UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Software

"Aplicación móvil de reconocimiento facial para el control de asistencia a clases de los estudiantes de la Universidad Técnica del Norte utilizando técnicas de Inteligencia Artificial"

Trabajo de Grado Previo a la Obtención del Título de Ingeniero en Software de la Universidad Técnica del Norte

Autor:

Guaichico Piñan Edison Geovanny

Director:

Ing. Marco Remigio Pusdá Chulde, PhD

Ibarra - Ecuador

2024

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
Cédula de identidad	100466013-8
Apellidos y nombres	Guaichico Piñan Edison Geovanny
Dirección	Imantag
E-mail	egguaichicop@utn.edu.ec
Teléfono móvil	0985802375
DATOS DE LA OBRA	
Título	APLICACIÓN MÓVIL DE RECONOCIMIENTO FACIAL PARA EL CONTROL DE ASISTENCIA A CLASES DE LOS ESTUDIANTES DE LA UNIVERSIDAD TÉCNICA DEL NORTE UTILIZANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL.
Autor	Guaichico Piñan Edison Geovanny
Fecha	16/02/2024
Programa	Pregrado
Titulo	Ingeniero en Software
Director	Ing. Marco Remigio Pusdá Chulde, PhD

CONSTANCIA

CONSTANCIA

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

.

Ibarra, a los 16 días del mes de febrero de 2024

EL AUTOR:

ESTUDIANTE

Edison Geovanny Guaichico Piñan

C.I: 1004660138

CERTIFICACIÓN

CERTIFICADO DEL DIRECTOR DE TRABAJO DE GRADO

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DEL DIRECTOR

Por medio de la presente yo PhD. Marco Remigio Pusdá Chulde, certifico que el **Sr. Edison Geovanny Guaichico Piñan**, portador de la cédula de ciudadanía Nro. **1004660138**. Ha trabajado en el desarrollo del proyecto titulación "APLICACIÓN MÓVIL DE RECONOCIMIENTO FACIAL PARA EL CONTROL DE ASISTENCIA A CLASES DE LOS ESTUDIANTES DE LA UNIVERSIDAD TÉCNICA DEL NORTE UTILIZANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL", previo a la obtención del título de ingeniería en Software, lo cual ha realizado en su totalidad con responsabilidad y esmero.

Es todo cuanto puedo certificar en honor a la verdad.

En la ciudad de Ibarra, a los 16 días del mes de febrero del 2024.

Atentamente

Ing. Marco Remigio Pusda Chulde, PhD

C.I: 0401200951

TUTOR DE TRABAJO DE GRADO

DEDICATORIA

Dedico este trabajo:

A la memoria de mi hermano Alex Guaichico quien partió de este mundo demasiado pronto; a pesar de su partida, su esencia estará presente en cada logro que alcance.

A mi padre Ángel Guaichico: quien, con su dedicación, amor y esfuerzo, me ha dado la oportunidad de tener una educación y alcanzar mis sueños. A mi madre María Piñan, por sus palabras de motivación y su apoyo incondicional; los cuales han sido la base para lograr esta meta. Mi anhelo es que este logro alcanzado les haga sentir tan orgullosos de mí, como yo lo estoy de tenerlos como padres.

A mis pequeños sobrinos Genesis, Aleksey y Ángeles, quienes dieron luz y alegría a mi vida en los momentos difíciles.

Edison Geovanny Guaichico

AGRADECIMIENTO

Agradezco infinitamente:

A Dios por darme la fuerza y la perseverancia para superar los desafíos y obstáculos presentados en el transcurso del camino hacia mi meta. También por haberme concedido la capacidad de aprender.

A mis padres: por ser los mejores inspiradores, por confiar, por inculcarme buenos valores y por creer en mis capacidades, que hoy me permiten decir que lo he logrado.

A la Universidad Técnica del Norte: especialmente a mis docentes, gracias por el tiempo y el conocimiento que han invertido en mí durante toda mi formación profesional.

Al mi director el PhD. Marco Pusdá, mi asesor el MSc. MacArthur Ortega y al PhD Iván García por ayudarme, aconsejarme y ser los guías en este trabajo de titulación.

Edison Geovanny Guaichico

RESUMEN

Hasta la actualidad se han realizados varios sistemas de registro de asistencia utilizando redes neuronales profundas, en su mayoría aplicaciones web. El objetivo de este trabajo fue desarrollar un nuevo método que ayude a reducir el tiempo en el registro de asistencia a clases de los estudiantes de la Universidad Técnica del Norte, utilizando redes neuronales convolucionales. Para dar solución al problema de exceso de tiempo se desarrolló una aplicación móvil de registro de asistencia implementado MTCNN un modelo bastante robusto para detección facial y MobileFaceNet. MobileFaceNet es un modelo utilizado para la extracción de incrustaciones faciales bastante ligera, y su precisión se considera similar a los mejores modelos actuales que realizan la misma tarea. La aplicación móvil captura imágenes cada segundo en tiempo real mediante la cámara principal del teléfono. Si se ha detectado un rostro, MobileFaceNet extrae las incrustaciones faciales de la imagen. Posteriormente mediante distancia euclidiana se verifica la identidad del estudiante, comparando el embedding predicho entre un conjunto de embedding previamente almacenado en una base de datos. La base de datos es un archivo json, que contiene la información y las incrustaciones faciales de cada estudiante. Con los estudiantes identificados se genera un reporte en Excel, el cual podrá ser utilizado por los docentes para registrar la asistencia, cargando el documento generado al portafolio académico de la Universidad. La sección de aprendizaje automático se analizó mediante el método CRISP-DM. Por otro lado, se utilizó la metodología en Cascada en el desarrollo de la aplicación móvil y, finalmente, se siguió las buenas prácticas establecidas por SWEBOK junto con las pruebas de caja negra para la validación de la aplicación. Las pruebas fueron realizadas a un grupo de 16 estudiantes, en un entorno controlado, en dos escenarios diferentes. En los dos escenarios los tiempos fueron mejores que realizarlo mediante el método tradicional. Por ende, se recomienda que este método pueda ser una solución viable para en un futuro considerar cambiar el método de registro de asistencia en la Universidad Técnica del Norte, en función de los resultados obtenidos.

ABSTRACT

To date, several attendance registration systems have been developed using deep neural networks, mostly web applications. The objective of this work was to develop a new method that helps to reduce the time in class attendance registration of students of the Universidad Técnica del Norte, using convolutional neural networks. To solve the problem of excess time, a mobile attendance registration application was developed using MTCNN, a robust model for face detection and MobileFaceNet. MobileFaceNet is a model used for fairly lightweight facial embedding extraction, and its accuracy is considered similar to the best current models performing the same task. The mobile application captures images every second in real time using the phone's main camera. If a face has been detected, MobileFaceNet extracts the facial inlays from the image. Subsequently using Euclidean distance, the student's identity is verified by comparing the predicted embedding between a set of embeddings previously stored in a database. The database is a json file, which contains the information and the facial embeddings of each student. With the identified students, an Excel report is generated, which can be used by teachers to record attendance, uploading the generated document to the University's academic portfolio. The machine learning section was analyzed using the CRISP-DM method. On the other hand, the Cascade methodology was used in the development of the mobile application and, finally, the good practices established by SWEBOK were followed along with black box testing for the validation of the application. The tests were performed on a group of 16 students, in a controlled environment, in two different scenarios. In both scenarios the times were better than the traditional method. Therefore, it is recommended that this method could be a viable solution to consider changing the attendance registration method at the Universidad Técnica del Norte in the future, based on the results obtained.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

Generalidades

Antecedentes

En la actualidad, los sistemas inteligentes han cobrado mayor relevancia frente a sistemas tradicionales, esto se debe a que las nuevas computadoras y/o máquinas utilizan nuevas tecnologías como la Inteligencia Artificial para mejorar sus actividades. Una de las ventajas de los sistemas inteligentes según Caro & López (2018), es que reduce el tiempo y el costo en la ejecución de alguna tarea, generando mayor nivel de confianza y eficiencia que los métodos tradicionales.

A continuación, se presenta las razones que motivaron al desarrollo de este trabajo de titulación, siendo la reducción de tiempo el principal motivo. Se comenzó con la identificación del problema, luego se definió los objetivos que se pretenden cumplir, el alcance que se pretende lograr y una justificación de fundamente el trabajo.

Tema

Aplicación móvil de reconocimiento facial para el control de asistencia a clases de los estudiantes de la Universidad Técnica del Norte utilizando técnicas de Inteligencia Artificial.

Problema

El alto consumo de tiempo por el uso del portafolio académico ha sido una de las falencias del sistema (SIIU) UTN; por lo general causado por la saturación de usuarios que acceden a la plataforma al mismo tiempo, ocasionando así una calidad de servicio limitada. Según Koné (2021), los sistemas informáticos deben estar disponibles y brindar el servicio a los usuarios en cualquier momento, lo cual implica que estén disponibles las 24 horas del día.

La impuntualidad de algunos estudiantes, quienes llegan tarde a las actividades académicas después de la hora límite establecido por la institución, ocasiona que se tenga registrar nuevamente la asistencia en el portafolio académico, perdiendo así un tiempo irrecuperable. Quishpe (2013), menciona que el tiempo de retraso aceptable en circunstancias normales, por lo general es de diez a quince minutos. El estudiante que ingresa tarde a clases afecta al desarrollo de la actividad lectiva, provocando retrasos y pérdida de concentración a los demás y a sí mismo (Centro Educativo La Amistad, 2023).

En ocasiones, durante el registro de asistencia, algunos docentes no verifican la autenticidad del estudiante que responde a la lista. Esto puede generar problemas de suplantación de identidad, es decir, un estudiante responde por alguien que no esté presente en el aula de clases.

Según una observación, al tomar asistencia de manera tradicional a un paralelo de estudiantes de la carrera de Software, se evidenció, que el consumo promedio de tiempo en realizar esta actividad es de 4 a 5 minutos en un grupo de 40 estudiantes. Este tiempo fue obtenido con un cronómetro, desde el momento que se abre el navegador hasta terminar de registrar al último estudiante.

Objetivos

Objetivo General

Desarrollar una aplicación móvil de reconocimiento facial para el control de asistencia a clases de los estudiantes de la Universidad Técnica del Norte utilizando técnicas de Inteligencia Artificial.

Objetivos Específicos

- Elaborar un marco teórico respecto a las técnicas de reconocimiento facial con Inteligencia Artificial y herramientas de desarrollo.
- Desarrollar una aplicación móvil que utilice una red neuronal convolucional de reconocimiento facial.
- Validar el funcionamiento de la aplicación móvil cumpliendo con los estándares y buenas prácticas establecidos en la guía SWEBOK.

Alcance

El alcance de este trabajo de titulación es diseñar y desarrollar una aplicación móvil de reconocimiento facial, para la identificación de estudiantes en tiempo real mediante el uso de una red neuronal convolucional. Con la finalidad de apoyar a los docentes en el proceso de control de asistencia de los estudiantes de CSOFT de la Universidad Técnica del Norte (UTN).

Los datos (imágenes) se obtuvieron empleando una cámara digital, posteriormente el tratamiento de las imágenes se lo realizó utilizando la librería Scikit-Learn de Python, debido a que algunos algoritmos como: RNA, SVM, K-NN, son muy sensibles a la escala desigual de datos. En la manipulación del modelo se utilizó las librerías de código abierto TensorFlow y Keras, los cuales, al ser libres, cuentan con una

amplia comunidad que aportan y ayudan a encontrar soluciones a diferentes dificultades.

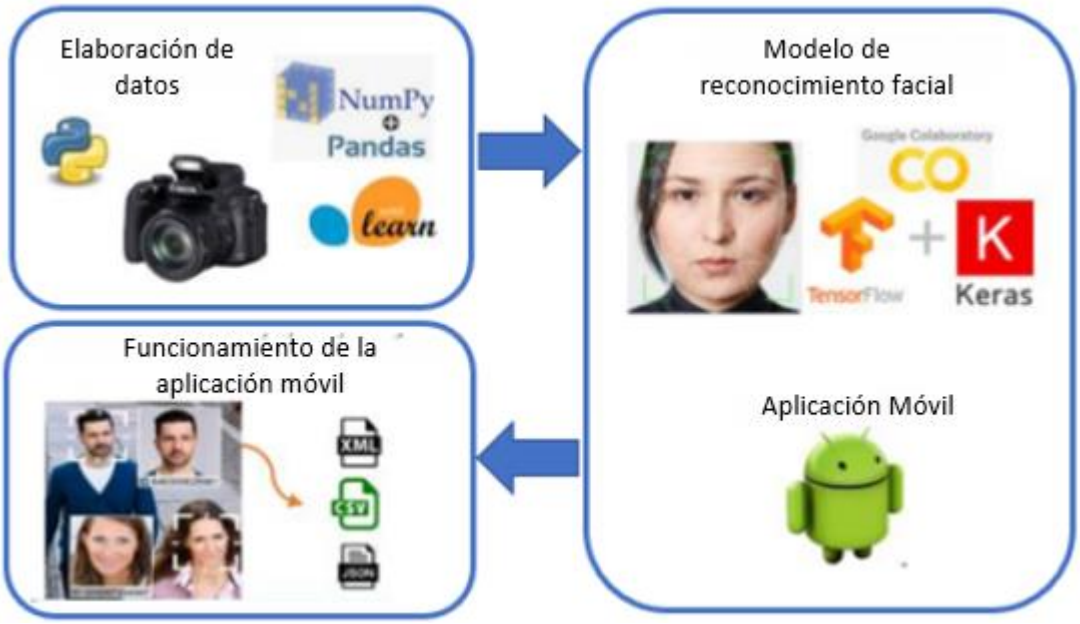
Para utilizar Python, se adquirió un plan de Iaas de Google Colab, con la finalidad de entrenar o probar los modelos de red neuronal de una manera más eficiente.

La aplicación móvil únicamente fue desarrollada para dispositivos Android, debido a la creciente demanda de este sistema operativo en los dispositivos móviles. Por otra parte, el desarrollo en Android ofrece algunos beneficios como: desarrollo a bajo costo, es decir que no es necesario pagar una licencia para desarrollar apps, a diferencia del sistema operativo IOS. Una de las razones de seleccionar un solo sistema operativo en lugar de ser multiplataforma, se debe a que en este proyecto se pretende desarrollar solo un prototipo funcional que permita comprobar que este método de registro de asistencia puede ser mejor que los métodos tradicionales

La comprobación del funcionamiento se lo realizó, identificando a los estudiantes de una asignatura en particular de la carrera de Software (UTN) en un ambiente cerrado. La aplicación móvil fue capaz de clasificar mediante reconocimiento facial a los estudiantes presentes, para luego, generar un reporte de dichos estudiantes mediante un documento de intercambio de datos común.

Figura 1

Alcance del Proyecto.



Nota: Elaboración propia.

Justificación

Actualmente la Universidad Técnica del Norte no cuenta con un método que permita llevar el control de asistencias de los estudiantes mediante un sistema de reconocimiento facial. Aunque se ha utilizado esta tecnología en otras tareas, tales como: sistemas biométricos para el registro de asistencia y control acceso de docentes a las aulas, restricción de acceso para ingreso y salida de docentes como de estudiantes en el campus universitario.

Una aplicación móvil de reconocimiento facial presentará mayor facilidad y mejorará los procesos en el control de asistencia, aprovechando las facilidades tecnológicas existentes en la actualidad, ayudando así a reducir el tiempo.

Este trabajo establece un punto de partida para el registro de asistencia de los estudiantes, el cual puede ser implementado en toda la comunidad UTN. La aplicación es una forma de colaboración hacia la Universidad Técnica del Norte.

.

Capítulo I.

Marco Teórico

En este capítulo se presenta conceptos generales sobre: Inteligencia Artificial, Aprendizaje Profundo, Reconocimiento Facial, se menciona las metodologías y las herramientas de programación a emplearse. También se ha realizado una recopilación bibliográfica de tesis e investigaciones previas, los cuales serán útiles para el sustento de este trabajo de titulación.

1.1 Control de asistencia a las clases.

En la actualidad la asistencia a clase y el compromiso de los estudiantes juegan un papel importante en la educación superior. Aunque la asistencia no es obligatoria, estudios han demostrado que los estudiantes que más asisten a las clases obtienen mejores resultados académicos (Lukkarinen et al., 2016). El control de asistencia consiste en gestionar y dar un seguimiento a la asistencia de los alumnos a las clases, registrando las ausencias en el sistema. En instituciones de educación superior existe un número mínimo de faltas para los estudiantes, pasado ese rango ellos podrían perder automáticamente el semestre (UTN, 2019).

1.2 Inteligencia Artificial (IA).

La Inteligencia Artificial fue adoptada por primera vez en 1956 en la Universidad de Dartmouth. La IA se refiere a la capacidad de las computadoras/máquinas para utilizar algoritmos y aprender de los datos por si solas aplicando las matemáticas, con el fin ejecutar tareas sin la intervención de inteligencia humana (Rouhiainen, 2018). Aunque la automatización de algunas tareas con la IA ha ocasionado pérdida de empleos en algunos sectores, al mismo tiempo, se ha creado nuevos puestos de trabajos en otros sectores beneficiándose de dicha tecnología.

Día a día aparecen nuevos descubrimientos y se ven muchas aplicaciones de esta tecnología tales como en la: Medicina, Robótica, Ingeniería, Educación, Matemáticas, Ley, Arquitectura, Seguridad, Bilogía, Gestión de la información, Industria del entretenimiento, Agrícola, Finanzas y muchas otras áreas.

1.3 Aprendizaje Automático (Machine Learning).

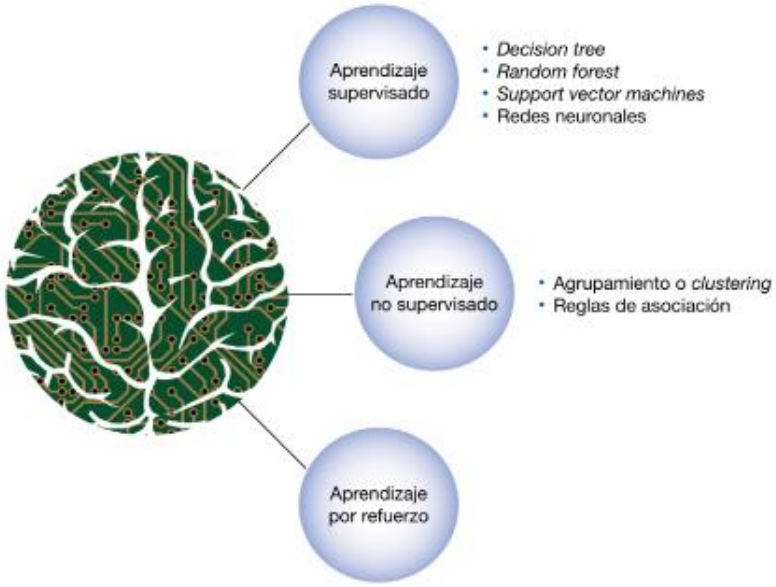
El aprendizaje automático es un campo de la inteligencia artificial orientada en la creación de algoritmos, que sean capaces de aprender patrones de datos por si solas

para hacer predicciones, sin estar programados para esa tarea (Rouhiainen, 2018). Esto ofrece una gran ventaja al controlar una gran cantidad de datos de manera más eficiente (ADP, 2019).

Dependiendo de los datos y la salida requerida se han propuesto diversos algoritmos de aprendizaje automático. Son tres las principales categorías en las que se divide el Machine Learning; aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo.

Figura 2

Aprendizajes de Machine Learning.



Nota: Reproducida de (Beunza & Puertas, 2019).

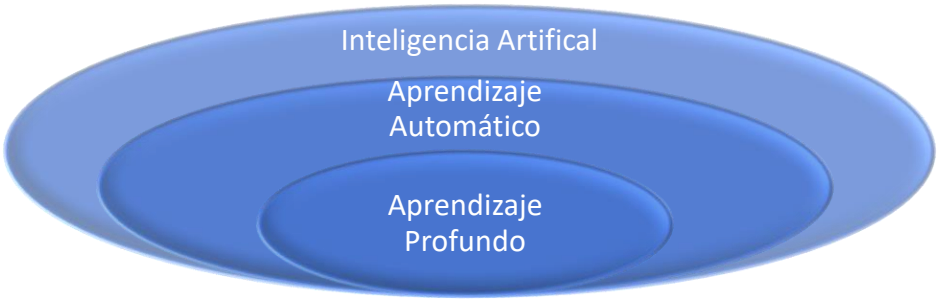
- **Aprendizaje Supervisado:** En este tipo de algoritmos se tiene la necesidad de contar con un conjunto de datos de entrada ya etiquetadas, para poder predecir una salida específica, el cual es útil cuando se conoce cuál y cómo será el resultado (Microsoft Azure, n.d.).
- **Aprendizaje no Supervisado:** Es un tipo de aprendizaje en el cual el modelo busca patrones similares en los datos de entrada sin etiquetar (Microsoft Azure, n.d.), sobre las categorías a las que deben pertenecer una salida.
- **Aprendizaje por Refuerzo:** Se basa en cómo un agente interactúa con su entorno, donde el modelo aprende a tomar decisiones autónomas sobre cuáles serían sus próximas acciones en función de sus aciertos y errores, sin antes haberle proporcionarle los datos de entrenamiento (IBM, n.d.); maximizando la recompensa a lo largo del tiempo a través de la experiencia.

1.4 Aprendizaje Profundo (Deep Learning).

El aprendizaje profundo es una técnica más avanzada dentro del aprendizaje automático, el cual imita las acciones del cerebro humano, lo que les permite a las computadoras/máquinas aprender características abstractas y complejas de datos no estructurados como: documentos, imágenes y textos. Con el aumento de la capacidad de procesamiento de los ordenadores y la disponibilidad de grandes conjuntos de datos, se ha impulsado más el desarrollo del aprendizaje profundo. Rosero (2019) menciona que, las redes neuronales con más de cinco capas no lineales son denominadas redes de aprendizaje profundo, estas capas no son creadas intencionalmente por especialistas humanos, sino que son aprendidas a partir de los datos, utilizando técnicas generales de aprendizaje.

Figura 3

Vínculo entre la IA, Machine Learning y Deep Learning.



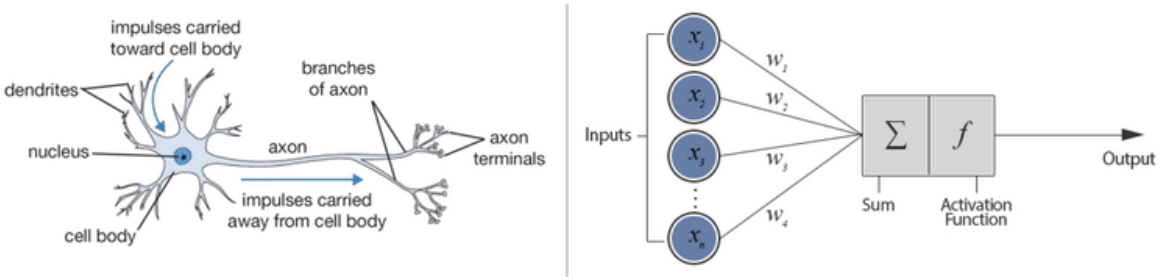
Nota: Adaptada de (Muñoz, 2021).

1.4.1 Redes Neuronales Artificiales (RNA).

Las redes neuronales artificiales son modelos matemáticos interconectadas entre sí, que imitan el funcionamiento de las redes neuronales biológicas del cerebro humano, a través de múltiples conexiones.

Figura 4

Red Neuronal Biológica VS Red Neuronal Artificial.



Nota: Reproducida de (DataScientest, 2021).

De acuerdo con Bonilla (2020), las redes neuronales artificiales son utilizados para aproximar o estimar funciones que dependan de grandes cantidades de datos de entradas, muchas de las cuales pueden no conocerse. Las redes neuronales artificiales y las redes neuronales biológicas tienen enfoques y aplicaciones diferentes, pero mantienes algunos conceptos relacionados.

Tabla 1

Red Neuronal Biológica versus una Red Neuronal Artificial.

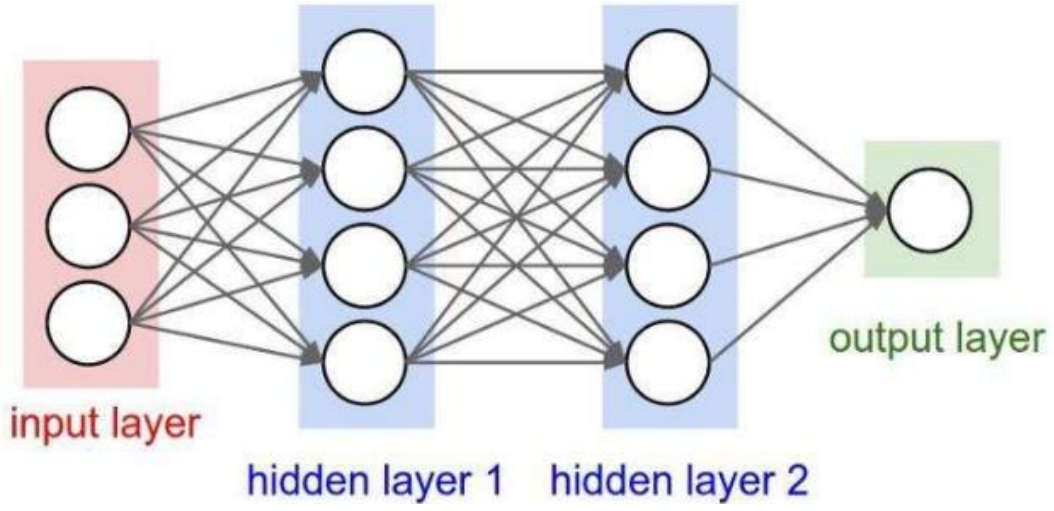
	Red neuronal biológica	**Red neuronal artificial**
Dendritas	Reciben señales eléctricas y químicas de otras neuronas	Entrada de datos
Sinapsis	Puntos de conexión entre las neuronas	Pesos establecidos para las conexiones
Núcleo	Recibe las señales desde las dendritas y las procesa para enviarlas al axón	Procesamiento y cálculo
Axón	Transmite las señales eléctricas a otras neuronas	Salida de datos

Nota: Elaboración propia.

En la Figura 5, se muestra la estructura esquemática de una red neuronal artificial.

Figura 5

*Estructura esquemática de una Red Neuronal Artificia*l



Nota: Reproducida de (Muñoz, 2021).

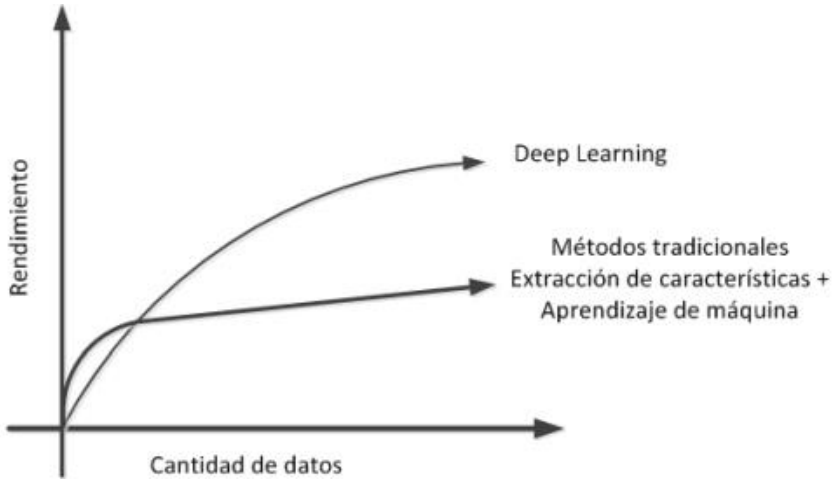
- **Capa de entrada (Input Layer)**: Capa que recibe directamente los datos de entrada.
- **Capas Ocultas (Hidden Layer):** Capas intermedias que procesan y extraen características de los datos de entrada y no tienen contacto directo con el entorno exterior.
- **Capa de salida (Output Layer):** Capa que produce la solución de la red.

1.4.2 Redes Neuronales Profundas (RNP).

Una red neuronal profunda es un tipo especial de red neuronal artificial que tiene múltiples capas ocultas entre la capa de entrada y la capa de salida, lo que les permite aprender patrones complejos. Mientras más neuronas y capas ocultas tengan la red, existe mayor precisión y mejor clasificación de los datos, sin embargo, el costo computacional es mayor (Untuña, 2022). Las RNP's no son necesariamente mejores de las RNA's, para escoger el uno o el otro, ambos se utilizan en problemas de aprendizaje automático, sin embargo, las RNP's son utilizadas para tareas complejas de procesamiento de imágenes y análisis de datos.

Figura 6

Rendimiento del Deep Learning VS Técnicas Tradicionales.



Nota: Reproducida de (Rosero, 2019).

1.4.3 Tipos de Redes Neuronales Profundas.

Existen algunos tipos de redes neuronales profundas, cada uno tiene sus propias características, ventajas y desventajas. Se menciona algunos tipos de RNP's más populares: las Redes Recurrentes basada en los trabajos de David Rumelhart de 1986, son conocidas por tener memoria artificial para procesar y obtener información de datos secuenciales o series temporales, es decir, su funcionamiento se realiza en bucle,

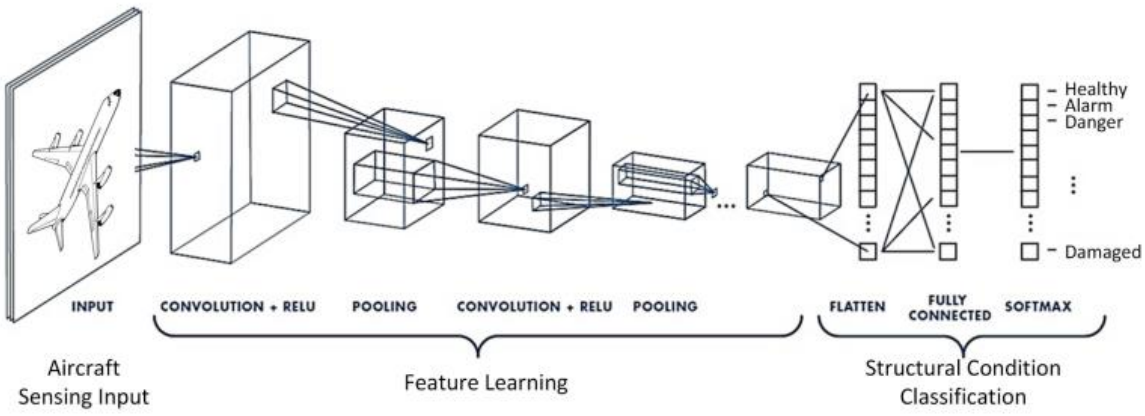
actualizando la información en cada iteración (Arana, 2021); las Redes Auto-encoder son utilizados para realizar tareas de extracción de características y reducción de dimensionalidad, estas redes son capaces de generar una reconstrucción de los datos hasta obtener algo similar a los originales en función de las características de entrada, conformado por un codificador (encoder) y un decodificador (decoder) (Matlab, 2022). En la actualidad se está hablando mucho de las Redes Transformers, lanzado en 2017 en el artículo "Attention Is All You Need", esta red se presentó como una alternativa al problema que existía con la traducción texto de un idioma a otro. El modelo Transformers se basa en la arquitectura codificador y decodificador con múltiples capas conocidas como atención, esto les permite procesar una mayor cantidad de datos en paralelo por la red en menor tiempo (Vaswani et al., 2017). Son considerados los modelos más potentes inventados hasta la fecha.

1.4.3.1 Redes Neuronales Convolucionales (CNN).

Las redes neuronales convolucionales son también un tipo de red neuronal artificial, utilizados en el procesamiento y clasificación de datos espaciales en forma tridimensional (ancho, largo y profundidad de capa), como imágenes y videos, Según Casado (2022), una red convolucional puede contener cientos de capas ocultas, que utilizan la convolución en al menos una de sus capas para la extracción de características. La Figura 7 muestra una red convolucional con varias capas ocultas que aplican filtros con diferentes resoluciones a una imagen, siendo la salida convolucionada, la entrada de la siguiente capa.

Figura 7

Red neuronal convolucional con varias capas.



Nota: Reproducida de (Rodríguez, 2022).

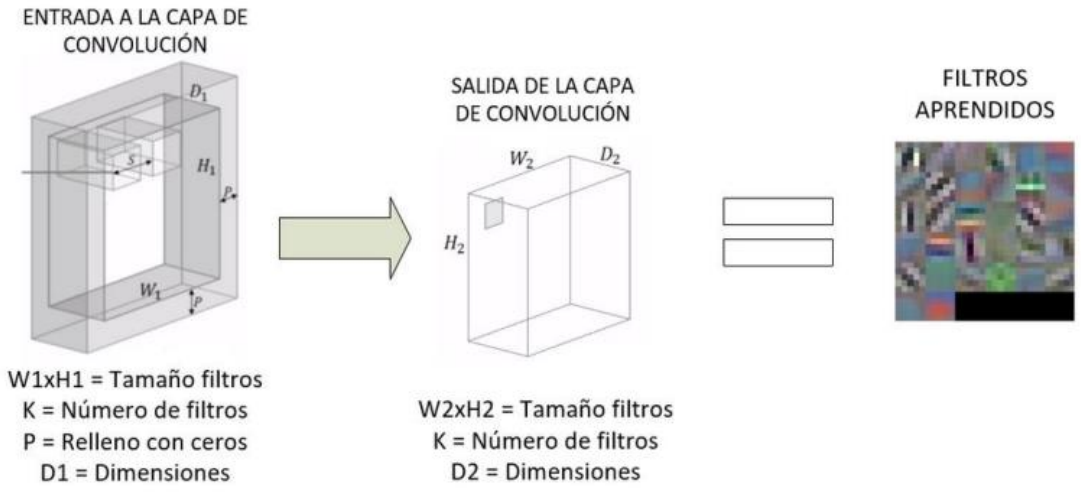
1.4.3.1.1 Estructura de una Red Neuronal Convolucional.

Una red convolucional típica consta de los siguientes componentes claves: capa de convolución, capa de pooling, capa fully connected y función de activación como los más importantes.

- **Capa de Convolución:** La capa de convolución es la piedra angular de las redes neuronales convolucionales. Según Rodríguez (2022) la capa de convolución es la capa oculta en donde para cada neurona se define los pesos de los filtros; este proceso es necesario para que el modelo aprenda. La capa de convolución consiste en un conjunto de filtros cuadrados de aprendizaje K (kernels). Cuando un filtro se desliza sobre la imagen y produce un producto punto a punto entre los valores de píxeles de la imagen, los resultados de esta operación se suman y guardan en la posición correspondiente a la imagen de salida. A medida que se desliza el filtro sobre la imagen, se van creando mapas de características que resumen las características más relevantes de la imagen de entrada (Rosero, 2019).

Figura 8

Capa de convolución.

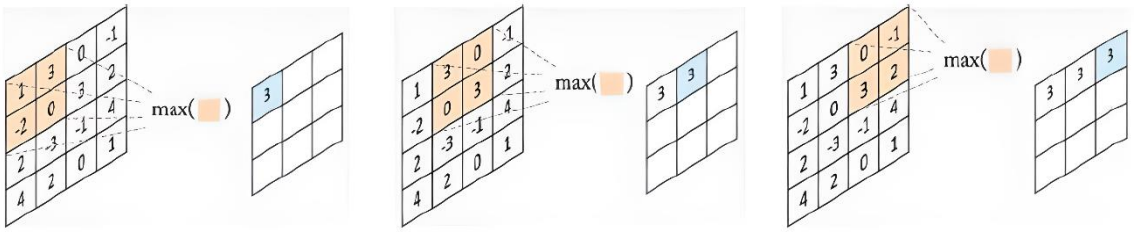


Nota: Reproducida de (Rodríguez, 2022).

- **Capa de Pooling:** También conocida como capa de agrupación, tiene como objetivo principal reducir la dimensión espacial de las representaciones, mientras se mantienen las características más importantes de la imagen, a través de un cálculo estadístico. Las técnicas de agrupación más conocidas son: averange-pooling y max-pooling, de los cuales max-pooling es la técnica más utilizada, debido a que reduce el tamaño del mapa significativamente (Indolia et al., 2018).

Figura 9

Operación de max-pooling en una región 2x2.



Nota: Reproducida de (Chacua, 2019).

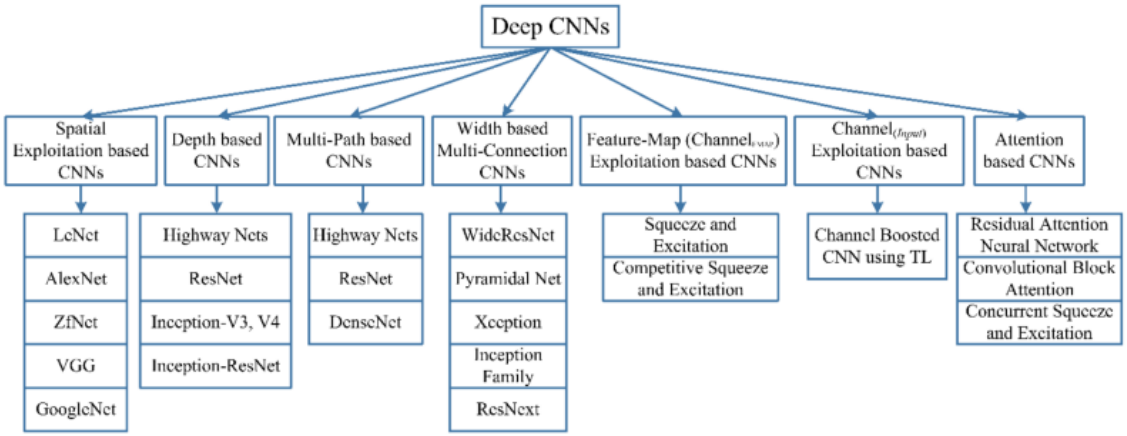
- **Capa Fully Connected (FC):** La capa FC está ubicado al final de la CNN y es exactamente igual a cualquier capa de una red neuronal artificial clásica, lo que diferencia a esta capa de las demás, es que esta capa está completamente conectada con todas las neuronas de la capa anterior (Bonilla, 2020). Cada neurona en la capa FC calcula una suma ponderada de los datos de entradas de la capa anterior, luego se aplica una función de activación (Softmax) para producir la salida. “Al final, la capa devuelve un vector de igual tamaño al número de clases, donde cada componente representa la probabilidad de que la imagen de entrada pertenezca a una clase en particular” (DataScientest, 2021).
- **Funciones de activación:** Las redes neuronales que utilizan solo la función lineal en su red no son capaces de resolver problemas complejos, debido a que aprenden únicamente relaciones lineales entre los datos. Desde años atrás se han estado investigando y se han descubierto nuevos métodos conocidos como funciones de activación que intentan solucionar ese problema. La función de activación es una operación matemática que permite agregar la no linealidad a la red, de esta manera la red ya no está limitada solo a un plano, sino que aprende relaciones no lineales en datos más complejos (Ringa Tech, 2022). Conforme se vaya agregando más capas a la red y con la selección adecuada de la función de activación, se obtiene buenos resultados.

1.4.4 Arquitecturas Convolucionales Destacadas.

Las arquitecturas convolucionales han optimizado y mejorado la manera en que las máquinas procesan e interpretan la información visual. Numerosas aplicaciones se han logrado implementado estas arquitecturas. Se presentan algunas de las arquitecturas convolucionales más populares inventadas a lo largo del tiempo.

Figura 10

Clasificación de las redes neuronales convolucionales.



Nota: Reproducida de (Quintero, 2020).

Estas arquitecturas pueden ser utilizados para resolver un sinfín de tareas, sin embargo, en este estudio se da más énfasis al reconocimiento facial. En la implementación de una CNN, la selección adecuada de la arquitectura es primordial para lograr un equilibrio entre precisión y eficiencia computacional. El problema radica, cuando son los dispositivos móviles donde se debe implementar la arquitectura, debido a que la capacidad de cálculo y la memoria, son los recursos más limitados en comparación con otros entornos tradicionales. Por lo tanto, es necesario tomar una decisión sobre si se requiere una red eficiente con precisión aceptable o se requiere una red precisa, aunque la eficiencia no sea tan buena. Más adelante se presenta arquitecturas por cada caso.

- **MobileNetV2:** MobileNetV2 es una arquitectura de red neuronal convolucional desarrollada por Google, diseñada específicamente para dispositivos móviles y aplicaciones con restricciones de recursos. La entrada y salida del bloque residual son capas delgadas de cuello de botella en la arquitectura de MobileNetV2, que permite una mayor profundidad sin gastar mucho dinero en computación (Sandler et al., 2018). A diferencia de los modelos residuales convencionales que utilizan representaciones expandidas en la entrada, MobileNetV2 utiliza convoluciones ligeras (separables) en profundidad para filtrar características en la capa de expansión intermedia, lo que reduce significativamente el número de cálculos en las operaciones.

- **FaceNet:** FaceNet es un modelo de aprendizaje profundo específicamente para reconocimiento facial, de alta precisión y capacidad discriminativo. Fue desarrollado por investigadores de Google y presentado en 2015 en el artículo titulado "*FaceNet: A Unified Embedding for Face Recognition and Clustering*", logrando una precisión de 99,63%. FaceNet aprende directamente un mapeo de imágenes a un espacio euclidiano compacto, basado en el aprendizaje de incrustaciones (representaciones vectoriales) de los rostros, donde las distancias corresponden directamente a las medidas de similitud facial. Este modelo utiliza redes convolucionales profundas entrenadas para optimizar las incrustaciones en lugar de las capas de cuello de botella intermedias (Asmara et al., 2022). FaceNet utiliza la función de pérdida triplete para entrenar la red. En esta función se comparan tres imágenes de rostros: ancla (el rostro a reconocer), positiva (diferentes imágenes de la misma persona) y negativa (imágenes de diferentes personas), la pérdida acerca el ancla a los valores positivos y la aleja de los valores negativos en el espacio de objetos (Schroff et al., 2015).

1.5 Reconocimiento Facial

El reconocimiento facial utiliza un análisis detallado de las características faciales de una persona y una comparación con imágenes almacenadas en un conjunto de datos para identificar automáticamente con precisión el rostro de una persona a partir de imágenes digitales (Garcés, 2017).

1.5.1 Historia del Reconocimiento Facial

El Reconocimiento Facial es una técnica que se ha utilizado desde hace algunas décadas atrás, pero se ha popularizado en los últimos años gracias a nuevos algoritmos mejorados y a la aparición del aprendizaje automático y el aprendizaje profundo.

Los inicios del Reconocimiento Facial se remontan a la década de 1960, donde el matemático Woodrow Wilson Bledsoe inventó un sistema para clasificar los rasgos faciales de la persona utilizando una tabla llamada RAND**.** En 1988, los investigadores Sirovich y Kirby aplicaron análisis de componentes principales (PCA) mediante el uso del álgebra lineal, a la dificultad de clasificar los rasgos del rostro humano. Años más

tarde en 1991, M. Turk y A. Pentland, mejoraron los resultados de la investigación y lo denominaron Eigenface. En el año 1991, Cheng y su equipo también presentaron el enfoque conocido como (LDA), este método buscaba identificar un espacio lineal que permita la máxima distinción entre dos categorías (rostro) de patrones. En 1997, Belheumeur introdujo un método que es una combinación de los métodos PCA y LDA para el reconocimiento facial conocido como Fisherface.

A principios de la década del 2000, se introdujo el reconocimiento facial basado en características locales, que utilizaba filtros artesanales como Gabor y LBP para identificar características discriminatorias. Por el 2010, aparecieron los descriptores locales basados en el aprendizaje de filtros y codificadores locales. El 2014 fue un año importante en la historia del reconocimiento facial, debido a que se reformuló el enfoque de investigación de esta tecnología. Fue el año en que el modelo DeepFace de Facebook, evaluado en el conjunto de datos de referencia de LFW, se acercó al rendimiento humano por primera vez alcanzando un 97,53%. La precisión del reconocimiento facial alcanzó el 99,80 % en solo tres años después de este importante avance. (Pokhrel, 2020).

1.5.2 Aplicaciones del Reconocimiento Facial

Existe una amplia variedad de usos del reconocimiento facial en distintas categorías. En la Tabla 2 se mencionan algunos escenarios de aplicación.

Tabla 2

Contextos de aplicación del Reconocimiento Facial.

Categoría	Contexto de Aplicación
Identificación facial	-Licencias de conducir. -Pasaportes. -Banca virtual.
Control de acceso	-Acceso a instalaciones. -Cajeros automáticos. -Acceso a centro de datos. -Acceso a la red informática. -Acceso a base de datos online.
Seguridad y Vigilancia	-Identificación de personas en aeropuertos. -Verificación de identidad en bancos. -Videovigilancia avanzada. -Detección de delincuentes. -Detección de fraudes de identidad.

Marketing	-Identificación de clientes. -Medición de respuesta emocional. -Anuncios publicitarios.
Medicina	-Diagnóstico enfermedades dermatológicas. -Identificación de pacientes.
Aplicaciones móviles y redes sociales	-Desbloqueo del teléfono. -Etiquetado de rostros para redes sociales.
Tarjetas inteligentes	-Seguridad de valor almacenado -Autenticación de usuario.

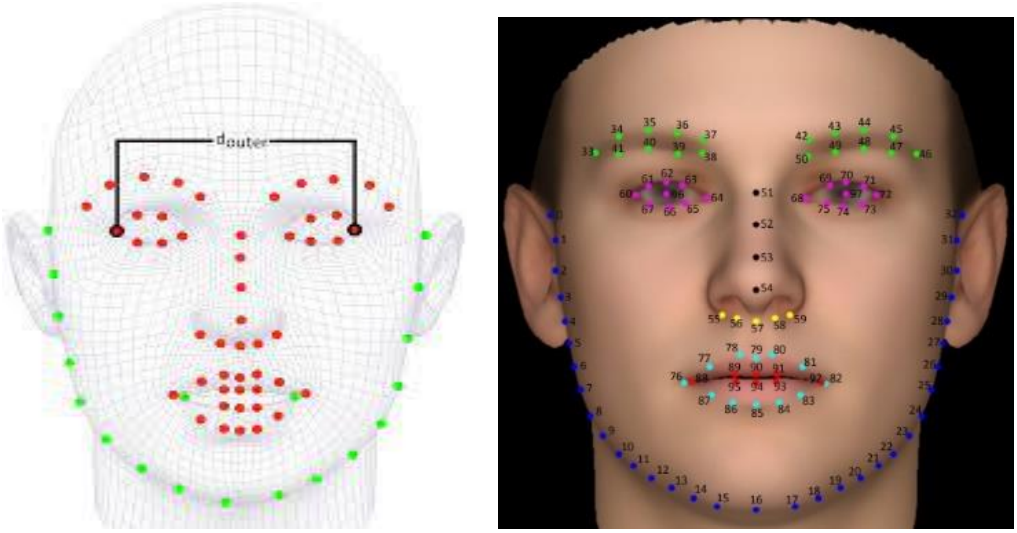
Nota: Adaptado de (Domínguez, 2017).

1.5.3 Características o Rasgos Faciales en el Reconocimiento Facial.

Las características faciales son esenciales en el reconocimiento facial debido a que estas son únicas para cada individuo. Aunque la apariencia, el peso, el estilo de peinado, etc, de una persona cambia con el tiempo por la edad; las características faciales básicas siguen siendo las mismas y pueden ser utilizadas para identificar a una persona. Los rasgos faciales más comunes en el reconocimiento facial son: la forma y tamaño del rostro, los ojos, las cejas, la nariz, la boca y la barbilla. Las características faciales se extraen, se procesan y se comparan para identificar a la persona, siendo las más importantes: la distancia entre ojos, la estructura de la nariz y la distancia entre labios (Saleem et al., 2021).

Figura 11

Rasgos Faciales mediante puntos referenciales.



Nota: Reproducida de (Cardona & Pineda, 2018).

1.5.4 Complicaciones en el Reconocimiento Facial.

Aunque el reconocimiento facial ha avanzado mucho en los últimos años, aún presenta ciertas limitaciones y problemas, que pueden dificultar su uso y precisión en determinadas situaciones. En la Tabla 3 se describen algunas de las principales complicaciones que se pueden presentar, por lo cual es recomendable tenerlos en cuenta al desarrollar un sistema de este tipo.

Tabla 3

Complicaciones en el Reconocimiento Facial.

Complicación	Descripción
Iluminación	Si la iluminación es demasiado brillante u oscura, las características faciales pueden no ser detectadas correctamente.
Cambios de apariencia facial.	Los cambios en la apariencia de una persona, como el uso de maquillaje, el cambio de peinado o el uso de gafas puede dificultar la identificación.
Posición y Orientación	Si la cabeza está inclinada hacia arriba o hacia abajo, o si se gira hacia un lado, el sistema no es capaz de detectar las características faciales.
Falsificaciones	Las imágenes falsas pueden engañar al sistema de reconocimiento facial.
Privacidad	Si se utiliza sin el consentimiento de las personas afectadas, se podría genera un problema.

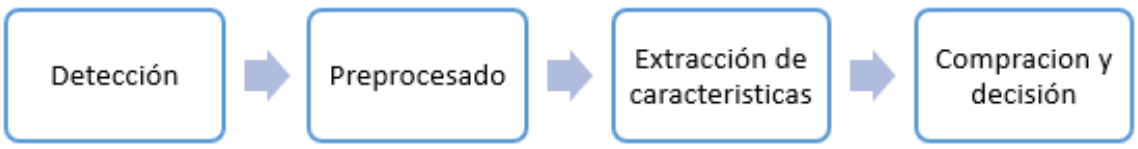
Nota: Elaboración propia

1.6 Fases en un Sistema de Reconocimiento Facial.

Para lograr un reconocimiento facial efectivo y preciso, cualquier sistema siguen cuatro fases principales: detección, preprocesado, extracción de características, comparación y decisión. En la Figura 12 se muestras dichas fases, más adelante se describen mejor cada una.

Figura 12

Diagrama general de un sistema de Reconocimiento Facial.



Fuente: Adaptado de (Domínguez, 2017).

1.6.1 Detección.

La detección es la primera etapa del proceso de reconocimiento facial y es fundamental para el correcto funcionamiento del sistema. Según Rosero (2019), la detección facial tiene como finalidad identificar si los objetos que aparecen en una imagen digital corresponden o no a un rostro. Esta fase identifica rasgos faciales en las imágenes para diferenciarlos del resto mediante el reconocimiento de patrones, como resultado se produce una serie de áreas donde es más probable encontrar un rostro, que generalmente son regiones rectangulares (Cardona & Pineda, 2018). En la actualidad existen algunos métodos para la detección facial, unos más complejos que otros tales como: detección basada en características como Viola-Jones, detección basada en puntos de referencia como Dlib, detección basada en histograma de gradientes como el descriptor HOG, detección basada en redes neuronales profundas como MTCNN y otros métodos más. Sin embargo, se mencionan algunas investigaciones donde proponen mejoras para la detección facial.

Gao & Yang (2022), proponen un nuevo algoritmo de detección facial el cual es una versión mejorada de TinyYOLOv3. El método se basa en aumentar el número de capas y en lugar de utilizar la convolución tradicional, utilizar la convolución profunda separable, donde fusionan las características de diferentes capas de red, logrando así obtener más información semántica. Por lo tanto, se garantiza la precisión en la detección facial y se reduce el tamaño de la red. En la investigación de Hangaragi et al. (2023), proponen un modelo que puede detectar y reconocer el rostro utilizando Face Mesh. El modelo Face Mesh funciona con diferentes tipos de fondo e iluminación, es decir, que captura imágenes de fondos no frontales. Méndez et al. (2019), presenta un método de frontalización de una imagen del rostro. Este método se basa en utilizar los puntos de referencia y un modelo elástico genérico 3D, para la alineación y reconstrucción en una imagen frontal desde un rostro completamente en perfil.

Luego de revisar algunos métodos para la detección facial, se menciona dos métodos bastante populares que realizan esta tarea.

1.6.1.1 Viola & Jones.

El algoritmo de Viola-Jones fue creado en 2001 por los investigadores Paul Viola y Michael Jones estudiantes de la Universidad de Cambridge. Viola-Jones es un algoritmo que encuentra las características más relevantes de rostro humano, utilizando

filtros que se asemejan a las funciones fundamentales de las transformadas HAAR, cuyos resultados se clasifican utilizando el algoritmo AdaBoos (Rojas et al., 2021). El algoritmo de Viola Jones se basa en tres conceptos principales: el primero es conocido como: Imagen Integral una representación de la imagen original que se utiliza para calcular rápidamente las características Haar, evitando la necesidad de recorrer todas las subventanas de la imagen original; el segundo es un algoritmo de aprendizaje basado en AdaBoost (*adaptative boosting),* que elige las características del objeto de interés en diferentes orientaciones y tamaños de un conjunto más grande, para construir un clasificador fuerte a partir de clasificadores débiles; es tercer concepto es la cascada de clasificadores, que combina los clasificadores más complejos y permite que el algoritmo descarte rápidamente las áreas de la imagen que no contienen el objeto de interés, reduciendo el tiempo (Viola & Jones, 2001). Cada clasificador en la cascada evalúa, si la región actual de la imagen de entrada es positiva (contiene una cara) o negativa. Si la región es evaluada como quizás, se envía al siguiente clasificador en la cascada y nuevamente se evalúa si existe o no un rostro.

Figura 13

Algoritmo de Paul Viola y Michael Jones.

Nota: Reproducida de (Rojas et al., 2021).

1.6.1.2 Red Convolucional en Cascada Multitarea (MTCNN).

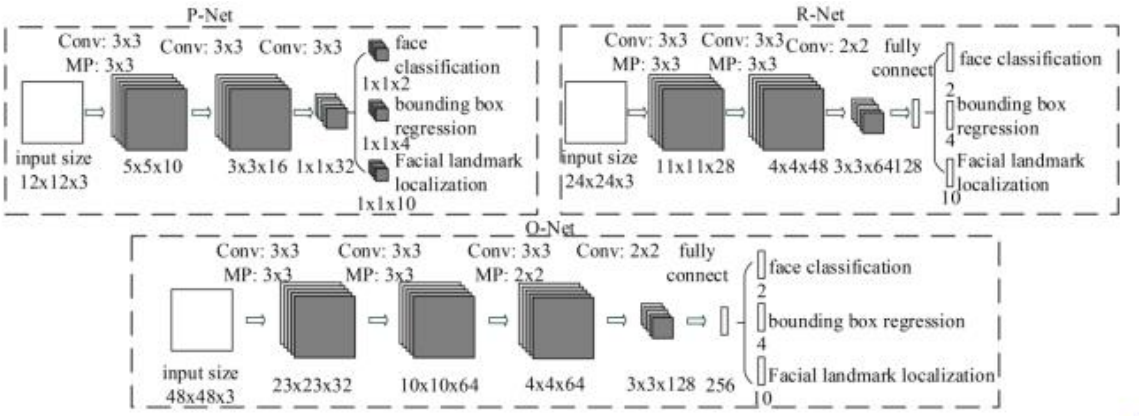
MTCNN es un modelo de detección facial presentada en 2015 en el artículo, "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks", por (Zhang et al., 2015). MTCNN es bastante utilizado en procesamiento de imágenes y en computer vision, debido a la excelente precisión en diversas condiciones en la detección de rostros.

La detección facial de este modelo consiste en tomar una imagen y convertirla a otra escala, con la finalidad de crear una pirámide de imágenes que servirá como entrada para la siguiente red en cascada (Gradilla, 2020). Las redes en cascada de este modelo son tres: la primera red Proporsal Network (Pnet), produce regiones faciales candidatas a través de una CNN poco profunda; luego una red más compleja Refine Network (Rnet), refina las regiones candidatas y filtra aquellas ventanas que no contienen rostros; finalmente la red Output Network (Onet), utiliza una CNN más robusta para refinar y alinear mejor el resultado de la red anterior, y así generar las puntuaciones de confianza del rostro detectado (Zhang et al., 2015).

La clasificación final de los rostros depende de las dos capas completamente conectadas de la red MTCNN. Una matriz de probabilidad para cada candidato se crea en la primera capa completamente conectada, mientras que la segunda capa completamente conectada crea un cuadro delimitador y puntos de referencia faciales finales.

Figura 14

Arquitectura MTCNN.

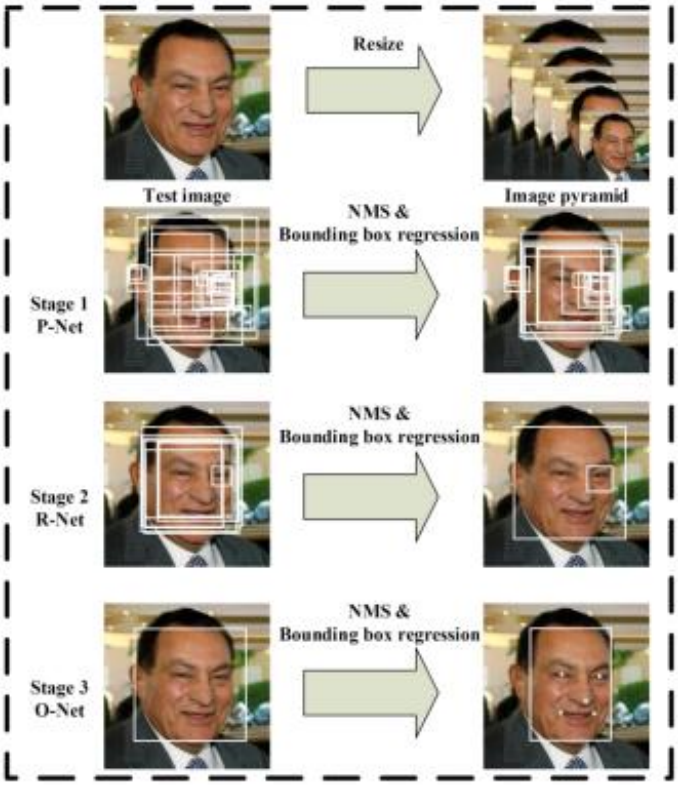


Nota: Reproducida de (Zhang et al., 2015).

La Figura 15, muestra cómo se procesa una imagen en el modelo MTCNN. Se puede observar que, a partir de la imagen original, que tiene una medida de 12x12x3 con un filtro de 3x3 y una profundidad de 10 capas, se crea una distribución en pirámide que se reescalan a diferentes dimensiones. De esta manera se genera una serie de versiones en nuevas escalas que pasan por las etapas, encontrando rostros de mayor tamaño en las etapas iniciales y rostros más pequeños en las etapas posteriores.

Figura 15.

Distribución en pirámide de una imagen aplicando MTCNN.



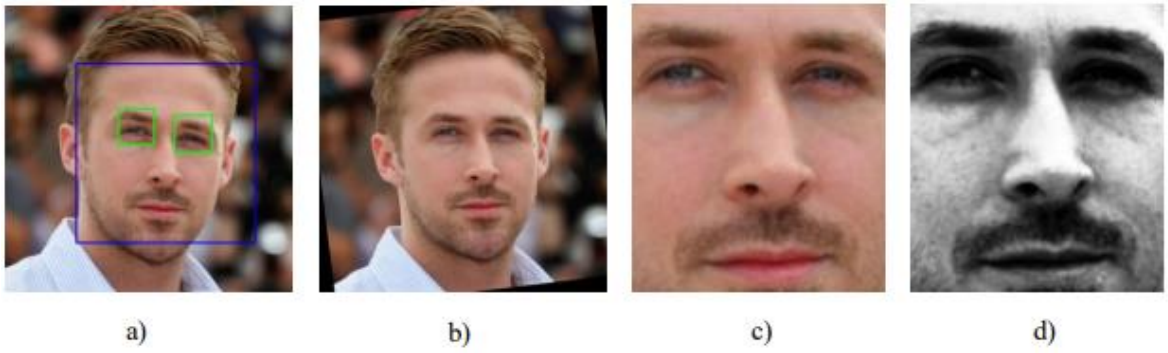
Nota: Reproducida de (Zhang et al., 2015).

1.6.2 Preprocesado.

El preprocesado se lo lleva a cabo con los rostros detectados en la fase anterior, de acuerdo con requerimientos del modelo CNN. En esta fase se realiza una serie de transformaciones a la imagen original con el fin de mejorar su calidad para la extracción de características.

Figura 16

Preprocesamiento de una imagen.



Nota: Adaptado de (Domínguez, 2017).

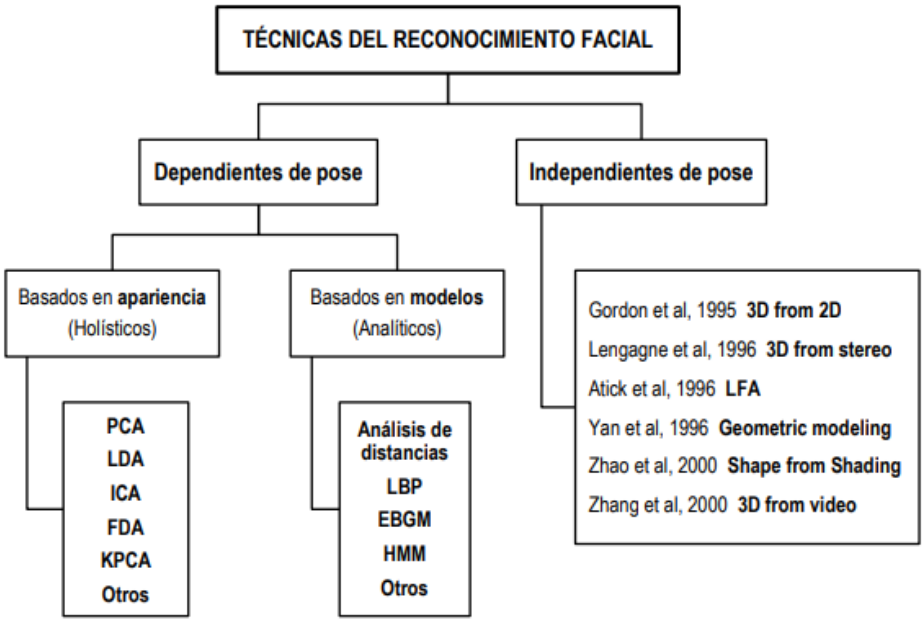
1.6.3 Extracción de Características.

Esta fase tiene como finalidad extraer información de ciertas zonas específicas, que describan la ubicación de un rostro, tales como: ojos, nariz, boca, cejas, mejillas, barbilla, entre otros, los cuales son utilizados en la identificación de la persona. Durante las últimas décadas se han desarrollado varios algoritmos para la extracción de características.

En la Figura 17 se puede observar cómo estos algoritmos están clasificados en dos grupos: las técnicas basadas en apariencia y las técnicas basadas en modelos. Sin embargo, en la actualidad las redes convolucionales profundas han estado ganando popularidad rápidamente, debido a que son más robustas por lo que aprenden características más complejas. Las técnicas independientes de pose no se revisarán en este trabajo.

Figura 17

Técnicas de reconocimiento facial.



Nota: Reproducida de (Domínguez, 2017).

1.6.3.1 Técnicas Basadas en Apariencia.

Las técnicas basadas en apariencia describen el cambio de textura del rostro para representar patrones en función de la información facial para identificar a la persona. Con este método, el reconocimiento facial se transforma en un problema de análisis de espacio, donde se pueden aplicar diferentes técnicas estadísticas para la extracción de características, aunque uno de los problemas es que para la fase de entrenamiento se

requiere un conjunto de muestras considerable para obtener buenos resultados (Domínguez, 2017).

- **EigenFaces:** Eigenfaces es un método que transforma las imágenes en una matriz (vector), la imagen de *n* x *n* píxeles la convierte en un vector de n^2 x *1*, cada vector se normaliza restando la media de todos los vectores "*vector promedio del rostro*"; luego se calcula la matriz de covarianza para obtener *eigenvectors*. Las eigenfaces se forman seleccionando *n* vectores con los valores más altos y calculando los eigenvectors de la matriz de covarianza, estas eigenfaces se combinan para crear un nuevo rostro en el reconocimiento facial (Espinoza, 2015). Finalmente, se utiliza la distancia euclidiana mínima para clasificar; la imagen con la menor distancia se considera el rostro de la imagen de entrada (Chacua, 2019). Se utiliza la descomposición en valores singulares *(SVD)* para reducir la dimensión de la matriz, debido a la necesidad de memoria.
- **FisherFaces:** Es un algoritmo de reconocimiento facial que utiliza el análisis de componentes (PCA) y el análisis discriminante lineal (LDA) para extraer características faciales relevantes que separe mejor entre las clases. Este método considera las imágenes de entrenamiento como clases, por lo que hay la misma cantidad de clases que personas. Una vez que se definen todas las clases, se calculan la matriz de dispersión entre clases y la matriz de dispersión dentro de clases. Se crea una matriz de proyección después de calcular estas matrices, donde cada columna es la base de un nuevo espacio conocido como Fisherfaces (Espinoza, 2015). Esta técnica es mejor que la anterior, sin embargo, resulta computacionalmente más costosa.

1.6.3.2 Técnicas Basadas en Modelos.

Las técnicas basadas en modelos se enfocan en modelar las características faciales empleando un modelo matemático con el fin de identificar a la persona utilizando las matemáticas y la estadística.

- **Local Binary Patterns (LBP):** LBP es un descriptor de textura útil para imágenes en escala de grises que delimita los pixeles cercanos según el valor del pixel actual. La idea principal del método de LBP es que la imagen del rostro puede ser analizada como una composición de pequeños patrones que se comportan de forma invariante respecto a las transformaciones en escala de grises (Jiménez, 2018).

"Para cada pixel en la vecindad determinará la intensidad de ella por medio de dos valores: 1 si el valor es mayor al del pixel central o 0 si el valor es menor al del pixel" (Espinoza, 2015). La técnica LBP es muy útil para el reconocimiento de objetos y la detección de rostros porque se considera invariante a la rotación y a la escala.

1.6.4 Comparación y Decisión.

La última fase desempeña un papel crucial en la identificación de la persona a partir de características faciales extraídas en etapas anteriores. El fin es medir la similitud entre las características faciales del rostro de entrada con las caracteristicas faciales de rostros conocidos que estarían en un conjunto de datos. Se puede emplear dos métodos diferentes para este proceso

1.6.4.1 Medidas de Similitud y Distancia.

- **Distancia Euclídea:** La distancia Euclidiana es una medida matemática antigua basada en el Teorema de Pitágoras, utilizada para calcular la distancia entre dos puntos en un espacio n-dimensional. La distancia euclidiana es la longitud de la línea recta entre dos puntos ($x1$, $y1$) y ($x2$, $y2$).
- **Distancia Bhattacharyya:** La distancia de Bhattacharyya es una proporción estadística que mide la cercanía entre distribuciones de probabilidad. Cuanto menor sea la distancia, más similar será la distribución.

1.6.4.2 Clasificadores.

- **K-Nearest Neighbours:** Este método se basa en encontrar los k vecinos más cercanos al objeto de interés, y clasificarlos en un grupo considerándolo parte de esa clase.
- **Support Vector Machines (SVM):** SVM utiliza modelos de aprendizaje supervisado para resolver problemas complejos de clasificación, regresión y detección de valores atípicos mediante la ejecución de transformaciones que maximiza la distancia entre las instancias de diferentes clases. La idea principal detrás de SVM es encontrar un hiperplano que mejor separe los puntos de datos en sus respectivas

clases, el hiperplano se localiza de tal manera que el mayor margen separa las clases consideradas (Vijay, 2022).

1.7 Herramientas de Desarrollo.

Cuando un carpintero va a construir un mueble necesita de las herramientas apropiadas para lograr obtener un buen producto. De la misma forma sucede en la programación, son necesarios las herramientas de desarrollo apropiadas para lograr construir un buen sistema.

1.7.1 Python.

Python un lenguaje de alto nivel, utiliza una sintaxis limpia y fácil de entender, siendo una buena opción para empezar aprender a programar. Una de las razones por las que muchos desarrolladores prefieren utilizar Python con respecto a otros lenguajes de programación es debido a que posee una amplia biblioteca estándar que abarca algunas áreas como: desarrollo web y móvil, análisis de datos, machine learning, automatización, desarrollo de juegos, etc.

1.7.1.1 Librerías de Python para Machine Learning.

Las librerías de Python son archivos específicos que pueden ser importados al código base para ser utilizado de acuerdo a las necesidades.

- **Numpy:** Es una biblioteca que se utiliza para la manipulación de datos numéricos, cuenta con una gran cantidad de funciones utilizadas en operaciones numéricas complejas con matrices y vectores.
- **TensorFlow:** Se utiliza para construir, entrenar y evaluar modelos complejos y escalables de Machine Learning y Deep Learning tanto en Python, C++, Java, etc.
- **Keras:** Es una librería diseñada para facilitar la creación de aplicaciones basadas en machine learning (Gonzáles, 2022). Keras se ejecuta sobre otros marcos de trabajo de aprendizaje profundo como TensorFlow.
- **Scikit-Learn:** Es una biblioteca lanzada en 2007 que proporciona herramientas eficientes para realizar tareas de aprendizaje automático como: clasificación, regresión, agrupación, reducción de dimensionalidad y más. Está construido sobre otras bibliotecas ampliamente utilizadas como NumPy, SciPy y Matplotlib.

1.7.2 Android Studio

Es el IDE oficial de Google y se ha convertido en la herramienta recomendada para desarrollar aplicaciones móviles para el sistema operativo Android. Android Studio ofrece una variedad de herramientas integradas que simplifican el desarrollo de proyectos de principio a fin en varios lenguajes de programación populares como: Java y Kotlin, Dart, JavaScript, PHP, etc. Facilita la creación de varias versiones de la misma aplicación, ofreciendo múltiples ventajas, como la posibilidad de crear versiones pagadas y gratuitas, así como varios dispositivos o almacenamiento de datos (Quisaguano et al., 2022).

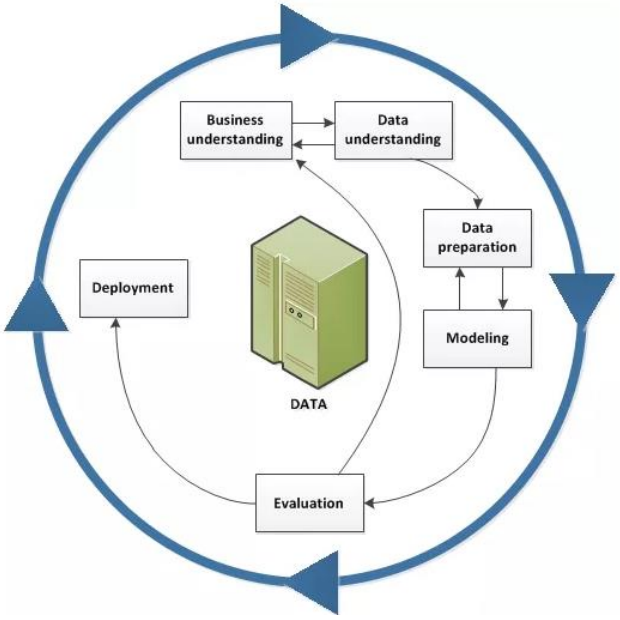
1.8 Metodologías

1.8.1 Metodología CRIPS-DM.

CRISP-DM es un marco que describe las fases para un proyecto de minería de datos o aprendizaje automático. La Figura 18 describe cada una de las fases a seguir.

Figura 18

Ciclo de vida de CRISP-DM.



Nota: Reproducida de (González, 2020).

Este marco está diseñado para ser genérica, completa y estable, por el nivel de detalle con el que se describen las tareas en cada fase, esto da a entender que cubre todo el proceso de desarrollo analítico del proyecto (Barrera & Barrera, 2022). La Tabla 4 describe las fases de la metodología CRISP-DM.

Tabla 4

Fases de la Metodología CRISP-DM.

Fases	Descripción
Comprensión del negocio	Se definen los objetivos y requisitos del proyecto.
Comprensión de los datos	Se recopilan, exploran, evalúan la calidad y la integridad de los datos.
Preparación de datos	Consiste en limpiar los datos y transfórmalos a un formato adecuado.
Modelado	Construcción de modelos.
Evaluación	Se evalúa los resultados obtenidos del modelo.
Implementación	El modelo final se implementa en el entorno de producción.

Nota: Elaboración propia.

1.8.2 Metodologías de Desarrollo de Software.

Una metodología nos de las pautas, procesos y técnicas para llevar un proceso disciplinado, estructurado y organizado en el desarrollo del software. El objetivo de utilizar las metodologías es mejorar la calidad y eficiencia del producto final, cumpliendo con los requisitos planteados al comienzo del proyecto. Se muestran las metodologías más utilizadas en el desarrollo de software.

1.8.2.1 Modelo Espiral o de Riesgos.

El modelo Espiral fue introducido por primera vez en 1986 por Barry Boehm. Este modelo describe el ciclo de vida del software a través de espirales repetitivos que se van incrementando en cada fase. El concepto de riesgo se refiere a la importancia de la gestión proactiva de riesgos para abordar la incertidumbre durante el desarrollo del software, esto permite probar el software antes de que este haya sido terminado completamente facilitando la mejora continua.

1.8.2.2 Modelo en V.

Este modelo fue desarrollado en 1993 por Alan Davis y es una metodología que divide el proceso de desarrollo de software en un enfoque lineal donde sus etapas están organizadas de manera secuencial en forma de una “V”. Este modelo es una variante del modelo en cascada y es utilizado en la verificación y validación de software. Cada fase

tiene una fase de desarrollo que se asocia a una fase de pruebas. Por ejemplo, la fase de definición de requerimientos se asocia con la fase de pruebas de aceptación.

1.8.2.3 Modelo en Cascada.

El modelo en cascada es un enfoque lineal tradicional que tuvo sus inicios en el año 1966 a 1970. El modelo en cascada es un proceso secuencial en el que cada etapa del ciclo de vida de desarrollo de software debe completarse antes de que pueda comenzar la siguiente etapa (Linkedin, 2023b). La principal desventaja de este modelo es que dificulta ver los resultados al finalizar cada fase, lo que dificulta las pruebas, provocando así un retraso en el proyecto. Es un modelo fácil de entender, pero es estricto para utilizarse en posibles cambios en el desarrollo, sin embargo, sigue siendo muy utilizado en la actualidad. El nombre de Cascada es debido a que el progreso se lo debe realizar de manera lineal.

1.8.3 Guía Swebook.

La guía Swebok (Software Engineering Body of Knowledge) es un documento elaborado por la IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) en 2004, que describe los conocimientos y habilidades que se espera que tenga un ingeniero de software. La versión 3.0 de la guía SWEBOK publicado en 2014 está organizada de quince áreas de conocimiento principales: Requisitos de software, Diseño de software, Construcción de software, Pruebas de software, Mantenimiento de software, Gestión de la configuración, Gestión de la Ingeniería de Software, Proceso de Ingeniería de Software, Herramientas y métodos de la Ingeniería de Software, Calidad del software, Práctica profesional de la Ingeniería de Software, Economía de la Ingeniería de Software, Fundamentos de Computación, Fundamentos Matemáticos y Fundamentos de Ingeniería. Cada área de conocimiento cubre una amplia gama de temas y actividades básicas relacionados con la ingeniería de software. Una de las ventajas de este documento es que se actualiza regularmente, reflejando las últimas tendencias y los avances en la disciplina de la ingeniería de software, aunque también existen desventajas y es que es complicado digerir todo el documento por cada actualización (Swebok, 2014).

1.9 Trabajos Relacionados.

En la Universidad del Norte de Colombia, se desarrolló un sistema para el registro de asistencia fundamentado en el procesamiento digital a un grupo de 25 personas, utilizando redes neuronales. Para la extracción de características que se utilizó una CNN tipo Resnet de 27 capas, embebida sobre la librería de Python "*Face_Recognition*". En cambio, para la detección facial utilizaron el descriptor HOG, debido a que este método localiza rostros en diferentes posiciones y cambios de iluminación. Mencionan que su sistema no será capaz de identificar personas con gorras, personas con lentes oscuros u otra obstrucción en el rostro, por ende, es un requisito contar con cámaras de resolución mayor a 352 x 240 pixeles. El sistema recibe dos imágenes por persona, una para la creación del conjunto de datos y otra para la asistencia. Para registrar la asistencia, se suben fotos de las personas en la reunión y el sistema analizará quién estaba presente y quién estaba ausente. Al final, el sistema creará un historial de asistencia. (Caro & López, 2018).

En la actualidad las empresas buscan nuevas soluciones de registro de asistencia, debido a que los mecanismos de asistencia tradicionales son susceptibles a fraudes y errores. En el artículo "*Real Time Face Recognition for Mobile Application Based on Mobilenetv2*" se presenta un sistema móvil para la gestión de asistencia empleado MobileNetV2, además se utiliza técnicas de anti-spoofing basado en puntos de referencia para prevenir el fraude. La detección facial fue llevada a cabo utilizando el marco BlazeFace. MobileNetV2 fue utilizada para procesar, clasificar y estructurar los datos de rostros en una base de datos local. El umbral establecido para comparar los rostros es de 0.8. La aplicación móvil fue construida con el lenguaje de programación Dart usando el marco Flutter. Para registrar un nuevo usuario se guarda el nombre y la imagen del rostro que puede ser capturada con la cámara desde el mismo aplicativo, después los resultados se almacenan en el dispositivo móvil como un archivo JSON. Luego, para el reconocimiento facial se ejecuta el modelo que está almacenado en un servidor en la nube, y este verificara si los rostros corresponden a la misma persona o no (Made et al., 2023).

En el artículo "*CHILD FACE RECOGNITION SYSTEM USING MOBILEFACENET*", se presenta un sistema de reconocimiento facial de niños utilizando la red neuronal convolucional MobileFaceNet. La elección del modelo se debe a que las CNN's profundas enfrentan varias falencias para ejecutarse en

aplicaciones móviles en tiempo real. El algoritmo Dlib se utiliza para detectar rostros y puntos de referencia faciales en las imágenes. Luego utilizan el modelo MobileFaceNet, una red neuronal convolucional liviana y eficiente diseñada específicamente para reconocimiento facial en dispositivos móviles y embebidos, en la extracción de características faciales de imágenes previamente procesadas. Posteriormente, utilizan el algoritmo de clasificación de vecinos más cercanos (KNN), para clasificar a los niños según las distancias de incrustación. Indican que el sistema logró una precisión del 96 % en los conjuntos de datos de rostros de niños (Shun & Aung, 2019).

Según el artículo "*Smart Attendance System Using Face Recognition*", los centros educativos necesitan un sistema robusto para el registro de la asistencia de los estudiantes. Por ende, el sistema propuesto utiliza el algoritmo de reconocimiento facial MobileFaceNet, para identificar a las personas y registrar su asistencia de manera eficiente y precisa en tiempo real. El dataset lo generaron extrayendo fotogramas de imágenes faciales, desde videos individuales de los estudiantes. Para la detección de rostros utilizaron un modelo de detección ultraligero "ultra_light_640.onnx" previamente entrenado, que ofrece una alta velocidad y una precisión aceptable. MobileFaceNet, logra una precisión de hasta el 85% en la identificación de rostros en un conjunto de datos etiquetados y el 90 % en la identificación de rostros en conjuntos de datos dados (Sabeenian et al., 2020).

En el artículo "*Smart Attendance Management System*" se presenta una solución innovadora para el seguimiento de la asistencia en un entorno educativo o laboral, dejando atrás el método tradicional de llamar a las personas por su nombre. La solución propuesta es un Sistema de Gestión de Asistencia Inteligente (SAMS en inglés), que utiliza tecnologías como el reconocimiento facial y la huella dactilar para registrar la asistencia de los estudiantes mediante listado automático. Un administrador del sistema realiza la inscripción de un nuevo usuario utilizando: la identificación; la imagen del rostro y las huellas digitales del usuario. El sistema utiliza la red de aprendizaje profundo MobileFaceNets para la extracción de características en el reconocimiento facial en tiempo real. Puede también utilizarse el lector de huellas para comparar las huellas del individuo con las registradas en la base de datos. Como resultado el sistema ha ayudado a aumentar la precisión y la velocidad en el registro de una asistencia en tiempo real (Albahrani, 2022).

Capítulo II

Desarrollo Experimental.

El presente capítulo desglosa el desarrollo del proyecto, aplicando la metodología CRISP-DM para obtener el modelo CNN de reconocimiento facial y la metodología en Cascada en el desarrollo de la aplicación móvil.

2.1 Requerimientos del Sistema.

Requerimientos del sistema se refiere a las especificaciones técnicas mínimas que un sistema informático debe tener y realizar durante su operación para lograr el objetivo planteado, cumpliendo con las expectativas de las partes interesadas. Estos requerimientos están vinculados con componentes de hardware, software, así como otro tipo de requerimientos, que definen la calidad del producto. Una buena elaboración de los requerimientos establece una ejecución exitosa del Software. En esta sección se establecen las bases sólidas, para la planificación, el posterior diseño y desarrollo de las funciones del sistema de reconocimiento facial. Estas especificaciones ayudarán a limitar el alcance del proyecto, así como, estimar el tiempo, los recursos y los costos, para de esta manera mitigar posibles riesgos en el desarrollo.

La Tabla 5 se ha creado teniendo en cuenta las consideraciones de la metodología en Cascada e incluyen: los requerimientos más importantes del sistema, los requerimientos de arquitectura y los requerimientos de las partes interesadas. La finalidad es presentar la información de modo que sea fácil de entender, permitiendo realizar una adecuada selección del software, hardware y algunos elementos específicos para el posterior diseño del sistema. Dicha Tabla muestra los acrónimos utilizados para describir brevemente cada requerimiento.

Tabla 5

Definición de Acrónimos

Acrónimo	Descripción
RFs	Requerimientos Funcionales
RNFs	Requerimientos no Funcionales
SRSH	Requerimientos de Arquitectura
StSR	Requerimientos de Stakeholder

Nota: Elaboración propia.

Las Tablas que describen los requerimientos del sistema deben tener una columna con el número de requerimiento, una columna con una descripción detallada del requerimiento y una columna con la prioridad del requerimiento. La prioridad es crucial, debido a que establece la importancia de un requerimiento con respecto a los demás; al ser el requerimiento más crítico debe ser el primero que debe desarrollarse. La Tabla 6 muestra los niveles de prioridad establecidos para definir los requerimientos del sistema. En el caso de que un requerimiento sea totalmente dependiente de otro, también se incluye una columna de relación.

Tabla 6

Prioridad de los Requerimientos del Sistema.

Prioridad	Descripción
Alta	Es un requisito esencial que debe ser considerado durante el proceso de desarrollo del sistema. Si no se implementa, puede afectar la funcionalidad.
Media	La decisión final del sistema puede verse afectada por la ausencia de este tipo de requerimiento, pero se puede omitir en situaciones de fuerza mayor.
Baja	Este requerimiento no debería tener un impacto significativo en la decisión final del sistema.

Nota: Adaptado de (Chacua, 2019) y (Portilla, 2018).

2.1.1 Requerimientos Funcionales.

Los requerimientos funcionales definen las diversas tareas y procesos que el sistema debe ser capaz de realizar. Representan las características y funcionalidades particulares que deben estar presentes en el sistema, para que cumplan con los objetivos establecidos y satisfaga las necesidades del usuario final. Los requerimientos funcionales describen que debe hacer y cómo se comportará el sistema en situaciones de interactuar con el usuario final. Por esta razón es crucial definir correctamente estos requerimientos, debido a que ayuda a diseñar soluciones, así como evitar posibles riesgos. Para garantizar el éxito de un proyecto, los requerimientos funcionales son esenciales, debido a que proporcionan una base clara y sólida sobre la cual se debe construir y evaluar el sistema o producto final. Los requerimientos funcionales más importantes del sistema se muestran listados en la Tabla 7.

Tabla 7

Requerimientos Funcionales del Sistema.

Requerimientos Funcionales del Sistema					
#	**Requerimiento**	**Prioridad**			**Relación**
		Alta	Media	Baja	
RF1	Autenticación al sistema mediante un Login	X			
RF1	Validar las credenciales del Usuario	X			
RF1	Posibilidad de modificar el Umbral para el cálculo de similitud facial		X		
RF2	Capturar la zona ROI del rostro de los estudiantes utilizado la cámara principal del dispositivo móvil	X			
RF3	Pintar la zona ROI del rostro en el dispositivo	X			
RF4	Recortar y procesar el rostro detectado	X			
RF5	Cálculo de similitud de las características del rostro predicho con las de la base de datos	X			
RF6	Pintar el nombre del estudiante identificado en el dispositivo.	X			
RF7	Posibilidad de agregar un nuevo estudiante al sistema de Reconocimiento Facial.	X			
RF8	Mostrar una lista de los estudiantes reconocidos.	X			
RF9	Generar un reporte en formato Excel.	X			

Nota: Elaboración propia.

2.1.2 Requerimientos no Funcionales.

Los requerimientos no funcionales, también conocidos como atributos de calidad del sistema a diferencia de los requerimientos funcionales, describen los criterios, las cualidades y se enfocan en cómo debe funcionar el sistema en lugar de qué debe hacer. Estos requerimientos impactan directamente la experiencia del usuario final. Los requerimientos no funcionales influyen directamente en la facilidad de uso, rendimiento eficiencia, escalabilidad, seguridad, etc, del sistema, para así garantizar que este cumpla con los objetivos establecidos. Sin embargo, estos requerimientos no son tan fáciles de medir y verificar como los requerimientos funcionales. La Tabla 8 describe los requerimientos no funcionales que se pretende que el sistema los cumpla.

Tabla 8

Requerimientos no Funcionales del Sistema.

Requerimientos no Funcionales del Sistema					
#	Requerimiento	Prioridad			Relación
		Alta	Media	Baja	
RNF1	La aplicación debe ser intuitiva y fácil de usar, con una interfaz sencilla.	X			
RNF2	La aplicación debe ser capaz de ejecutarse de manera eficiente y en tiempo real.	X			RF1
RNF3	La aplicación debe tener una alta precisión.	X			
RNF4	La aplicación debe proteger los datos de los usuarios.	X			

Nota: Elaboración propia.

2.1.3 Requerimientos de Arquitectura.

Los requerimientos de arquitectura son una parte crucial del proceso de desarrollo de software, debido a que definen el diseño y la organización general del sistema; facilitando la selección del hardware y software a emplearse en el proyecto.

Tabla 9

Requerimientos de Arquitectura del Sistema.

Requerimientos de Arquitectura del Sistema					
#	Requerimiento	Prioridad			Relación
		Alta	Media	Baja	
Requerimiento de Hardware					
SRSH1	Cámara de buena resolución para la extracción de imágenes.	X			
SRSH2	RAM mínima del dispositivo móvil: 4 GB o superior.	X			
SRSH3	Cámara principal de 48 MP o superior del dispositivo móvil.	X			
SRSH4	Duración de batería del dispositivo móvil.	X			
SRSH5	Procesador del dispositivo móvil que ofrezca una buena potencia de 2.4 GHz o superior	X			

Requerimiento de Software		
SRSH6	El sistema operativo y el lenguaje de programación deben ser de código abierto.	X
SRSH7	Se requiere un software que permita ejecutar código Python en un servidor de la nube	X
SRSH8	Se requiere un software que permita usar GPU para el procesamiento de código	X
SRSH9	Sistema operativo Android, mínimo: Android 11.0 o superior	X
SRSH 10	El lenguaje de programación Java debe ser compatible con las bibliotecas de aprendizaje automático.	X

Nota: Elaboración propia.

2.1.4 Requerimientos del Stakeholder.

Los interesados son un grupo de personas que tienen un interés directo en el resultado obtenido por el desarrollo del proyecto. El objetivo de la definición de los requisitos de los stakeholder es determinar las necesidades de los interesados por el sistema. La Tabla 10 menciona a los implicados en el desarrollo de este proyecto.

Tabla 10

Lista de stakeholder.

#	Lista de Stakeholders
1	Estudiantes y docentes de la FICA
2	MsC. Marco Pusdá (Director del Trabajo de Titulación)
3	MsC. MacArtur Ortega (CoDirector del Trabajo de Titulación)
4	Edison Guaichico (Desarrollador del proyecto)

Nota: Elaboración propia.

Posteriormente, la Tabla 11 muestra las necesidades y restricciones de los Stakeholders con respecto al sistema.

Tabla 11

Requerimientos de Stakeholders.

Requerimientos de Stakeholders					
#	Requerimiento	Prioridad			Relación
		Alta	Media	Baja	
Requerimientos Operacionales					
StSR1	El sistema debe implementarse en una aplicación móvil.	X			
StSR2	La aplicación móvil debe ser capaz de funcionar sin conexión a la red	X			
Requerimientos de Usuario					
StSR3	El sistema debe contar con mecanismos claros de notificación de cualquier evento.	X			

Nota: Elaboración propia.

2.2 Introducción al Desarrollo del Proyecto.

Los principios fundamentales que guiaron el desarrollo de este sistema se presentan en esta sección. Se destaca su propósito y se enfatiza en los aspectos más relevantes e importantes, los beneficiarios a los que se dirige y los objetivos que se buscan alcanzar a través de su implementación.

2.2.1 Propósito del Sistema.

El propósito principal de este proyecto fue proporcionar una solución adecuada al problema del alto consumo de tiempo en el registro de asistencia, utilizando técnicas de Inteligencia Artificial. Los métodos tradicionales de control de asistencia como, nombrar a cada individuo uno por uno, requieren de bastante tiempo y son propensos a errores y a la falta de interés de los estudiantes.

El sistema propuesto es una aplicación móvil para el registro de asistencia de los estudiantes mediante la utilización del reconocimiento facial. La aplicación debe generar un reporte de los estudiantes identificados; posteriormente ese reporte podrá ser utilizado por el docente para registrar la asistencia automáticamente en el portafolio académico de la Universidad. Esta tecnología podría cambiar la forma en que se realiza esta actividad, proporcionando una solución eficiente a los métodos tradicionales utilizados actualmente.

El fin del proyecto fue evaluar la eficacia de la aplicación móvil en el registro de asistencia en comparación con los métodos convencionales. Para así corroborar que este método puede ser una solución viable o considerarlo como una alternativa al método tradicional.

2.2.2 Beneficiarios.

Los beneficiarios del sistema comprenden a las siguientes personas.

- **Docentes:** Son quienes registran la asistencia de los estudiantes y se beneficiarán de la aplicación porque ayudará a reducir el tiempo, que puede dedicarlas a otras actividades.
- **Estudiantes**: Son los principales actores en este proyecto.

Se espera que, en el futuro, todas las facultades de la Universidad Técnica del Norte sean beneficiarios del proyecto.

2.2.3 Selección de Hardware y Software.

La correcta selección del hardware y software se ha convertido en un factor importante para lograr el éxito en el desarrollo del producto tecnológico, dado que estas decisiones pueden afectar a la compactibilidad y rendimiento de los sistemas. Debido a la rápida evolución de la tecnología, existen una amplia variedad de opciones a escoger.

En este trabajo, para la selección del hardware y software se consideró, primero que sean los recursos óptimos, capacidad de las tecnologías, así como la facilidad de acceso y uso.

2.2.3.1 Hardware.

La selección de Hardware se realizó teniendo en cuenta la Tabla 9, sobre los Requerimientos de Arquitectura.

2.2.3.1.1 Cámara para la Extracción de Imágenes.

En este proyecto se utilizó la cámara Canon Rebel EOS T5 para grabar videos de cada estudiante. Dicha cámara permite configurar la apertura del diafragma, controlar la velocidad de obturación y muchas otras facilidades. Esta cámara produce imágenes y videos de alta calidad al permitir el control manual sobre las fotografías y los videos. Los detalles de la cámara se muestran en la Tabla 12.

Tabla 12

Especificaciones de la cámara Canon Rebel EOS T5.

Especificaciones	Propiedades
Modelo	Canon Rebel EOS T5
Pantalla	TFT
Resolución	18 MXP
Lente	18-55+75-300
Peso	435 g
Corrección de exposición a la luz	+-5EV(1/2; 1/3 EV step)
Procesador de imagen	DIGIC 4

Nota: Elaboración propia.

2.2.3.1.2 Dispositivo Móvil.

La selección del dispositivo móvil fue de suma importancia para lograr resolver el problema planteado en este proyecto. Esto se debe a que la aplicación móvil es exigente y requiere de un buen dispositivo para poder ejecutarse correctamente. Los requerimientos para seleccionar el dispositivo móvil fueron que tenga una excelente resolución de cámara y buena capacidad de RAM. Actualmente en el mercado existen varios dispositivos de gama media que cumplen con los requerimientos que la aplicación de reconocimiento facial establece. Además, existen dispositivos móviles que incluso ya cuentan con un procesador para ejecutar más rápido las tareas. Las especificaciones del dispositivo móvil seleccionado se describen en la Tabla 13.

Tabla 13

Especificaciones del Dispositivo Móvil.

Especificaciones	Propiedades
Nombre	HONOR 90
Modelo	REA-NX9
Cámara	Principal: 200 megapíxeles, f/1.9 Gran angular: 12 megapíxeles, f/2.2 Profundidad: 2 megapíxeles, f/2.4
Núm. **Compilación**	7.1.0.184(C60E5ER3P1) GPU Turbo

Versión Android	13
Procesador	Qualcomm Snapdragon 7 Gen 1
RAM	8.0GB+5.0 GB (HONOR RAM TURBO)
Resolución	2664 x 1200
Versión de banda base	00049,00049

Nota: Elaboración propia.

2.2.3.2 Software.

La selección del Software también se realizó teniendo en cuenta la Tabla 9, sobre los Requerimientos de Arquitectura.

2.2.3.2.1 Herramientas de Desarrollo.

La selección del Software en el desarrollo de un sistema informático es fundamental para garantizar la calidad, la seguridad y el buen rendimiento del sistema.

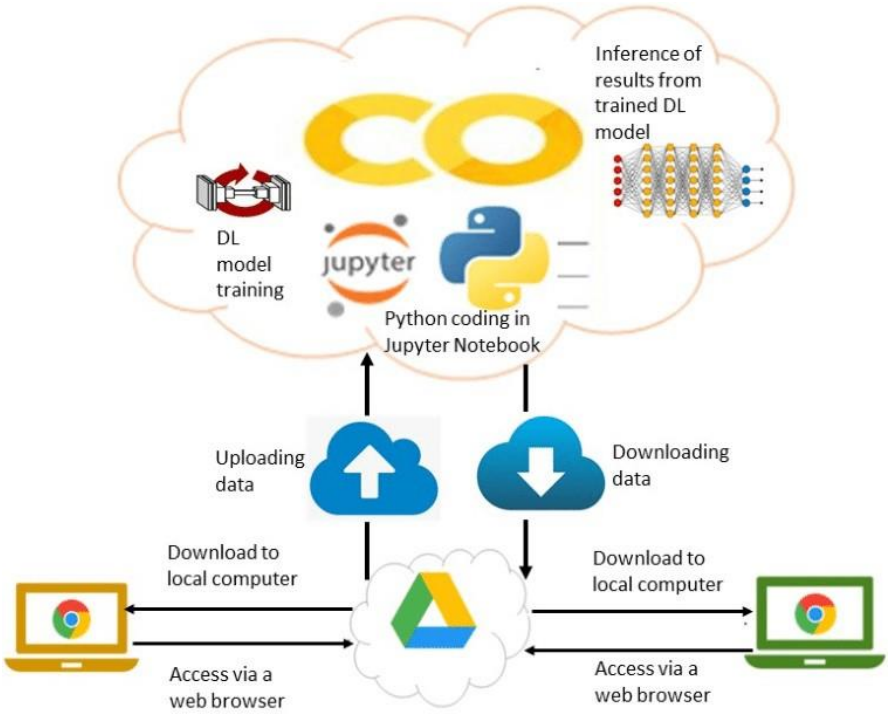
- *Google Colab / Python.*

Google Colab es una plataforma gratuita en la nube de Google que permite la creación y ejecución de código Python de manera sencilla y colaborativa. Su uso es bastante simple, únicamente es necesario una cuenta de Gmail y se tendrá acceso a dicha plataforma. Aunque la versión gratuita de esta plataforma tiene recursos limitados, existe un plan de pago que ofrece unidades de procesamiento más robustos, como GPU y TPU. Una de razones principales para la selección de dicha plataforma, fue su fácil integración a Google Drive.

Python es un lenguaje de programación conocido por su sencillez y legibilidad para entender y escribir código de alto nivel. Es considerado uno de los mejores lenguajes de programación. Este lenguaje es utilizado para programar en diferentes campos como: desarrollo web y móvil; análisis de datos; aprendizaje automático; seguridad informática; etc, gracias a su extensa librería. Además, al ser un lenguaje de programación interpretado varios servidores como: Google, Amazon, Microsoft Azure, Kaggle, etc, presentan entornos donde se puede utilizar fácilmente este lenguaje sin instalar nada por debajo, lo que lo convierte en un lenguaje bastante accesible.

Figura 19

Python para Machine Learning en Google Colab.



Nota: Reproducida de (Ekanayake, 2022).

- *Android Studio / Java / CameraX.*

Como el IDE oficial de Google, Android Studio es el mejor entorno para desarrollar aplicaciones de Android. Este entorno facilita la creación de emuladores (dispositivos móviles virtuales) de manera rápida y sencilla, lo que lo convierte en un IDE bastante adecuado para este proyecto.

Por otra parte, la elección de Java se debe a su robustez como lenguaje de programación, además de que cuenta con una gran base de comunidad activa y amplios recursos en línea que se pueden utilizar. En Android Studio es posible utilizar Java o Kotlin para desarrollar aplicaciones móviles. Sin embargo, una de las razones por la que se decidió usar Java como lenguaje de programación, fue por tener más experiencia en él.

CameraX es una biblioteca Jetpack de Google, utilizada para administrar y manejar la funcionalidad de la cámara en las aplicaciones de Android mediante una API simplificada basada en la API de Camera2. CameraX reduce significativamente la cantidad de código necesario para implementar las funcionalidades de la cámara en el desarrollo de aplicaciones en Android, al proporcionar una interfaz unificada y abstraer las complejidades asociadas (Zeeshan, 2020).

Figura 20

CameraX en Android.

Nota: Reproducida de (Zeeshan, 2020).

- *ScreenStream.*

ScreenStream es una aplicación móvil gratuita disponible en Google Play que permite la transmisión de la pantalla de un dispositivo Android a una computadora u otro dispositivo móvil de manera sencilla. La aplicación genera una dirección Ip local que se puede pegar en un navegador, y este comienza automáticamente la transmisión en tiempo real.

Figura 21

Transmisión de pantalla por Screen Stream.

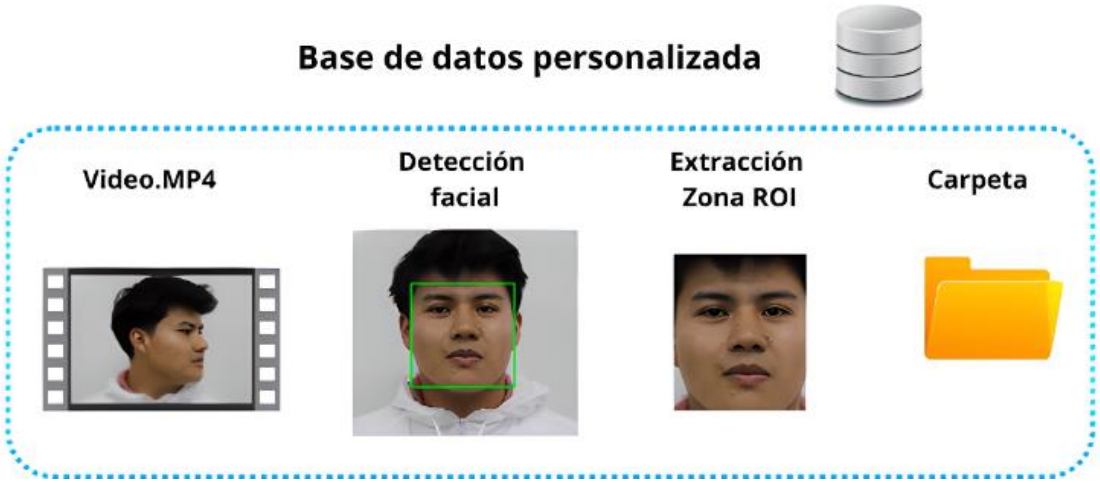
Nota: Reproducida de (alfanoTV, 2020).

2.3 Elaboración del Dataset

La elaboración del dataset se considera unos de los procesos más importantes dentro del desarrollo de modelos de Machine Learning y Deep Learning. Un dataset o conjunto de datos no es más que una colección de muestras (imágenes, texto, audio u otros tipos de información), los cuales se utilizan para entrenar, validar y probar un modelo CNN. Hay que tener en cuenta que es fundamental crear un dataset equilibrado, es decir, contar con cantidades similares de muestras para cada clase. Caso contrario el modelo producirá sesgos y no predecirá correctamente. La Figura 22 presenta un diagrama donde se puede observar la etapa de elaboración de los datos.

Figura 22

Diagrama del proceso de Elaboración del Dataset.



Nota: Elaboración propia.

2.3.1 Recolección de Video.

Se consideró varios métodos para la elaboración del dataset de imágenes. La primera opción planteada fue tomar 100 fotografías individuales por persona, sin embargo, cada fotografía debía ser tomado por separado, lo que requeriría mucho esfuerzo y tiempo. La segunda opción fue grabar un video de 25 segundos que muestre a la persona en varias posiciones. Esta idea fue más efectiva debido a que se podrían obtener múltiples fotogramas (imágenes) de un solo video, lo que permitiría crear una base de datos grande en menos tiempo.

Después de analizar las opciones antes mencionadas se optó por utilizar la segunda opción, en donde se siguió los siguientes pasos:

- **Preparación:** Es importante tener una iluminación adecuada y un fondo neutral, de manera que el video sea claro y consistente.
- **Posiciones y movimientos:** Planificar las posiciones y movimientos que se desea capturar en los videos como: expresiones faciales variadas y movimientos como inclinación de lado a lado de la cabeza.
- **Grabación de los videos:** Para la grabación de los videos se utilizó la cámara especificada en la Tabla 12. El proceso consistió en grabar videos de aproximadamente 25 segundos, asegurándose de capturar las diferentes posiciones y movimientos previamente planificados.

2.3.2 Construcción del Dataset de Imágenes.

Después de recopilar los videos de los estudiantes quienes serán los protagonistas en este proyecto de la aplicación de reconocimiento facial, el siguiente paso es adquirir imágenes de rostros a partir de los videos recopilados, para después etiquetarlos adecuadamente.

El etiquetado es un proceso que consiste en asignar categorías a los datos. Cada dato individual (imagen de rostro), se asocia a una clase o grupo dependiendo de su contenido, lo que permite el aprendizaje supervisado y la evaluación del rendimiento del modelo CNN. Además, las etiquetas mejoran la efectividad de los algoritmos en la identificación y clasificación de patrones visuales, lo que es esencial para interpretar resultados y garantizar que los modelos sean adecuados para una variedad de contextos. Este proceso es una labor un poco tediosa, por ende, para obtener el correcto proceso de etiquetado aún es necesario la supervisión humana.

2.3.2.1 Detección Facial.

Por cada fotograma (imágenes individuales en secuencia) del video, se identificó la zona ROI (*Región de Interés*) utilizando un modelo de detección facial. Python ofrece múltiples librerías para realizar esta actividad, tales como: Haar Cascade, DLib, Face_Recognition, etc. Sin embargo, estos modelos de detección facial identificaban algunos falsos positivos, por lo cual no fueron tomados en cuenta. Por esa razón se implementó MTCNN, un modelo de detección facial bastante robusto que utiliza una CNN profunda compuesto por tres etapas P-Net, R-Net y O-Net para mejorar la detección, en la sección 1.6.1.2 se explica más a detalle su arquitectura. Además de detectar la región facial, MTCNN también es capaz de identificar cinco puntos faciales

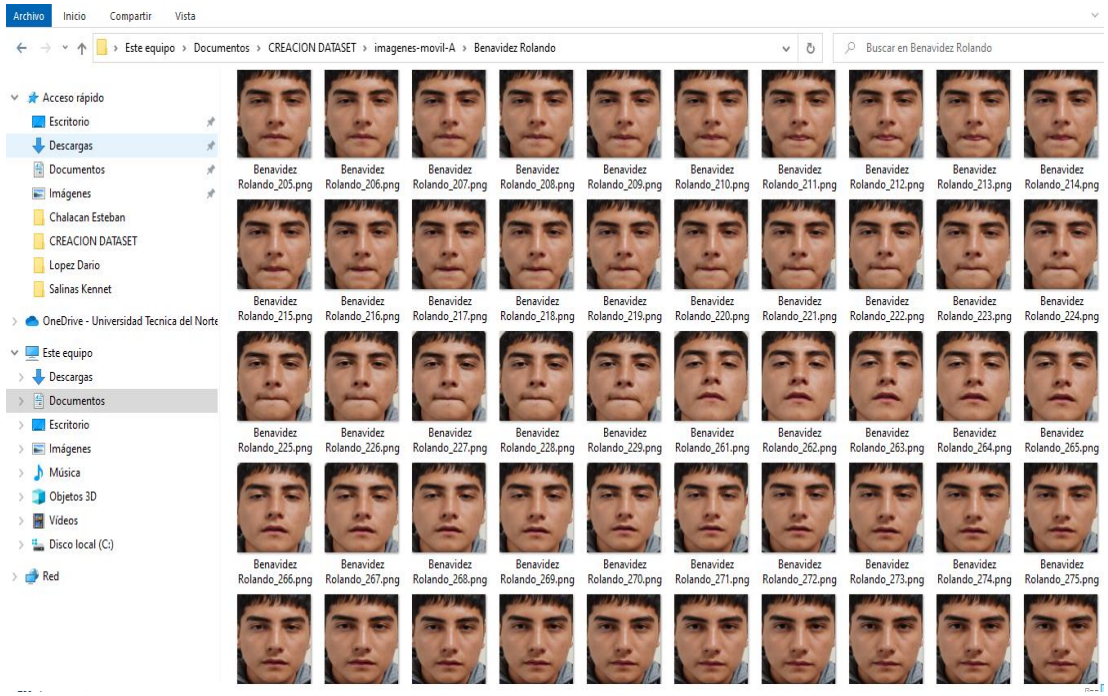
(landmarks faciales); estos son puntos anatómicos claves en el rostro tales como: ojos, nariz y boca. Este detector es accesible fácilmente desde Python y permitió detectar con gran precisión los rostros en el video. Por cada fotograma el modelo de detección facial devuelve las coordenadas del boundingBox que rodea el rostro, prediciendo que es una cara.

Con las coordenadas del rostro detectado, se procedió a recortar y a redimensionar la zona ROI, de acuerdo a los requerimientos del modelo CNN. Luego la imagen del rostro fue guardada en una capeta en formato “PNG”, esto se debe a que: “PNG utiliza un algoritmo de compresión sin pérdidas que tiene la capacidad de almacenar hasta 8 bits de información adicionales en cada pixel (transparencia) a diferencia del formato JPG” (Idento, 2015, como se citó en Chacua, 2019), lo que podría mejorar ligeramente la precisión con respecto a otros formatos como: JPE, JPEG, etc.

Las carpetas (clases) se crearon con el nombre del video (nombre de estudiante), dentro de dichas carpetas se guardaron las imágenes recortadas. El proceso se ejecutó hasta que se cumpla con la condición de extracción de 750 imágenes por cada clase.

Figura 23

Base de datos de imágenes en carpeta.



Nota: Elaboración propia.

La ejecución del programa continuó hasta terminar con todos los videos de los estudiantes. De esta forma fue que se logró generar el dataset personalizado correctamente etiquetadas. Aunque, posteriormente se tuvo que eliminar manualmente algunas imágenes de mala calidad, debido a que, al ser imágenes obtenidas desde un video, hubo algunas imágenes que contenían bastante ruido por lo cual fueron desechadas. En la Figura 24 se muestra una pequeña porción de código utilizado para realizar esa actividad.

Finalmente, luego de ya tener listo el dataset de imágenes correctamente etiquetas, se procedió a subir toda la carpeta a Google Drive, pues como se mencionó anteriormente, Google Colab fue la plataforma seleccionada para utilizar Python y realizar cualquier proceso con el modelo CNN, debido a su facilidad de uso.

Figura 24

Script "align_data.py" para la Elaboración del Dataset.

```
videos_path = path_base+"\\Estudiantes"    #Ruta de los videos por cada estudiante.

for index, video in enumerate(os.listdir(videos_path)):    #Lectura del directorio de los videos.
    file_name, extension = os.path.splitext(video)         #Separación del nombre y la extension del archivo.
    person_path = os.path.join(path_base, "fatltas", file_name) #Concatenación para crear la clase o grupo.

    if not os.path.exists(person_path):  #Condición que verifica si existe o no un directorio con ese nombre.
        print('Carpeta creada:', person_path)
        os.makedirs(person_path)         # Creación del nuevo directorio.

    cap = cv2.VideoCapture(os.path.join(videos_path, video)) #Lectura de los videos.

    detector = MTCNN()         # Cargar el detector desde MTCNN.
    countImg = 0 #Variable utilizado para nombrar las imágenes de los rostros.

    window_size = (800, 600)    # Configurar el tamaño del frame


    while True:
        ret, frame = cap.read()                  #Captura del frame en el video.

        if ret == False: break
        frame = imutils.resize(frame, width=640)       #Redimensiona la imagen.
        auxFrame = frame.copy()                        #Crea una copia de la imagen.
        faces = detector.detect_faces(frame)           #Rostros detectados con el Detector DLIB.

        for face in faces:                             #Recorrer todos los rostros detectados.
            x, y, w, h = face['box'] #           #Extrae las coordenadas y dimensiones de la cara detectada
            rostro = auxFrame[y:y + h, x:x + w]        #Asigna a la variable rostro una submatriz de la imagen.
            rostro = cv2.resize(rostro, (112, 112))
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

            # Guardar imagen recortada (rostro)
            cv2.imwrite(person_path + '/{}_{}.png'.format(file_name,  countImg), rostro)
            countImg += 1
```

Nota: Elaboración propia.

2.4 Modelo Extractor de Características

Esta sección es de suma importancia en el desarrollo del proyecto, debido a que especifica el modelo CNN que se utilizó como extractor de características faciales.

El reconocimiento facial puede considerase como un problema de clasificación, si se tiene una base de datos con imágenes etiquetadas y se trata de identificar a qué clase (a qué persona) pertenece un rostro desconocido. Afortunadamente, hay CNN's previamente entrenadas que se pueden utilizar para resolver dicho problema, tales como: MobileNet, VGGNet, Resnet, EfficientNet, Inception Resnet, etc. Para lograrlo, se debe configurar las capas más altas de la red, ajustar los parámetros de acuerdo con las necesidades y entrenarlo con el conjunto de datos personalizado, de esta manera el modelo será capaz de clasificar entre las clases. Sin embargo, este método puede tener ciertas desventajas como: ¿Qué ocurre si se desea que el modelo reconozca a una nueva persona con la que no ha sido entrenado?; en ese caso, se tendría que entrenar nuevamente el modelo con las imágenes de la nueva persona, consumiendo más tiempo. En algunas situaciones el método de clasificación puede ser la mejor solución para resolver el problema, no obstante, en este proyecto, este enfoque no es el método más viable, debido a que se quiere dejar la posibilidad de agregar un nuevo individuo al sistema.

En general, con el sistema de reconocimiento facial, lo que se pretende es encontrar una coincidencia entre una entrada desconocida y un conjunto de entradas conocidas previamente almacenadas. Es decir, comparar la similitud entre las representaciones vectoriales de las características faciales de diferentes personas utilizando alguna función de similitud. Ventajosamente existen algunos modelos que pueden utilizarse como extractor de características faciales tales como: Facenet, MobileNetV2, etc. Sin embargo, se cae en la situación de que el modelo debe ser implementada en un dispositivo móvil, revisar sección 1.4.4 donde se explica sobre las limitaciones del entorno móvil. Ambos modelos pueden ser utilizados para el proceso de reconocimiento facial, no obstante, Facenet es una red CNN bastante profunda, por lo que dificultaría la eficiencia de la aplicación móvil. En cambio, MobileNetV2, aunque es un modelo específico para dispositivos móviles, no es un modelo destinado para abordar problemas relacionados con el reconocimiento facial, pero puede ser adaptado tal como lo hicieron (Made et al., 2023).

Afortunadamente existe un modelo CNN "*MobileFaceNet"*, que combina las ventajas de ambos modelos mencionados anteriormente. Este modelo es super ligero y ha demostrado tener una precisión bastante buena.

Es así como, luego de haber analizado las ventajas que ofrece, evaluado las limitaciones del entorno móvil descritos en la seccion 1.4.4 y cumpliendo los requerimientos no funcionales establecidos en la Tabla 8; se ha decido utilizar MobileFaceNet como modelo extractor de características faciales para la aplicación móvil de reconocimiento facial.

2.4.1 MobileFaceNet.

MobileFaceNet es una red neuronal profunda, utilizada ampliamente como modelo extractor de características faciales en dispositivos móviles y embebidos; debido a que es una red bastante ligera y su precisión se considera similar a los mejores modelos actuales que realizan la misma tarea.

El modelo fue presentado en el artículo "*MobileFaceNets: Efficient CNNs for Accurate RealTime Face Verification on Mobile Devices*", por (S. Chen et al., 2018); como un método para superar las deficiencias de otros modelos presentados años anteriores. MobileFaceNet utiliza menos de 1 millón de parámetros con apenas 4,0 MB de tamaño, y supera en precisión y velocidad de ejecución en tiempo real a MobileNetV2, bajo las mismas condiciones experimentales. El autor menciona que la red principal de MobileFaceNet (112 x 112) alcanzó una precisión del 99,55 % en LFW, superando a MobilenetV2 de (Sandler et al., 2018) que alcanzó 98.58% y casi igualando a Facenet de (Schroff et al., 2015) con 99.63%, en el mismo dataset de imágenes.

La dimensión del vector de características faciales (embedding) que devuelve MobileFaceNet es de 192. En el vector se extraen un conjunto de incrustaciones faciales únicas de cada individuo para el reconocimiento facial, revisar sección 1.6.3. Posteriormente, las características faciales obtenidas se comparan y determinan la identidad de la persona, revisar sección 1.6.4 donde se describen los métodos de comparación. MobileFaceNet también identifica otros rasgos faciales de alto nivel a partir de una imagen completa del rostro, es así como, el vector embedding de 192 valores es una representación compacta de dichas características de alto nivel.

Luego de conocer un poco sobre MobileFaceNet en la siguiente sección se explica la arquitectura de este modelo,

2.4.1.1 Arquitectura de MobileFaceNet.

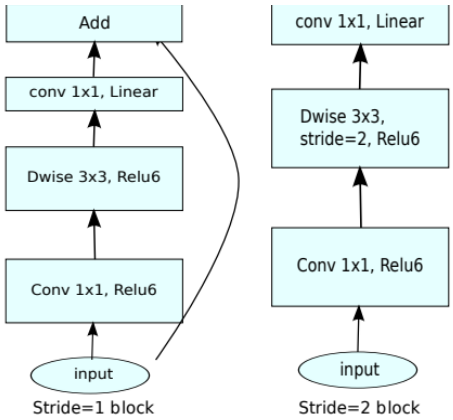
MobileFaceNet se basa en la utilización de cuellos de botella residuales invertidos que tiene la arquitectura MobileNetV2. Los cuellos de botella utilizan bloques de convolución separable en profundidad que ayudan a reducir la dimensionalidad, los parámetros y la complejidad de la red. MobileNetV2 se define como el esqueleto sobre el cual está construido MobileFaceNet. Para mejorar la comprensión, se proporciona una explicación de la estructura de MobileNetV2.

2.4.1.1.1 MobileNetV2

En la Figura 25 se puede observar que son principalmente dos bloques residuales invertidos con una combinación de niveles de cuellos de botella que conforman la arquitectura de MobileNetV2. El primer bloque en la arquitectura es el bloque de cuello de botella con zancada (stride) 1 y otro con zancada (stride) 2, donde para cada bloque existe 3 capas de convolución. La zancada 1 se emplea en las primeras capas de la red para capturar características de bajo y medio nivel de la imagen de entrada, sin cambiar mucho la dimensionalidad espacial en las representaciones intermedias. En cambio, la zancada 2 ayuda a reducir a la mitad la dimensionalidad espacial de la entrada, debido a que el filtro se mueve de dos en dos en cada paso. Estos bloques residuales invertidos con cuellos de botella lineales son configurados sobre MobileNetV1 (Kumar & Bansal, 2023). Dicho método de cálculo es útil para aprender características en diferentes niveles, es así como, se reduce la cantidad de cálculos innecesarios en las capas posteriores de la red.

Figura 25

Arquitectura de MobileNetV2.



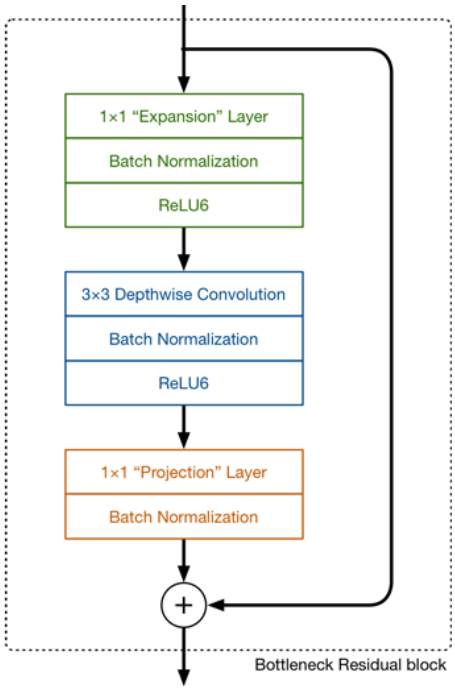
Nota: Reproducida de (Sandler et al., 2018).

Bloque Residual Invertido.

Los bloques residuales invertidos son una mejora a los bloques residuales tradicionales. Este es el bloque más importarte de la arquitectura MobileNetV2. Los bloques residuales invertidos se caracterizan por tener una convolución para reducción de tamaño en la primera capa, seguida de una convolución de expansión en la siguiente capa. Estos bloques ayudan a reducir tamaño de la imagen sin perder demasiado detalle, evitando así, el problema de la desaparición del gradiente durante el entrenamiento. Los bloques residuales invertidos son una adaptación de las conexiones residuales de la arquitectura Resnet. La inversión hace referencia al hecho de que la expansión y la contracción de las dimensiones espaciales (ancho y alto) se invierten con respecto a un bloque residual tradicional. En la Figura 26 se puede visualizar las capas más importantes que interfieren en un bloque residual con cuello de botella.

Figura 26

Bloque Residual de Cuello de Botella.



Nota: Reproducida de (Hollemansby, 2018).

Como se observó en la Figura 25, cada bloque residual tiene una capa de normalización por lotes y una función de activación (ReLU6), excepto la capa de proyección. La capa de proyección solo tiene normalización por lotes debido a que la salida es de baja dimensión.

- *Capa de expansión convolucional inicial de tamaño 1x1 con activación ReLU6.*

Su propósito es expandir la cantidad de canales en los datos antes de que entren en la convolución profunda, mediante una operación de convolución 1x1. La capa de expansión siempre tiene más canales de salida que canales de entrada; hace prácticamente lo opuesto a la capa de proyección (Hollemansby, 2018). Esta acción se realiza aplicando el factor de expansión predeterminado 6.

- *Capa convolucional en profundidad (Dwise) de tamaño 3x3 con activación ReLU6.*

La convolución en profundidad aplica una convolución profunda separable de 3x3 a un único filtro en cada canal de la imagen de entrada, en lugar de para todos como se lo hace en la convolución normal. Esto significa que cada uno de los canales de entrada se convolucionan con un único filtro y la salida de todos los canales se concatena al final para dar la salida (Ehtesham, 2022).

- *Capa proyección convolucional de tamaño 1x1 sin activación (lineal).*

El filtro de convolución 1 × 1 se utiliza para lograr no linealidad en un modelo (Shukla & Tiwari, 2022). La capa de proyección convolucional de tamaño 1x1 lo que hace es reducir a la mitad la dimensionalidad de las características (número de filtros utilizados en las capas de convolución). Esto permite que el bloque residual invertido se integre fácilmente en la red, aprendiendo características más complejas sin aumentar la cantidad de parámetros. La salida de la capa de proyección convolucional de tamaño 1x1 es un mapa de características que tiene un espacio de dimensiones más bajas.

2.4.1.1.2 Cambios Notables de MobileFaceNet.

Después de conocer un poco sobre MobileNetV2, a continuación, se describen los cambios más notables realizados en MobileFaceNet a partir de la arquitectura MobileNetV2.

- *Convolución global en profundidad.*

Una de las principales contribuciones que los autores de MobileFaceNet presentaron fue que, sustituyeron la capa de agrupación promedio global (GAPool) por la capa convolucional de profundidad global (GDConv), es así como se puede obtener una representación facial más discriminante (Pérez et al., 2023). Una capa GDConv es una capa de convolución en profundidad con un tamaño de núcleo igual al tamaño de

entrada, pad = 0 y zancada = 1 (S. Chen et al., 2018). La ecuación del cálculo de la capa en profundidad se presenta a continuación.

Ecuación 1

Ecuación para la capa de convolución en profundidad global.

$$G_m = \Sigma_{i,j}\ K_{i,j,m} \cdot F_{i,j,m}$$

Nota: Adaptada de (S. Chen et al., 2018).

Donde:

G: es la salida de tamaño $1 \times 1 \times M$.

K: es la profundidad núcleo de convolución de tamaño $W \times H \times M$.

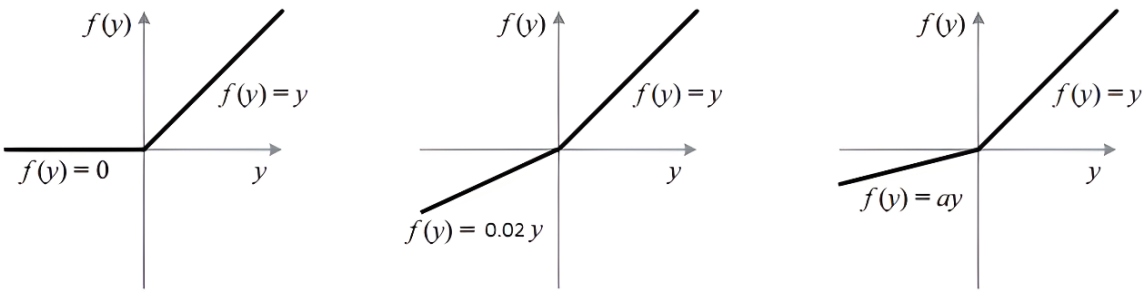
F: es el mapa de características de entrada de tamaño $W \times H \times M$.

- *Función de Activación.*

Las funciones de activación ReLU, LeakyReLU y ReLU6 han demostrado buenas soluciones en problemas de aprendizaje profundo. Sin embargo, en el estudio de (S. Chen et al., 2018), autores de MobileFaceNet, mejoraron aún más el modelo utilizando la función de activación Unidad Linear Rectificada Paramétrica (PReLU) de (He et al., 2015), para aplicar la no linealidad a la red. PReLU mejora el ajuste del modelo con un costo computacional adicional casi nulo y poco riesgo de sobreajuste (He et al., 2015). PReLU que es capaz de aprender el parámetro de pendiente mediante retropropagación con un aumento insignificante en el costo de capacitación (Shaurya, 2019). La Figura 27 muestra las diferentes variaciones de la función de activación ReLU.

Figura 27

Funciones activación ReLU, LeakyReLU y PReLU.



Nota: Reproducida de (Shaurya, 2019).

- *Función de Perdida*

MobileFaceNet está entrenado mediante la pérdida de margen angular aditiva (ArcFace) para obtener características altamente discriminativas para el reconocimiento facial. En comparación con la pérdida Softmax, ArcFace optimiza la diversidad máxima para muestras entre clases y la similitud para muestras intraclase, además se puede implementar fácilmente con baja sobrecarga computacional (Shun & Aung, 2019).

2.4.1.1.3 Arquitectura.

Luego de revisar temas importantes sobre el modelo, finalmente en la Figura 28 se muestra la arquitectura general de MobileFaceNet.

Figura 28

Arquitectura de MobileFaceNet.

Input	Operator	*t*	*c*	*n*	*s*
$112^2 \times 3$	conv3x3	-	64	1	2
$56^2 \times 64$	depthwise conv3x3	-	64	1	1
$56^2 \times 64$	bottleneck	2	64	5	2
$28^2 \times 64$	bottleneck	4	128	1	2
$14^2 \times 128$	bottleneck	2	128	6	1
$14^2 \times 128$	bottleneck	4	128	1	2
$7^2 \times 128$	bottleneck	2	128	2	1
$7^2 \times 128$	conv1x1	-	512	1	1
$7^2 \times 512$	linear GDConv7x7	-	512	1	1
$1^2 \times 512$	linear conv1x1	-	128	1	1

Nota: Reproducida de (S. Chen et al., 2018).

De acuerdo con la Figura 28, cada línea contiene una secuencia de operadores que se repite n veces. El número c de canales de salida de todas las capas de la misma secuencia es el mismo. Cada secuencia tiene un paso s en su capa inicial, mientras que todas las demás secuencias utilizan el paso 1. Los núcleos de 3 × 3 se utilizan para todas las convoluciones espaciales en los cuellos de botella. Así mismo el tamaño de entrada siempre recibe el factor de expansión t (S. Chen et al., 2018).

Para terminar, MobileFaceNet utiliza 0,99 millones de parámetros y presenta factores de expansión de cuellos de botella más pequeños que los de MobileNetV2. Para reducir aún más el costo computacional y la cantidad de parámetros; la resolución de entrada de la imagen que ingresa al modelo se ha establecido a 112 x 122, a diferencia de MobileNetV2 que tiene una resolución de entrada de 224 x 224. Los autores han entrenado MobileFaceNet en la base de datos de imágenes MS-Celeb-1M. MS-Celeb-

1M es un conjunto de datos extenso que consta de 100.000 identidades, donde cada identidad tiene alrededor de 100 imágenes faciales.

Estas fueron algunas de las características más relevantes de la arquitectura MobileFaceNet detalladas en este proyecto, para más información revisar el artículo de (S. Chen et al., 2018).

2.5 Modelo de análisis aplicando la Metodología CRISP-DM.

En esta sección se analiza cómo fue aplicado la metodología CRISP-DM en la elección del modelo de reconocimiento facial. Esta metodología se enfatizó más a profundidad en la sección 1.8.1, específicamente.

- *Entendimiento del negocio.*

La primera fase de la metodología CRISP-DM consiste en comprender y profundizar los objetivos que se pretende cumplir en el contexto general del proyecto, para así implementar una solución efectiva. Los trabajos relacionados fueron los documentos utilizados para establecer la base de cómo resolver el problema expuesto, además que ayudaron a comprender el procedimiento que se debe seguir. Es así que, la alternativa sugerida para solventar el problema expuesto fue desarrollar una aplicación para el registro de asistencia utilizado una red neuronal profunda ligera, pero a la vez que sea precisa, con el fin de reducir el tiempo que se tarda en ejecutar dicha actividad. Esta idea se dio a conocer en la sección 2.2.1, al explicar el propósito del sistema.

- *Recopilación y comprensión de datos.*

La segunda fase de la metodología CRISP-DM se refiere a comprender como elaborar el conjunto de datos a utilizarse en el entrenamiento y en la evaluación de un modelo de Machine Learning o Deep Learning. La recolección, construcción de los datos y su relación con el negocio son algunas de las tareas que se realizan en esta etapa. Esta fase se tomó en cuenta en la sección 2.3.1, en la elaboración del conjunto de imágenes.

- *Preparación de datos.*

La preparación de los datos depende principalmente del modelo CNN que se va a utilizar, pues cada uno cuenta con diferentes requerimientos. Sin embargo, los pasos principales que se deben seguir para la correcta preparación de los datos son: el preprocesamiento, la normalización y finalmente aplicar el aumento de los datos. Es

importante destacar que la calidad de los datos con los que se entrena el modelo influye significativamente en la calidad del modelo final.

- *Elección del modelo.*

En la actualidad existen varios modelos que se pueden utilizar para resolver tareas de reconocimiento facial, por esta razón, la elección del modelo correcto es una de las partes más complicadas de realizarse. El modelo seleccionado para la aplicación móvil de reconocimiento facial fue MobileFaceNet. Este modelo fue tomado como base de (sirius-ai, 2019). La razón por la cual se utilizó este modelo y no se entrenó uno desde cero, se debe a que el modelo seleccionado tiene buena precisión, por lo que, no se vió viable entrenar nuevamente para obtener algo similar.

- *Evaluación.*

MobileFaceNet es uno de los mejores modelos utilizados en tareas de reconocimiento facial, debido a tiene una excelente precisión en la predicción de los embeddings. Según sirius-ai, 2019), MobileFaceNet alcanzó una precisión de 99.4 % de precisión en la base de datos de LFW. Es así que, con el modelo seleccionado se realizó las pruebas necesarias con el conjunto de datos personalizado antes de implementarlo en la aplicación móvil.

- *Despliegue*

La fase final de la metodología CRISP-DM, define la etapa en el que el modelo elegido y evaluado se pone a producción en un entorno operativo para su utilización. Esta fase se tiene en cuenta en la sección 2.6.2.3, cuando el modelo es utilizado como extractor de características faciales en la aplicación de reconocimiento facial.

2.6 Desarrollo de la Aplicación Móvil.

Luego de haber seleccionado el modelo para el reconocimiento facial y convertido en un formato ideal (*tflite*) para dispositivos móviles, se inició con el desarrollo de la aplicación móvil, teniendo en cuenta la metodología en Cascada. Cabe desatacar que la aplicación móvil se creó en Android Studio utilizando Java y las herramientas de desarrollo descritas en la sección 2.2.3.2.1.

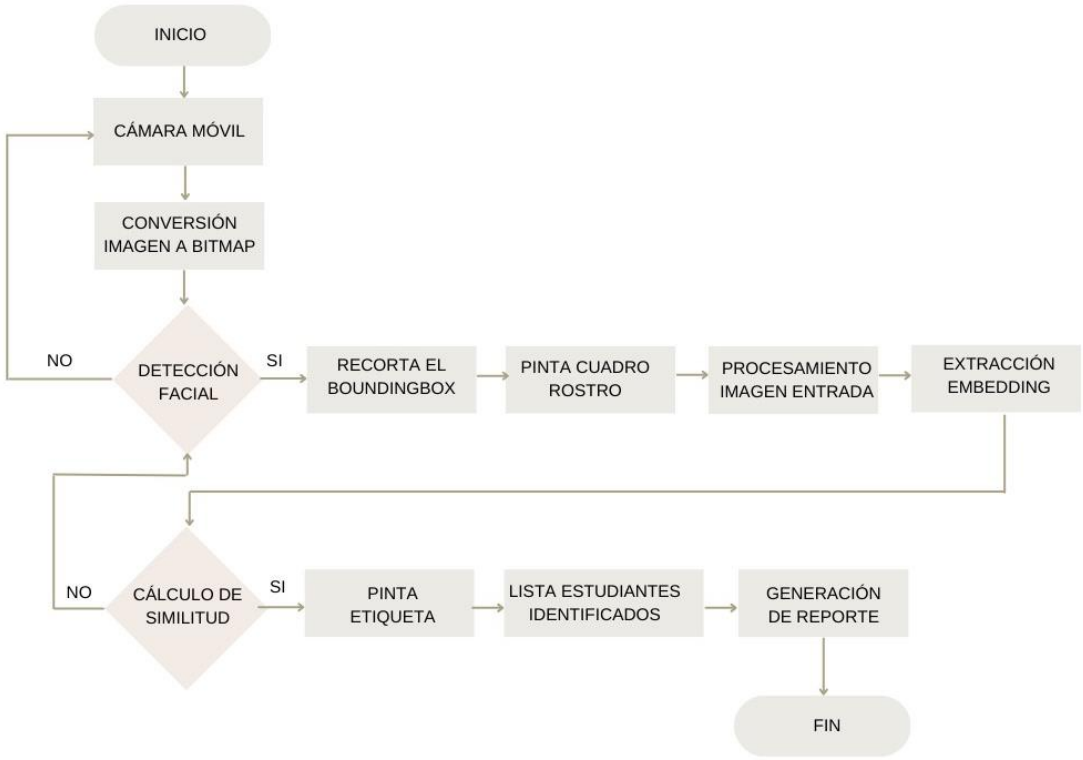
2.6.1 Diseño del Sistema.

El diseño del sistema es una fase crucial en el desarrollo del software, debido a que define detalladamente la arquitectura del sistema. Esta etapa es esencial para interpretar los requerimientos establecidos, para así lograr desarrollar una solución

efectiva. A la hora de codificar, el diseño del sistema ayuda a entender cómo será el flujo de navegación, evitando estar desorientado. El diseño del sistema sirve como intermediario entre la creación de software y su ejecución exitosa. Basado en el diseño es eventual tomar decisiones importantes sobre la estructura, la asignación de recursos, la gestión de datos, la seguridad y la interconexión de componentes del sistema. La eficiencia, el rendimiento y la confiabilidad están directamente influenciados por la calidad del diseño. La Figura 29 presenta un diagrama de flujo que representa la arquitectura general del sistema de reconocimiento facial.

Figura 29

Diagrama de flujo del Sistema de Reconocimiento Facial.



Nota: Elaboración propia.

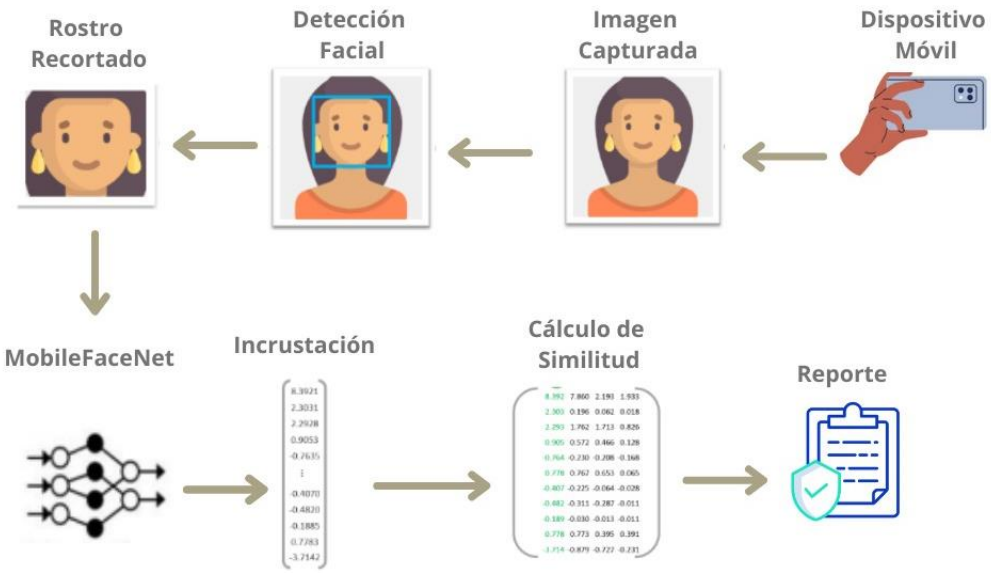
En la Figura 29, el diagrama de flujo describe el funcionamiento del sistema de reconocimiento facial. El proceso inicia con la aplicación móvil que captura fotogramas cada segundo en tiempo real mediante la cámara principal del teléfono. Posteriormente por cada fotograma (*imagen*) capturado, se convierte a un formato que pueda ser procesado por la aplicación (*bitmap*) para luego realizar la detección facial. Si se ha detectado un rostro el flujo continua, y la aplicación procede a recortar, preparar e ingresar el rostro recortado al modelo MobileFaceNet para la extracción de las características faciales. Posteriormente mediante un cálculo de similitud se verifica la identidad del estudiante, comparando el embedding predicho con un conjunto de

embedding previamente almacenado en una base de datos. Los estudiantes identificados por el sistema se agregan a una lista para generar un reporte en Excel.

Así mismo, también se ha elaborado un diagrama que ayuda a comprender mejor como es el funcionamiento de la aplicación móvil de reconocimiento facial. La Figura 30 se basó en el diagrama de flujo de la Figura 29, por lo que ambos representan lo mismo.

Figura 30

Representación gráfica del Sistema de Reconocimiento Facial.



Nota: Elaboración propia.

2.6.2 Proceso integral de MobileFaceNet en la Aplicación Móvil.

Finalmente, la codificación de la aplicación móvil se dió inicio teniendo en cuenta el propósito del sistema descritas en la sección 2.2.1, los requerimientos establecidos en la Tabla 7 y el diseño del sistema desarrollado en la sección 2.6.1. Son cuatro etapas esenciales que se va a tomar en cuenta para documentarse en este trabajo: detección, preprocesamiento, extracción de características, comparación y decisión. Estas fases fueron detalladas mejor en la sección 1.6.

2.6.2.1 Detección

De acuerdo con, Rosero (2019), la detección facial tiene como finalidad identificar si los objetos que aparecen en una imagen digital corresponden o no a un rostro. La detección de rostros se probó utilizando dos métodos: Haar Cascade y MTCNN. Ambos detectores cumplían con su función, sin embargo, MTCNN demostró

tener mejor precisión en la detección con respecto al otro detector. Para comprender el funcionamiento de modelo MTCNN se recomienda revisar la sección 2.6.1.2.

El modelo MTCNN fue tomado como base de (Chen. J, 2018) y fue adaptado a las necesidades de la aplicación. En formato "*pb*" fue como se obtuvo el modelo desde el repositorio. Protocol Buffers (pb) son formatos de serialización de datos que estructuran y permiten la transmisión y el almacenamiento de los datos de manera más eficiente y compacta que otros tipos de archivos.

Luego de cargar el modelo al proyecto, la configuración de los pesos se los realiza en la clase *MTCNN.java*. En la Figura 31 se presenta una pequeña porción de código del código implementado.

Figura 31

Configuración del modelo MTCNN en Android Studio.

```
1 usage
private float factor=0.709f;
1 usage
private float PNetThreshold=0.6f; //Umbral para PNET
1 usage
private float RNetThreshold=0.7f; //Umbral para RNET
1 usage
private float ONetThreshold=0.7f; //Umbral para ONET
no usages
private static final String MODEL_FILE  = "file:///android_asset/mtcnn_freezed_model.pb";
1 usage
private static final String   PNetInName  ="pnet/input:0";
3 usages
private static final String[] PNetOutName =new String[]{"pnet/prob1:0","pnet/conv4-2/BiasAdd:0"};
1 usage
private static final String   RNetInName  ="rnet/input:0";
3 usages
private static final String[] RNetOutName =new String[]{ "rnet/prob1:0","rnet/conv5-2/conv5-2:0",};
1 usage
private static final String   ONetInName  ="onet/input:0";
4 usages
private static final String[] ONetOutName =new String[]{ "onet/prob1:0","onet/conv6-2/conv6-2:0","onet/conv6-3/conv6-3:0"};
```

Nota: Elaboración propia.

El proceso de detección comienza con la transformación de la imagen digital capturada por la cámara del teléfono en un formato que el detector MTCNN pueda procesar. La cámara (CameraX) devuelve la imagen en formato ImageProxy por lo que es necesario convertirlo antes a Bitmap. Un Bitmap es un tipo de dato que representa la información de una imagen, pixel por pixel en una cuadricula bidimensional. La imagen convertida en Bitmap pasa por cada una de las tres redes hasta lograr identificar uno o varios rostros, estos pasos del modelo se mencionan en la sección 1.6.1.2. El resultado final son las coordenadas de cuadros delimitadores que rodean las caras identificadas en la imagen, así como la ubicación de 5 landmarks faciales. Con esa información se

procedió a pintar la zona ROI del rostro en la aplicación móvil, tal como se refleja en la Figura 33. La Figura 32 muestra una fracción de código de las tres redes que conforman el detector MTCNN.

Figura 32

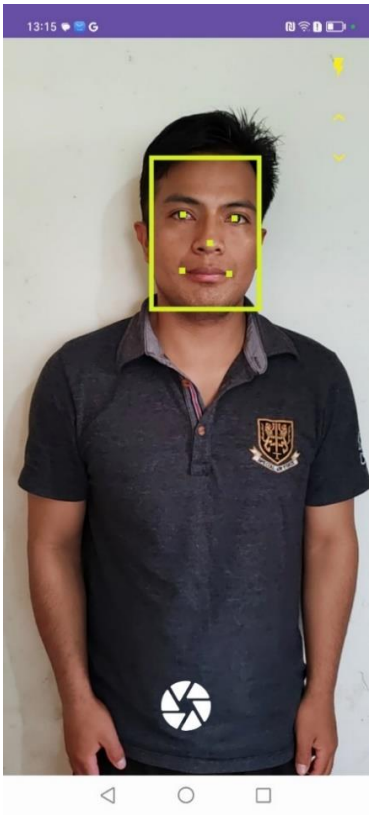
Fragmento de código detección facial en java con MTCNN.

```
public Vector<Box> detectFaces(Bitmap bitmap,int minFaceSize) {
    long t_start = System.currentTimeMillis();
    //【1】PNet: Genera cuadros delimitadores candidatos iniciales para caras.
    Vector<Box> boxes=PNet(bitmap,minFaceSize);
    //Garantizar que los cuadros delimitadores permanezcan dentro del rango de las dimensiones de la imagen.
    square_limit(boxes,bitmap.getWidth(),bitmap.getHeight());

    //【2】RNet: Refina los cuadros candidatos generados por PNet.
    boxes=RNet(bitmap,boxes);
    //Garantizar que los cuadros delimitadores permanezcan dentro del rango de las dimensiones de la imagen.
    square_limit(boxes,bitmap.getWidth(),bitmap.getHeight());

    //【3】ONet: Refina aún más los cuadros delimitadores y proporciona los puntos referenciales.
    boxes=ONet(bitmap,boxes);

    Log.i( tag: "MTCNN", msg: "Tiempo de ejecución:"+(System.currentTimeMillis()-t_start));
    lastProcessTime=(System.currentTimeMillis()-t_start);
    return  boxes;
}
```

Nota: Elaboración propia.

Figura 33

Detección de Facial con 5 puntos de referencia desde la Aplicación Móvil.



Nota: Elaboración propia.

2.6.2.2 Preprocesamiento.

El preprocesamiento se centra en preparar y adaptar las imágenes de los rostros detectados en la fase anterior, asegurando que estén en un formato adecuado, antes de ingresarlas al modelo extractor de características. Es importante transformar la imagen de entrada a las mismas características con las cuales fue entrenado el modelo. Es decir, si el modelo fue entrenado con imágenes en color RGB de 112 x 112 pixeles, pues la imagen de entrada al modelo también debe cumplir con esas características caso contrario la aplicación fallará.

- **Alineación del rostro:** Con las coordenadas de los cuadros delimitadores, resultado de la fase anterior, **s**e recorta la zona ROI (rostro) y se redimensiona a 112 x 112 pixeles, de acuerdo con los requerimientos del modelo MobileFaceNet.

2.6.2.3 Extracción de Características.

La extracción de características se utiliza para extraer información facial de una imagen, lo cual es esencial para realizar la comparación (Domínguez, 2017); esto se explicó en la sección 1.6.3. Con la imagen de entrada ya preprocesada en la fase anterior, se crea una variable de tipo ByteBuffer antes de pasar los datos al modelo. Esta variable permite reservar un bloque de memoria de tamaño especifico que contiene los datos de la imagen. Posteriormente la información de la imagen preprocesada se normaliza y se almacena en esa variable. Esto es necesario debido a que el modelo "*MobileFaceNet.tflite*" requiere ese dato para poder inferir. La variable de tipo ByteBuffer ingresa al modelo y pasa por todas las capas convolucionales, hasta dar como resultado un vector de características que representan características específicas de la estructura facial. En la Figura 34 se presenta una pequeña porción de código.

Figura 34

Extracción de características en Android Studio.

```
try {
    float[][] result = new float[1][Utils.NUM_CLASSES_B]; //Matriz vacio con el tamaño de todas las clases

    int[] imagePixels = new int[Utils.IMAGE_HEIGHT * Utils.IMAGE_WIDTH]; //Vector para almacenar los pixeles de la imagen
    ByteBuffer byteBuffer = convertBitmapByteBuffer(bitmap, //Imagen de entrada
                                                    Utils.imageDataMobileNetV2_B, //Almacena datos de entrada de la imagen
                                                    imagePixels,
                                                    Utils.IMAGE_WIDTH, //Ancho de la imagen
                                                    Utils.IMAGE_HEIGHT); //Alto de la imagen
    Utils.interpreterMobileNetV2_B.run(byteBuffer, result); //Ejecuta la inferencia
    float[] embedding=result[0]; //Caracteristicas extraidas por el modelo
}catch (Exception e){
    Log.i( tag: "FACE RECOGNIZER", msg: "Error face recognizer_B"+e.getMessage());
}
```

Nota: Elaboración propia.

2.6.2.4 Comparación y Decisión.

La última fase del sistema de reconocimiento facial implica que el resultado obtenido en la fase anterior, (embedding) se someta a un proceso de comparación para poder tomar una decisión precisa. Uno de los métodos son los clasificadores; los clasificadores separan los datos en una categoría específica, como la identificación de una persona, evaluando las características del vector. Así mismo se utilizan métodos de cálculo de similitud, que hacen que las características faciales extraídas resultado del modelo, se comparen con representaciones existentes previamente calculados que están almacenados en una base de datos, dando una puntuación de similitud que ayuda a tomar decisiones. Para mejor comprensión acerca de estos métodos de comparación se recomienda revisar la sección 1.6.4. La robustez y la precisión en la fase de comparación y decisión es esencial para garantizar la efectividad y la confiabilidad de los sistemas de reconocimiento facial en diversas aplicaciones.

El método utilizado para realizar la comparación de vectores en la aplicación móvil fue la distancia euclidiana. La distancia euclidiana es una métrica utilizada para medir la distancia (similitud) entre dos puntos (vectores) en un espacio euclidiano, donde la menor distancia indica una mayor similitud entre vectores. La distancia entre vectores se obtiene después de restar las coordenadas de los puntos, elevar al cuadrado cada diferencia, sumar esas cantidades y luego obtener la raíz cuadrada del resultado. La Ecuación 2 define la fórmula de la distancia euclidiana.

Ecuación 2

Fórmula de la Distancia Euclidiana.

$$d(p1, p2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Nota: Adaptado de (Domínguez, 2017).

El umbral por defecto para realizar la comparación fue de 0.80. Si la distancia mínima es inferior al umbral establecido se considera coincidencia o semejanza entre los vectores. Un umbral demasiado alto podría aumentar la sensibilidad, pero también aumentarían los falsos positivos, en cambio un umbral demasiado bajo aumentaría los falsos negativos. No obstante, la aplicación móvil cuenta con la opción de aumentar o reducir el valor del umbral de acuerdo con las necesidades requeridas.

Figura 35

Cálculo de la Distancia Euclidiana.

```
// Iteración a través de los registros de los embeddings
for (Embedding registro : embeddings) {
    List<Float> values = registro.getValues(); // Obtener el embedding del registro actual

    // Calcular la distancia euclidiana entre el registro actual y el vector de probabilidades (probs)
    float dist = calculateDistance(probs, values);

    // Actualizar la distancia mínima y las variables asociadas si la distancia actual es menor
    if (dist < min_distance) {
        min_distance = dist;
        name = registro.estudiante();
        cedula = registro.getCedula();
    }
}

// Comparación según la distancia mínima y el umbral establecido
if (min_distance < THRESHOLD) {
    // Si la distancia mínima está por debajo del umbral se agrega el estudiante a la lista
    if (!students.contains(cedula)) {
        students.add(cedula);
    }

    mPersonClass = name;
} else {
    mPersonClass = "Unknown";
}
```

Nota: Elaboración propia.

2.7 Desarrollo de Pruebas.

La fase pruebas establece un paso esencial para el correcto funcionamiento del sistema. Esto se debe a que las pruebas no solo verifican la funcionalidad, sino que también pueden encontrar posibles errores que deben corregirse. La detección temprana de fallos ayuda a optimizar el tiempo y recursos invertidos en el desarrollo. Además, en esta fase es posible comprobar que si se está cumpliendo los requerimientos del sistema establecidos.

Esta aplicación es un prototipo desarrollado para comprobar el tiempo que se demora en registrar la asistencia utilizando la aplicación móvil y el método tradicional. Es así que, para realizar las pruebas se ha considerado un grupo de estudiantes matriculados legalmente en el periodo académico Octubre 2023 - Febrero 2024, que cursan la materia de Lógica y Algoritmos de Programación. Las pruebas fueron realizadas en un ambiente controlado "*salón de clases*", conformado por un grupo de 16 estudiantes, es decir que el lugar cuenta con óptimas condiciones de iluminación. Las pruebas se llevaron a cabo en dos escenarios diferentes.

2.7.1 Primer Escenario.

El primer escenario de la prueba consistió en que la persona encargada de registrar la asistencia (en este caso el docente), recorre los lugares donde se encuentran los estudiantes enfocando la cámara del teléfono móvil. Es importante destacar que la aplicación móvil de reconocimiento facial funciona de manera independiente de la orientación del teléfono, como se puede observar en la Figura 36, sin embargo, la prueba se realizó con el teléfono en orientación vertical.

Figura 36

Proceso de registro de asistencia.

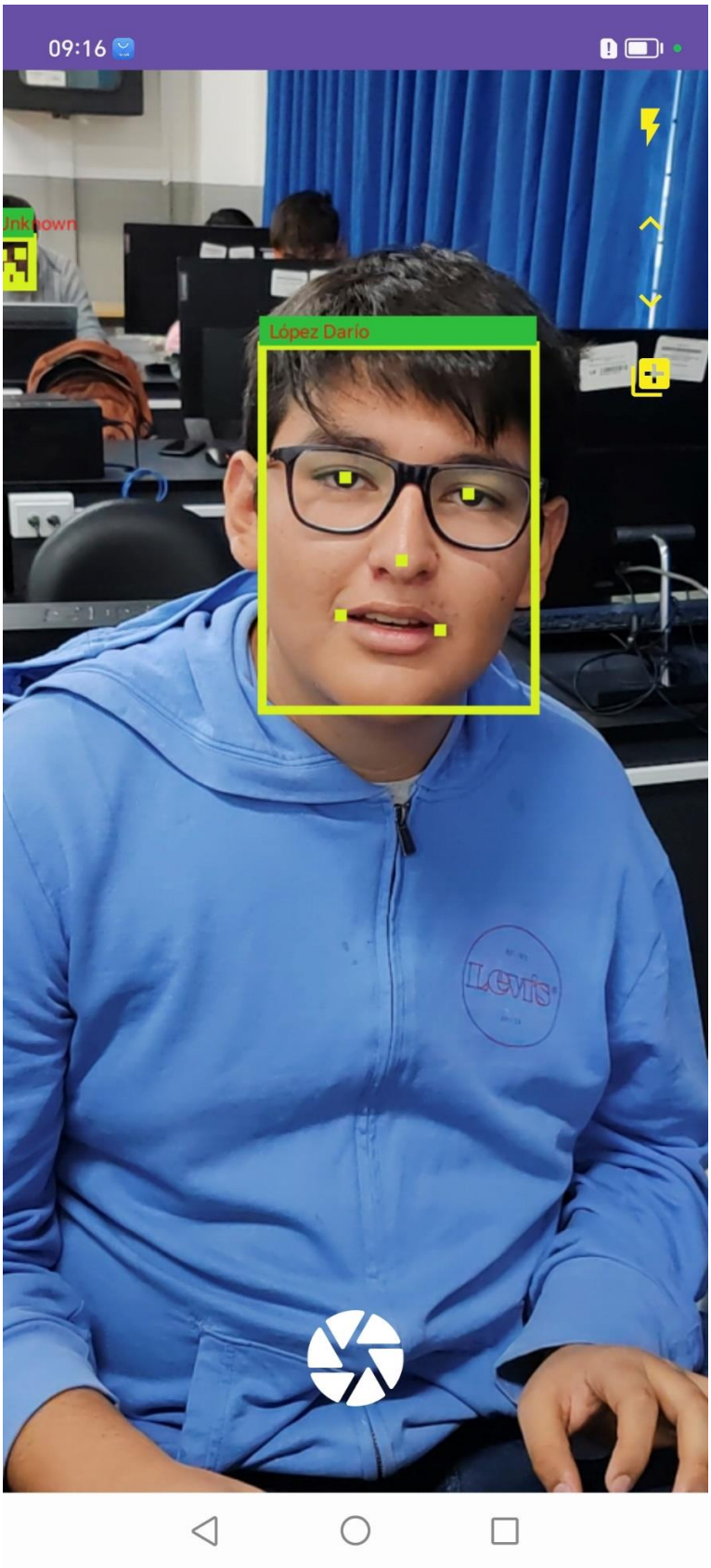
Nota: Elaboración propia.

El tiempo que se tardó en pasar registrando la asistencia de todos los estudiantes con la aplicación fue de 1.14 minutos. Este tiempo fue obtenido con un cronometro, desde que se abre la aplicación en el teléfono, hasta finalizar con la generación del reporte. Sin embargo, este tiempo es una estimación, debido a que el tiempo puede cambiar según las circunstancias del momento; es decir, casos como cuando al docente se le dificulte movilizarse por el aula o que la aplicación no reconozca de inmediato al estudiante, en esos casos el tiempo puede variar.

En las Figuras 37, 38, 39 y 40 se muestra la ejecución en tiempo real de la aplicación móvil durante el registro de asistencia de algunos estudiantes.

Figura 37

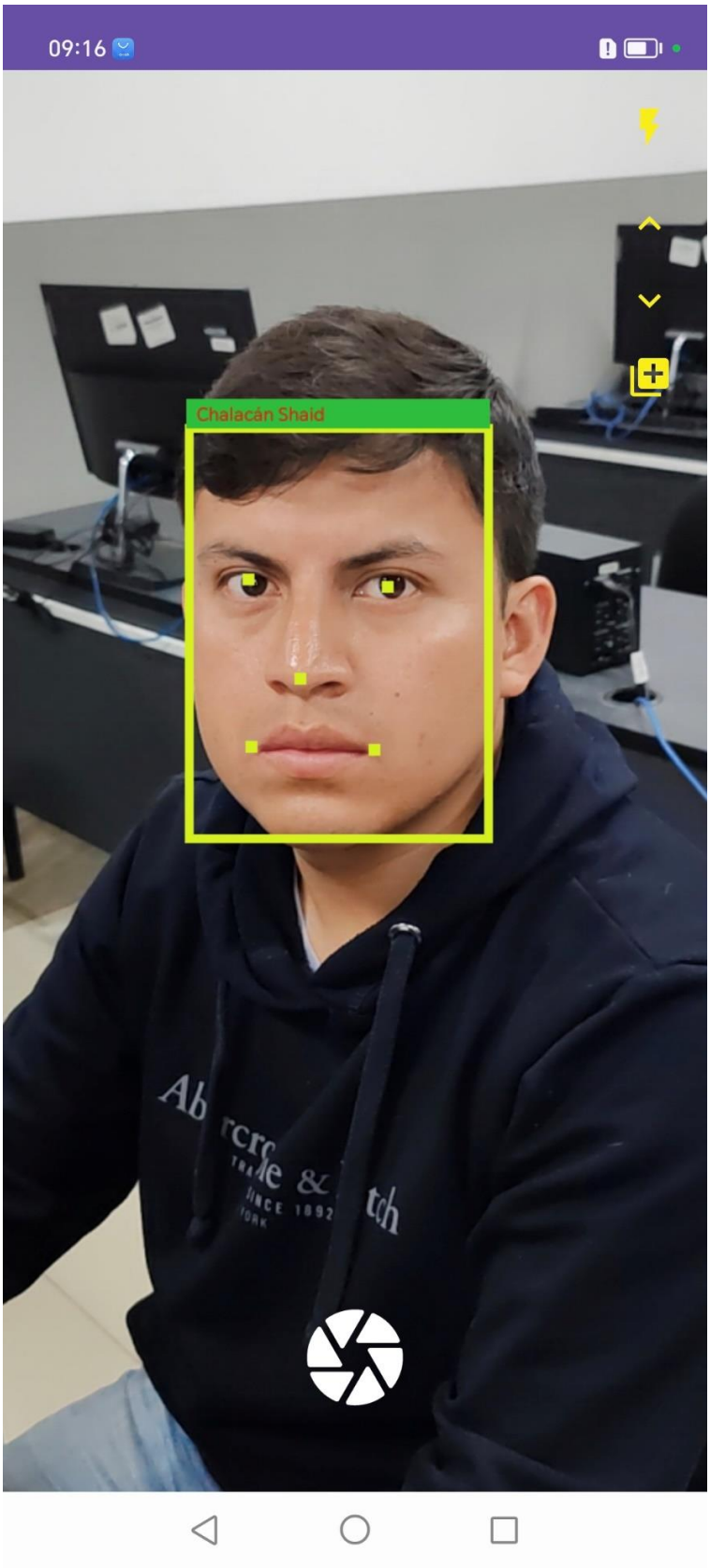
Verificación Facial estudiante 1.



Nota: Elaboración propia.

Figura 38

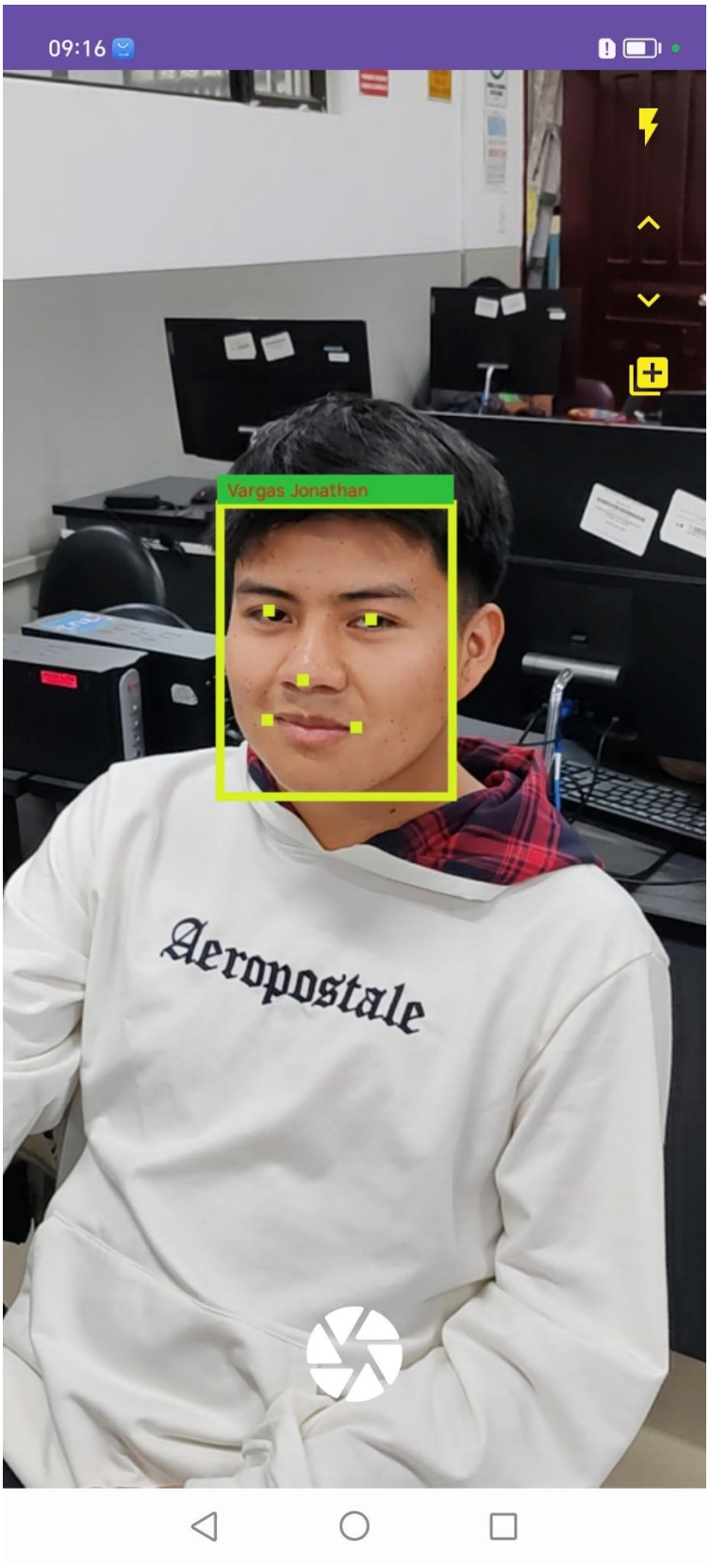
Verificación Facial estudiante 2.



Nota: Elaboración propia.

Figura 39

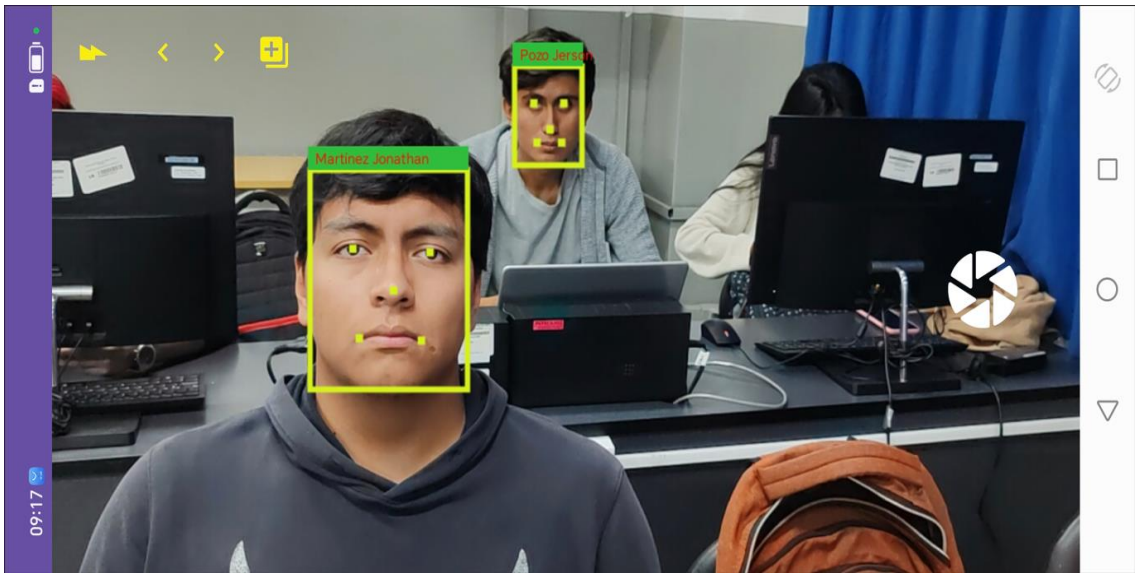
Verificación Facial estudiante 3.



Nota: Elaboración propia.

Figura 40

Verificación Facial de dos estudiantes.



Nota: Elaboración propia.

2.7.2 Segundo escenario.

En cambio, en el segundo escenario, el método de registro de asistencia fue diferente. En ese escenario los estudiantes formaron una columna y cada uno pasó frente al teléfono. Así mismo el tiempo fue obtenido con un cronometro, desde que se abre la aplicación en el teléfono, hasta finalizar con la generación del reporte. Es así como, el tiempo tardado en realizar el registro de asistencia en el segundo escenario fue de 55.3 segundos.

En los ambos escenarios la distancia entre el teléfono y el rostro del estudiante rondaba de 1 a 1.5 metros, dando buenos resultados. Cabe destacar que a mayor distancia el modelo empezó a presentar falsos positivos, debido a que la imagen de entrada empezaba a hacerse pequeña. Sin embargo, esto es posible solucionarlo adicionado imágenes variadas a la base de datos de los vectores de características, para posteriormente compáralo con la función de similitud.

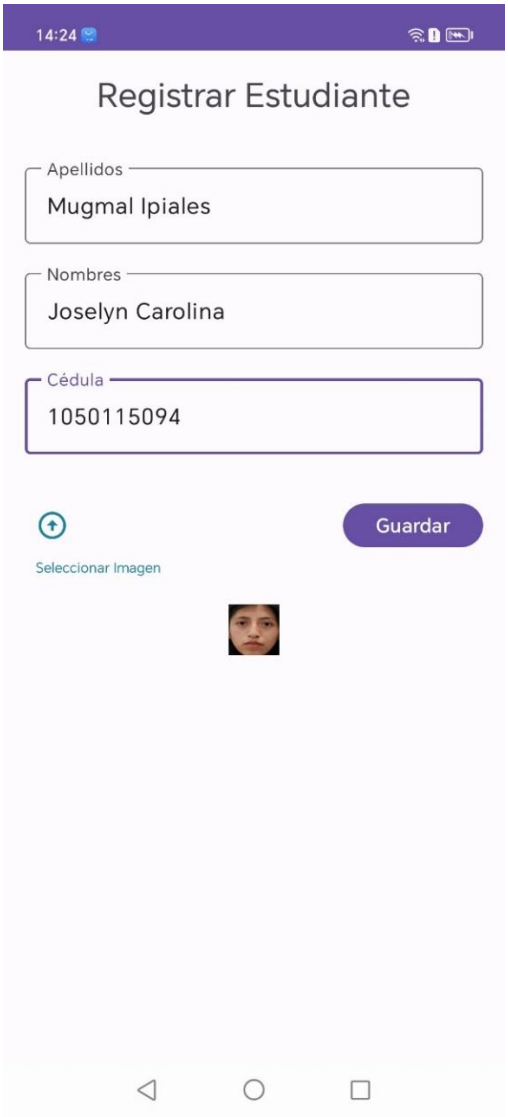
En las pruebas del primer escenario, hubo un caso en el que la aplicación móvil no pudo identificar de inmediato a un estudiante que usaba lentes. Esto pudo suceder debido a que la fotografía utilizada para registrar al estudiante en el sistema reflejaba brillo de la luz en los lentes. No obstante, ese error fue corregido.

2.7.3 Adicción de un nuevo estudiante a la Base de Datos.

La aplicación móvil se probó con un grupo de 16 estudiantes; sin embargo, la aplicación tiene la opción de agregar un nuevo estudiante al sistema solo con sus datos personales y una imagen de su rostro. La aplicación calcula el vector de características de esa imagen y lo almacena en la base de datos junto con la información personal del estudiante. El nuevo registro se guarda en un archivo json que se encuentra en el mismo directorio del teléfono. No obstante, para trabajos futuros se recomienda utilizar una base de datos más robusta. Aquí fue donde también se utilizó el dataset de imágenes elaboradas en la sección 2.3.2. El procedimiento para registrar un nuevo estudiante se muestra en la Figura 41.

Figura 41

Registro de un nuevo estudiante.



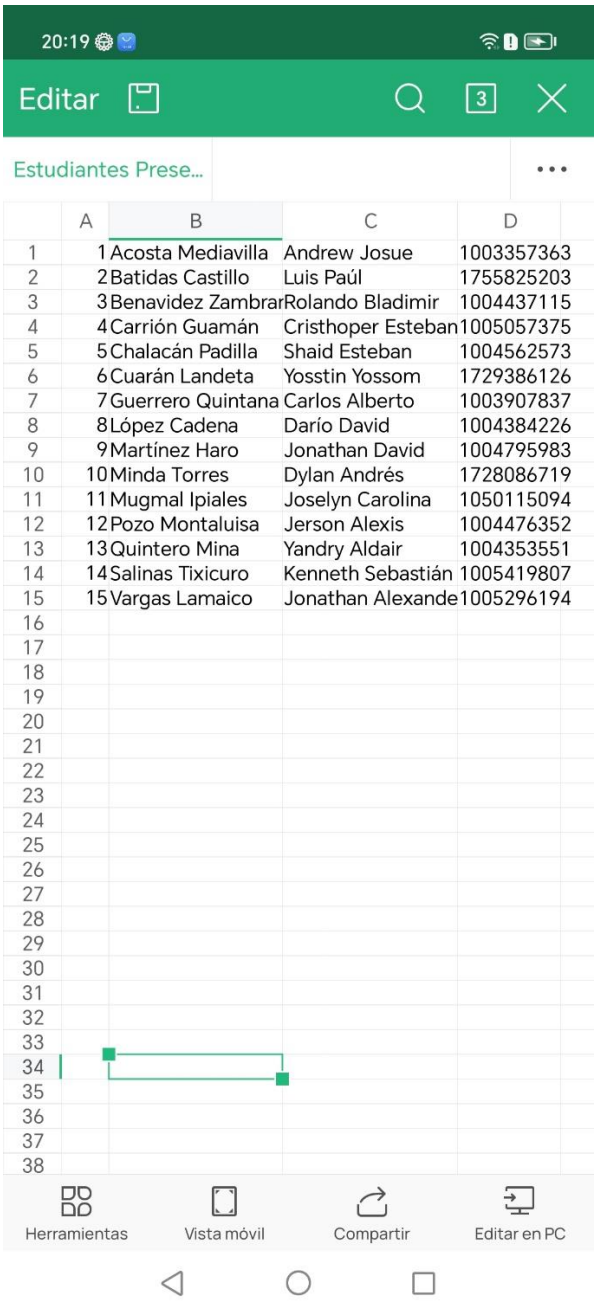
Nota: Elaboración propia.

2.7.4 Reporte.

El reporte describe la información de los estudiantes identificados por la aplicación móvil de reconocimiento facial, los cuales son considerados como estudiantes presentes (estudiantes que asistieron a clases). Este reporte es un documento en formato "*.xls*", que incluye los nombres, apellidos y cédula del estudiante. En la Figura 42 se presenta el reporte generado por la aplicación, del primer escenario de pruebas.

Figura 42

Reporte generado por la Aplicación.



	A	B	C	D
1	1	Acosta Mediavilla	Andrew Josue	1003357363
2	2	Batidas Castillo	Luis Paúl	1755825203
3	3	Benavidez Zambrar	Rolando Bladimir	1004437115
4	4	Carrión Guamán	Cristhoper Esteban	1005057375
5	5	Chalacán Padilla	Shaid Esteban	1004562573
6	6	Cuarán Landeta	Yosstin Yossom	1729386126
7	7	Guerrero Quintana	Carlos Alberto	1003907837
8	8	López Cadena	Darío David	1004384226
9	9	Martínez Haro	Jonathan David	1004795983
10	10	Minda Torres	Dylan Andrés	1728086719
11	11	Mugmal Ipiales	Joselyn Carolina	1050115094
12	12	Pozo Montaluisa	Jerson Alexis	1004476352
13	13	Quintero Mina	Yandry Aldair	1004353551
14	14	Salinas Tixicuro	Kenneth Sebastián	1005419807
15	15	Vargas Lamaico	Jonathan Alexande	1005296194

Nota: Elaboración propia.

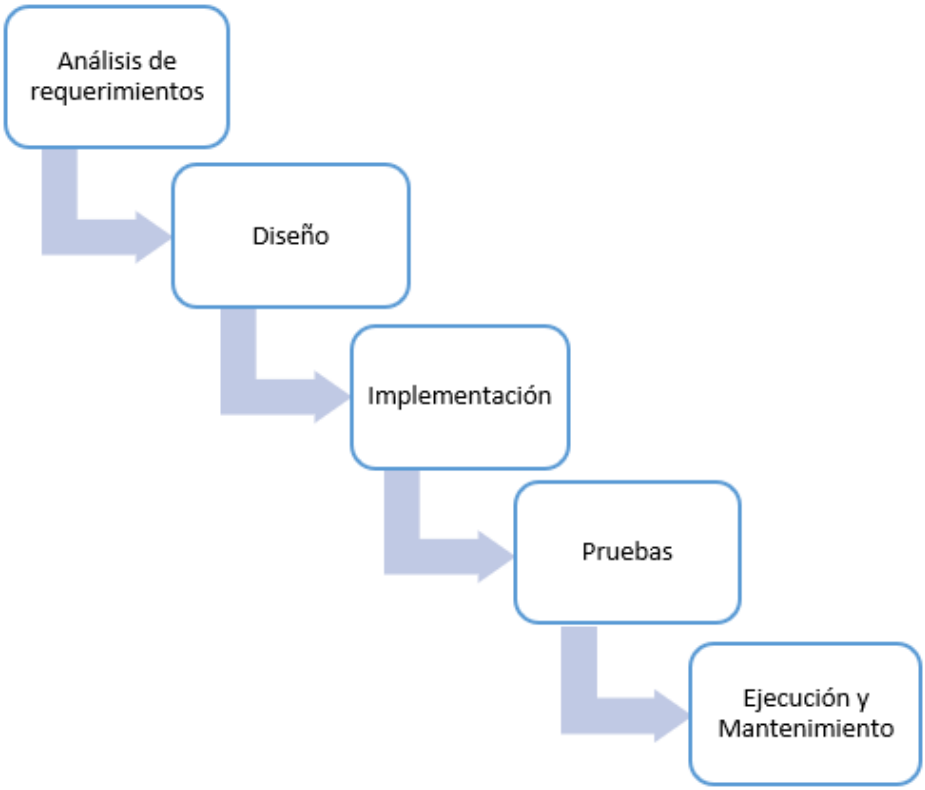
2.8 Modelo de análisis aplicando la Metodología en Cascada.

Se optó por utilizar la metodología en Cascada en el desarrollo de la aplicación móvil, debido a que es un método adecuado para proyectos con requisitos bien definidos. El modelo en cascada es un proceso secuencial que sigue un enfoque top-down, en el que cada etapa del ciclo de vida de desarrollo de software debe completarse antes de que pueda comenzar la siguiente etapa (Linkedin, 2023b). Esto permite al equipo de desarrollo encontrar y corregir errores antes de que afecten al resto del proyecto.

Aunque existen otras metodologías agiles como Scrum, esta metodología sigue siendo ampliamente utilizada en la actualidad. La metodología proporciona una estructura clara y permite una planificación detallada del proyecto. Esta metodología divide el proceso en fases bien definidas.

Figura. 46

Modelo lineal en Cascada.



Nota: Adaptado de (Cardenas, 2017).

Más adelante se describe donde fue utilizada cada fase de la metodología durante el desarrollo de la aplicación móvil de reconocimiento facial.

- *Análisis de requerimiento*

En esta fase, se recomienda recopilar y establecer todos los requerimientos del sistema, debido a que es necesario tener una comprensión clara de las necesidades a resolver antes de avanzar a la siguiente fase. Lo que se busca es minimizar los cambios en el sistema, luego que el desarrollador ya se encuentre en fases posteriores. Este es una de las fases más importantes de la metodología debido a que establece el alcance, presupuesto y el éxito del proyecto. Los requerimientos establecidos para el desarrollo de este sistema están descritos en la sección 2.1.

- *Diseño*

Esta fase interpreta los requerimientos establecidos en la fase anterior y es fundamental para la elaboración de la arquitectura y la interfaz del sistema. La documentación o diagramas generados en esta fase sirven de guía para la codificación. Esta fase fue aplicada en la sección 2.6.1 donde se diseña la arquitectura de la aplicación móvil.

- *Implementación*

En esta fase se traduce a código fuente los requerimientos recopilados durante la primera fase, además de basarse en el diseño del sistema. Es esencial asegurarse que el código se adhiera estrictamente al diseño para que el sistema funcione correctamente. Esta fase se tuvo en cuenta en la sección 2.6.2, durante el desarrollo de la aplicación móvil de reconocimiento facial.

- *Pruebas*

Después de finalizar la fase de implementación, sigue la fase de pruebas que consiste en verificar el correcto funcionamiento de la aplicación y encontrar posibles fallos que deben corregirse lo más pronto posible. El objetivo es garantizar el rendimiento y estabilidad del software antes de implementarlo definitivamente. Esta sección fue tomada en cuenta en la sección 2.7

- *Ejecución y mantenimiento*

Finalmente, en la etapa de ejecución y mantenimiento se administran actualizaciones, correcciones de errores y asistencia técnica. En esta etapa se abordan los problemas que requieran mejoras. No obstante, esta fase no se ha tomado en cuenta en este proyecto.

Capítulo 3.

Validación del Sistema.

La validación es un proceso que busca verificar que el producto (software) final, cumpla con los requerimientos especificados al inicio del desarrollo del proyecto. Para realizar la validación existen muchos métodos disponibles, es cuestión de buscar cuál de ellos se adapta más a las necesidades propias. La validación de la aplicación móvil de reconocimiento facial en este trabajo, se lo ha realizado siguiendo las buenas prácticas de la Guía de SWEBOK.

3.1 Validación del Modelo.

En esta sección se presenta el método utilizado en la validación, específicamente del rendimiento del modelo seleccionado.

3.1.1 Métricas de evaluación a partir de la Matriz de Confusión.

El modelo MobileFaceNet, fue tomado como base de (sirius-ai, 2019); el autor a su vez se basó en la investigación de (S. Chen et al., 2018). Sin embargo, en el repositorio solo se indicó la métrica accurracy, por lo que, utilizando el mismo conjunto de datos LFW en el cual el autor obtuvo esa métrica, se evaluó el modelo para obtener las demás métricas. Así mismo, el modelo también fue evaluado utilizando el dataset de imágenes elaborado en la sección 2.3.

De una matriz de confusión se puede obtener otras métricas de evaluación, que proporcionan información más detallada sobre el rendimiento del modelo. La matriz de confusión está compuesta de los siguientes componentes.

- *TP:* Son instancias verdaderas clasificadas como positivo (si es un rostro).
- *TN:* Son instancias verdaderas clasificadas como negativo (no es un rostro).
- *FP:* Son instancias falsas clasificadas como positivo, es decir predice como verdadero cuando no lo es.
- *FN:* Son instancias falsas clasificadas como negativo, es decir predice como falso cuando no lo es.

Existen algunas métricas de evaluación, no obstante, en este trabajo solo se tomaron en cuenta las siguientes métricas.

3.1.1.1 Exactitud (Accuracy).

El accuracy es una métrica que calcula la tasa de precisión de las predicciones correctas en un conjunto de datos. Esta métrica es utilizada para definir la eficacia de un modelo, sin embargo, puede resultar engañosa cuando la distribución de clases es desequilibrada (Bressler, 2022). La exactitud se calcula utilizando la Ecuación 3.

Ecuación 3

Fórmula de Exactitud.

$$accurracy = \frac{TP + TN}{TP + FN + FP + TN}$$

Nota: Adaptado de (Rodríguez, 2022).

3.1.1.2 Tasa de Error.

La tasa de error es una métrica que calcula la proporción de predicciones incorrectas basando en los falsos positivos y falsos negativos en un conjunto de datos. La tasa de error se calcula utilizando la Ecuación 4.

Ecuación 4

Fórmula de la Tasa de Error.

$$error = \frac{FP + FN}{TP + FN + FP + TN}$$

Nota: Adaptado de (Chacua, 2019).

3.1.1.3 Especificidad.

La especificidad calcula la eficiencia del modelo en identificar todas las instancias negativas detectadas. La especificidad se calcula utilizando la Ecuación 5.

Ecuación 5

Fórmula de la Especificidad.

$$especifidad = \frac{TN}{TN + FP}$$

Nota: Adaptado de (Chacua, 2019).

3.1.1.4 Sensibilidad (Recall).

La sensibilidad es una métrica que calcula la eficiencia del modelo para identificar todas las instancias positivas detectadas. La sensibilidad se calcula utilizando la Ecuación 6.

Ecuación 6

Fórmula de la Sensibilidad.

$$recall = \frac{TP}{TP + FN}$$

Nota: Adaptado de (Rodríguez, 2022).

La Tabla 14 muestra una comparación del modelo, evaluada en la base de datos de LFW y en la base de datos personalizada.

Tabla 14

Tabla comparativa de métricas de evaluación de MobileFaceNet.

	BBDD Personalizado	**BBDD LFW**
Accuracy	98.9%	99.4%
Tasa de error	0.1%	0.52%
Sensibilidad	98.2%	99.5%
Especificidad	99.4%	99.4%

Nota: Elaboración propia.

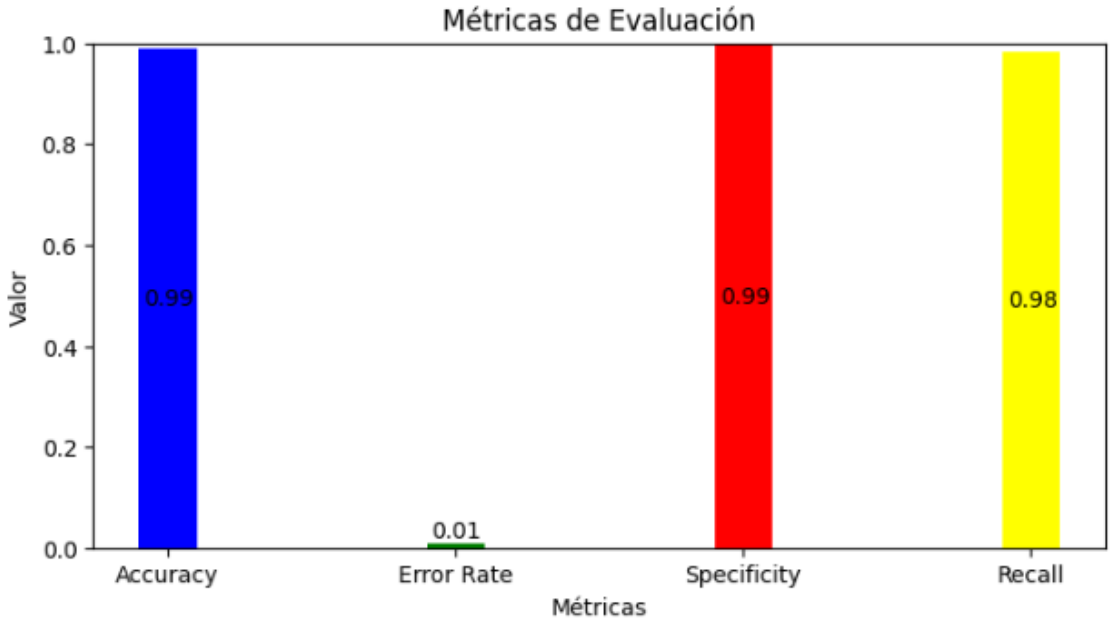
Los resultados de las métricas de evaluación descritas en la Tabla 14 son bastante buenos en ambos grupos de datos. Para garantizar que el modelo sea capaz de generalizar en diferentes contextos, es fundamental comprender como es su rendimiento en datos con los que no ha sido entrenado.

La Figura 43 muestra los resultados de las métricas de evaluación del modelo MobileFaceNet utilizando el conjunto de datos personalizado. Donde la barra azul fue calculada utilizando la Ecuación 3, la barra verde utilizando la Ecuación 4, la barra roja utilizando la Ecuación 5 y la barra azul utilizando la Ecuación 6. Esta Figura representa los valores obtenidos en la Tabla 14.

Así mismo también se obtuvo el gráfico del modelo utilizando la base de datos LFW, sin embargo, no fue tomada en cuenta en este proyecto ya que los valores fueron bastantes similares al dataset personalizado. Se dió más énfasis al gráfico del modelo evaluado con las imágenes de los rostros de los estudiantes a los que se pretendía aplicar el reconocimiento facial.

Figura 43

Gráfico métricas de evaluación.



Nota: Elaboración propia.

La Figura 44 muestra una parte del código utilizado en la obtención de las métricas de evaluación aplicando las ecuaciones 3, 4, 5 y 6; para obtener la Figura 43.

Figura 44

Cálculo de las métricas de evaluación.

```
def metricas(predictions_frame):
    accuracy = []
    recall = []
    error_rate_list = []
    specificity_list = []

    for index, row in predictions_frame.iterrows():
        actual_labels = np.argmax(row['Actual'])  # Convertir one-hot a etiquetas numéricas
        predicted_label = np.argmax(row['Predictions'])

        # Calcular los componentes para calcular las métricas
        tp = np.sum((actual_labels == i) and (predicted_label == i) for i in range(len(row['Predictions'])))
        tn = np.sum((actual_labels != i) and (predicted_label != i) for i in range(len(row['Predictions'])))
        fp = np.sum((actual_labels != i) and (predicted_label == i) for i in range(len(row['Predictions'])))
        fn = np.sum((actual_labels == i) and (predicted_label != i) for i in range(len(row['Predictions'])))

        total_preds = len(row['Predictions'])
        accuracy.append((tp + tn) / total_preds if total_preds > 0 else 0)
        recall.append(tp / (tp + fn) if (tp + fn) > 0 else 0)
        error_rate_list.append(1 - accuracy[-1])
        specificity_list.append(tn / (tn + fp) if (tn + fp) > 0 else 0)

    return {
        'accuracy': np.mean(accuracy),
        'recall': np.mean(recall),
        'error_rate': np.mean(error_rate_list),
        'specificity': np.mean(specificity_list),
    }
```

Nota: Elaboración propia.

3.2 Validación del sistema mediante la Guía SWEBOK.

SWEBOK es un documento promovido por el IEEE Computer Society que describe un conjunto de conocimientos y buenas prácticas que los ingenieros de software deben adoptar para desarrollar un software de calidad (Swebok, 2014). El documento fue diseñado para guiar la educación y formación en la ingeniería de software, no obstante, su alcance es bastante amplio por lo que, puede ser utilizado como marco de referencia para evaluar cualquier proyecto de software. Este documento abarca 15 áreas principales en su edición 2014; para conocer las áreas que abarca se recomienda revisar la sección 1.8.3. Sin embargo, en este proyecto se hizo más énfasis en el área de *Pruebas de Software*. Se describe los pasos que se ha seguido, las herramientas y técnicas utilizadas para evaluar la aplicación enfocado en dos aspectos: funcionalidad y rendimiento.

3.2.1 Evaluación de Funcionalidad.

- **Objetivo**

Evaluar la funcionalidad externa de la aplicación móvil de registro de asistencia que utiliza reconocimiento facial.

- **Método de Evaluación.**

Pruebas de caja negra: Las pruebas de caja negra, es un método donde se evalúa literalmente el comportamiento funcional del sistema sin hacer énfasis en su implementación interna (*código fuente*). A diferencia de otros métodos, este método es fácil de usar y ayuda a ahorrar tiempo.

- **Plan de Pruebas.**

Descripción General: El uso de la aplicación comienza cuando un docente inicia sesión con sus credenciales. Para realizar el registro de asistencia mediante reconocimiento facial en tiempo real, el usuario autenticado (docente) debe enfocar la cámara del dispositivo móvil hacia el estudiante. Después de completar el proceso con todos los estudiantes, la aplicación debe generar un reporte en un formato específico de los individuos identificados. Para más detalle del funcionamiento de la aplicación se recomienda revisar la sección: 2.21 y 2.6.1.

La Tabla 15 presenta un conjunto de casos de prueba donde se describen las funcionalidades que se desea evaluar y que el sistema debe cumplirlos. Un caso de prueba es una condición especifica que se diseña con el fin de comprobar que sus resultados sean los esperados, cumpliendo con los requerimientos establecidos en la Tabla 7.

Tabla 15

Casos de prueba para Caja Negra.

Caso de prueba	Descripción	Fecha	Datos de entrada	Resultado esperado	Procedimientos especiales requeridos
RF01	Instalación de la aplicación en el teléfono	27/12/2023	Archivo .apk	La aplicación debe estar instalado correctamente en el teléfono.	Aplicación solo para Android
RF02	Autenticación nombre de usuario y contraseña	27/12/2023	Usuario y Contraseña	Autenticación exitosa del usuario	Llenar los campos obligatoriamente
RF03	El archivo MTCNN se carga en el dispositivo	27/12/2023	Archivo MTCNN.pb	Carga exitosa del modelo MTCNN	N/A
RF04	El archivo MobileFaceNet se carga en el dispositivo	27/12/2023	Archivo MobileFaceNet.tflite	Carga exitosa del modelo MobileFaceNet.	N/A
RF05	Reconocimiento de la identidad de un estudiante	08/01/2024	Imagen del rostro	Reconocimiento exitoso del individuo	N/A
RF06	Registro de un nuevo estudiante	27/12/2023	Datos personales y una fotografía	Registro exitoso del estudiante.	N/A
RF07	Generación de reporte en Excel.	08/01/2024	Lista de nombres	Generación exitosa de reporte	N/A

Nota: Elaboración propia.

- **Ejecución de Validación.**

Las plantillas para las pruebas de caja negra fueron obtenidas de (PMOinformatica, 2017). Con los casos de prueba ya definidos, el siguiente paso fue evaluar el cumplimiento de cada caso, señalándolo con un estado. También se adjunta la clasificación de algunos docentes, quienes revisaron la aplicación. En la Tabla 16 se muestra la evaluación de los casos de prueba.

Tabla 16

Análisis de los casos de prueba.

Caso de Prueba	Resultado obtenido	Estado	Docente 1	Docente 2	Observaciones
RF01	Aplicación móvil instalado correctamente en el teléfono	Cumple	✓	✓	N/A
RF02	Autenticación exitosa del usuario	Cumple	✓	✓	N/A
RF03	Carga exitosa del modelo MTCNN	Cumple	✓	✓	N/A
RF04	Carga exitosa del modelo MobileNetV2	Cumple	✓	✓	N/A
RF05	Reconocimiento exitoso del individuo	Cumple	✓	✓	N/A
RF06	Registro exitoso del estudiante.	Cumple	✓	✓	N/A
RF07	Generación exitosa de reporte	Cumple	✓	✓	N/A

Nota: Elaboración propia.

- **Análisis de Resultados**

Las pruebas de caja negra ayudan a identificar posibles errores de integración, los cuales necesitan corregirse lo más pronto posible. Una de las ventajas de estas pruebas es que se puede ir evaluado, incluso antes que el sistema este completamente finalizado.

Los resultados de la prueba de caja negra realizados a la aplicación fueron positivos. La funcionalidad de la aplicación móvil fue bastante satisfactoria, todos los casos establecidos en la Tabla 15 se cumplieron correctamente. No obstante, la aplicación es escalable por lo que, se puede ir mejorando y añadiendo nuevas funcionalidades.

3.2.2 Evaluación de Rendimiento.

- **Objetivo**

Evaluar el rendimiento de la aplicación móvil de registro de asistencia que utiliza reconocimiento facial.

- **Método de Evaluación.**

Pruebas de rendimiento: Las pruebas de rendimiento evalúan cómo se comporta una aplicación bajo diferentes condiciones, con el fin de detectar posibles cuellos de botellas, anticipándose a problemas antes de que afecten el rendimiento general del sistema.

Android Studio Profiler: Android Profiler es una herramienta integrada en el IDE Android Studio desde la versión 3.0 y versiones posteriores (Developers, 2023). Esta herramienta es usada para diagnosticar el rendimiento, la memoria y el CPU de las aplicaciones Android, mientras se ejecuta. Android Profiler ayuda a los desarrolladores a identificar, como una aplicación consume recursos, y en el caso de que el consumo sea excesivo buscar la manera optimizarlo.

- **Plan de Pruebas.**

Descripción General: Para medir el rendimiento de una aplicación, es necesario establecer una serie de métricas claves, tales como: el tiempo de respuesta, la escalabilidad, la capacidad y la estabilidad de la aplicación (Linkedin, 2023a). Sin embargo, cabe destacar que la aplicación será utilizada únicamente por un usuario unas pocas veces, por ende, no existe la probabilidad de concurrencia. Lo que se pretende evaluar es cómo se comporta la aplicación al ejecutar modelos CNN’s, en especial el modelo MTCNN. Es así como, las métricas establecidas fueron: el consumo de la memoria RAM, el consumo de la CPU y el consumo de la energía de la aplicación.

- **Ejecución de pruebas y Análisis de Resultados.**

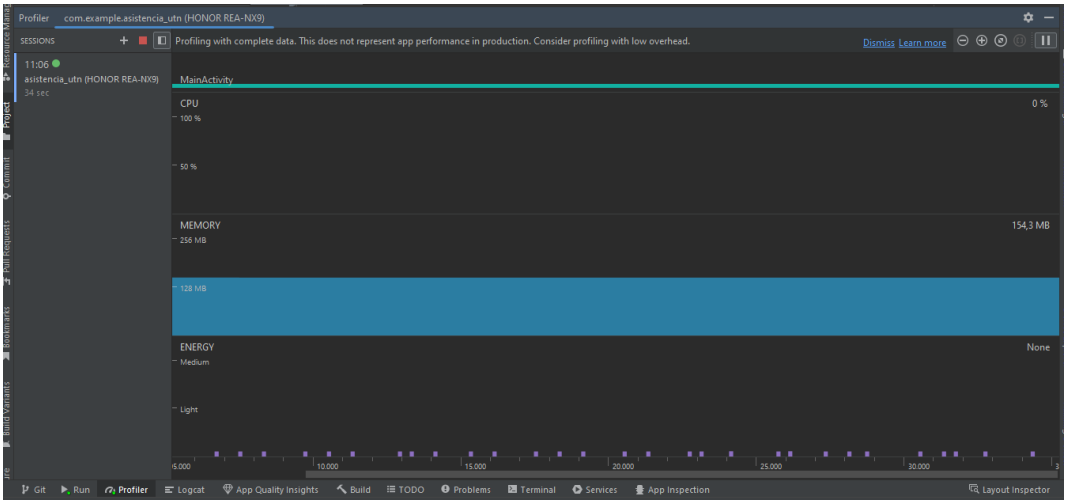
Una vez establecidas las métricas de evaluación, se realizó la prueba que consistió en ejecutar la aplicación, activar la opción de Profiler incorporada en Android

Studio, luego empezar a utilizar la aplicación normalmente. La herramienta muestra cómo cambia el consumo de recursos mediante un gráfico. Ese gráfico ayuda a comprender si el aplicativo esta funcionado correctamente y si no es el caso se debe optimizarlo.

- *Cuando se inicia la aplicación.*

Figura 45

Pruebas de rendimiento 1 con Android Profiler.



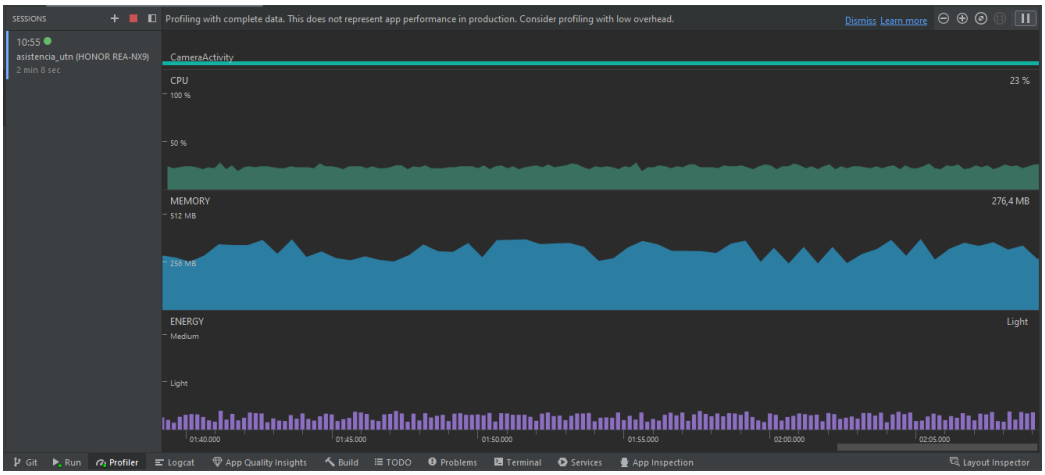
Nota: Elaboración propia.

Cuando se inicia la aplicación se observa que los valores de rendimiento del CPU, memoria y energía son bajos, esto se debe a que la aplicación a un no ha ejecutado ninguna tarea por lo que los recursos están sin utilizarse.

- *Cuando se está utilizando la aplicación.*

Figura 46

Pruebas de rendimiento 2 con Android Profiler.



Nota: Elaboración propia.

La prueba fue realizada durante un lapso de 1 minuto. Sin embargo, esta prueba es solo una simulación del real rendimiento, donde los valores podrían variar un poco. La Figura 45 mostró un cambio bastante notorio de los números en el consumo de recursos, esto es consecuencia de que la aplicación ya empieza a utilizar los modelos CNN's y la cámara del dispositivo. No obstante, actualmente ya existen dispositivos móviles con buenos recursos de hardware, por lo que las aplicaciones pueden ejecutarse sin ningún problema.

En conclusión, después de las pruebas de funcionalidad y rendimiento realizadas, se destaca los resultados favorables de la aplicación, no obstante, como se mencionó anteriormente, la aplicación móvil puede mejorarse aún más.

3.3 Análisis de Resultados.

Es crucial destacar que el objetivo de este proyecto es crear un prototipo de aplicación móvil para el registro de asistencia que permita la reducción de tiempo que se tarda en realizar esa actividad. Es por ello que, durante las pruebas realizadas se ha tomado el tiempo consumido en registrar la asistencia utilizando el método tradicional y la aplicación móvil. Para de esta manera corroborar si es factible utilizar este nuevo método, o caso contrario establecerlo como una opción adicional que facilitaría el proceso.

La Tabla 17 muestra los tiempos obtenidos de cada caso, los cuales fueron registrados empleado un cronómetro. Cabe destacar que dicha Tabla solo muestra el tiempo del registro de la asistencia.

Tabla 17

Tiempos de registro de asistencia.

Método	Tiempo	Observación
Registro de asistencia utilizando el portafolio académico	1.38 min.	Tiempo que el docente registra la asistencia de inicio a fin.
Registro de asistencia utilizando la aplicación móvil, escenario 1	1.14 min.	Tiempo desde que se abre la aplicación desde el teléfono hasta generar el reporte.
Registro de asistencia utilizando la aplicación móvil escenario 2	55.3 seg.	Tiempo desde que se abre la aplicación desde el teléfono hasta generar el reporte.

Nota: Elaboración propia.

La Tabla 18 presenta tiempos individuales de transición de algunos estudiantes. En el caso del registro de asistencia con la aplicación móvil fueron dos escenarios, esas pruebas fueron documentadas en la sección 2.7. El objetivo de esto fue crear una ecuación que permitiese calcular el tiempo necesario para registrar la asistencia con la aplicación ante cualquier situación, ya sea con un número pequeño o grande de estudiantes.

Tabla 18

Tiempos individuales de registro de asistencia.

Método	Tiempo 1	Tiempo 2	Tiempo 3	Tiempo 4	Tiempo 5	Tiempo 6
Tiempos individuales utilizando el portafolio académico	4.49 seg	5.15 seg	5.17 seg	5.39 seg	4.53 seg	5.26 seg
Tiempos individuales utilizando la aplicación móvil, escenario 1	3.58 seg	4.03 seg	3.43 seg	4.00 seg	4.11 seg	4.06 seg
Tiempos individuales utilizando la aplicación móvil escenario 2	2.57 seg	2.55 seg	2.49 seg	2.58 seg	2.48 seg	3.01 seg

Nota: Elaboración propia.

La Tabla 18, mostró los tiempos de registro capturados de algunos estudiantes, esos valores representan el tiempo en el que el docente solo hace referencia a ese estudiante, desde el inicio hasta el inicio del siguiente. El enfoque es la misma en los tres casos.

Sin embargo, hay que destacar que esos tiempos son una estimación, debido a que el tiempo puede variar según las circunstancias del momento. En el primer caso el tiempo puede variar, cuando el docente desea registrar la asistencia del estudiante y este

no responde de inmediato, teniendo que volver a nombrarlo. En el segundo caso el tiempo puede variar, cuando al docente se le dificulte movilizarse por el aula o que la aplicación no reconozca de inmediato al estudiante. En el tercer caso el tiempo puede variar, cuando la aplicación no reconozca de inmediato al estudiante. Aunque los modelos de última generación como las CNN's han logrado una clasificación a nivel humano para determinadas tareas, todavía parecen tener problemas con imágenes de baja resolución (Chacua, 2019).

El tiempo de registro de la asistencia con la aplicación móvil, puede reducirse aún más si se mejora los métodos presentados en este proyecto. Revisar los escenarios de pruebas documentadas en la sección 2.7. Es así como, basado en los tiempos obtenidos en la Tabla 18, se presenta la Ecuación 7, que ayudaría a conocer el tiempo que la aplicación demoraría en registrar la asistencia en un grupo de estudiantes.

Ecuación 7

Cálculo de tiempo para registro de asistencia.

$$T(N) = p\ x\ N$$

Nota: Elaboración propia.

Donde:

N: Número de estudiantes.

P: Tiempo promedio por estudiante: Escenario 1 = 3.49; Escenario 2 = 2.37.

Este proyecto queda abierto para futuros trabajos, debido a que el registro de asistencia mediante la aplicación móvil aún no ha sido terminado por completo. En este trabajo se ha creado una aplicación que registra la asistencia, hasta generar un reporte de los estudiantes identificados, que serviría para regístralos después en el portafolio académico de la Universidad. No obstante, aún falta implementar un módulo que cargue al portafolio académico el reporte generado y que los registre automáticamente.

Bajo esas circunstancias, el tiempo de registro de asistencia tiende a subir. En el caso de registro de asistencia utilizando el método tradicional, el docente tiene que iniciar sesión en el portafolio académico, el cual consta de varios pasos. En cambio, con la aplicación móvil se debe encontrar un método que realice la misma acción.

La Tabla 19 muestra el tiempo que lleva en registrar la asistencia completa de los estudiantes en el método tradicional.

Tabla 19

Tiempos de registro de asistencia por método tradicional.

Método	Tiempo ingreso al portafolio académico	Tiempo registro asistencia	Tiempo Total
Registro de asistencia utilizando el portafolio académico	1.28 min.	1.38 min	3.06 min
Registro de asistencia utilizando la aplicación móvil, escenario 1	X	1.14 min.	X
Registro de asistencia utilizando la aplicación móvil escenario 2	X	55.3. seg	X

Nota: Elaboración propia.

Después de las comparaciones realizada en las Tablas 17 y 18, se observó que el método de registro de asistencia utilizando la aplicación móvil, lleva cierta ventaja al método tradicional. En la Tabla 19 se pretendió realizar una comparación general de los dos métodos, sin embargo, la implementación del registro asistencia en el portafolio académico con la aplicación aún no está implementada. No obstante, para trabajos futuros es un reto lograr que el registro de asistencia con la aplicación móvil sea mejor que con el método tradicional.

Por otra parte, el modelo MobileFaceNet mostró resultados muy buenos en la extracción de características, además, que se ejecutó sin ningún problema en el teléfono.

3.4 Limitaciones del Sistema.

Las limitaciones son restricciones que pueden afectar el funcionamiento y la eficacia de una aplicación, por lo tanto, es importante considerarlas antes de tomar una decisión.

La aplicación se ejecutó en dos dispositivos móviles diferentes para comprobar la funcionalidad y la eficiencia. Allí se observó ciertas limitaciones pueden impactar al correcto funcionamiento.

- **Cámara del Dispositivo.**

Si la imagen capturada a través de la cámara del teléfono no es de buena calidad puede afectar la precisión en el reconocimiento facial. Esto se debe a que las capas del modelo encargadas de la extracción de características pueden complicarse a la hora de procesar, si la imagen es de baja resolución. Aunque se logre la detección facial en imágenes de baja calidad no es garantía de que se logre la correcta identificación del individuo. En la Figura 46, se presenta como fue la calidad de imagen que ingresa al modelo utilizando dos teléfonos diferentes, la primera de gama baja y la segunda de gama media.

Figura 47

Captura de rostro realizadas con el Teléfono.

a) Tecno Spark 10C b) Honor 90

Nota: Elaboración propia.

La Figura (42a) fue el resultado de utilizar una cámara de 16 MP, mientras que la Figura (42b) fue tomada con el teléfono especificado en la Tabla 13. Con una cámara de alta resolución se captura más detalle en una imagen, por ende, mejora la precisión en la identificación. Pero incluso con estas especificaciones, la aplicación aún podría encontrar dificultad en la identificación, esta vez asociadas a condiciones de luz o movimientos del rostro.

- **Variación en la Posición.**

Para solventar este problema se recomienda utilizar DataAumentation durante el entrenamiento de los modelos CNN's. Este método ayuda a generar imágenes aleatorias variadas desde una imagen base. Sin embargo, cuando hay cambios de apariencia y variación moderada del individuo, es difícil que el sistema sea tan preciso.

El reconocimiento facial funciona correctamente en la aplicación móvil, tan solo con registrar al estudiante con sus credenciales y una fotografía. Sin embargo, se recomienda registrar al estudiante con diversas fotografías en diferentes posiciones, de esta manera el modelo tendrá más opciones con las que puede comparar y así dar un resultado más certero.

- **Dispositivos con bajos recurso de hardware.**

Además de la cámara, otra limitación en dispositivos móviles son los recursos del hardware especialmente en términos de procesamiento y memoria. Esto puede afectar la eficiencia general de la aplicación y el consumo de la batería. Es importante comprender que la aplicación utiliza dos modelos convoluciones, la primera para la detección facial (MTCNN) y la segunda para la extracción de características (MobileFaceNet). Con el segundo modelo no hay inconvenientes, debido a que es una red ligera diseñada específicamente para dispositivos móviles. El problema surge con el primer modelo, debido a que es un modelo que cuenta con tres redes internas profundas. En esta situación, el dispositivo no es capaz de mantener el ritmo para procesar las imágenes en tiempo real, lo que resulta parecido a una grabación en cámara lenta. En la Tabla 20 se muestra cómo influye la capacidad de recursos de hardware en la ejecución de la aplicación en dos dispositivos.

Tabla 20

Cálculo de tiempo entre dispositivos.

Procesos	Tecno Spark 10C	Honor 90
Abrir la aplicación	1.69 segundos	0.93 segundos
Preparación para detección facial	Cargar modelo 1.85 segundos	Cargar modelo 0.33 segundos
Frecuencia de actualización	0.8 seg. durante 4 segundos	0.4 seg. durante 1 segundos

Nota: Elaboración propia.

Basado en la Tabla 20, un teléfono con buenos recursos de hardware presenta mejores resultados que con un teléfono de bajos recursos.

Capítulo 4.

Conclusiones y Recomendaciones.

Conclusiones

Actualmente el reconocimiento facial es un campo bastante aplicado en distintas áreas laborales; investigadores han realizado diferentes aplicaciones de esta tecnología. Para el desarrollo de este proyecto, el marco teórico ha establecido una comprensión de base sólida, que permitió contextualizar, adquirir nuevos conocimientos y fundamentar el trabajo; proporcionando conocimiento teórico basado en trabajos anteriores. Los trabajos relacionados ayudaron a seleccionar la técnica más adecuada, con el cual se pudo resolver el problema planteado en este proyecto; es así como se estableció un esquema de confianza para seleccionar a MobileFaceNet como una solución eficiente para el desarrollo de la aplicación de reconocimiento facial, debido a que esta tecnología se ejecuta sin problemas en dispositivos móviles.

La elección de modelo MobileFaceNet fue la mejor elección que se realizó para el reconocimiento facial, debido a que este modelo se ejecuta ligeramente y tiene una excelente precisión en la extracción de incrustaciones faciales en dispositivos móviles. Sin embargo, fue la tarea que más tiempo consumió, ya que su implementación no es la misma que la de Python, en la que casi todo tiene un método ya desarrollado, pues en Java se requieren una serie de operaciones para obtener el mismo resultado.

Los métodos analizados para la detección facial fueron Haar Cascade y MTCNN, sin embargo, el segundo método evidenció mejores resultados, que la primera. No existe inconveniente en utilizar Haar Cascade, no obstante, este detector es más propenso a detectar falsos positivos; eso se evidenció en la aplicación. MTCNN ofrece excelentes resultados, aunque es una red bastante profunda, por lo que, si se tiene la facilidad de utilizar un dispositivo móvil con buenos recursos de hardware, el modelo sería una excelente elección.

La aplicación de la metodología CRISP-DM y la metodología en Cascada, permitieron seguir un conjunto de buenas prácticas por separado que ayudaron a: planificar, diseñar y construir de manera organizado la aplicación móvil de reconocimiento facial, contribuyendo a evitar posibles errores a futuro.

Con los resultados obtenidos de las pruebas realizadas a la aplicación móvil de reconocimiento facial se puede apreciar que, este método de registro de asistencia es más efectivo que el método tradicional en términos de consumo de tiempo. El tiempo de registro con la aplicación se reduciría aún más de los resultados obtenidos en las pruebas realizadas, utilizando otras técnicas como, agrupamiento de estudiantes. Es así como el presente proyecto es una solución viable, para en un futuro considerar cambiar el método de registro de asistencia en la Universidad Técnica del Norte.

El reporte generado de los estudiantes identificados por la aplicación móvil, los cuales se consideran, estudiantes que asistieron a clases, se guarda en el directorio del mismo teléfono. Ese reporte es el documento que se debe cargar al portafolio académico de la Universidad para registrar la asistencia completamente, sin embargo, ese parte no ha sido incluida en el alcance de este proyecto. Por ende, este proyecto que abierto para futuros trabajos.

Después de las pruebas realizadas se evidenció que la aplicación presenta mejores resultados de eficiencia, al implementarlos en dispositivos móviles con buenos recursos de hardware. Así mismo, la capacidad de la cámara cumple un factor muy importante a la hora de presentar los resultados.

La aplicación efectiva de la Guía SWEBOK y las pruebas de caja negra ofrecen un método simple de evaluación del sistema. SWEBOK es un documento diseñado para guiar la educación y formación en la ingeniería de software, no obstante, su alcance es bastante amplio por lo que, puede ser utilizado como marco de referencia para evaluar cualquier proyecto de software. Al incorporar pruebas de caja negra, se garantiza una evaluación efectiva de la funcionalidad externa de un sistema, sin requerir conocimiento interno del código.

Recomendaciones.

Con el cambio acelerado de la tecnología en la actualidad, es fundamental conocer los fundamentos teóricos antes utilizar algoritmos de inteligencia artificial en trabajos grandes. Estos conocimientos, darán al desarrollador las herramientas necesarias para anticiparse a posibles desafíos que puedan surgir. Afortunadamente existe mucha información en línea que se puede utilizar.

Se recomienda no utilizar la aplicación móvil a más de 3 metros de distancia, ya que la cámara del dispositivo no captura correctamente los rostros y eso podría generar

falsos positivos. Cabe desatacar que los estudiantes identificados se almacenan en una lista, y en base a esa lista se genera el reporte. Por ende, no debe existir margen de error.

El umbral establecido es de 0.8, con ese valor se realiza correctamente la comparación de las incrustaciones faciales, por ende, no se recomienda subir o bajar el valor. Un umbral demasiado alto podría aumentar la sensibilidad, pero también aumentarían los falsos positivos, en cambio un umbral demasiado bajo aumentaría los falsos negativos.

Se recomienda mantener lo más estable posible el dispositivo móvil durante el proceso de reconocimiento facial. Esto permitirá a la aplicación capturar mejor la imagen y procesarlos lo más rápido posible. Caso contrario la aplicación tardará en lograr identificar al individuo, debido a que las imágenes que estaría capturando tendrían movimientos.

Antes de seleccionar un modelo CNN para utilizarlo como solución a cualquier problema planteado, es recomendable también conocer otras alternativas relevantes. La finalidad de esto es comprender cómo ha sido su desempeño en trabajos anteriores, y así corroborar que la elección seleccionada sea la más factible. La investigación permite conocer como se ha desenvuelto el modelo en diferentes entornos para así constatar que cumple con los requerimientos establecidos. Esto al final facilita su evaluación.

Capítulo 5.

Análisis Económico.

Este capítulo presenta el análisis económico del proyecto: *Aplicación móvil de reconocimiento facial para el control de asistencia a clases de los estudiantes de la Universidad Técnica del Norte utilizando técnicas de Inteligencia Artificial*, en el cual se considera los montos referenciales del presupuesto invertido en los diferentes componentes de hardware y software. En la Tabla 21 se indican los elementos físicos utilizados.

Tabla 21

Gastos referenciales de hardware.

Descripción	Cantidad	Valor Unitario	Valor Total
Laptop	1	0	0
Cámara Canon Rebel EOS T5	1	0	0
Trípode	1	25	25
Teléfono móvil	1	430	430
		TOTAL	455

Nota: Elaboración propia.

En la Tabla 21 se indican los recursos de software utilizados.

Tabla 22

Gastos referenciales de software.

Descripción	Cantidad	Valor Unitario	Valor Total
Plan Google Colab PRO+	1	100	100
		TOTAL	100

Nota: Elaboración propia.

En total para el desarrollo de este proyecto se invirtió alrededor de 555 dólares, sumando el total de los gatos de elementos de hardware y de software.

Referencias

ADP. (2019). *Qué es Machine Learning, cómo funciona y a qué se aplica*.

Albahrani, A. (2022). *Smart Attendance Management System*. https://www.researchgate.net/publication/361607151_Smart_Attendance_Management_System

alfanoTV. (2020). *Screen Stream Over HTTP para Android – alfanoTV*. https://alfanotv.com/apps/screen-stream-over-http-para-android/

Arana, C. (2021). *Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales*.

Asmara, R. A., Sayudha, B., Mentari, M., Budiman, R. P. P., Handayani, A. N., Ridwan, M., & Arhandi, P. P. (2022). Face Recognition Using ArcFace and FaceNet in Google Cloud Platform For Attendance System Mobile Application. In *Proceedings of the 2022 Annual Technology, Applied Science and Engineering Conference (ATASEC 2022)* (pp. 134–144). Atlantis Press International BV. https://doi.org/10.2991/978-94-6463-106-7_13

Barrera, J.-, & Barrera, K. (2022). *DESARROLLO DE UNA APLICACIÓN HÍBRIDA INTELIGENTE Y UN MODELO DE ANÁLISIS DE DATOS EN FASES PARA RESUMIR Y REPRESENTAR LOS COMENTARIOS DE CLIENTES SOBRE PRODUCTOS Y SERVICIOS DE BARES Y RESTAURANTES MEDIANTE DJANGO, IONIC, ANGULAR, DEEP LEARNING Y REDES SOCIALES*.

Beunza, J.-, & Puertas, E. (2019). *Manual práctico de inteligencia artificial en entornos sanitarios*.

Bonilla, C. (2020). *REDES CONVOLUCIONALES*.

Bressler, N. (2022). *How to Check the Accuracy of Your Machine Learning Model*. https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/

Cardenas, M. (2017). *Evaluación de metodologías de desarrollo de Software utilizadas para valorar el impacto de un proyecto de software, durante un semestre de clases*.

Cardona, A.-, & Pineda, F. (2018). Reconocimiento de rostros en tiempo real sobre dispositivos móviles de bajo costo. *Lámpsakos*, *20*, 30–39. https://doi.org/10.21501/21454086.2938

Caro, D.-, & López, A. (2018). *Sistema Inteligente para el Registro de Asistencia Basado en Procesamiento Digital de Imágenes y Redes Neuronales Convolucionales*.

Casado, N. (2022). *REDES NEURONALES CONVOLUCIONALES Y APLICACIONES*.

Centro Educativo La Amistad. (2023). *La Puntualidad habla de nosotros*.

Chacua, B. (2019). *Diseño de un sistema prototipo de reconocimiento facial para la identificación de personas en la Facultad de Ingeniería en Ciencias Aplicadas (FICA) de la Universidad Técnica del Norte utilizando técnicas de Inteligencia Artificial*.

Chen, J. (2018). *MTCNN4Android*. https://github.com/vcvycy/MTCNN4Android/tree/master

Chen, S., Liu, Y., Gao, X., & Han, Z. (2018). MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10996 LNCS*, 428–438. https://doi.org/10.1007/978-3-319-97909-0_46

DataScientest. (2021). *Convolutional Neural Network : definición y funcionamiento*.

Developers. (2023). *Perfila el rendimiento de tu app | Android Studio | Android Developers*. https://developer.android.com/studio/profile?hl=es-419

Domínguez, S. (2017). *Reconocimiento facial mediante el Análisis de Componentes Principales (PCA)*.

Ehtesham, Z. (2022). *Explained: MobileNet V1*. *MobileNet V1*. https://medium.com/mlearning-ai/explained-mobilenet-v1-bb700abfe824

Ekanayake, B. (2022). A DEEP LEARNING-BASED BUILDING DEFECTS DETECTION TOOL FOR SUSTAINABILITY MONITORING. *World Construction Symposium*, 2–13. https://doi.org/10.31705/WCS.2022.1

Espinoza, D. (2015). *Reconocimiento Facial*.

Gao, J.-, & Yang, T. (2022). Face detection algorithm based on improved TinyYOLOv3 and attention mechanism. *Computer Communications*, *181*, 329–337. https://doi.org/10.1016/J.COMCOM.2021.10.023

Garcés, A. (2017). *SISTEMA DE RECONOCIMIENTO FACIAL CON VISIÓN ARTIFICIAL PARA APOYAR AL ECU-911 CON LA IDENTIFICACIÓN DE PERSONAS EN LA LISTA DE LOS MÁS BUSCADOS*.

Gonzáles, L. (2022). *Herramientas de Python para Machine Learning.*

González, R. (2020). *"Sistema de Reconocimiento Facial con Deep Learning."*

Gradilla, R. (2020). *Multi-task Cascaded Convolutional Networks (MTCNN) for Face Detection and Facial Landmark Alignment | by Rosa Gradilla | Medium*. https://medium.com/@iselagradilla94/multi-task-cascaded-convolutional-networks-mtcnn-for-face-detection-and-facial-landmark-alignment-7c21e8007923

Hangaragi, S.-, Singh, T.-, & N, N. (2023). Face Detection and Recognition Using Face Mesh and Deep Neural Network. *Procedia Computer Science*, *218*, 741–749. https://doi.org/10.1016/J.PROCS.2023.01.054

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. http://arxiv.org/abs/1502.01852

Hollemansby, M. (2018). *MobileNet version 2*. https://machinethink.net/blog/mobilenet-v2/

IBM. (n.d.). *¿Qué es machine learning?* Retrieved January 22, 2024, from https://www.ibm.com/mx-es/topics/machine-learning

Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. *Procedia Computer Science*, *132*, 679–688. https://doi.org/10.1016/J.PROCS.2018.05.069

Jiménez, I. (2018). *Reconocimiento facial basado en redes neuronales convolucionales*.

Koné, D. (2021). *High Availability Systems*.

Kumar, B. A., & Bansal, M. (2023). Face Mask Detection on Photo and Real-Time Video Images Using Caffe-MobileNetV2 Transfer Learning. *Applied Sciences 2023, Vol. 13, Page 935*, *13*(2), 935. https://doi.org/10.3390/APP13020935

Linkedin. (2023a). *"Pruebas de rendimiento: Cómo medir, identificar y optimizar el rendimiento de una aplicación."* https://es.linkedin.com/pulse/pruebas-de-rendimiento-c%C3%B3mo-medir-identificar-y-optimizar-el-una

Linkedin. (2023b). *Y tú qué eliges, metodología ágil o de cascada*. https://es.linkedin.com/pulse/y-t%C3%BA-qu%C3%A9-eliges-metodolog%C3%ADa-%C3%A1gil-o-de-cascada-practum

Lukkarinen, A., Koivukangas, P., & Seppälä, T. (2016). Relationship between Class Attendance and Student Performance. *Procedia - Social and Behavioral Sciences*, *228*, 341–347. https://doi.org/10.1016/J.SBSPRO.2016.07.051

Made, I., Sandhiyasa, S., & Waas, D. V. (2023). Real Time Face Recognition for Mobile Application Based on Mobilenetv2. *Jurnal Multidisiplin Madani*, *3*(9), 1855–1864. https://doi.org/10.55927/MUDIMA.V3I9.5924

Matlab. (2022). *Autoencoders (Autocodificadores)*.

Méndez, N.-, Nicolás, M. A.-, & Méndez, H. (2019). Frontalización de imágenes de rostro de perfil basada en puntos característicos y en el uso de un Modelo 3D Genérico Elástico. *Ingeniería Electrónica, Automática y Comunicaciones*, *40*(3), 72–78.

Microsoft Azure. (n.d.). *Algoritmos de aprendizaje automático*. Retrieved January 22, 2024, from https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/what-are-machine-learning-algorithms

Muñoz, E. (2021). *Desarrollo de un sistema de control de acceso de personal empleando reconocimiento facial respaldado con técnicas de aprendizaje profundo*.

Pérez, F., Olivares-Mercado, J., Sanchez-Perez, G., Benitez-Garcia, G., Prudente-Tixteco, L., & Lopez-Garcia, O. (2023). Analysis of Real-Time Face-Verification Methods for Surveillance Applications. *Journal of Imaging*, *9*(2). https://doi.org/10.3390/JIMAGING9020021

PMOinformatica. (2017). *Pruebas de caja negra: Ejemplos - La Oficina de Proyectos de Informática*. http://www.pmoinformatica.com/2017/02/pruebas-de-caja-negra-ejemplos.html

Pokhrel, S. (2020). *Did you know Facial Recognition existed in 1960s?* https://medium.com/analytics-vidhya/did-you-know-facial-recognition-existed-in-1960s-2f0961e1bf5b

Portilla, L. (2018). *Detector de alcoholemia para conductores que analiza variables faciales y ambientales del automóvil mediante el aprendizaje automático supervisado para la reducción de accidentes de tránsito*.

Quintero, M. (2020). *Análisis comparativo de técnicas de deep learning para el reconocimiento de rostros en escenarios abiertos*.

Quisaguano, L., Camalle, T. G., & Toca, J. W. (2022). Análisis comparativo de entornos de desarrollo móvil. *Ciencia Latina Revista Científica Multidisciplinar*, *6*(4), 4478–4498. https://doi.org/10.37811/cl_rcm.v6i4.2950

Quishpe, D. (2013). *LA PUNTUALIDAD ESCOLAR Y SU INCIDENCIA EN EL APRENDIZAJE DEL CUARTO GRADO DE EDUCACIÓN BÁSICA DE LA ESCUELA FISCAL MIXTA MANUELA JIMÉNEZ DEL CANTÓN PILLARO PROVINCIA DE TUNGURAHUA*.

Ringa Tech. (2022). *Funciones de activación a detalle (Redes neuronales)*. https://www.youtube.com/watch?v=_0wdproot34

Rodríguez, I. (2022). *Evaluación de distintas arquitecturas de redes neuronales aplicadas a la clasificación de datos tabulares*.

Rojas, U.-, Goñi, J.-, & Paredez, F. (2021). *Reconocimiento de expresiones faciales y características personales como herramienta para identificar personas en un sistema de transporte público*.

Rosero, S. (2019). *Reconocimiento facial: técnicas tradicionales y técnicas de aprendizaje profundo, un análisis*.

Rouhiainen, L. (2018). *INTELIGENCIA ARTIFICIAL 101 COSAS QUE DEBES SABER HOY SOBRE NUESTRO FUTURO INTELIGENCIA ARTIFICIAL*.

Sabeenian, D., Aravind, S., Arunkumar, P., Harrish Joshua, P., & Eswarraj, G. (2020). Smart Attendance System Using Face Recognition. *Article in Journal of Advanced Research in Dynamical and Control Systems*, *12*, 2020. https://doi.org/10.5373/JARDCS/V12SP5/20201860

Saleem, S., Shiney, J., Priestly Shan, B., & Kumar Mishra, V. (2021). Face recognition using facial features. *Materials Today: Proceedings*. https://doi.org/10.1016/J.MATPR.2021.07.402

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018a). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. http://arxiv.org/abs/1801.04381

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018b). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. http://arxiv.org/abs/1801.04381

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). *FaceNet: A Unified Embedding for Face Recognition and Clustering*. https://doi.org/10.1109/CVPR.2015.7298682

Shaurya, G. (2019). *PReLU activation*. 2019. https://medium.com/@shauryagoel/prelu-activation-e294bb21fefa

Shukla, R. K., & Tiwari, A. K. (2022). Masked Face Recognition Using MobileNet V2 with Transfer Learning. *Computer Systems Science and Engineering*, *45*(1), 293–309. https://doi.org/10.32604/CSSE.2023.027986

Shun, M., & Aung, O. (2019). *Child Face Recognition System Using Mobilefacenet.pdf*. 2019. https://meral.edu.mm/record/6187/preview/%20Child%20Face%20Recognition%20System%20Using%20Mobilefacenet.pdf

sirius-ai. (2019). *MobileFaceNet_TF: Tensorflow implementation for MobileFaceNet*. https://github.com/sirius-ai/MobileFaceNet_TF?tab=readme-ov-file

Swebok. (2014). *Guide to the Software Engineering Body of Knowledge SWEBOK ® A Project of the IEEE Computer Society*.

Untuña, V. (2022). *SISTEMA DE CONTROL DE ACCESO POR MEDIO DE RECONOCIMIENTO FACIAL CON USO DE MASCARILLA Y MONITOREO DE TEMPERATURA*.

UTN. (2019). *REGLAMENTO DE REGIMEN ACADEMICO UTN*. https://legislacion.utn.edu.ec/wp-content/uploads/2019/01/REGLAMENTO-DE-REGIMEN-ACADEMICO-UTN.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, *2017-December*, 5999–6009. https://arxiv.org/abs/1706.03762v5

Vijay, K. (2022). *All You Need to Know About Support Vector Machines*.

Viola P, & Jones M. (2001). *Robust Real-time Object Detection. SECOND INTERNATIONAL WORKSHOP ON STATISTICAL AND COMPUTATIONAL THEORIES OF*. https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-IJCV-01.pdf

Zeeshan, E. (2020). *How To Use And Configure CameraX In Android Applications*. https://zeeshan-elahi.medium.com/how-to-use-and-configure-camerax-in-android-applications-15d1db2576e5

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2015). *Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks*. https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf