UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Prototipo de sistema móvil para reconocimiento de personas requeridas por organismos de seguridad y control a través de reconocimiento facial aplicando Redes Neuronales Convolucionales y Transferencia de Aprendizaje.

Samy Amaru Conejo Gualacata

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 20 de diciembre de 2022

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Prototipo de sistema móvil para reconocimiento de personas requeridas por organismos de seguridad y control a través de reconocimiento facial aplicando Redes Neuronales Convolucionales y Transferencia de Aprendizaje.

Samy Amaru Conejo Gualacata

Nombre del profesor, Título académico **Felipe Grijalva, PhD.**

Quito, 20 de diciembre de 2022

© DERECHOS DE AUTOR

Nombres y apellidos: Samy Amaru Conejo Gualacata

Código: 00202787

Cédula de identidad: 1005029937

Lugar y fecha: Quito, 20 de diciembre de 2022

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

El reconocimiento facial es un método fundamental en la ciencia de datos relacionada con la detección de rostros, autenticación y monitoreo. Esta tecnología permite aprovechar la disponibilidad de bases de datos de imágenes para diversos fines. Organismos como la Interpol aplican el reconocimiento facial haciendo uso de una base de datos compartida con más de 179 países para identificar personas en procesos de investigación para fines de seguridad internacional. El objetivo de este proyecto es modelar una arquitectura de Red Neuronal Convolucional aplicando Transferencia de Aprendizaje y probar su funcionamiento en un dispositivo móvil para detectar a 10 participantes en un simulacro de control de seguridad. El framework para el desarrollo móvil es Flutter que nos permite usar el API de Visión de Google para el proceso de detección de rostros. El modelo de RNC generado es compatible con Flutter gracias al complemento de TensorFlow Lite y es desplegable en dispositivos iOS y Android. La motivación es mostrar como las tecnologías modernas nos permiten crear sistemas sofisticados, portables e inteligentes con aplicaciones reales, específicamente en temas de seguridad. En el estudio se observa que la arquitectura escogida, MobileNetV2, presenta una tasa de exactitud del 99.9 % sobre el set de prueba.

Palabras clave: Redes Neuronales Convolucionales, Transferencia de Aprendizaje, Vision por Computadora, MobileNetV2, Imagenet, TensorFlow, Keras, Google ML, Flutter, iOS.

ABSTRACT

Face recognition is a fundamental method in data science related to face detection, authentication, and monitoring. This technology makes it possible to take advantage of the availability of image databases for various purposes. Organizations such as Interpol apply face recognition using a database shared with more than 179 countries to identify people in investigation processes for international security purposes. The objective of this project is to model a Convolutional Neural Network architecture applying Transfer Learning and test its operation in a mobile device to detect 10 participants in a security control simulation. The framework for the mobile development is Flutter which allows us to use the Google Vision API for the face detection process. The generated RNC model is compatible with Flutter thanks to the TensorFlow Lite add-on and is deployable on iOS and Android devices. The motivation is to show how modern technologies allow us to create sophisticated, portable, and intelligent systems with real applications, specifically in security issues. The study shows that the chosen architecture, MobileNetV2, presents an accuracy rate of 99.9 % on the test set.

Keywords: Convolutional Neuronal Networks, Transfer Learning, Computer Vision, MobileNetV2, Imagenet, TensorFlow, Keras, Google ML, Flutter, iOS.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En Ecuador para 2020 se instalaron 78 cámaras de video vigilancia en el Centro Histórico de Quito que cuentan la tecnología para reconocimiento facial misma que ha permitido detectar hasta 23 de julio de 2020 a 45 personas que han incurrido en delitos (Bravo, 2020). La ciudad de Guayaquil en 2021 firmó un contrato para adquirir 100 cámaras de vigilancia de seguridad, software de reconocimiento y servidores con almacenamiento de 225 terabytes (Mella, 2021). El uso de esta tecnología ha generado discusiones relacionadas a la retención, manipulación y procesamiento de datos personales como el rostro. El problema de fondo está relacionado con la normativa necesaria que regula el uso de la información. En este sentido el uso se enfoca en construir bases de datos con personas con antecedentes criminales que representan un riesgo para la seguridad del país y no busca mantener información de toda la población en general. La ONU enfatiza que es de vital importancia usar la Inteligencia Artificial siguiendo altos estándares y normas para que la información sea acertada además de estar respaldada desde la parte jurídica (ONU, 2021). La detección y reconocimiento facial que se propone se implementa bajo el estudio de Redes Neuronales Convolucionales.

La innovación tecnológica moderna ha facilitado la creación de información multimedia en masa que es potencialmente útil para estudios científicos. Estos desarrollos permiten reflexionar sobre el panorama a futuro con el objetivo de crear sistemas cada vez más sofisticados para realizar tareas que implican inteligencia humana (Davis & Ellis, 1964). El reconocimiento inteligente de rostros tiene un sin número de aplicaciones en temas de seguridad, salud e incluso entretenimiento (Harrington, 2012). Organismos como la Interpol aplican el reconocimiento facial haciendo uso de una base de datos compartida con más de 179 países para identificar personas en procesos de investigación para fines de seguridad

internacional. La detección es el proceso por el cual un sistema inteligente es capaz de identificar un rostro humano mientras el reconocimiento implica determinar la identidad del rostro detectado (Zheng et al, 2017).

El área de Deep Learning constantemente busca aplicar algoritmos para mejorar el rendimiento de sistemas y la precisión de los resultados. Para casos de reconocimiento facial, el proceso incluye pre-procesamiento de imágenes de rostros, extracción de características y clasificación. El estudio de las Redes Neuronales Convolucionales ha permitido diseñar y evaluar sistemas de reconocimiento en tiempo real y su aplicación en el campo de la visión artificial ha mejorado significativamente su efectividad (Zheng et al, 2017). Las RNC son útiles para aprender características de bajo a alto nivel por capas y de forma automática aplicando técnicas de Transferencia de Aprendizaje incrementando resultados positivos y entrenando modelos en menos tiempo (Saad et al, 2017). Estas técnicas resultan útiles cuando no se dispone de un set de datos entrenado lo suficientemente grande, reduce la complejidad del proceso de computación y ofrece una clasificación precisa.

En este escenario, este proyecto está enfocado en diseñar e implementar un prototipo de aplicación móvil que permita la detección de personas requeridas por organismos o entes de control de seguridad generando un modelo de ML usando reconocimiento facial con técnicas de Redes Neuronales Convolucionales y Transferencia de Aprendizaje. El proceso se encuentra enmarcado en los ***fundamentos del diseño de ingeniería***: Reconocimiento y definición del problema, recolección de información (Estado del Arte), planteamiento de la configuración experimental óptima, estrategia y materiales necesarios, diseño de análisis y evaluación del modelo propuesto (Haik, 2011).

OBJETIVOS

Objetivo General

- Implementar un prototipo de sistema móvil de seguridad para detección de personas requerida por entes de control a través de reconocimiento facial con Redes Neuronales Convolucionales y Transferencia de Aprendizaje.

Objetivos Específicos

- Crear una base de datos de imágenes que contengan rostros de personas conocidas.
- Modelar una arquitectura de Red Neuronal Convolucional pre-entrenada para uso en reconocimiento facial.
- Implementar el modelo en un sistema móvil.

ESTADO DEL ARTE

La información técnica de los sistemas de detección y reconocimiento biométrico utilizados por los organismos de seguridad y control en Ecuador y el mundo es protegida. En Ecuador se sabe que el principal software utilizado por laboratorios de Criminalística y Ciencias Forenses y el ECU 911 usan AVIS+F que permite comparar rostros y voces para determinar la identidad de una persona en investigación (Policía Ecuador, 2020). El sistema permite detectar similitudes y proporciona información sobre rasgos físicos comparando dos imágenes de entrada (Notimundo, 2016). La policía metropolitana de Reino Unido utiliza un sistema de reconocimiento facial en directo que utiliza algoritmos desarrollado por su Laboratorio Nacional de Física (NPL) y probados por el Instituto Nacional de Normas y Tecnología de Estados Unidos (NIST) (MET, s.f).

El desarrollo y validación de algoritmos aplicando Deep Learning permiten automatizar los sistemas de detección. Gulshan, Coram, et al (2016) propusieron un algoritmo para detección temprana de retinopatía diabética y edema macular diabético en fotografías de fondo de retina. Se usaron imágenes de EyePACS en Estados Unidos y 3 centros de salud especializados en ojos en India. El algoritmo de optimización propuesto para entrenar la red fue una implementación distribuida de descenso de gradiente estocástico propuesto por Dean, et al (2012). La detección de objetos no tiene una limitación de estudio. Kagaya, Aizawa y Ogawa (2014) consiguieron la detección y clasificación de alimentos basado en detección de imágenes creando un dataset donde se almacenaron imágenes publicadas por un sitio llamado Food Log durante dos meses. Demostraron que para su análisis RNC tuvo mejor precisión que SVM. Se configuró local response normalization (LRN) para el proceso de normalización después de las capas de pooling y se aplicó 6-fold cross validation. Para la comparación se probó SVM con spatial pyramid matching, GIST y un paquete de ScSPM.

Coşkun, Uçar, et al (2017) aplicaron una modificación a la arquitectura de RNC añadiendo dos operaciones de normalización, una a la primera capa convolucional y la siguiente para la capa final. La estructura consta de capas convoluciones, pooling y ReLu. Se usó MatConvNet para diseñar la red con un dataset de rostros de Georgia Tech. La red se entrenó para 35 epochs y la clasificación se concluyó con Softmax Classifier. Hu y Peng (2015) desarrollaron XFace, un sistema móvil de reconocimiento de rostros para Android. Para la detección del rostro se usó LBP-based frontal detector y métodos ROI con detección de ojos. Para las pruebas se incluyeron Eigenfaces y Fisherfaces. La implementación del sistema se basó principalmente con OpenCV SDK y Android SDK con Java. Se usó un dataset de 15 clases con mínimo 10 rostros.

Los dispositivos inteligentes modernos cuentan con sistemas de reconocimiento facial pero el uso de la mascarilla creó un nuevo reto para los investigadores. Kocacinar, Tas et al (2022), para 2021 pusieron a prueba las estrategias y tecnologías modernas para crear un sistema de reconocimiento facial para rostros con mascarilla en el contexto de la pandemia por Covid-19. Se creó un dataset de imágenes de personas con mascarilla con el fin de enfocar el análisis en la parte superior del rostro. Se incluyeron tres capas para el modelo de detección: masked, unmasked, improper mask-wearing. Los módulos VGG-16 y MobileNetV2 fueron los más precisos: 99.81% y 99.6% respectivamente. Para la implementación móvil se usó TensorFlowLite. Chowanda y Sutoyo (2019) presentaron la implementación de un modelo de Deep Learning para reconocimiento de rostros en dispositivos móviles usando CoreML. Se probaron dos arquitecturas: VGG-19 y Google Inception V3. Los modelos se convirtieron a modelos ligeros usando CoreMLTools junto con Python. La implementación se realizó en un sistema iOS con Swift y CoreML SDK. La precisión fue de 78.7% para VGG-19 y 78.8% para Inception V3.

Machine Learning

El aprendizaje automático o Machine Learning consiste en algoritmos matemáticos sofisticados y modelos estadísticos que son aplicables a computadores para realizar tareas específicas sin necesidad de ser programados explícitamente para ello. Las aplicaciones de estos algoritmos se usan en minería de datos, procesamiento de imágenes, analítica predictiva y más. La ventaja de usar ML es que una vez que los algoritmos aprenden a manejar los datos, el proceso se puede realizar de forma automática (Batta, 2020).

Aprendizaje Supervisado

Este paradigma de aprendizaje automático consiste en aprender una función que asigna una entrada a una salida basada en pares de ejemplos de entrada – salida. El grupo de datos de entrada este etiquetado y se divide en subconjuntos de entrenamiento y de prueba. El subconjunto de entrenamiento contiene una o varias variables que requieren predecirse. El objetivo del aprendizaje supervisado es construir un sistema artificial que pueda aprender el mapping entre la entrada y la salida, y consiga predecir la salida del sistema dada una nueva entrada (Liu y Wu, 2012).

Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales son un tipo particular de red Deep Learning. Para generalizar el concepto es importante entender que una red neuronal se conforma por varias neuronas artificiales que se encuentran conectadas entre sí a través de capas o layers. Entre capas es posible el intercambio de información. La unidad más simple en el contexto del procesamiento de una red neuronal es el *perceptrón o neurona artificial*, y se representa como $y = Wx + b$ (Gosh et al, 2020). En esta expresión x representa una característica, W los pesos o weights y b el sesgo o bias. El resultado de esta operación será evaluado por una función de activación que permite realizar una inferencia o predicción.

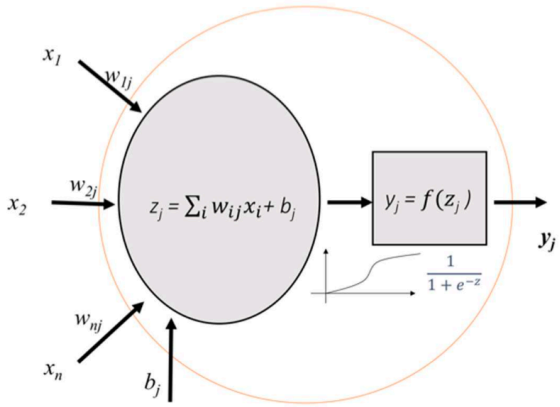

Figura 1. Perceptrón con función de activación sigmoide.

Las Redes Neuronales Convolucionales están inspiradas en el cortex visual animal. Su uso es común en aplicaciones de procesamiento de imágenes, video y NPL. Están compuestas por varias capas que siguen el siguiente esquema. Una capa de entrada o *Input Layer*, varias capas ocultas o *Hidden Layers* y una capa de salida o *Output Layer*. Entre las capas ocultas destacan las convolucionales, agrupación o polling y las densamente conectadas o dense – fully connected (Albawi et al, 2017).

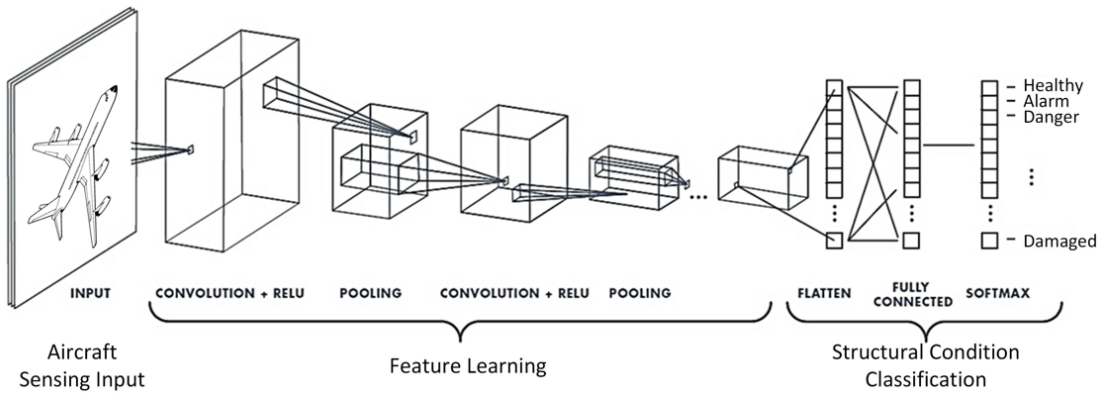

Figura 2. Capas de una red convolucional simple.

La capa de convolución recibe como entradas regiones de pixeles de una imagen. El proceso de recorrido se hace de izquierda a derecha y hacia abajo aplicando filtros en el proceso. La operación de convolución es el resultado de aplicar el filtro a la imagen. Como resultado la

imagen reduce información de los bordes considerando relevantes los pixeles del centro de la imagen. La operación de convolución realiza una multiplicación entre la entrada y el filtro elemento a elemento y suma los resultados parciales. El proceso continúa siguiendo el patrón descrito anteriormente hasta completar la matriz.

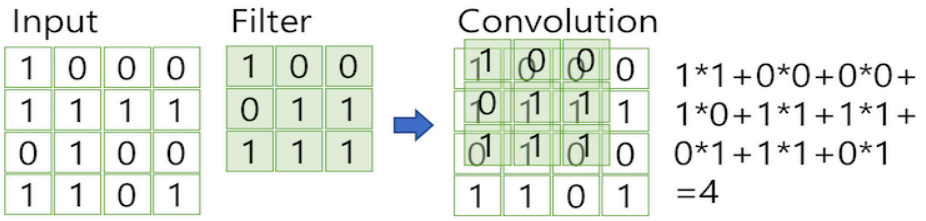

Figura 3. Convolución dado un input y un filtro 3x3.

La capa de agrupación máxima o Max Polling se encarga de detectar características múltiples dado un conjunto o grupo de pixeles aplicando un filtro. Dada una matriz de pixeles selecciona los pixeles máximos dentro de la imagen; mantiene la relación de espacio de la matriz de imagen original.

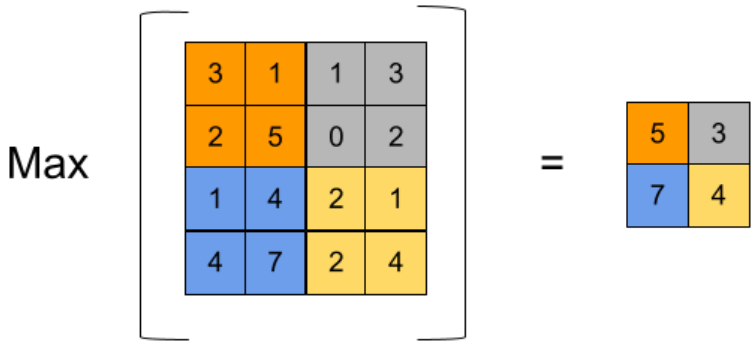

Figura 4. Max polling dado un input y un filtro de 2 x 2.

La capa de aplanamiento o Flatten se encarga de convertir una matriz de dimensiones n en un vector plano de dimensión 1.

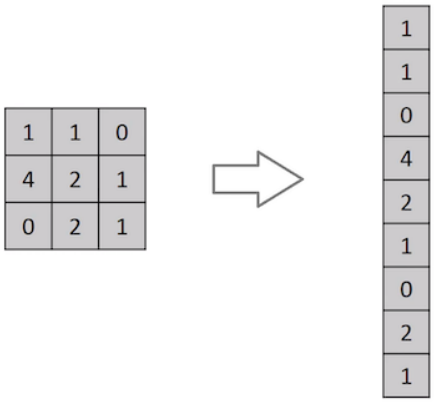

Figura 5. Flatten layer dado una matriz input.

La capa de abandono o Dropout desconecta ciertas neuronas al azar para evitar problemas relacionados al sobre-entrenamiento.

La capa densa o Dense se encarga de conectar cada neurona de la capa precedente hacia cada neurona de la capa en operación.

Adicional a las capas tenemos las **funciones de activación** que se encargan de disponer los outputs o salidas de una capa hacia la siguiente. Estas funciones se aplican según el contexto del problema. Entre las más usadas están:

- *Sigmoide*: usualmente para problemas de clasificación binarias. El output de esta función tiene valores entre 0 y 1.
- *Relu*: Activa solo los pesos de ciertas neuronas evitando problemas de gradiente de fuga. Anula valores negativos de entrada y mantiene los positivos.
- *Softmax*: usualmente para problemas de clasificación multiclase. Permite predecir más de 2 clases.

Transferencia de Aprendizaje

La tecnología de aprendizaje de maquina moderna a alcanzado grandes logros, pero todavía presenta ciertas limitaciones para escenarios reales del mundo. El escenario prometedor e ideal para el aprendizaje de maquina se da cuanto la cantidad de instancias o datos a entrenar se encuentran adecuadamente etiquetados y además presentan una distribución similar a la de los datos de prueba (Zhuang, 2021). Pero, la realidad es que en muchos escenarios reales recolectar suficientes datos de entrenamiento es costoso y consume tiempo. Por ejemplo, es posible aprovechar el conocimiento de un modelo para reconocer automóviles y aplicar transferencia de aprendizaje para conseguir reconocer camiones.

El aprendizaje por transferencia aplica un esquema de transferencia de conocimientos entre diferentes dominios re-utilizando los pesos extraídos de un proceso de entrenamiento previo complejo y agregando capas superiores entrenadas con una tarea nueva pero relacionada. El prerrequisito para aplicar esta metodología es que debe haber una conexión entre dos actividades de aprendizaje (Tan et al, 2018). Aunque la mayoría de los algoritmos de aprendizaje automático están diseñados para abordar tareas individuales, el desarrollo de algoritmos que faciliten el aprendizaje por transferencia es un tema de interés permanente en la comunidad del aprendizaje automático (Zhuang, 2021). La transferencia de aprendizaje permite la utilización de conjuntos de datos de menor tamaño. Sus ventajas se muestran con la disminución de tiempo de aprendizaje y recursos de cómputo.

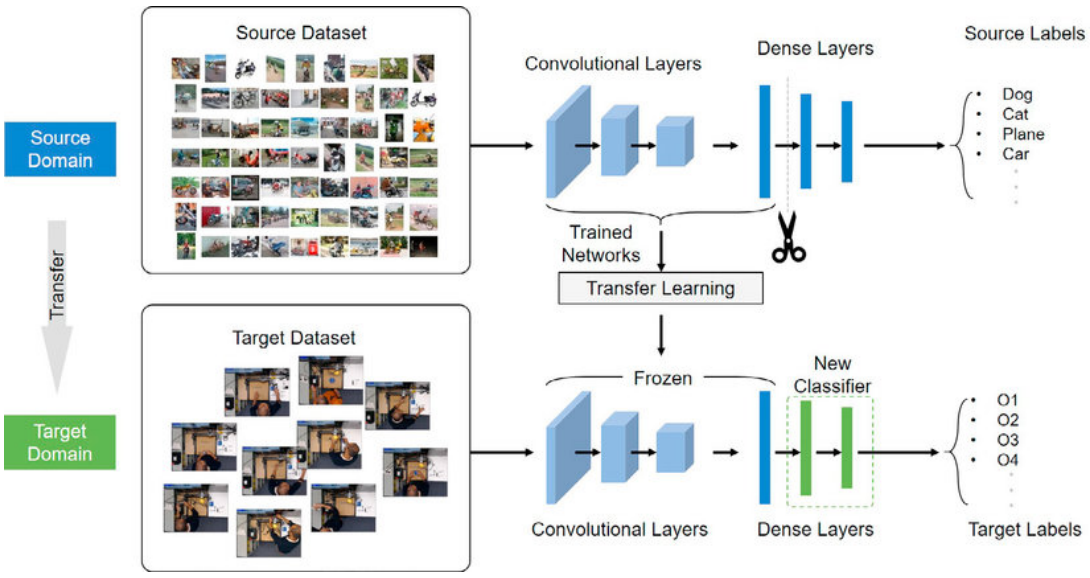

Figura 6. Flujo metodología de Transferencia de Aprendizaje.

Talukdar, Gupta, et al (2018) propusieron el uso de imágenes abstractas y redes neuronales convolucionales pre-entrenadas para demostrar el mejor performance en tareas de detección de objetos. Se implementó dynamic stacking, pseudo random placement, distractor noise para arquitecturas Faster-RCNN, R-FCN y SSD. En el campo del sector industrial la detección de fallos en equipos eléctricos es importante. Pérez, Risco & Casaverde (2021) propusieron aplicar transferencia de aprendizaje para la clasificación binaria de imágenes térmicas aplicando fine tuning para arquitecturas como: AlexNet, VGG16, VGG19, MobilNetV2, entre otros. Se contó con un dataset de 815 imágenes organizadas siguiendo un esquema estratificado y validando el rendimiento con la prueba de Friedman. Los resultados para las diferentes arquitecturas superaron el 85% contra las pruebas en modelos sin entrenamiento previo.

Yin, Yu, et al (2019) propusieron Feature Transfer Learning (FTL) para entrenar clasificadores de reconocimiento facial menos sesgados adaptando la distribución de características UR para tratar el desbalance de datos. Prakash, Thenmoezhi, et al (2019) propusieron un método

automatizado para reconocimiento facial usando los pesos el modelo pre-entrando VGG-16 sobre la base de datos de Imagenet para entrenar las imágenes de su base de datos. Las características extraídas sirven como input de una capa *Fully Connected* con función de activación *softmax* para la clasificación.

MATERIALES Y MÉTODOS

El proceso propuesto inicia recolectando los rostros de 10 personas participantes simulando personas requeridas por entes de control y seguridad. Con los videos recolectados se procede a procesar cada uno para obtener frames de imágenes. Para la detección de rostros y creación de las imágenes de rostros del dataset se usó MTCNN. Siguiendo las directrices de la **norma ISO 23053:2022** se aplicará el split de los folders aplicando Stratified Cross Validation con k = 10 y se evaluará el modelo con las métricas de Accuracy, Precision, Recall y AUC-ROC. Se construirán los plots Acurracy vs Epoch, Loss vs Epoch y una matriz de confusión. Se entrenará el modelo RNC escogido para aplicar la técnica de Transferencia de Aprendizaje con la base de datos generada y se aplicará Fine – Tuning siguiendo las especificaciones de la **norma ISO 23053:2022** para transferencia de aprendizaje.

El mejor modelo seleccionado será el que muestre el mejor rendimiento entre los 10 modelos resultantes del esquema Stratified Cross Validation y se procederá a la conversión a una versión compatible con el dispositivo móvil. Se diseñará el GUI del aplicativo móvil y se implementará el modelo entrenado en el sistema. En el dispositivo se configurará la detección de rostros usando librerías de Machine Learning para que las imágenes capturadas sean procesadas y enviadas como entrada para el modelo RNC. La figura 7 muestra un diagrama de bloques de la metodología aplicada.

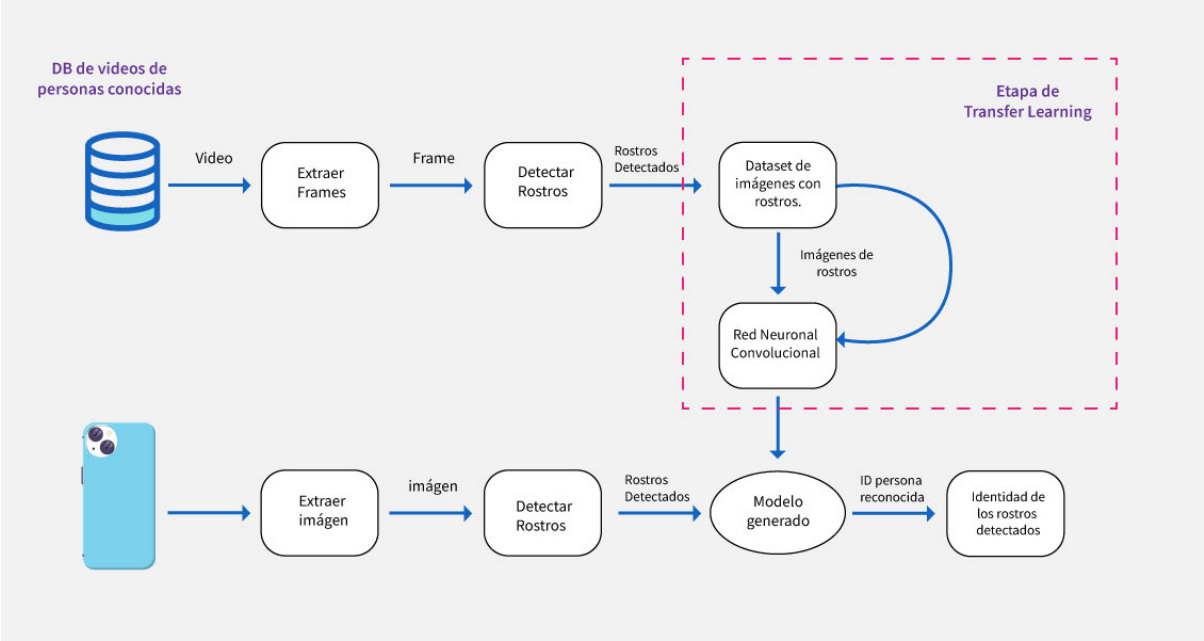

Figura 7. Metodología del modelado de la red e interacción con el dispositivo móvil.

Base de Datos

Para la construcción de la base de datos se contó con la participación de 10 personas mayores de edad. Cada una recibió una introducción sobre el tema del proyecto integrador y el uso que se va a dar a su información. El objetivo de esta etapa es almacenar los rostros de cada participante en archivos de imagen en formato png. Se realizo la captura de 10 videos de entre 15 y 20 segundos por cada persona. Para la grabación del video se utilizó un iPhone 12 Pro con la configuración de video en 1080p a 30fps y sin filtros. Los videos se guardaron en formato .MOV.

Fotogramas

Con los videos obtenidos por cada participante se procedió a procesar cada uno para extraer los frames que constituyen cada video. Por ejemplo: si consideramos la configuración de 30fps y un video de 15 segundos obtendremos un total de 450 frames para ese video. Esta lógica se

aplicó para los 10 videos de cada participante. Para conseguir esto se usó la librería para visión artificial *OpenCV* para Python 3. De esta forma se consiguió la primera etapa de construcción de la base de datos.

Detección de rostros

Por cada fotograma es necesario extraer únicamente los rostros. Para esto se consideró usar la red neural convolucional en cascada multitarea *MTCNN*. Esta red se utiliza para detección y posicionamiento de puntos clave de rostros. Las redes que componen *MTCNN* son Proposal Network (P-Net), Refine Network (R-Net) y Output Network (O-Net) (Young et al, 2020). Para el output del proceso se configuró una dimensión de imagen de 224 x 224 píxeles en formato png al tratarse de un formato de compresión sin pérdida. El pipeline de este marco se diagrama en la Figura 8. La extracción de rostros de cada fotograma por la red neuronal fue el proceso más extenso en términos de tiempo de procesamiento en esta etapa. Para esta sección se usó Google Colab en su versión Pro habilitando el entorno de ejecución para usar sus GPUs disponibles. El script diseñado para la extracción de fotogramas y detección de rostros se encuentra como ***Rostros.ipynb*** y está disponible en los anexos.

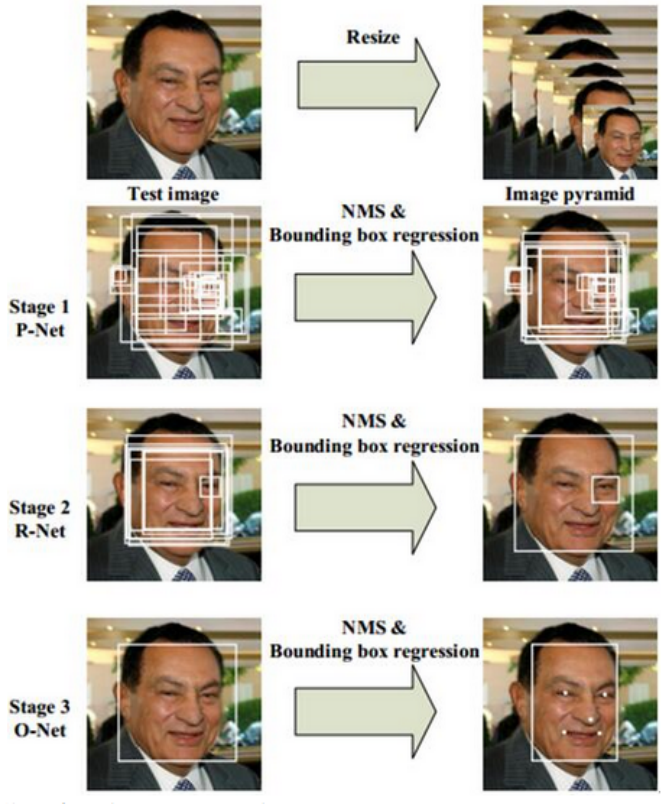

Figura 8. Flujo de procesamiento para detección de rostros con MTCNN

Con los resultados se creó el directorio donde se almacenan los datos finales para el proceso de entrenamiento, validación y prueba del modelo. El set de entrenamiento de cada persona está formado por 7 carpetas de rostros por persona. El set de validación está formado por 2 carpetas de rostros por persona. El set de prueba está formado por 1 carpeta de rostros por persona. El enlace al repositorio de la base de datos se incluye en los anexos del documento. El esquema de los directorios, la cantidad de imágenes y el tamaño en disco se muestra en la figura 9.

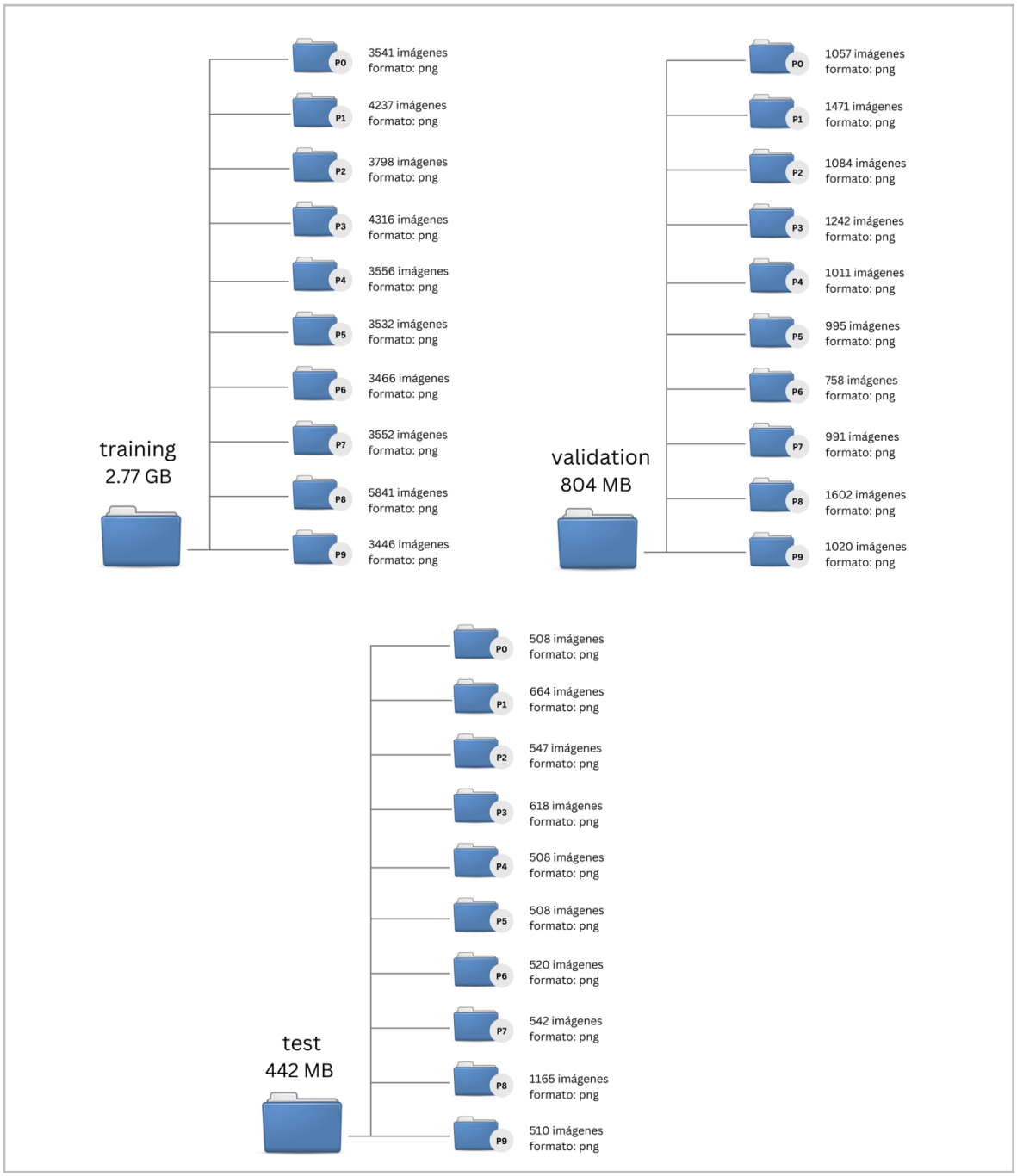

Figura 9. Estructura de los directorios training, validation y test.

Modelo Deep Learning

MobileNetV2

El modelo MobileNetV2 implementa una arquitectura de convolución profunda separable que permite reducir considerablemente la cantidad de cómputo. Su estructura consiste en un esquema residual invertido donde los enlaces de tipo residual se ubican entre los layers de bottleneck con Depthwise, Bath Normalization y funciones de activación de tipo ReLU. Presenta una capa de convolución completa en el inicio con 32 kernels o filtros seguida por 19 capas de bottleneck residuales. Cada línea describe una secuencia de 1 o más capas idénticas (paso de módulo), repetidas n veces. Todas las capas en la misma secuencia tienen el mismo número c de canales de salida. La primera capa de cada secuencia tiene un paso s y todos los demás usan el paso 1. Todas las circunvoluciones espaciales usan kernels de 3 × 3. El factor de expansión t siempre se aplica al tamaño de entrada. (Sandler et al, 2018). Una de sus características principales es el gran desempeño que presenta su módulo convolucional en dispositivos móviles reduciendo el consumo de memoria durante las inferencias.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figura 10. Arquitectura MobileNetV2

Configuración Experimental

Stratified Cross Validation

Para aplicar el esquema de partición con Stratified Cross Validation con k = 10 se combinaron los folders de training y validation en una única carpeta llamada data. Los archivos de cada carpeta de training y validation fueron renombrados de tal forma que el nombre del archivo contiene el nombre de la persona a la que corresponde y el número de imagen. Esto nos permitió crear un archivo csv llamado “images” donde podemos mapear el nombre del archivo con la clase a la que pertenece. Esto nos permite usar la librería de SckiLearn y hacer el split sobre los datos del fichero y por cada fold usar el flow_from_dataframe del ImageGenerator para poder cargar a memoria unicamente las imágenes seleccionados por el SCV para cada fold en cada iteración. La implementacion detallada del esquema estratificado se encuentra en el archivo face_recognition_cv.py en el repositorio de GitHub.

	filename	label
0	p0_0.png	p0
1	p0_1.png	p0
2	p0_2.png	p0
3	p0_3.png	p0
4	p0_4.png	p0
5	p0_5.png	p0
6	p0_6.png	p0
7	p0_7.png	p0

Figura 11. Estructura del fichero de mapeo nombre de archivo – clase.

Preprocesamiento de datos

Como parte de la preparación del dataset para el entrenamiento se tomó en cuenta el input que acepta la arquitectura MobileNetV2. Se requieren imágenes con dimensión 224 x 224 pixeles

y un valor de 3 para el canal RGB (224 x 224 x 3). Como resultado de la extracción de rostros en el proceso de creación de la base de todos se configuró el código para que las detecciones realizadas por MTCNN sean extraídas en el formato requerido por la red neuronal usando la librería CV2. Por cada una de las imágenes en el proceso de configuración de los conjuntos de entrenamiento, validación y prueba se escalaron las imágenes en un factor de 1/255. Esto considerando que cada imagen está formada por pixeles en rango de 0-255, con esto conseguimos escalar los pixeles en un rango de 0 -1 lo que permite que el modelo trate a todas las imágenes de igual forma. La tasa de aprendizaje también se puede ver afectada por escalas de pixeles demasiado altas para ciertas imágenes y demasiados bajos para otras, esto afecta directamente en el cálculo del loss.

Entrenamiento

Sobre esta arquitectura se implementó la estrategia de Transferencia de Aprendizaje descartando las capas superiores con el fin de entrenar el modelo para detectar las clases que nosotros necesitamos diseñando nuestras propias capas. Respecto a los pesos del modelo se trabajó sobre los resultados de Imagenet. MobileNetV2 trabaja en su capa de entrada con imágenes de 224 x 244 pixeles y con canal de color 3 (RGB). Bajo experimentación con el fin de mejorar el accuracy de nuestro modelo se descongelaron 8 capas inferiores.

```
# MobileNetV2, not including top layers to enable Transfer Learning

MobileNet = MobileNetV2(weights='imagenet',
                        include_top=False,
                        input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3))

# We freeze layers, except 8 last layers.
count = 0
for layer in MobileNet.layers:
    if count < 146:
        layer.trainable = False
    count += 1
```

Figura 12. Configuración para Transferencia de Aprendizaje sobre MobileNetV2

Para las capas superiores se experimentó con varias configuraciones considerando que usamos el modelo pre-entrenado. Las capas descongeladas son Conv2D con funciones de activación tipo ReLU, y capas de normalización tipo Batch Normalization. Sobre estas capas se agregó nuestra configuración. Una capa GlobalAveragePooling2D que consiste en aplicar pooling, pero en un contexto espacial. La siguiente capa es tipo Dense con 256 neuronas y función de activación tipo ReLU con el fin de ir reduciendo dimensión para en el output incluir la última capa de tipo Dense con 10 neuronas dado que buscamos clasificar 10 clases con una función de activación tipo softmax.

```
# our top custom layers.
def lw(bottom_model, num_classes):
    top_model = bottom_model.output
    top_model = GlobalAveragePooling2D()(top_model)
    top_model = Dense(256,activation='relu')(top_model)
    top_model = Dense(NUM_CLASSES,activation='softmax')(top_model)
    return top_model
```

Figura 13. Capas superiores para entrenamiento sobre 10 clases.

Para el entrenamiento se configuraron callbacks con el fin de tener control sobre este proceso y evitar que el modelo continue entrenando sin mejorar en sus métricas. ***ModelCheckPoint*** para poder monitorear los mejores valores para la métrica escogida al final de cada epoch, para nuestro caso se observó validation y training accuracy. Para controlar la tasa de aprendizaje se usó ***ReduceLROnPlateu*** esta función permite especificar el número de épocas sobre las cuales reducir o no la tasa de aprendizaje, para nuestro caso se especificó en 2. El factor de reducción es de 0.5. ***EarlyStopping*** es una optimización que nos permite prevenir el overtraining sin

afectar el accuracy del modelo de esta forma dejamos que el entrenamiento se detenga cuando se considere oportuno, la métrica a monitorear es accuracy y la paciencia se configuró en 5.

Como optimizador se experimentó con Adam. Como función de costo o ***loss*** se especificó ***categorical crossentropy*** al tratarse de un problema de clasificación multiclase. Para las épocas se probó con un rango de 100, un ***learning rate*** de 0.0001 y un ***batch size*** de 64; considerando los callbacks el entrenamiento con los mejores resultados sin sobre-entrenamiento termina antes de las 100 épocas.

```
# compile and fit model with our configuration, we use lr = 10-4, Adam opt

model.compile(
    optimizer=optimizers.Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    train_generator,
    epochs = EPOCHS,
    callbacks = callbacks,
    validation_data=val_generator
)
```

Figura 14. Optimizador y función de costo.

Conversión del modelo

Finalmente se exporto el modelo en dos formatos: .h5 y .tflite. El modelo lite servirá para la implementación del clasificador en un dispositivo móvil. Para la conversión lite se utilizó la librería TFLiteConverter. Para obtener un modelo eficiente en esta etapa es importante escoger una arquitectura apropiada. El tamaño en memoria del modelo .h5 es de 19.4MB y el modelo. tflite es de 10.2MB.

```
# we can export the model

saved_model_dir = 'mobile'
tf.saved_model.save(model, saved_model_dir)

# we converte .h5 model to .tflite to mobile on device implementation.
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()

with open('mobile/model.tflite', 'wb') as f:
  f.write(tflite_model)
```

Figura 15. Configuración de TFLiteConverter para conversión del modelo.

La carpeta con los scripts del proceso de entramiento, validación y prueba con nombre ***face_recognition.ipynb*** y ***face_recognition_cv.py*** y los archivos exportados ***face_recognized.h5*** y ***modelo.tflite*** se encuentran adjuntos en el repositorio de GitHub en la sección de anexos.

Implementación del modelo en dispositivo móvil

Con el modelo. tflite convertido en la parte de modelado de la Red Neuronal Convolucional se busca construir un prototipo de aplicación móvil que pueda realizar inferencias basadas en fotogramas que no han sido parte del proceso de entrenamiento ni validación y poner a prueba el modelo. El framework escogido para este desarrollo fue Google Flutter. La aplicación móvil nos permite seleccionar imágenes de la galería o usar la cámara del dispositivo para en primera instancia detectar rostros y a través de un proceso de preprocesamiento de imagen enviar un input al modelo TensorFlow Lite con la configuración necesaria.

Flutter SDK

Este SDK nos permite crear aplicaciones de alto rendimiento y fidelidad para plataformas móviles como iOS, Android e incluso web a partir de un único código base. Su lanzamiento fue en mayo de 2017 con la primera versión estable 1.0 publicada en diciembre 2018. Fluttter es un framework declarativo lo que implica que se construye el IU reflejando el estado actual de la aplicación. El diseño de Flutter está basado en capas que independientemente contienen librerías una sobre otra. El esquema que usa para la renderización se base en Widgets que pueden ser mutables o inmutables. El lenguaje de programación que se usa es Dart.

- *Versión de desarrollo:*
    - Flutter 3.0.5
    - Dart 2.17.6

Google's ML Kit for Flutter

Es un paquete de Aprendizaje Automático manejado por Google que permite a los desarrolladores aprovechar tecnologías para crear aplicaciones para iOS y Android y consumir APIs relacionadas a Visión por Computadora y Natural Processing Language. Para nuestro

caso usamos el API para detección de rostros que nos permite identificar rostros, rasgos y contornos faciales en imágenes.

- *Versión de desarrollo:*
    - Google ML Kit Face Detection Module 0.4.0

```
final FaceDetector faceDetector = FaceDetector(
  options: FaceDetectorOptions(performanceMode: FaceDetectorMode.accurate),
); // FaceDetector
```

Figura 16. Inicialización del objeto Face Detector de Google ML Kit.

TensorFlow Lite for Flutter

Un complemento de Flutter para acceder a la API de TensorFlow Lite. Admite clasificación de imágenes, detección de objetos (SSD y YOLO), Pix2Pix y Deeplab y PoseNet en iOS y Android. Con esta herramienta podemos cargar el modelo creado con Python y TensorFlow en el dispositivo móvil y realizar inferencias con ayuda del detector de rostros de Google ML Kit.

- Versión de desarrollo:
    - Tflite 1.1.1

```
// Load model.ftlite
Future loadModel() async {
  try {
    String res;
    res = (await Tflite.loadModel(
      model: "assets/model.tflite",
      labels: "assets/labels.txt",
      // useGpuDelegate: true,
    ))!;
  } on PlatformException {
    print('Failed to load model.');
  }
}
```

Figura 17. Carga del modelo TensorFlow Lite a memoria.

Componentes adicionales

Componente	Descripción	Versión
Image	Preprocesamiento de imágenes.	3.2.2
Gallery Saver	Guarda imágenes de formal local.	2.3.2
Image Picker	Selección de imagen: cámara, galería.	0.8.6
Camera	Acceso a hardware (cámara) nativa.	0.10

Tabla 1. Componentes adicionales para el manejo de imágenes en dispositivo móvil.

Casos de uso

El diagrama de usos nos permite mostrar la forma en la que el sistema móvil interactuará con los agentes. Como elementos del diagrama tenemos a una entidad externas principal: persona que requiere escanear el rostro o administrador. El administrador del aplicativo es capaz de tomar una fotografía del individuo directamente desde el sistema o utilizar una imagen previamente capturada desde la galería del dispositivo y realizar el proceso de clasificación.

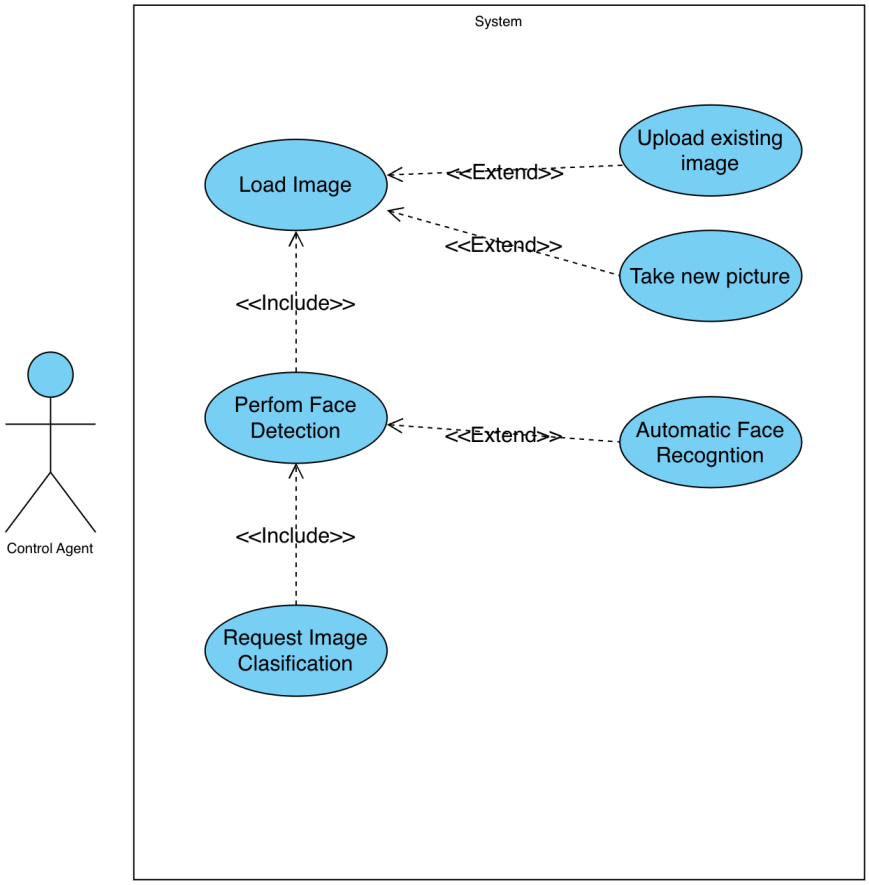

Figura 18. Diagrama de usos sistema móvil de reconocimiento facial.

Diagrama de actividades

Con el diagrama de actividades buscamos mostrar el flujo de la aplicación. Nos permite especificar el comportamiento y dinámica. El prototipo únicamente cuanta con una clase de entrada que se muestra en pantalla para seleccionar el origen de la imagen. El agente de control puede seleccionar entre cámara o galería para obtener la fotografía e iniciar el proceso de detección y clasificación. La detección de rostros se realiza con *Google Face Detector* del paquete ML y la clasificación se usa cargando el modelo entrenado usando el componente de *TensorFlowLite para Flutter*.

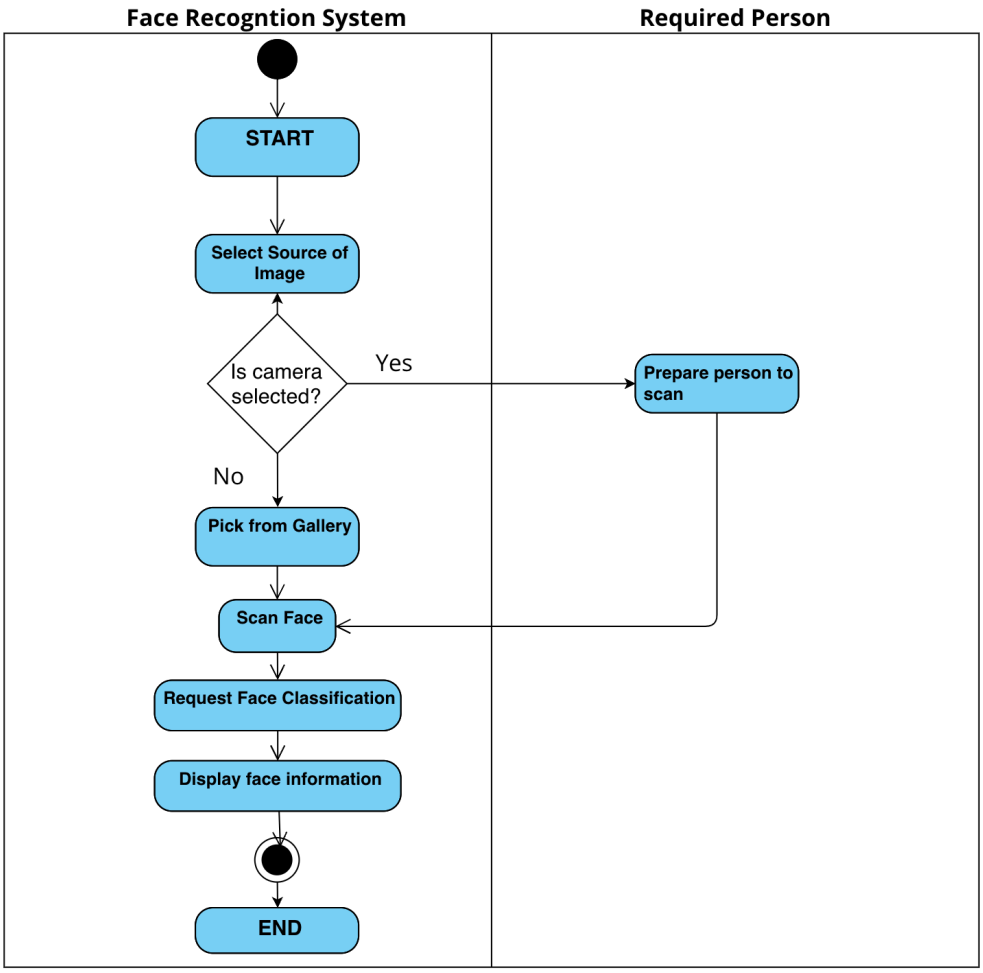

Figura 19. Diagrama de actividades para sistema de reconocimiento facial.

RESULTADOS

El proyecto integrador se dividió en dos etapas principales. La primera consistió en el entrenamiento de un modelo de Red Neuronal Convolucional aplicando Transferencia de Aprendizaje siguiendo un esquema estratificado. Para la parte final, con el mejor modelo obtenido se procedió con el desarrollo de un prototipo de aplicativo móvil para implementar el modelo y realizar inferencias. El entrenamiento finalmente se dio sobre una MacBook Pro M1 Pro -16 GB de RAM, con CPU de 10 núcleos y GPU integrado de 16 núcleos.

- ***Modelo***

El mejor modelo resultado del entrenamiento y validación con Stratified Cross Validation se determinó en el fold 1. El valor de accuracy sobre los datos de validación para este modelo fue de 0.9996.

Fold	Accuracy	Precision	Recall	AUC
1	**0.9996**	**0.9990**	**0.9983**	**0.9998**
2	0.9994	0.9959	0.9972	0.9997
3	0.9970	0.9981	0.9968	0.9986
4	0.9983	0.9991	0.9987	0.9968
5	0.9977	0.9983	0.9993	0.9995
6	0.9978	0.9984	0.9991	0.9993
7	0.9983	0.9991	0.9988	0.9993
8	0.9991	0.9992	0.9979	0.9982
9	0.9995	0.9973	0.9981	0.9997
10	0.9990	0.9990	0.9992	0.9998

Figura 20. Accuracy, Precision, Recall, AUC entre 10 folds.

A continuación, se muestran los resultados obtenidos por el clasificador. En el caso del training y validation podemos ver en la ***figura 21*** como las predicciones correctas superan el umbral de 0.98 rápidamente a partir de la época 1 y se mantiene en 0.99. En general se puede evidenciar que el accuracy de los datos de validación no bajan de 0.99 lo cual para nuestro contexto es bueno.

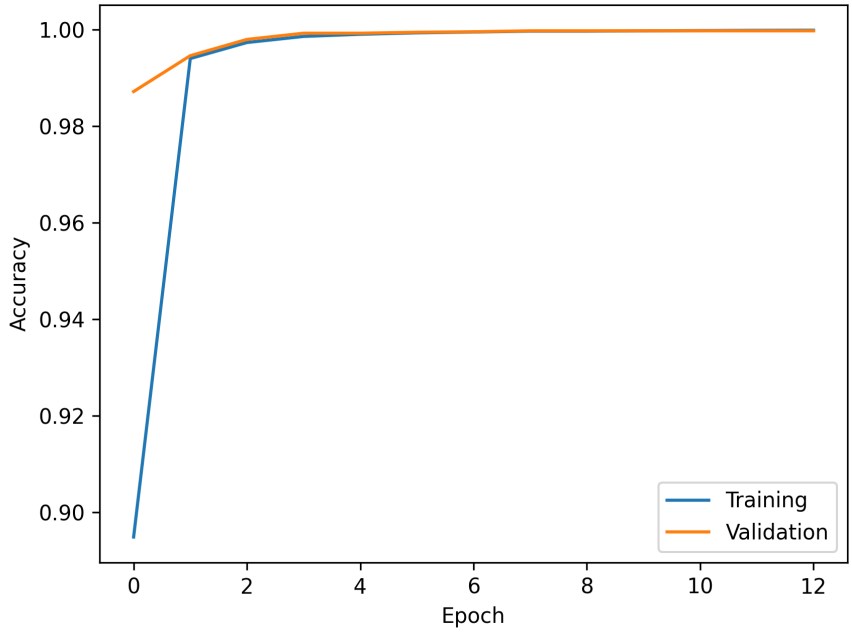

Figura 21. Accuracy vs Epoch para el modelo entrenado.

El valor de ***loss*** representa los errores en nuestro modelo. El plot de Loss vs Epoch nos permite determinar si estamos haciendo overfitting. En nuestro caso podemos observar que la curva tiende a converger rápidamente alrededor de la época 2 y 4 antes de estabilizarse. Para nuestro caso se utilizó como optimizador Adam con función de loss: Categorical Cross Entrophy. El proceso de optimización busca minimizar el valor de loss con el entrenamiento. Se puede evidenciar que la distancia entre la curva de training y validation es mínima hasta la época 12.

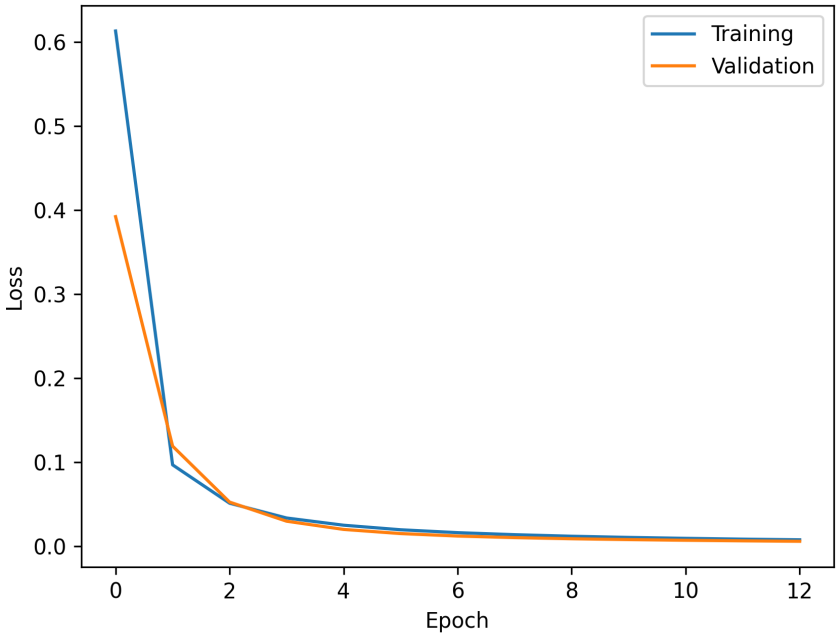

Figura 22. Loss vs Epoch para el modelo entrenado.

Se evaluó el modelo con el set de prueba con 6090 imágenes para 10 clases diferentes y se obtuvieron los siguientes resultados para distintas métricas. Para nuestro esquema consideramos como relevante la métrica de accuracy. Evaluado el conjunto de prueba que no fue parte del entrenamiento ni validación se determinó un valor de **0.999** para la métrica accuracy general que se ajusta al umbral previamente establecido.

```
Metrics Model  1
Accuracy   : 0.9995073891625615
```

Figura 23. Resultados de accuracy para set de prueba.

La matriz de confusión nos muestra el comportamiento de las predicciones entre los diferentes participantes (clases). Podemos detectar si el clasificador tuvo errores en su inferencia con relación a otras clases. De la ***figura 24*** podemos observar para el conjunto de prueba la clase

que tuvo desaciertos más relevantes fue Samy. De 1165 imágenes de prueba, 15 fueron clasificados como Maru y 4 fueron clasificados como Fausto. Para la clase Jenifer con 618 imágenes hubo 1 desacierto siendo clasificada como Ale. Finalmente, la clase Fausto con 660 imágenes de prueba tuvo 1 clasificación errónea.

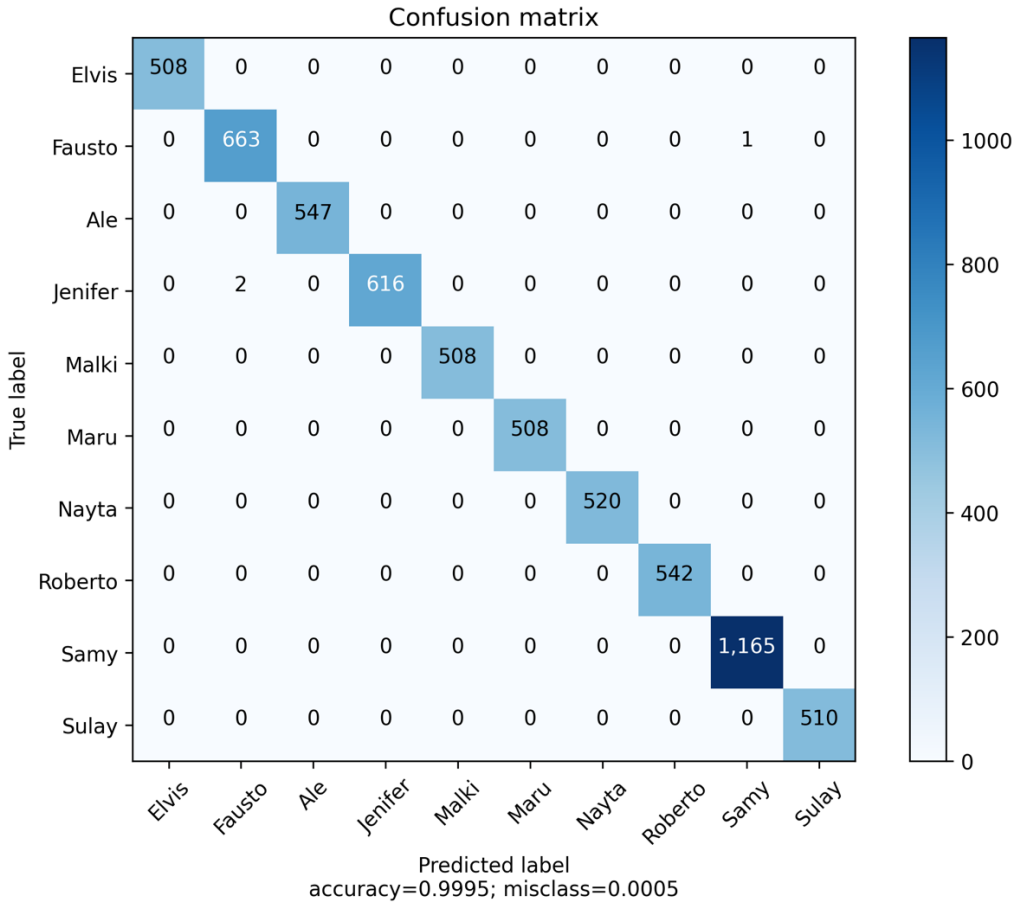

Figura 24. Matriz de confusión de predicciones en el conjunto de pruebas.

Como segunda parte se inició con el desarrollo de la aplicación móvil para usar el modelo exportado anteriormente. Para nuestro caso se usó el ambiente configurado para el sistema operativo iOS de Apple. El prototipo fue evaluado en un iPhone 12 Pro con iOS 16.1.2

- ***Prototipo***

Pantalla de inicio

En la pantalla de inicio se muestran las opciones que el usuario tiene para seleccionar la imagen de donde se va a detectar y reconocer el rostro. Las opciones se despliegan en dos Floating Action Buttons. El primero botón enciende la cámara y el segundo permite abrir la galería del dispositivo.

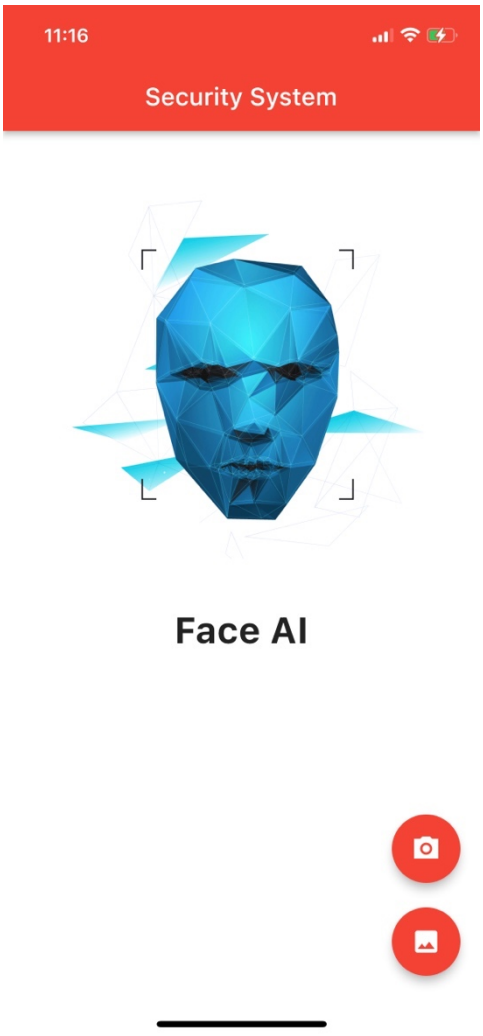

Figura 25. Pantalla de inicio de Face AI.

El sistema móvil se encarga de procesar la imagen recibida por una de las dos fuentes y se realiza un preprocesamiento. Se elimina correctamente la metadata innecesaria para el proceso de clasificación y se normaliza la imagen. Las capturas de las ***figuras 26 y 27*** muestran correctamente el proceso de selección.

Figura 26. Captura de imagen desde cámara del dispositivo.

Figura 27. Captura de imagen desde galería de fotos.

Una vez que la imagen es seleccionada el motor de detección de rostros de Google ML se encarga de detectar la ubicación del rostro dentro de un cuadro. Las coordenadas generadas por este complemento nos permiten correctamente recortar la imagen y redimensionarlas para el input requerido por el modelo (224 x 224 x 3). El modelo recibe como input la imagen del rostro y se encarga de realizar una inferencia de la clase a la que pertenece. En la siguiente pantalla se muestra el nivel de confidencia de la predicción, el nombre de la clase, identificación e información relacionada a la clase encontrada.

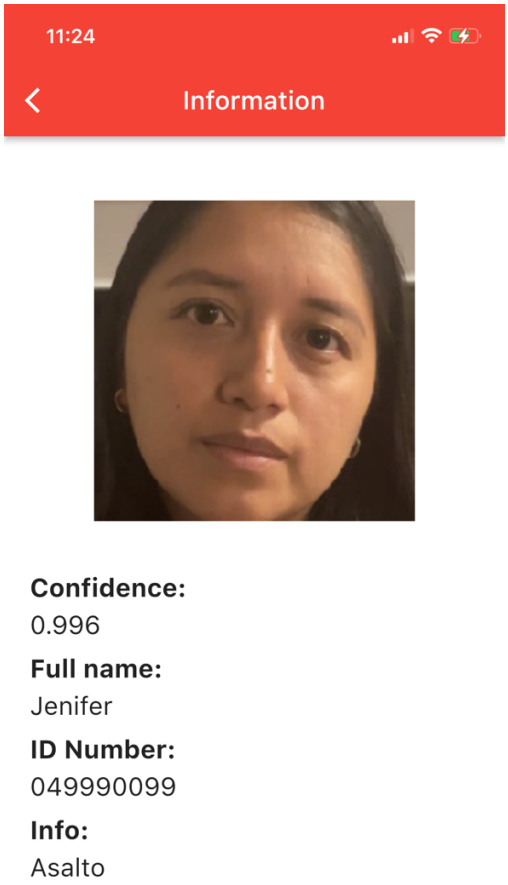

Figura 28. Resultados de la predicción realizada por el modelo entrenado.

CONCLUSIONES

La implementación del modelo pre-entrando, aplicando Redes Neuronales Convolucionales y Transferencia de Aprendizaje se realizó con éxito. El modelado de la arquitectura requirió de una serie de modificaciones y pruebas para alcanzar el umbral de exactitud de 98% alcanzando un porcentaje de 99.9% para el set de prueba. Estos resultados tienen relación directa con el uso de un esquema estratificado para el entrenamiento y validación. Las ventajas de usar un modelo pre-entrenado en nuestro caso con *Imagenet* redujo el tiempo de cómputo; el número de épocas requeridas fue de 12. La creación del dataset de personas requeridas y el primer preprocesamiento para extracción de rostros se realizó con MTCNN y los resultados fueron los esperados. Para todos los videos se logró extraer los dataframes y por cada dataframe finalmente se almacenó únicamente el rostro encontrado en la imagen. Para aumentar el accuracy se decidió descongelar 8 capas adicionales del modelo MobileNetV2 para mejorar la generalización de las predicciones lo que resultó en una mejora evidente. Es importante mencionar que, aunque en primera instancia se propuso trabajar con Google Colab, las prestaciones y limitaciones como cantidad de RAM y unidades computacionales disponibles no fueron suficientes para este proyecto. El performance del nuevo chip M1 de Apple demostró que tiene ventajas en tiempo de entrenamiento sobre los servicios que ofrece Google en su versión gratuita para nuestro caso.

La implementación del prototipo móvil se desarrolló con el framework Flutter que es propiedad de Google. Una de las principales ventajas de trabajar con este esquema fue el tiempo de desarrollo. Con un único código base se pudo obtener un aplicativo funcional y con rendimiento nativo para iOS en corto tiempo. Sin embargo, si se desea obtener el instalador para Android solo se requiere una configuración adicional en términos de permisos para acceder al hardware del dispositivo. Los complementos principales para el manejo de

imágenes, detección de rostros y clasificación fueron Google ML Kit con el módulo de detección de rostros y TensorFlow Lite para poder cargar el modelo entrenado previamente. Fue importante manejar correctamente el módulo de imagen de Flutter para realizar tareas de preprocesamiento como eliminación de metadata innecesaria y normalización de pixeles. El detector de rostros de Google facilitó la ubicación de las coordenadas de ubicación del rostro en la imagen para proceder con el recorte. Se logró demostrar que el módulo para Flutter de TensorFlow es funcional para problemas de detección de rostros y que, a pesar de requerir de un proceso de conversión previo, la calidad del modelo no se ve afectada. Cabe recalcar que iOS tiene frameworks nativos para proyectos de entrenamiento como es el caso de CoreML, pero todavía tienen limitaciones para el diseño de arquitecturas personalizadas. Para nuestro caso al tratarse de un desarrollo multiplataforma se trabajó con módulos compatibles para iOS y Android como es el caso de TensorFlow Lite y los resultados fueros los esperados. El prototipo es capaz de cargar imágenes, detectar rostros, y solicitar su clasificación.

De esta forma se alcanzaron los objetivos propuestos para este proyecto integrador demostrando que todo el contenido académico recibido a lo largo de la carrera de Ciencias de la Computación fue aplicable para solucionar un problema real. Como trabajo futuro se propone poner a prueba otras arquitecturas como VGG-16 e Inception V3 sobre un dataset similar y verificar su funcionamiento en un ambiente móvil nativo.

ANEXOS

- **Repositorio de GitHub con los archivos de desarrollo:**
    - https://github.com/SamyConejo/Face_AI

REFERENCIAS BIBLIOGRÁFICAS

Albawi, S., Mohammed A., & Al-Zawi, S. (2017). *"Understanding of a convolutional neural network".* 2017 International Conference on Engineering and Technology (ICET). pp. 1-6. Doi: 10.1109/ICEngTechnol.2017.8308186.

Batta, M. (2020). Machine Learning Algorithms: A Review. *International Journal of Science and Research (IJSR)*, 9(1), 1174–1179. Doi 10.21275/ ART20203995

Bravo, D. (23 de julio de 2020). *Cámaras de reconocimiento facial identifican a 45 personas que han incurrido en delitos en Quito*. El Comercio. https://www.elcomercio.com/actualidad/seguridad/camaras-reconocimiento-facial-delitos-quito.html

Chowanda, A., Sutoyo, R. (2019). "Convolutional neural network for face recognition in mobile phones". *ICIC Express Letters*, vol 13, pp 569-574. Doi 10.24507/icicel.13.07.569.

Coşkun, M., Uçar, A., Yildirim, Ö., & Demir, Y. (2017). "Face recognition based on convolutional neural network". *2017 International Conference on Modern Electrical and Energy Systems (MEES).* pp 376-379. Doi: 10.1109/MEES.2017.8248937.

Davis, M., Ellis, T. (August 1964). "The RAND Tablet: A Man-Machine Graphical Communication Device - Memorandum rm-4122-arpa*". RAND Corporation*. Doi: 10.1145/1464052.1464080.

Dean, J., Corrado, G., Monga, R., et al. (2012). "Large Scale Distributed Deep Networks". *Advances in neural information processing system,* vol 1, pp 1223-1231. Doi: 0.5555/2999134.2999271

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). *Fundamental Concepts of Convolutional Neural Network*. Doi: 10.1007/978-3-030-32644-9_36.

Gulshan, V., Peng, L., Coram, M., et al. (2016). "Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs". *JAMA*, vol 316(22), pp 2402–2410. Doi:10.1001/jama.2016.17216

Harrington, P. (2012). "Machine learning in action". Shelter Island, N.Y.: *Manning Publications Co.*

Haik, Y. and M., S.T.M. (2011) *Engineering design process*. Stamford, CT: Cengage Learning.

Hu, J., Peng, L., & Zheng, L. (2015). "XFace: A Face Recognition System for Android Mobile Phones". *2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications*. Doi:10.1109/cpsna.2015.12

International Organization for Standardization. (2022). Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML) (ISO 23053:2022). https://www.iso.org/standard/74438.html

Kagaya, H., Aizawa, K., and Ogawa, M. (2014). "Food Detection and Recognition Using Convolutional Neural Network". *Proceedings of the ACM International Conference on Multimedia - MM '14*. Doi:10.1145/2647868.2654970

Kocacinar, B., Tas, B., Patlar, A., Fatma, C., & Cagatay, M. (2022). "A Real-Time CNN-based Lightweight Mobile Masked Face Recognition System", *IEEE Access,* 2022 [Internet]. Doi 10.1109/ACCESS.2022.3182055.

Liu, Q., Wu, Y. (2012). Supervised Learning. In: Seel, N.M. (eds) *Encyclopedia of the Sciences of Learning.* Springer, Boston, MA. Doi 10.1007/978-1-4419-1428-6_451

Mella, C. (28 de septiembre de 2021). *Alcaldía de Guayaquil compró cámaras para reconocimiento facial que no usa.* Primicias. https://www.primicias.ec/noticias/sociedad/municipio-guayaquil-compro-camaras-reconocimiento-facial-ecuador/

MET. (s.f). Facial Recognition. https://www.met.police.uk/advice/advice-and-information/fr/facial-recognition

Notimundo. (07 de junio de 2016). *Audio y video: herramientas de apoyo en Criminalística.* https://notimundo.com.ec/audio-video-herramientas-apoyo-criminalistica/

Pérez-Aguilar, D., Risco-Ramos, R., & Casaverde-Pacherrez, L. (2021). *Transfer learning en la clasificación binaria de imágenes térmicas*. Ingenius. Revista de Ciencia y Tecnología, (26),71-85. Doi: https://doi.org/10.17163/ings.n26.2021.07

Policía Ecuador [@PoliciaEcuador]. (21 de enero de 2021). *TECNOLOGÍA PARA SU SEGURIDAD | Durante la reunión mantenida con funcionarios de @ForensesEC y el @ECU911_ se incorporó el Sistema de Identificación de Voz e Imagen Facial AVIS+F que facilitará el reconocimiento facial mediante video* [Tweet]. Twitter. https://twitter.com/PoliciaEcuador/status/1219681185103826946?s=20&t=jVQGOdvQmN8n38KoVFcLQw

Saad, A. et al. (2017). "Understanding of a convolutional neural network." *2017 International Conference on Engineering and Technology (ICET).* pp 1-6. Doi:10.1109/ICENGTECHNOL.2017.8308186

Sandler, Mark & Howard, Andrew & Zhu, Menglong & Zhmoginov, Andrey & Chen, Liang-Chieh. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 4510-4520. Doi: 10.1109/CVPR.2018.00474.

Talukdar, Jonti & Gupta, S. & Rajpura, P. & Hegde, Ravi. (2018). *Transfer Learning for Object Detection using State-of-the-Art Deep Neural Networks*. 78-83. Doi: 10.1109/SPIN.2018.8474198.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C. (2018). *A Survey on Deep Transfer Learning*. Artificial Neural Networks and Machine Learning – ICANN 2018. ICANN 2018. vol 11141. pp 270-279. Doi: 10.1007/978-3-030-01424-7_27

ONU. (15 de septiembre de 2021). *Los riesgos de la inteligencia artificial para la privacidad exigen medidas urgentes –Bachelet*. https://www.ohchr.org/es/press-releases/2021/09/artificial-intelligence-risks-privacy-demand-urgent-action-bachelet

Yin, Xi & Yu, Xiang & Sohn, Kihyuk & Liu, Xiaoming & Chandraker, Manmohan. (2019). *Feature Transfer Learning for Face Recognition with Under-Represented Data*. 5697-5706. Doi: 10.1109/CVPR.2019.00585.

Yuan, M., Nikouei, Y., Fitwi, A., Chen, Y., & Dong, Y. (2020). Minor Privacy Protection Through Real-time Video Processing at the Edge.

Zheng, N., Loizou, G., Jiang, X., Lan, X. & Li, X. (2017). "Computer vision and pattern recognition", *International Journal of Computer Mathematics*. vol 84:9. pp 1265-1266. Doi: 10.1080/00207160701303912

Zhuang, F., et al. (2021). "A Comprehensive Survey on Transfer Learning". *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43-76. Doi: 10.1109/JPROC.2020.3004555.