección clasificador de imágenes																				
Desarrollo aplicación Android																				
Entrenamiento clasificador de imágenes																				
Experimentación																				
Reajuste del clasificador y la aplicación																				
Resultados																				
Redacción de documentación																				

Ilustración 77-Cronograma de tiempo dedicado por fase

A II.2 COSTES DEL PROYECTO

Para realizar un análisis de los costes que puede significar la realización de un proyecto de esta índole, se desglosa el listado en una primera parte del equipo

hardware utilizado para el desarrollo y pruebas de la aplicación, seguido de una segunda parte orientada al coste laboral de desarrollo de la aplicación.

Como listado hardware del que disponer para realizar el proyecto se considerará el utilizado en la elaboración del TFG tasando los equipos a fecha actual de 2022. Al cálculo del coste total se estima que un equipo tiene una duración aproximada de 3 años, por lo que se calcula la parte proporcional al tiempo de desarrollo indicándose entre paréntesis.

EQUIPO HARDWARE	COSTE (€)
Ordenador sobremesa HP Omen 25L GT12	1500 €
Pantalla Acer K242HQL	110 €
Periféricos (Teclado + Ratón)	50 €
Smarthphone Google Pixel 3	500 €
COSTE TOTAL HARDWARE	2160 € (300 €)

Debido a herramientas de código libre se ahorra en la obtención de licencias de programas con los que se consigue las mismas prestaciones. Sin embargo, la licencia del sistema operativo y las herramientas ofimáticas para estudiar los resultados requieren de la necesidad de contratar licencias por meses o años. Igual que anteriormente, se calcula también la parte de coste proporcional al tiempo de desarrollo.

EQUIPO SOFTWARE	COSTE (€)
Windows 10 Home	150 €
Microsoft Office	70 €
COSTE TOTAL SOFTWARE	220 € (92 €)

Finalmente contemplando un desarrollo idéntico como una tarea para una compañía, se estima el sueldo básico que un ingeniero junior percibe para obtener el coste de la mano de obra aproximada. En base a las horas estimadas, agrupándolas en un horario de jornada completa de 40h se quedan en 12 semanas y media. Como sueldo bruto de referencia se obtiene el de 25000€ anuales, presentado por la

empresa tecnológica de Indra[14]. El coste de la mano de obra proporcional a dicho tiempo de trabajo se muestra en la tabla siguiente.

MANO DE OBRA	COSTE (€)
Sueldo Ingeniero Junior (500h)	6250 € Brutos
COSTE TOTAL MANO DE OBRA	6250 €

Aunando el coste hardware, software y humano necesario para el desarrollo de un proyecto siguiendo la hoja de ruta de este TFG, concluye en un gasto total de 6642 €; resultando en un proyecto económico gracias a la apuesta por herramientas de código libre.

PARTES IMPLICADAS	COSTE (€)
Equipo Hardware	300 €
Equipo Software	92€
Mano de obra	6250 €
COSTE TOTAL	6642 €

[14] Glassdoor, Página comparadora de sueldos profesionales.
https://www.glassdoor.es/Sueldos/ingeniero-junior-sueldo-SRCH_KO0,16.htm

ANEXO III – MANUAL DE USUARIO

Poniendo el broche final a la documentación del TFG se realiza una breve explicación del uso de la aplicación al usuario, como se realiza de forma similar en la opción que incluye la propia aplicación. Esta explicación se acompaña de imágenes para comprender la interfaz y las funciones.

A III.1 INICIO

Pulsando sobre el icono de la aplicación en el menú de aplicaciones del dispositivo o en su acceso directo en pantalla, se inicia automáticamente la aplicación, iniciando con una presentación.

Ilustración 78-Pantalla inicio

Esta presentación inicial muestra las partes participantes en la aplicación y solicita los permisos necesarios, que el usuario tendrá que aceptar para usar la aplicación.

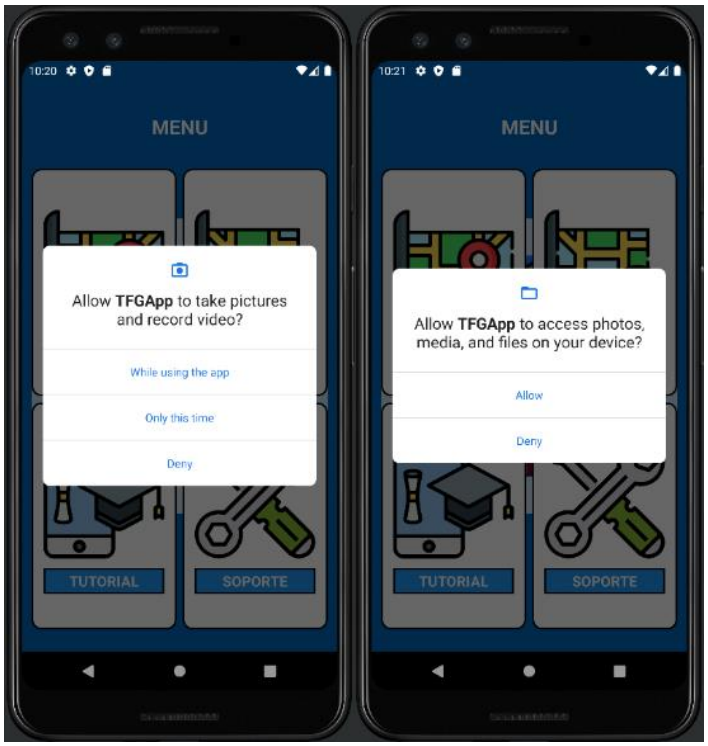


Ilustración 79-Solicitud de permisos

A III.2 MENÚ

Finalizada la presentación inicial, se muestra el menú con las cuatro utilidades implementadas.

- Localización
- Mapa
- Tutorial
- Soporte

Para iniciar cualquiera de ellas se pulsa sobre su botón vinculado identificado por su título e icono relativo a la funcionalidad. Se regresará a la misma pantalla finalizada cada opción, pulsando el botón de retroceder de la navegación del propio dispositivo.

Ilustración 80-Pantalla menú

A III.3 LOCALIZACIÓN

Seleccionada la localización se abre automáticamente la cámara para capturar el entorno donde se quiere localizar el usuario, intentando reflejar elementos que puedan ser característicos del área en cuestión.

Cuando se realiza la fotografía se cambia el interfaz de la cámara por las opciones de aceptar la imagen tomada, rechazar la imagen o tomar otra imagen.

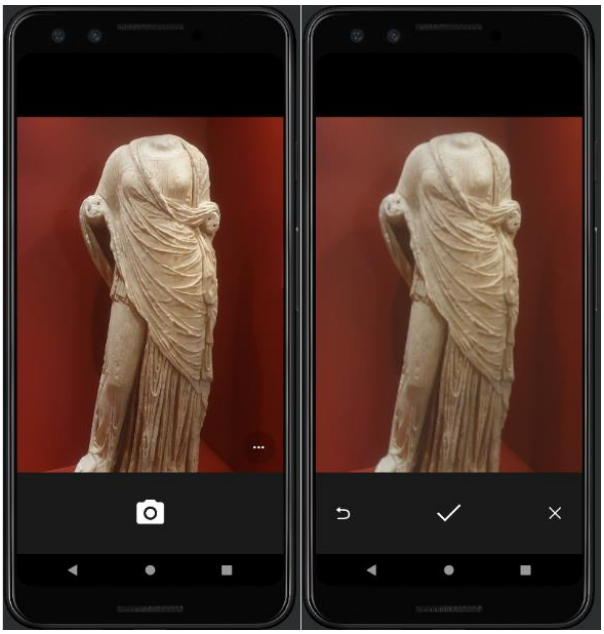


Ilustración 81-Pantalla cámara

Tras aceptar la foto realizada se inicia el proceso de inferencia denotado por una animación de carga. Cuando finaliza se refleja la opción con mayor probabilidad, indicando está acompañada de una tonalidad en señal de consideración de calidad del resultado; y se pone una marca indicando la localización predicha sobre el mapa. Junto al indicador se despliega un cuadro que presenta información vinculada a la ubicación.

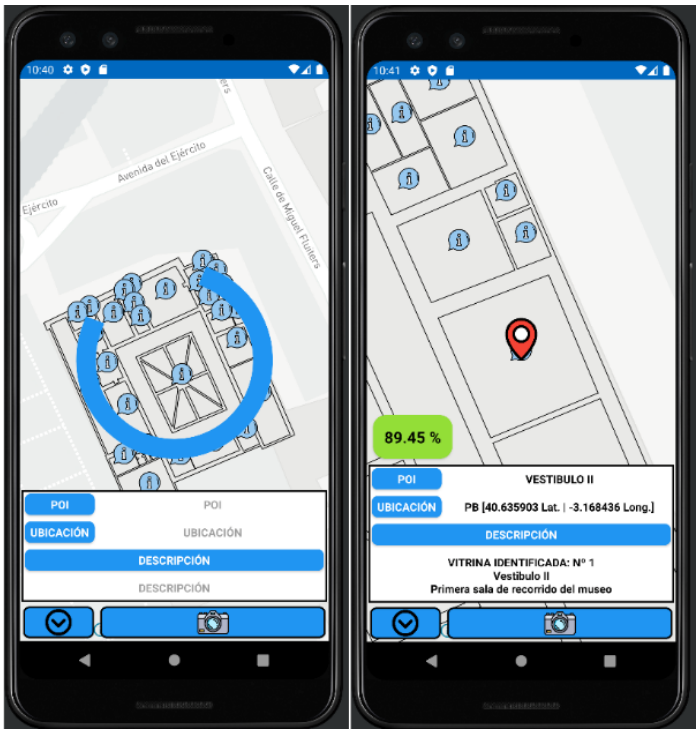


Ilustración 82-Pantalla localización

A continuación, se puede repetir el proceso pulsando sobre el botón con una cámara o iniciar la libre navegación sobre el mapa.

A III.4 MAPA

Es la opción que despliega el mapa con toda la información disponible sobre este para que el usuario pueda navegar libremente sobre este. También se llega a esta opción al final de la función de localización.

Consta de las opciones de *zoom* o desplazamiento con la interacción de los gestos que se realizan sobre el mapa, acompañado de la navegación entre los distintos niveles con los botones ligados a cada altura del plano.

Con la interacción sobre los puntos de interés mostrados se puede comprobar la información relacionada a estos como se realiza en la anterior función.

Ilustración 83-Pantalla mapa

A III.5 TUTORIAL

Como se está realizando en este anexo, se muestra un listado ordenado de vistas formadas por una imagen y texto con el fin de tener un recurso de consulta del funcionamiento de la aplicación en la misma. Deslizando con el dedo se pueden recorrer.

Ilustración 84-Pantalla tutorial

A III.6 SOPORTE

Opción orientada a la puesta en marcha de la aplicación en un entorno real, con el fin de recoger y realizar el soporte de incidencias que se puedan ocasionar con el uso de la aplicación por parte de los usuarios.

Se presentaría la información y facilidades para la puesta en contacto con el soporte técnico.

A III.7 LEYENDA

Para facilitar la comprensión al usuario se realiza una relación entre iconos característicos y significados que ayudan a identificar los distintos tipos de información o acciones que se pueden realizar.

A continuación, se muestra la relación icono-significado de los elementos más destacables.

ICONO	SIGNIFICADO
	Localización, opción para localizarse usando el modelo de inferencia con las fotografías tomadas
	Mapa, opción para el despliegue del mapa para la navegación libre
	Tutorial, opción para el repaso de los conceptos básicos de uso de la aplicación
	Soporte, punto de información para el soporte técnico de la aplicación
	Punto de interés (POI), identificando cada posible localización
	Localización, identificando selección manual o resultado de predicción de una ubicación del mapa
	Mostrar/Ocultar, identificando la opción de reducir la información en pantalla o mostrándola de nuevo
	Cámara, identificando la opción de localizarse a partir de la toma de imágenes con la cámara

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá