Universidad
Tecnológica
de Pereira

PROGRAMA DE INGENIERÍAS

TRABAJO DE GRADO

RECONOCIMIENTO FACIAL PARA LA AUTOMATIZACIÓN DEL REGISTRO DE ASISTENCIA A CLASES.

Diego Fernando Legarda Delgado

Oscar Andrés Loaiza Pabón

Director

Hernán Felipe García Arias

Ph.D. en Ingeniería

Abril de 2022

Resumen

En el presente trabajo se realizó el desarrollo del sistema piloto de registro automático de asistencia a clases presenciales FR-ARCA, basado en técnicas de detección y reconocimiento facial, utilizando modelos de aprendizaje profundo. Se diseñó el modelo del sistema con una arquitectura modular de diferentes herramientas integradas adecuadamente para resolver las consideraciones previstas. El uso y la integración de contenedores, el lenguaje de programación Python con los frameworks FastAPI y Django, el uso de frameworks para Machine Learning como Keras, TensorFlow, PyTorch, OpenCV y MXNet y protocolos de comunicación REST y MQTT permitieron superar retos que tiene la ingeniería de software para implementar inteligencia artificial (IA) en software de producción. Con la extracción de características principales de rostro y su almacenamiento en bases de datos (DB) se realiza la identificación de identidades a través del cálculo de similitud entre los vectores de características (embeddings) por lo cual no se necesita reentrenar las redes neuronales convolucionales cuando ingresan nuevos aprendices a la institución. Se probaron modelos preentrenados de detección de rostros como MTCNN, RetinaFace y SCRFD, y modelos de reconocimiento de rostros como VGG-Face, FaceNet y ArtFace, en el sistema FR-ARCA y fueron evaluados con experimentos que permitieron validar y configurar el sistema para lograr excelentes resultados en la exactitud del registro de asistencia a clases.

Agradecimientos

Este trabajo no se habría podido realizar sin la colaboración de muchas personas que nos han brindado su ayuda y sus conocimientos.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

INTRODUCCIÓN

El Servicio Nacional de Aprendizaje SENA en sus clases presenciales debe registrar asistencia de los aprendices según articulo 15 numeral 1 del reglamento del aprendiz (Avance Jurídico Casa Editorial Ltda, 2022). Para ello actualmente se usan métodos tradicionales como registrar manualmente la asistencia en formato físico, llamar nombres, firmar documentos, archivos digitales y uso de la plataforma institucional; estos métodos consumen tiempo de clase y no permiten unificar eficientemente la información recolectada, lo cual dificulta descubrir información que puede resultar útil para mejorar el bienestar de los aprendices y los procesos de la formación profesional integral.

Los avances significativos en tecnologías de la inteligencia artificial, especialmente en visión por computadora, han incrementado su importancia en muchas áreas y se ha considerado un campo de investigación interesante en los últimos años (Du, Shi, Zeng, y Mei, 2020). El reconocimiento facial es una técnica biométrica que debido a su bajo nivel de intrusión, rapidez y comodidad de uso, cada vez tiene mayor presencia en la automatización de procesos de seguridad como la autenticación en los dispositivos móviles, en los aeropuertos para el control migratorio, como también para registros de asistencia en entornos laborales o institucionales. (Ortiz, Hernandez Beleño, Moreno, Mauledoux, y Avilés Sánchez, 2018). Estos métodos son cada vez más precisos con el surgimiento de nuevas técnicas, pero su adopción todavía está en desarrollo.

La solución propuesta en este proyecto usa herramientas de Deep Learning, especialmente las redes neuronales convolucionales (CNN), las cuales se han convertido en una de las técnicas más populares para resolver problemas relacionados con clasificación de imágenes, detección de objetos, reconocimiento de rostros entre otros (Wu, He, Sun, y Tan, 2015).

En este trabajo se presenta el proceso a través del cual se desarrolló el sistema piloto de registro automático de asistencia a clases presenciales basado en técnicas de detección y reconocimiento facial, utilizando modelos de aprendizaje profundo para el Centro Atención Sector Agropecuario SENA de Risaralda, al cual llamamos FR-ARCA por sus siglas en inglés Facial Recognition for Automatic Registration of Class Attendance. El cual permite el registro de asistencia a clases en ambientes de formación internos como externos, ya que incluye módulos que permiten usar diferentes métodos de captura y envío de imágenes a la API de detección y reconocimiento de rostros.

En el primer capítulo se describe y formula el problema de investigación propuesto en este proyecto: ¿Cómo implementar un sistema automático de registro de asistencia a clases presenciales para el Centro Atención Sector Agropecuario SENA, a través de tecnologías de detección y reconocimiento facial?. Con base a ello se formulan los objetivos, se presenta la justificación y el alcance de la investigación.

En el segundo capítulo se presenta el estado del arte de los principales conjuntos de datos, modelos de detección y reconocimiento de rostros como también la descripción de diferentes tipos de sistemas de reconocimiento de rostros, herramientas y soluciones del mercado. Por último, se describen algunos retos de la ingeniería de software (SE) aplicada al machine learning (ML).

En el capítulo tres se presenta el marco teórico, en el cual se describen las bases de la investigación. Estas incluyen las variables principales del estudio: Detección y reconocimiento de rostros, Inteligencia Artificial (IA), redes neuronales convolucionales (CNN), aprendizaje de una sola vez, medidas de similitud. Así mismo, se describen los otros elementos del proceso de desarrollo de software: Metodologías de desarrollo, protocolos de trasferencia de datos e infraestructura y alojamiento.

El cuarto capítulo presenta la metodología de desarrollo de software para el análisis, diseño y desarrollo del sistema FR-ARCA como también la caja de herramientas utilizada en cada uno de los módulos que lo componen. En las secciones de preparación de datos, experimentos y métricas se describen las metodologías usadas para realizar la evaluación al sistema desarrollado FR-ARCA.

En el capítulo quinto, inicialmente se presenta el modelo realizado para el sistema piloto de registro automático de asistencia a clases presenciales basado en técnicas de detección y reconocimiento facial, FR-ARCA, luego se describen las principales consideraciones y las propuestas de solución que se tuvieron en cuenta para el diseño y desarrollo del sistema, como también las técnicas implementadas para realizar el sistema de reconocimiento facial sin reentrenamiento de redes neuronales convolucionales (CNN). Se presentan los modelos de detección y reconocimiento de rostros seleccionados en este proyecto con la descripción de sus características principales y la arquitectura del sistema FR-ARCA la cual describe cada uno de los módulos propuestos. Asimismo se presentan los diseños que describen los diagramas de flujo para los procesos principales del sistema FR-ARCA.

En el capítulo sexto se presentan, analizan y discuten los resultados obtenidos en la serie de experimentos realizados con el objetivo de medir la exactitud del sistema, al identificar los aprendices para registrar su asistencia a clases. Adicionalmente, se muestran y explican las interfaces de usuario y los módulos del sistema desarrollado.

En el capítulo siete se resaltan las conclusiones y se realizan propuestas para futuras investigaciones. Finalmente, se incluyen anexos que complementarán la información con detalle representativo de lo descrito durante la investigación.

1.1. DESCRIPCIÓN DEL PROBLEMA

En las instituciones educativas, por lo general, el registro de asistencia a clases se realiza de formas tradicionales como: formatos en físico y múltiples formatos digitales individuales, lo que no permite disponer fácilmente de la información detallada y consolidada de la asistencia de los aprendices, para ser analizada de manera oportuna. Es clave para la administración académica de una institución educativa, resolver interrogantes como: ¿Cuántos y cuáles aprendices asistieron a la clase? ¿Cuántos y cuáles aprendices llegan a tiempo a la hora de inicio a la clase? ¿Cuánto tiempo se tardan en llegar? ¿Cuántos y cuáles aprendices se salen antes que termine la clase?

El Servicio Nacional de Aprendizaje SENA (Congreso de la república de Colombia, 1994), tiene 33 regionales en el país, en las cuales se distribuyen 117 centros de formación especializados en diferentes líneas tecnológicas. En Risaralda existen tres, el Centro Atención Sector

Agropecuario, el Centro de Comercio y Servicios y el Centro de Diseño e Innovación Tecnológica Industrial. Nuestro caso de estudio sucede en el Centro Atención Sector Agropecuario, en el cual, para el mes de mayo del 2020 según la base de datos SofiaPlus se tiene un total de 286 grupos activos en modalidad presencial y 153 Instructores.

El SENA en sus clases presenciales debe registrar asistencia de los aprendices, según articulo 15 numeral 1 del reglamento del aprendiz (Avance Jurídico Casa Editorial Ltda, 2022), puesto que las inasistencias del aprendiz generan acciones correctivas definidas por la institución para garantizar la calidad en los procesos formación. Para ello actualmente se usan métodos tradicionales como registrar la asistencia en formato físico, archivos digitales y plataforma institucional.

Se realizó una encuesta voluntaria en la cual respondieron 111 instructores del Centro Atención Sector Agropecuario (Legarda, D. Loaiza, O, 2020), al analizar las respuestas se obtuvieron los siguientes resultados:

- El 75,7 % registra asistencia siempre, 17,1 % casi siempre, 6,3 % pocas veces y 0,9 % nunca.
- El 42,1 % de los instructores utiliza un medio físico para registrar la asistencia.
- El 57,9 % de los instructores registra la asistencia en múltiples medios digitales.
- El 46,8 % registra si el aprendiz llega a tiempo, el 48,6 % registra cuando un aprendiz se retira de clase,
- El 13,1 % de los instructores registra la asistencia a clases en la plataforma SenaSofiaPlus.
- El 32,4 % de instructores del equipo ejecutor entregan un informe del control de asistencias al coordinador del grupo, de forma periódica.
- El 27,9 % de los instructores conoce la asistencia de los aprendices en las clases de los otros instructores siendo parte del grupo ejecutor de la formación.
- El 36,9 % consolida el registro de asistencias a clases de un grupo de forma periódica.

Para obtener información confiable del registro de la asistencia a clases, esta debe registrarse siempre y en el momento en que se imparte la formación. Además, la información

preferiblemente debe estar centralizada para facilitar su análisis y trazabilidad, también es importante contar con copias de seguridad como medios de respaldo.

Por lo descrito anteriormente se evidencia que el Centro Atención Sector Agropecuario SENA de Risaralda, no cuenta con un método de registro de asistencia a clases que permita unificar la información y esto dificulta investigar diferentes variables para descubrir información que puede resultar útil para mejorar el bienestar de los aprendices y los procesos de la formación profesional integral.

1.2. FORMULAR EL PROBLEMA

Múltiples estudios (Ojala, Pietikainen, y Harwood, 1994) (Heikkilä, Pietikäinen, y Schmid, 2009) (Kaur, Nazir, y Manik, 2021) demuestran que la asistencia a clases influye en variables como la recepción correcta de los contenidos y la retención. Algunas de las principales consecuencias que han sido asociadas al ausentismo o inasistencia, refieren un impacto directo en el desempeño académico del estudiante, afectando la adquisición de conocimientos y habilidades matemáticas, verbales y de orden complejo.

El registro de asistencia a clases que actualmente se lleva en el Centro Atención Sector Agropecuario SENA de Risaralda, dificulta obtener información detallada, consolidada y en tiempo oportuno. Este problema se podría solucionar con una herramienta que registre automáticamente la asistencia a clases presenciales en bases de datos para posteriormente ser consultada.

Para automatizar el registro de la asistencia de los aprendices, se necesitaría la identificación del individuo, lo cual se puede lograr utilizando sistemas de reconocimiento biométricos como: huella dactilar, geometría de la mano, iris, voz, rostro, entre otros. De los anteriores, el reconociendo facial detecta e identifica a la persona sin intervención humana, logrando que el proceso de identificación pueda automatizarse.

El reconocimiento facial es un problema interesante y desafiante ya que permite el desarrollo de aplicaciones importantes en muchas áreas, como la autenticación para el acceso a los sistemas de seguridad, la identificación personal, reconocimiento de delincuentes, entre otros (Kirchberg, Jesorsky, y Frischholz, 2002). La identificación de características faciales

(Dubuisson y Jain, 1994) ha recibido un fuerte impulso gracias al avance en la tecnología de vídeo multimedia, propiciándose así un aumento de cámaras en los lugares de trabajo, hogar y dispositivos móviles con un reducido costo.

Lo anterior da pie al problema de investigación propuesto en este proyecto: ¿Cómo implementar un sistema de registro automático de asistencia a clases presenciales, para el Centro Atención Sector Agropecuario SENA de Risaralda, a través de tecnologías de detección y reconocimiento facial?

1.3. OBJETIVOS DE LA INVESTIGACIÓN

1.3.1. Objetivo General

Desarrollar un sistema piloto de registro automático de asistencia a clases presenciales basado en técnicas de detección y reconocimiento facial, utilizando modelos de aprendizaje profundo. Caso de Estudio Centro Atención Sector Agropecuario SENA – Risaralda.

1.3.2. Objetivos Específicos

1. Implementar un modelo automático de aprendizaje profundo para la detección y reconocimiento de rostros que permita identificar la identidad del aprendiz SENA del Centro Atención Sector Agropecuario - Risaralda.
2. Construir un sistema piloto para registro de nuevos grupos, registro automático de asistencia del aprendiz y generación de reportes de asistencia a clases presenciales, que utilice el modelo de reconocimiento facial para la validación de identidad.
3. Evaluar el desempeño del sistema de registro automático de asistencia a clases utilizando casos de pruebas simulados y entornos reales que permitan tener un valor agregado en la exactitud de los reconocimientos de identidad y disminuyan los casos de falsos positivos derivados de reconocimientos erróneos.

1.4. JUSTIFICACIÓN DE LA INVESTIGACIÓN

La cuarta revolución industrial, está generando una transformación en la humanidad, sin precedentes, obligando a cambiar la forma en que se vive, se trabaja y se relacionan los unos

con los otros. Ésta se diferencia de las anteriores revoluciones por la velocidad, amplitud, profundidad e impacto (Lin, Kung, y Lin, 1997).

La manera como las personas acceden a la información, se educan, trabajan, generan ingresos y se interrelacionan, está siendo transformada radicalmente debido a las Tecnologías de la Información y las Comunicaciones (TIC). Para estar a la vanguardia, las instituciones deben considerar la investigación y el uso de innovaciones que llegan con la cuarta revolución industrial, como la robótica, las impresiones 3D, la automatización, inteligencia artificial, big data, entre otras. Estas innovaciones no solo ayudan a mejorar procesos, sino también al descubrimiento de conocimientos ocultos.

Como se menciona en la descripción del problema, el proceso de registro de asistencia a clases que actualmente se realiza en el Centro Atención Sector Agropecuario Regional Risaralda, no provee datos consolidados, que faciliten investigar diferentes variables como tendencias comportamentales, patrones y correlaciones; en general, descubrir información adicional que puede resultar útil para mejorar el bienestar de los aprendices y los procesos de la formación profesional integral. Por lo tanto, se hace necesario en la institución construir una herramienta tecnológica que permita registrar la asistencia a clases evitando las anteriores desventajas del actual proceso.

El rostro contiene un conjunto de rasgos altamente discriminativos, por el cual se pueden identificar individuos a simple vista y que ha sido utilizado para determinar la identidad en diversos ámbitos. La capacidad de automatizar este proceso consiguiendo resultados incluso mejores que los humanos ha sido el objeto de numerosas investigaciones, obteniéndose así numerosas técnicas de reconocimiento que convierten al reconocimiento facial en un sistema bastante fiable. (Ordieres-Meré y cols., 2006) Por ejemplo, Google presenta FaceNet (Sufyanu, Mohamad, Yusuf, y Nuhu, 2016), el cual tiene en promedio una exactitud de clasificación del 98.87 %. Facebook presenta otro método DeepFace (Mahto y Yadav, 2014) que logra en promedio una exactitud de clasificación del 97,35 %. CyberLink presenta a FaceMe (Deriche, 2008) y tiene en promedio una exactitud de clasificación del 99.82 % en el conjunto de datos rostros LFW . Se debe tener en cuenta que el resultado de los experimentos anteriores se realizó con pruebas en ambientes controlados. Las condiciones de iluminación, factores de oclusión, pose y la distancia de los individuos frente a la cámara en un entorno real pueden hacer que los resultados cambien y la precisión sea inferior a la reportada.

En la actualidad, uno de los retos del reconocimiento facial es llevar la fiabilidad obtenida en entornos controlados, a la variabilidad e instantaneidad de los entornos reales, esto lleva a enfrentar desafíos técnicos, como: implementar tecnologías de aprendizaje profundo en una aplicación de software, la adquisición de imágenes en diferentes espacios académicos (Jolliffe y Springer-Verlag, 2002), variación de iluminación, pose o punto de vista, envejecimiento, accesorios en el rostro, expresión facial, tono de la piel, entre otros.

Con el creciente aumento del rendimiento de la tecnología y la aparición de nuevas técnicas que lo optimizan, así como la carga computacional de los algoritmos de reconocimiento, renace la posibilidad de desarrollar sistemas eficientes de reconocimiento de rostros, en entornos no controlados y con mayor precisión que en sus inicios. Esto ha fortalecido el uso de esta tecnología para solucionar diferentes problemas en entornos interdisciplinarios, este proyecto se enfoca en realizar aportes en el entorno educativo, desarrollando un sistema piloto que pueda usar el reconocimiento de rostros para identificar a los aprendices y registrar la asistencia a clases presenciales automáticamente y en tiempo oportuno, generando así información consolidada para posteriores estudios, permitiendo dar respuesta a la problemática planteada.

Capítulo 2

ESTADO DEL ARTE

2.1. DATASET DE IMÁGENES

En esta parte, se describen todos los conjuntos de datos que se han utilizado para la validación del rendimiento de los modelos que tendrá el sistema que se implementó. Estas bases de datos de imágenes generalmente son las más usadas para los procesos de entrenamiento, prueba y validación de redes neuronales convolucionales.

2.1.1. LFW

Labeled Faces in the Wild, es una base de datos de fotografías de rostros diseñada para estudiar el problema del reconocimiento de rostros sin restricciones. El conjunto de datos contiene más de 13.000 imágenes de rostros recopilados de la web. Cada cara ha sido etiquetada con el nombre de la persona representada. 1680 de las personas fotografiadas tienen dos o más fotos distintas en el conjunto de datos. La única restricción en estos rostros es que fueron detectados por el detector de rostros Viola-Jones. Se pueden encontrar más detalles en el informe técnico a continuación (Huang, Mattar, Berg, y Learned-Miller, 2008).

Figura 2.1: Pares positivos en LFW.
(Huang y cols., 2008)

2.1.2. CFP-FP

Celebrities in Frontal-Profile in the Wild, es una recopilación de imágenes de celebridades en vistas frontales y de perfil. El conjunto de datos contiene 10 imágenes frontales y 4 de perfil de 500 individuos. Al igual que LFW, tiene definido 10 divisiones, cada uno contiene 350 pares iguales y 350 no iguales (Sengupta y cols., 2016).

Figura 2.2: Imágenes de muestra de Celebrities in Frontal-Profile in the Wild CFPW.
(Sengupta y cols., 2016)

2.1.3. AgeDB-30

AgeDB contiene 16.488 imágenes de varias personas famosas, como actores/actrices, escritores, científicos, políticos, etc. Cada imagen está anotada con respecto a la identidad, la edad y el atributo de género. Existen un total de 568 materias distintas. El promedio de imágenes por sujeto es de 29. La edad mínima y máxima es de 1 y 101, respectivamente. El rango de edad promedio para cada sujeto es de 50,3 años (Moschoglou y cols., 2017).

Figura 2.3: Random images from the AgeDB "in-the-wild" database.
(Moschoglou y cols., 2017)

2.1.4. CALFW

Cross-Age LFW (CALFW), una renovación de LFW, conjunto de datos incluye mejoras en agregando diferentes poses, iluminaciones, oclusiones, expresiones y la diferencia de edad entre pares positivos. La diferencia de edad de la mayoría de las parejas positivas en LFW es inferior a 10 años, mientras que la de la mayoría de las parejas negativas es superior a 10 años. En CALFW, no existe un límite claro para distinguir los dos tipos de parejas, por lo que la diferencia de edad no se puede determinar. una gran influencia en la verificación facial en CALFW (Zheng, Deng, y Hu, 2017).

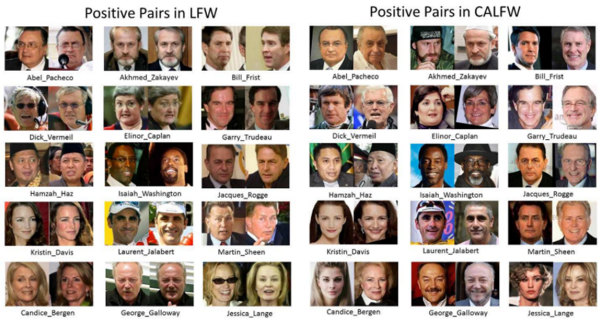


Figura 2.4: La comparación de pares positivos en LFW y CALFW. Comparado con LFW, los pares positivos en CALFW contienen edades obvias diferencia.
(Zheng y cols., 2017)

2.1.5. CPLFW

Cross-Pose LFW, construida para agregar influencia de pose en el reconocimiento facial. Considerando que, en la literatura de psicología, los pares de identidad diferente deben mostrar personas del mismo género y raza, los pares negativos en CPLFW se seleccionan utilizando personas del mismo género y raza. Esto asegura que el rendimiento en la base de datos indique la verdadera capacidad de distinguir a las personas utilizando sus identidades en lugar de depender de las diferencias de género y raza (Zheng y Deng, 2018).

La base de datos incluye 13.233 imágenes de rostros de 5.749 personas. Con fines de comparación, el conjunto de datos se separó en 10 subconjuntos no repetitivos de pares de imágenes para la validación cruzada. Cada subconjunto contiene 300 pares positivos (dos imágenes de la misma persona) y 300 pares negativos (dos imágenes de diferentes personas).

Figura 2.5: La comparación de pares positivos en LFW y CPLFW. En comparación con LFW, los pares positivos en CPLFW contienen una diferencia de pose obvia.
(Zheng y Deng, 2018)

2.1.6. CASIA-WebFace

Este dataset se utiliza para tareas de identificación y verificación de rostros. El conjunto de datos contiene 494.414 imágenes de rostros de 10.575 identidades reales recopiladas de la web (Yi, Lei, Liao, y Li, 2014).

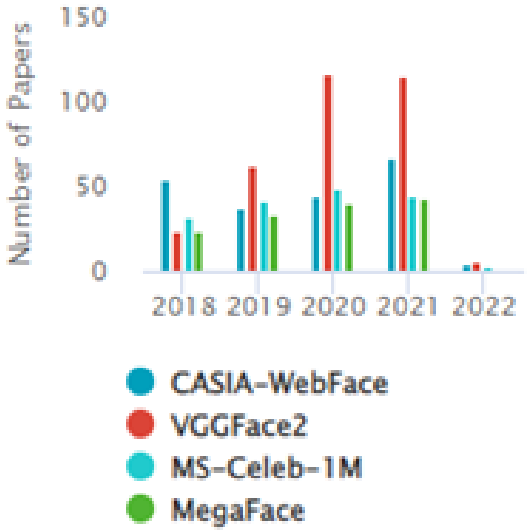


Figura 2.6: Cantidad de artículos publicados por año para los datasets.
(Yi y cols., 2014)

2.1.7. Glint360K

El conjunto de datos publicado contiene 17 millones de imágenes de 360.000 personas, que es el conjunto de capacitación más grande y limpio hasta ahora en el mundo académico;

Figura 2.7: Ejemplo de imágenes de CASIA-WebFace.
(Yi y cols., 2014)

Este conjunto de datos sale de la depuración y combinación de los dataset Celeb-500k (Cao, Li y Zhang 2018) y MS1MV2 (An y cols., 2021).

2.1.8. WebFace600k

Es un subconjunto de datos obtenidos a partir del dataset WebFace260M, este dataset contiene 600 mil personas. WebFace260M contiene 4 millones de identidades ruidosas de 260 millones de caras (WebFace260M) y 2 millones de identidades limpias de 42 millones de caras (WebFace42M) (Zhu y cols., 2021).

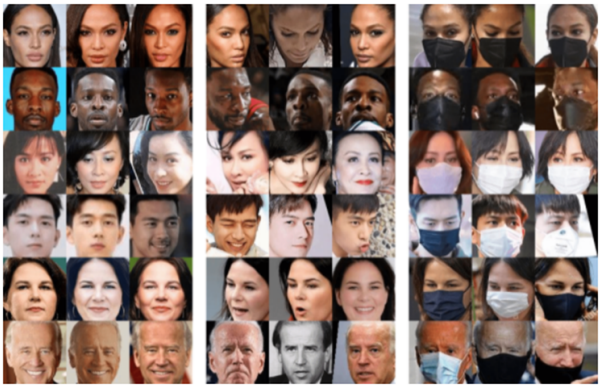


Figura 2.8: Tres columnas de caras tienen atributos de controlado, salvaje y enmascarado.
(Face-benchmark.org, s.f.)

2.2. MODELOS DE DETECCIÓN

En la clasificación de imágenes en visión por computadora se toma una imagen y se predice el objeto en una imagen, mientras que la detección de objetos no solo predice el objeto, sino que también se encuentra su ubicación en términos de cuadros delimitadores. Por ejemplo, cuando se construye un clasificador de rostros, se toma una imagen de entrada y se predice si contiene un rostro, mientras que un modelo de detección de objetos también indicaría la ubicación del rostro encontrado.

A continuación, se listan algunos detectores de rostros dentro del estado del arte.

2.2.1. RetinaFace

Se presenta en el 2020 como un método de localización de rostros multinivel singleshot, en el que se unifica la predicción de cuadros faciales, la localización de puntos de referencia faciales en 2D y la regresión de vértices en 3D; con el fin de obtener la regresión de puntos en el plano de la imagen (Deng, Guo, Ververas, Kotsia, y Zafeiriou, 2020).

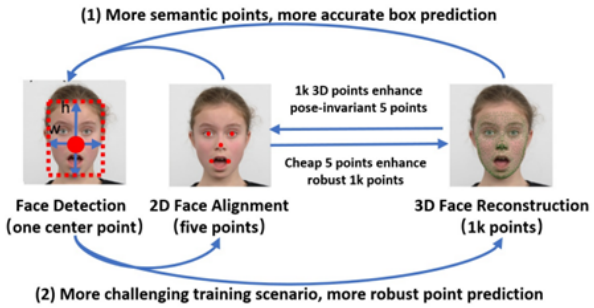


Figura 2.9: Tres tareas de localización de rostros tienen diferentes niveles de detalle, pero comparten el mismo objetivo: predicción precisa de puntos en el plano de la imagen.
(Deng y cols., 2020)

2.2.2. SCRFD

Sample and Computation Redistribution for Efficient Face Detection (SCRFD). El muestreo de datos de entrenamiento y las estrategias de distribución de cómputo son las claves para una detección de rostros eficiente y precisa. Debido a lo anterior, se presenta dos métodos simples pero efectivos: La redistribución de muestras (SR), que aumenta las muestras de capacitación para las etapas más necesarias, según las estadísticas de los conjuntos de

datos de referencia; y la Redistribución de Computación (CR), que reasigna la computación entre la columna vertebral, el cuello y la cabeza del modelo, con base en una metodología de búsqueda meticulosamente definida (J. Guo, Deng, Lattas, y Zafeiriou, 2021).

2.2.3. MTCNN

MTCNN (Redes convolucionales en cascada multitarea) es una red neuronal convolucional en cascada multitarea, que se utiliza para tratar simultáneamente la detección de rostros y el posicionamiento de puntos clave de rostros. En MTCNN se realizan tres etapas, para las que se necesita redimensionar la imagen a diferentes escalas para construir una pirámide de imágenes. En la primera etapa se utiliza una red convolucional que detecta ventanas de caras candidatas. A continuación, se utiliza otra red neuronal convolucional que descarta un gran número de candidatos en los que no existen rostros. Finalmente, una última red convolucional trata de identificar cualidades de los candidatos donde existe realmente un rostro, identificando las posiciones de cinco puntos de referencia faciales: uno en cada ojo, otro en la punta de la nariz y los dos restantes en las comisuras de los labios (García, Castro, Gutiérrez, y Fernández, 2017).

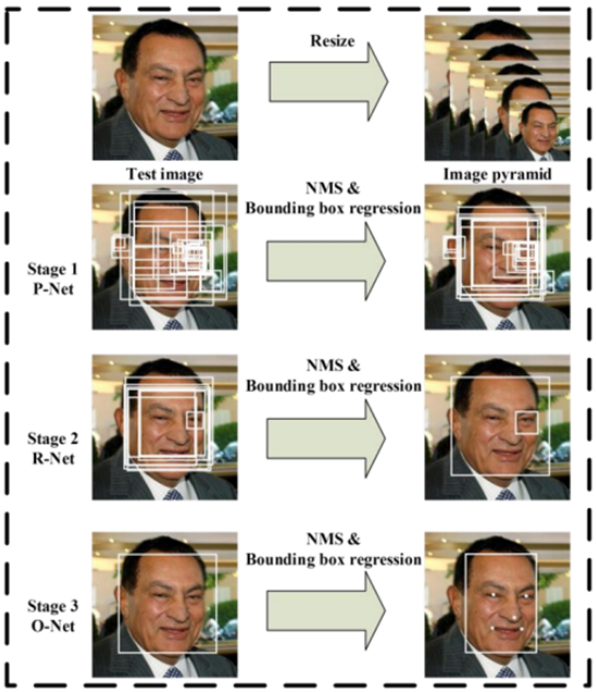


Figura 2.10: Canalización del marco en cascada que incluye redes convolucionales profundas multitarea de tres etapas.

(K. Zhang, Zhang, Li, y Qiao, 2016)

En la figura 2.10 las ventanas candidatas se producen a través de una red de propuestas rápida (P-Net). Después de eso, se refinan los candidatos en la siguiente etapa a través de una Red de Refinamiento (R-Net). En la tercera etapa, The Output Network (O-Net) produce el cuadro delimitador final y la posición de los puntos de referencia faciales.

2.2.4. Single Shot Detector (SSD)

El modelo SSD detecta el objeto en un solo paso sobre la imagen de entrada, a diferencia de otros modelos que atraviesan la imagen más de una vez para obtener una detección de salida. SSD se compone de 2 partes: Modelo Backbone y cabeza SSD (Adith, 2021).

El ***modelo backbone*** es una típica red de clasificación de imágenes previamente entrenada que funciona como extractor de mapas de características. Las capas de clasificación de imagen final del modelo se eliminan para brindar solo los mapas de características extraídos.

La ***cabeza SSD*** se compone de un par de capas convolucionales apiladas juntas y se agrega a la parte superior del modelo backbone. Esto nos da la salida como los cuadros delimitadores sobre los objetos. Con estas capas convolucionales detectan los diversos objetos en la imagen.

2.2.5. Librerías Para Detección De Rostros

Para la detección de rostros, quizás los modelos de aprendizaje profundo funcionan mejor. Pero la detección de rostros también existía antes de la aparición del aprendizaje profundo. Anteriormente, los descriptores de características clásicos y los clasificadores lineales eran una muy buena solución para la detección de rostros, en esta sección nombraremos 3 librerías o frameworks que analizamos para el desarrollo del proyecto.

Dlib HoG: El histograma de gradientes orientados (HOG) es un descriptor de características utilizado en la visión artificial y el procesamiento de imágenes con el fin de detectar objetos. Actualmente se encuentra implementado en diferentes librerías para la dirección de rostros, siendo un ejemplo la librería Dlib (Rath, 2021).

MediaPipe: MediaPipe Face Detection es una solución ultrarrápida de detección de rostros que viene con 6 puntos de referencia y compatibilidad con múltiples rostros. Se basa en BlazeFace, un detector de rostros liviano y de buen rendimiento diseñado para la inferencia

de GPU móvil (Google, s.f.). El rendimiento en tiempo real del detector permite que se aplique a cualquier experiencia de visor en vivo que requiera una región facial de interés precisa como entrada para otros modelos específicos.

BlazeFace utiliza una red de extracción de funciones liviana inspirada en MobileNetV1/V2 , pero distinta de ella, un esquema de anclaje compatible con GPU modificado de Single Shot MultiBox Detector (SSD) y una estrategia de resolución de vínculos mejorada alternativa a la supresión no máxima (Bazarevsky, Kartynnik, Vakunov, Raveendran, y Grundmann, 2019).

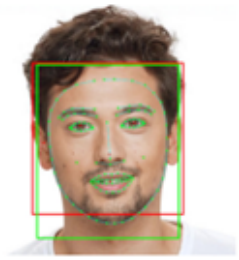


Figura 2.11: Ejemplo de canalización (mejor visto en color). Rojo: salida BlazeFace. Verde: resultado del modelo específico de la tarea.
(Bazarevsky y cols., 2019)

OpenCV: Es una biblioteca libre de visión artificial originalmente desarrollada por Intel. OpenCV significa Open Computer Vision (Visión Artificial Abierta). Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en una gran cantidad de aplicaciones, y hasta 2020 se la sigue mencionando como la biblioteca más popular de visión artificial. Detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes, son sólo algunos ejemplos de aplicaciones de OpenCV.

2.3. METODOS DE RECONOCIMIENTO

Para el problema de reconocimiento facial donde se pretende identificar individuos, existen numerosos mecanismos, de los cuales las redes neuronales han obtenido resultados de precisión muy altos. En la actualidad para mejorar la precisión en el reconocimiento se han establecido métodos para el reconocimiento, los cuales son modificaciones a las arquitecturas o modelos de redes neuronales existentes, como se evidencia en la tabla de la Figura 2.13.

A continuación, describiremos algunos métodos y modelos que dentro del estado del arte han permitido obtener una precisión alta para la tarea de reconocimiento facial.

2.3.1. Arcface

Additive Angular Margin Loss , es una función de pérdida utilizada en tareas de reconocimiento facial. El softmax se utiliza tradicionalmente en estas tareas. Sin embargo, la función de pérdida de softmax no optimiza explícitamente la incorporación de características para imponer una mayor similitud para las muestras intraclase y diversidad para las muestras interclase, lo que da como resultado una brecha de rendimiento para el reconocimiento facial profundo bajo grandes variaciones de apariencia de intraclase (Deng, Guo, Xue, y Zafeiriou, 2019). En la figura 1 se muestra que la distancia entre elementos de clases diferentes es alta, mientras que los elementos de la misma clase, pero con grandes variaciones de apariencia se encuentran muy próximos o con distancia corta.

Figura 2.12: Según la normalización del centro y la función, todas las identidades se distribuyen en una hiperesfera.

(Deng, Guo, Xue, y Zafeiriou, 2019)

2.3.2. Partial FC

Se presenta como un método de entrenamiento de clasificación distribuida dispersa con equilibrio de carga, el cual es capaz de usar una máquina con solo 8 GPU Nvidia Tesla V100 para implementar el entrenamiento en un conjunto de datos de reconocimiento facial con hasta 29 millones de identidades (Sun, Wang, y Tang, 2014b). En el artículo referenciado en la

Method	Public. Time	Loss	Architecture	Number of Networks	Training Set	Accuracy±Std(%)
DeepFace [20]	2014	softmax	Alexnet	3	Facebook (4.4M,4K)	97.35±0.25
DeepID2 [21]	2014	contrastive loss	Alexnet	25	CelebFaces+ (0.2M,10K)	99.15±0.13
DeepID3 [36]	2015	contrastive loss	VGGNet-10	50	CelebFaces+ (0.2M,10K)	99.53±0.10
FaceNet [38]	2015	triplet loss	GoogleNet-24	1	Google (500M,10M)	99.63±0.09
Baidu [58]	2015	triplet loss	CNN-9	10	Baidu (1.2M,18K)	99.77
VGGface [37]	2015	triplet loss	VGGNet-16	1	VGGface (2.6M,2.6K)	98.95
light-CNN [85]	2015	softmax	light CNN	1	MS-Celeb-1M (8.4M,100K)	98.8
Center Loss [101]	2016	center loss	Lenet+-7	1	CASIA-WebFace, CACD2000, Celebrity+ (0.7M,17K)	99.28
L-softmax [104]	2016	L-softmax	VGGNet-18	1	CASIA-WebFace (0.49M,10K)	98.71
Range Loss [82]	2016	range loss	VGGNet-16	1	MS-Celeb-1M, CASIA-WebFace (5M,100K)	99.52
L2-softmax [109]	2017	L2-softmax	ResNet-101	1	MS-Celeb-1M (3.7M,58K)	99.78
Normface [110]	2017	contrastive loss	ResNet-28	1	CASIA-WebFace (0.49M,10K)	99.19
CoCo loss [112]	2017	CoCo loss	-	1	MS-Celeb-1M (3M,80K)	99.86
vMF loss [115]	2017	vMF loss	ResNet-27	1	MS-Celeb-1M (4.6M,60K)	99.58
Marginal Loss [116]	2017	marginal loss	ResNet-27	1	MS-Celeb-1M (4M,80K)	99.48
SphereFace [84]	2017	A-softmax	ResNet-64	1	CASIA-WebFace (0.49M,10K)	99.42
CCL [113]	2018	center invariant loss	ResNet-27	1	CASIA-WebFace (0.49M,10K)	99.12
AMS loss [105]	2018	AMS loss	ResNet-20	1	CASIA-WebFace (0.49M,10K)	99.12
Cosface [107]	2018	cosface	ResNet-64	1	CASIA-WebFace (0.49M,10K)	99.33
Arcface [106]	2018	arcface	ResNet-100	1	MS-Celeb-1M (3.8M,85K)	99.83
Ring loss [117]	2018	Ring loss	ResNet-64	1	MS-Celeb-1M (3.5M,31K)	99.50

Figura 2.13: La precisión de diferentes métodos evaluados en el conjunto de datos LFW. (Wang y Deng, 2021)

fuente anterior, se indica que con el método Partitial FC, se logró aumentar la velocidad de entrenamiento 2.5 veces en un conjunto de entrenamiento con 4 millones de identidades y se aumentó el número máximo de identidades admitidas hasta 8 veces respecto a los modelos expuestos.

2.3.3. VPL

Variational Prototype Learning (VPL), representa cada clase como una distribución en lugar de un punto en el espacio latente. Con VPL se puede simular comparaciones de muestra a muestra dentro del marco de clasificación, alentando al solucionador SGD (Stochastic gradient descent) a ser más exploratorio, al tiempo que aumenta el rendimiento (Sun y cols., 2014b).

2.3.4. DeepID

DeepID es un sistema de reconocimiento facial profundo multivista (MVDFR), que permite abordar el desafío que propone la naturaleza 3D de la cabeza humana, dado que la apariencia facial de un mismo individuo presenta alta variabilidad intraclase cuando se pro-

Method	ICCV21-MFR					AgeDB	CFPFP	LFW	IJB-C
	All	African	Caucasian	South Asian	East Asian				
CASIA	**36.79**	42.55	55.83	49.62	19.61	99.45	95.21	94.90	87.22
CASIA+pfc	**37.11**	38.93	53.82	48.67	19.93	99.37	95.43	94.60	84.97
VGGFace	**38.58**	35.26	54.30	44.08	24.10	99.55	97.41	95.08	91.22
VGGFace+pfc	**40.67**	36.77	60.18	49.04	24.26	99.68	98.53	95.40	92.49
GlintAsian	**62.66**	49.53	64.83	57.98	61.74	99.58	93.19	95.40	91.50
GlintAsian+pfc	**63.15**	50.37	65.23	57.94	61.82	99.65	93.03	95.23	91.14
MS1MV2	**77.70**	74.60	84.13	82.04	51.10	99.83	98.08	98.08	96.14
MS1MV2+pfc	**77.74**	74.73	84.88	82.80	52.51	99.78	98.07	98.02	96.08
MegafaceMS1M	**78.37**	74.14	82.25	77.22	60.20	99.75	97.56	97.40	95.35
MegafaceMS1M+pfc	**78.77**	73.69	82.95	78.79	57.57	99.80	97.87	97.73	95.40
MS1MV3	**82.52**	77.17	87.03	86.01	60.62	99.80	98.53	98.27	96.58
MS1MV3+pfc	**81.68**	78.13	87.29	85.54	58.93	99.80	98.44	98.17	96.43
Glint360K	**86.79**	84.75	91.41	90.09	66.17	99.82	99.14	98.45	97.13
Glint360K+pfc	**87.08**	85.27	91.62	90.54	66.81	99.82	99.14	98.45	97.02
Webface12m	**90.57**	89.36	94.18	92.36	73.85	99.80	99.20	98.10	97.12
Webface12m+pfc	**89.95**	89.30	94.02	92.38	73.01	99.82	99.14	98.12	97.01

Figura 2.14: Comparación de aplicar Partitial FC en datasets de celebridades.
(An y cols., 2021)

yecta en plano de la imagen de la cámara. En este contexto, múltiples imágenes 2D de cada sujeto bajo diferentes puntos de vista se introducen en la red neuronal profunda propuesta con un diseño único para volver a expresar las características faciales en un descriptor facial único y más compacto, que a su vez produce una imagen más informativa y manera abstracta para la identificación de rostros utilizando redes neuronales convolucionales (Sun y cols., 2014b).

2.3.5. Facenet

FaceNet se basa en aprender una incrustación euclidiana (embedding) por imagen utilizando una red convolucional profunda. La red está entrenada de manera que las distancias L2 al cuadrado en el espacio del embedding se correspondan directamente con la similitud de rostros: los rostros de la misma persona tienen distancias pequeñas y los rostros de personas distintas tienen distancias grandes. Una vez que se ha producido el embedding, las tareas de verificación de rostros simplemente implican pasar por un umbral la distancia entre las dos incrustaciones; el reconocimiento se convierte en un problema de clasificación k-NN; y el agrupamiento se puede lograr utilizando técnicas estándar como k-means o agrupamiento aglomerativo (Schroff, Kalenichenko, y Philbin, 2015).

2.3.6. FbDeepFace

El método DeepFace, presentado en el 2014, es la base del aprendizaje profundo en el campo del reconocimiento facial. Para la tarea de reconocimiento facial, DeepFace realiza un proceso de detección, alineación, extracción y clasificación. La precisión en el reconocimiento obtenido por el método fue del 97.35 % en el conjunto de datos LFW, en gran medida debido al proceso de alineación (Taigman, Yang, Ranzato, y Wolf, 2014).

2.3.7. VGGFace

Desarrollado en el Grupo de Geometría Visual (VGG) de la Universidad de Oxford, es la aplicación de la arquitectura muy profunda de ConvNet VGG-16 (Simonyan & Zisserman 2014). Entrenada en una base de datos de 2,6 millones de imágenes de rostros y compuesta por 2622 identidades únicas, la base de datos utilizada se compone de hasta mil instancias de cada sujeto. El modelo está configurado para tomar una imagen RGB de 224 x 224 de tamaño fijo como entrada; como una forma de preprocesamiento, inicialmente normalizan en el centro todas las imágenes de entrenamiento (Parkhi, Vedaldi, y Zisserman, 2015) (Bukar y Ugail, 2017).

2.4. SISTEMAS DE RECONOCIMIENTO

2.4.1. TIPOS DE SISTEMAS DE RECONOCIMIENTO

En el mercado actual existen una gran variedad de soluciones de reconocimiento facial, gratuitas y de pago. Algunas de las soluciones están listas para ser usadas sin contar con habilidades en Machine Learning y otras necesitan mucho más tiempo y experiencia. Por esta razón, si se realiza una clasificación se encuentran los siguientes tipos, cada uno con sus propias ventajas y desventajas:

- **Motores de reconocimiento facial basados en software como servicio (SaaS).** Un proveedor de servicios de reconocimiento facial maneja todo, desde mantenerse al día con la tecnología de machine learning hasta administrar y brindar soporte a servidores de alta carga. No se requiere realizar el proceso de entrenamiento, el proveedor ya ha realizado este trabajo, lo que se debe hacer es integrar el software con sus sistemas de TI a través de una API.

  A pesar de las muchas ventajas al usar los motores de reconocimiento facial basados en SaaS, estas soluciones suelen ser más costosas de implementar, ya que todo lo maneja

el proveedor. También se necesita una conexión a Internet estable, ya que se enviarán imágenes pesadas a un servidor en algún lugar de Internet. Adicionalmente se puede llegar a tener problemas de seguridad y privacidad, debido que los datos que el software recolecte se deben enviar a una empresa de terceros donde no se puede controlar lo que hagan con estos.

- **Soluciones API REST auto hospedadas.** Este tipo de solución permite almacenar los datos en infraestructura TI propia, creando sistemas que funcionen sin conexión a internet. La infraestructura propia ayuda a tener el control de la ubicación de las imágenes que son de los datos más pesados.

  Las API REST auto hospedadas comparten muchas de las ventajas del tipo anterior, sin embargo, para personalizarlas a las necesidades particulares se deben tener buenos conocimientos en tecnologías de Machine Learning y desarrollo de software, además se tiene que administrar todos los elementos de la infraestructura tecnológica, siendo indispensable contar con las habilidades para esta tarea. En la mayoría de los casos, los servidores donde se ejecutan los servicios de la API REST se entregan como contenedores Docker, por lo que es muy fácil su administración y crecimiento.

- **Marcos y bibliotecas de código abierto.** Este tipo de solución por lo general suele ser gratuita, sin embargo, se necesitará algo de experiencia en Machine Learning para el uso de las bibliotecas. Otra de las consideraciones a tener en cuenta es que, a diferencia de las soluciones anteriores, esta solución implica que se construirá el software por completo, por lo que se debe dedicar tiempo en la arquitectura de microservicios en caso de querer que el sistema se acople a diferentes aplicaciones personalizadas, sin embargo, tendremos un sistema de vanguardia que podemos administrar y personalizar por completo.

Aunque dentro de las alternativas se encuentran herramientas gratuitas, se debe tener en cuenta que para construir un sistema de reconocimiento facial con las exigencias del mercado y a la altura de software comercial que aborde la problemática que estemos interesados en cubrir, existe un conjunto de costos asociados a la infraestructura tecnológica que soportará al sistema. Para lograr reducir los costos en la implementación, debemos diseñar sistemas que integren herramientas que se encuentren en estos tres tipos de solución.

2.4.2. HERRAMIENTAS DE RECONOCIMIENTO FACIAL OPEN SOURCE

2.4.2.1. Ageitgey/face_recognition

Es una biblioteca gratuita para reconocimiento, esta librería puede usarse como un módulo de Python para realizar los proyectos de software o usarse directamente con su interfaz de línea de comandos. Para incluir esta librería e iniciar un proyecto de reconocimiento facial, se puede instalar de forma sencilla con el sistema de gestión de paquetes PIP o haciendo uso de contenedores Docker en caso de necesitar construir un sistema bajo una arquitectura de servidores.

Pese a que la biblioteca se instala fácilmente y su manipulación es relativamente sencilla, existen mejores alternativas al tratarse de precisión debido que utiliza como modelo de reconocimiento facial ResNet con 27 capas de convolución, obteniendo una precisión de 99,38 % en el conjunto de datos LFW (Huang y cols., 2008) (Geitgey, s.f.).

2.4.2.2. DeepFace

Es una biblioteca hibrida de reconocimiento facial que envuelve modelos de última generación: VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace, DeepID, ArcFace y Dlib (Serengil y Ozpinar, 2020).

DeepFace proporciona herramientas para el reconocimiento facial, verificación facial, detección facial, detección de puntos de referencia faciales, similitud, reconocimiento de edad y género. La solución es escalable y cuenta con un sistema de administración de roles de usuario que permite controlar fácilmente quién tiene acceso a los servicios de reconocimiento facial.

2.4.2.3. InsightFace

Utiliza uno de los métodos más recientes y precisos para la detección de rostros (SCRFD) y el reconocimiento de rostros (SubCenter-ArcFace). InsightFace implementa en código abierto de manera eficiente una rica variedad de algoritmos de reconocimiento facial, detección facial y alineación facial, que se optimizan tanto para el entrenamiento como para el despliegue. Basada principalmente en PyTorch y MXNet (Deepinsight, 2022b). (Deepinsight, 2022b).

2.4.3. SISTEMAS DE RECONOCIMIENTO FACIAL

2.4.3.1. Exadel CompreFace

"Exadel CompreFace es un servicio de reconocimiento facial gratuito y de código abierto que se puede integrar fácilmente en cualquier sistema sin conocimientos previos de aprendizaje automático. CompreFace proporciona REST API para reconocimiento facial, verificación facial, detección de rostros, detección de puntos de referencia, reconocimiento de edad y género y se implementa fácilmente con Docker" (Exadel, s.f.).

Debido a que tiene una API REST, no necesita ser un experto en machine learning para implementarla; es muy fácil integrarlo en su sistema. La solución es escalable, por lo que puede reconocer caras simultáneamente en varias transmisiones de video. CompreFace también tiene una interfaz de usuario sencilla para administrar roles de usuario y colecciones de rostros. CompreFace permite seleccionar diferentes modelos para el reconocimiento facial, uno de ellos es FaceNet con precisión LFW del 99,63 % (Schroff y cols., 2015).

2.4.3.2. InsightFace-REST

Este es otro repositorio prometedor creado en 2019, aunque el desarrollo activo solo comenzó en octubre de 2020. Al igual que CompreFace, esta es una solución basada en Docker que proporciona una API REST conveniente para el reconocimiento facial y convertir módulos ONNX y TensorRT mediante Docker (SthPhoenix, 2021).

2.4.3.3. Amazon Rekognition

Es un servicio que facilita la incorporación de un potente análisis visual a sus aplicaciones (Amazon Inc., s.f.-b). Rekognition Image realiza reconocimiento de imágenes que detecta objetos, escenas y rostros; extrae texto, reconoce a personas famosas e identifica contenido inapropiado en imágenes. También le permite realizar búsquedas y comparar rostros. Utiliza modelos de redes neuronales profundas para detectar y etiquetar miles de objetos y escenas en sus imágenes, y constantemente Amazon Web Service(AWS) añade nuevas etiquetas y características de reconocimiento facial al servicio. Rekognition Images cuenta con las siguientes características (Amazon Inc., s.f.-a):

- Detección de objetos y escenas
- Reconocimiento facial
- Análisis facial

- Comparación de rostros
- Detección de imágenes no seguras
- Reconocimiento de famosos
- Texto en imágenes
- Detección de equipo de protección personal (epp)
- Administración a través de la api, la consola o la línea de comandos
- Seguridad administrativa
- Resumen de la api
- Revisión humana

2.4.3.4. Azure Face

Dentro de los servicios cognitivos de Azure encontramos Azure Face, que es un API que permite incorporar reconocimiento facial en aplicaciones de software. No es necesario tener conocimientos de aprendizaje automático debido que hace parte del tipo de *motores de reconocimiento facial basados en software como servicio (SaaS)*. Las características incluyen la detección de rostros que percibe características y atributos faciales (como una mascarilla, gafas o la ubicación del rostro) en una imagen y la identificación de una persona por una coincidencia con su repositorio privado o por el identificador de la foto. El servicio Azure Face ofrece algoritmos de IA que permiten usar las siguientes herramientas (Microsoft, s.f.):

- ***Detección y análisis de rostros*** API Detect detecta rostros humanos en una imagen y devuelve las coordenadas del rectángulo en sus ubicaciones. También devuelve un id. único que representa los datos del rostro almacenado. Esto se usa en operaciones posteriores para identificar o verificar rostros.

- ***Identificación*** La identificación facial puede abordar la coincidencia de uno a varios, de una imagen del rostro con un conjunto de rostros en un repositorio seguro. Los candidatos de coincidencia se devuelven en función de la coincidencia de sus datos faciales con el rostro de la consulta.

- ***Comprobación*** La operación de comprobación responde a la pregunta: ¿Estos dos rostros pertenecen a la misma persona?, la verificación es una coincidencia uno a uno, de una imagen del rostro con otro rostro de un repositorio seguro o una foto para verificar que son la misma persona.

- ***Búsqueda de rostros similares*** La operación de búsqueda realiza una coincidencia facial entre un rostro objetivo y un conjunto de rostros candidatos, y busca un conjunto más reducido de rostros parecidos al rostro objetivo. Esto resulta útil para realizar una búsqueda de rostros por imagen.
- ***Agrupación de rostros*** La operación de agrupación divide un conjunto de rostros desconocidos en varios grupos más pequeños en función de la similitud. Cada grupo es un subconjunto apropiado separado del conjunto original de rostros. También devuelve una única matriz "messyGroup"que contiene los identificadores de rostros para los que no se han encontrado similitudes.

2.5. DESAFÍOS AL IMPLEMENTAR Y DESPLEGAR MODELOS DE MACHINE LEARNING EN UN PRODUCTO DE SOFTWARE

El articulo A Software Engineering Perspective on Engineering Machine Learning Systems: State of the Art and Challenges (Giray, 2020) selecciona sistemáticamente un grupo de 141 investigaciones del contexto, realiza un análisis cuantitativo y cualitativo utilizando los datos extraídos de estos estudios donde se obtiene como resultados que la naturaleza no determinista de los sistemas ML complica todos los aspectos de SE en la ingeniería de sistemas ML. A pesar del creciente interés a partir de 2018, los resultados revelan que ninguno de los aspectos de SE tiene un conjunto maduro de herramientas y técnicas. Las pruebas son, con mucho, el área más popular entre los investigadores. Incluso para probar sistemas ML, los ingenieros solo tienen algunos prototipos de herramientas y propuestas de solución con pruebas experimentales débiles. Los investigadores deben realizar experimentos y estudios de casos, idealmente en entornos industriales, para comprender mejor estos desafíos y proponer soluciones.

Hacemos un resumen destacando algunos de los desafíos identificados en el estudio (Giray, 2020) que se tuvieron en cuenta para desarrollar el sistema piloto.

2.5.1. Desarrollo de Software:

La práctica de desarrollo de modelos de ML difiere del desarrollo de software tradicional debido a su dependencia de datos, incertidumbre y requisitos de experimentación. Las

organizaciones, que continuamente desarrollan e implementan sistemas ML, deben tener un proceso adecuado para respaldar el desarrollo, las pruebas y la implementación altamente iterativos de modelos ML (Sapp, 2017).

2.5.2. Manejo de datos de gran volumen:

Los patrones arquitectónicos distribuidos se utilizan ampliamente en el diseño de sistemas de ML para hacer frente a datos de gran volumen, lo que genera una complejidad adicional en el diseño arquitectónico y (Wan, Xia, Lo, y Murphy, 2021). Anderson (Anderson, 2015) enfatiza la importancia primordial del modelado de datos en la ingeniería de sistemas intensivos en datos que son escalables, robustos y eficientes, de Souza Nascimento et al. (Nascimento y cols., 2019) plantean la dificultad de lograrlo.

2.5.3. Manejo de datos:

La preparación de datos es un grupo de actividades vital e inevitable para desarrollar sistemas de ML (Sapp, 2017). Descubrir, acceder, recopilar, limpiar y transformar datos es un desafío y requiere mucho tiempo (Amershi y cols., 2019). Las canalizaciones de datos son artefactos esenciales, cuya versión debe controlarse, probarse, implementarse y mantenerse (Danilo Sato, 2019).

2.5.4. Comprender los algoritmos, las técnicas y las bibliotecas de ML:

Los desarrolladores usan algoritmos, técnicas y bibliotecas para producir modelos (Danilo Sato, 2019), enfrentando desafíos para comprenderlos, especialmente los relacionados con Deep Learning (Alshangiti, Sapkota, Murukannaiah, Liu, y Yu, 2019) (T. Zhang, Gao, Ma, Lyu, y Kim, 2019) (R. Zhang y cols., 2020).

2.5.5. Manejo de modelos:

Este esfuerzo de desarrollo implica algunas subactividades, como la ingeniería de funciones, el entrenamiento de modelos, la evaluación de modelos y su implementación (Amershi y cols., 2019). El desarrollo de modelos puede requerir muchas iteraciones y experimentación, incluidos muchos circuitos de retroalimentación entre subactividades (Amershi y cols., 2019) (Pham y cols., 2020). Para los modelos DL, la ingeniería de características se realiza implícitamente durante el entrenamiento del modelo (Amershi y cols., 2019).

2.5.6. Tratar con las dependencias:

Construir componentes altamente cohesivos y débilmente acoplados es una forma poderosa de manejar la complejidad en el desarrollo de software tradicional (Larman y Kruchten, 2002). Es posible encapsular el comportamiento y los datos asociados en componentes modulares. Por otro lado, los componentes de ML generalmente dependen de datos externos (Sculley y cols., 2015). Los resultados de un modelo ML pueden ser utilizados por muchos otros componentes implícitamente. En tales casos, los cambios en el modelo ML pueden afectar negativamente a los componentes dependientes (Belani, Vukovic, y Car, 2019) (Lwakatare, Raj, Bosch, Olsson, y Crnkovic, 2019). Por lo tanto, es vital considerar los impactos negativos potenciales de tales dependencias durante el desarrollo.

2.5.7. Modelos de reutilización:

La reutilización en el desarrollo de software tradicional es vital para aumentar la productividad y la calidad y disminuir el tiempo y el costo de desarrollo (Ravichandran y Rothenberger, 2003). Si bien la reutilización de bibliotecas de ML es muy común, efectiva y eficiente, la reutilización de modelos de ML en diferentes dominios o sistemas no es sencilla (Alvarez-Rodríguez, Zuñiga, Moreno, y Llorens, 2019) (Amershi y cols., 2019) (Baier, Jöhren, y Seebacher, 2019) (Q. Guo y cols., 2019). Adaptar la implementación de una red neuronal para una tarea diferente (T. Zhang y cols., 2019) o transferir una solución construida a otro dominio (Baier y cols., 2019) sigue siendo un desafío. Usar modelos pre entrenados del mismo dominio es otra estrategia de reutilización en ML debido a la demanda de recursos que el proceso de entrenamiento de redes neuronales.

2.5.8. Entorno de desarrollo, las herramientas y la infraestructura:

La presencia de herramientas de datos y programación diversas e incompatibles es una gran preocupación para los sistemas de ML de ingeniería (Kim, Zimmermann, DeLine, y Begel, 2018). Un conjunto heterogéneo de herramientas desafía a los ingenieros en muchas tareas, como el procesamiento de datos, la configuración del entorno y la implementación de modelos (Alshangiti y cols., 2019). Esta heterogeneidad provoca discrepancias potenciales entre los entornos de desarrollo, control de calidad y producción (R. Zhang y cols., 2020). Aunque las imágenes de Docker con todo el software deseado preinstalado pueden ayudar hasta cierto punto (R. Zhang y cols., 2020), los investigadores reconocieron la importancia de tener una infraestructura de desarrollo de sistemas de ML adecuada (Baier y cols., 2019), pre-

feriblemente integrando el desarrollo de ML con soporte en la infraestructura de desarrollo de software tradicional (Alvarez-Rodríguez y cols., 2019) (Amershi y cols., 2019). La escalabilidad es un problema real para muchos proyectos de ML, de hecho, es necesario asegurarse de que sus modelos sean capaces de escalar y satisfacer los aumentos de rendimiento y la demanda de aplicaciones en la producción.

2.5.9. Rendimiento:

La toma de decisiones incorrecta durante el desarrollo del modelo ML puede provocar un error de desbordamiento en la memoria de la GPU. Además, muchos de los procesos de producción de Machine Learning se basan hoy en día en las unidades de procesamiento gráficos o GPU. Sin embargo, estos equipos son caros, lo que fácilmente añade otra capa de complejidad a la tarea de escalar los sistemas de Machine Learning.

Capítulo 3

MARCO TEÓRICO

3.1. DETECCIÓN Y RECONOCIMIENTO DE ROSTROS

3.1.1. DETECCIÓN DE ROSTROS

La detección de rostros es una de las tareas dentro del campo de la detección de objetos por computadora, en este proceso el sistema debe localizar la posición del rostro de una o varias personas en una imagen o fotograma. Teniendo en cuenta que en la imagen no contiene solo el rosto, existen diferentes objetos que harón difícil la tarea (Sotaquira, s.f.).

La localización del rostro consiste en encontrar las coordenadas del rostro dentro de la imagen, término que se denomina *bounding box*, para cada rostro el sistema debe generar su respectivo bounding box.

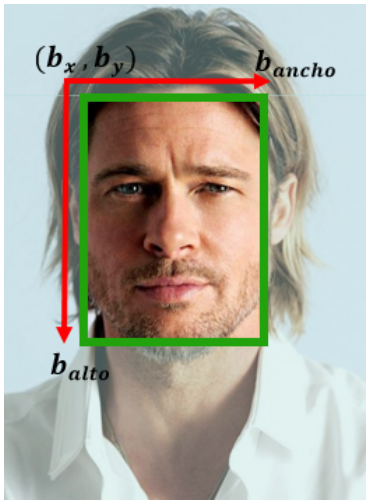


Figura 3.1: Bounding box, proporcionado por el modelo BlazeFace. Indicando el origen, el ancho y el alto.

A nivel computacional realizar la detección de rostros es una tarea compleja, debido que se deben detectar los rostros independientemente de la escala, dado que en una imagen pueden existir rostros ubicados a diferentes distancias de la cámara, generando rostros de diferentes tamaños en una sola imagen. Otras variables que deben considerarse en la tarea de detección son: la oclusión, donde solo es observable una parte del rostro; las expresiones o variación de iluminación.

3.1.1.1. La detección de rostros usando procesamiento de imágenes

Para lograr la detección automática se han ideado diferentes métodos de detección y los primeros fueron desarrollados a comienzos del siglo XXI, donde hacían uso de técnicas basadas en visión por computadora y procesamiento de imágenes.

Los algoritmos tradicionales realizan un barrido a través de la imagen y en cada posición del recorrido se intentan detectar patrones que permitan diferenciar el rostro de otros objetos, ya sea usando algunos filtros básicos, como en el caso del algoritmo de Viola and Jones (Viola y Jones, 2005), o analizando las diferencias entre tonalidades claras y oscuras en la imagen, como en el caso del algoritmo de histograma de gradientes orientados (Dalal y Triggs, 2005). El problema de este tipo de métodos es que se requieren condiciones ideales para que el rostro sea detectado correctamente, cuando hay algún tipo de rotación o cambios en la iluminación, simplemente no logran hacer la detección (Sotaquira, s.f.).

Actualmente se pueden distinguir cuatro grandes categorías de métodos de detección de caras, basados en el conocimiento, en caracteres invariantes, basados en plantillas y en apariencia (Yang, Kriegman, y Ahuja, 2002).

- **Métodos basados en el conocimiento:** Estos métodos representan las técnicas de detección de rostros que se basan en una serie de reglas previas definidas por la persona que quiere hacer la detección. Se definen una serie de características sobre las caras a detectar (forma de la cabeza, dos ojos, nariz, labios, etc.). Esto puede suponer un problema, debido que, si estas reglas son muy generales, el resultado de una búsqueda en imágenes donde no hay rostros, puede ser positivo y además puede encontrar una cantidad elevada de rostros. En el caso en que las reglas establecidas sean muy específicas posiblemente también aparezcan problemas ya que el resultado de la detección será muy bajo (Silva, 2018).

- **Métodos basados en caracteres invariantes:** Estos métodos utilizan como punto de referencia el color de la piel y la textura; el problema que supone aplicar estos métodos es que si en la imagen aparece ruido o diferentes condiciones de iluminación el algoritmo aplicado no funcionará correctamente. Si se utiliza el color de la piel, los algoritmos que utilizan toda la gama de colores tienen mejor resultado que los que utilizan una escala de grises (Silva, 2018).

- **Métodos basados en plantillas:** Estos métodos modelan geométricamente la forma del objeto. Las plantillas son las componentes básicas como por ejemplo círculos, elipses, entre otras [5]. Las correlaciones entre una imagen de entrada y los patrones almacenados se calculan para la detección. Estos métodos se han utilizado tanto para la localización como para la detección de rostros (Yang y cols., 2002).

- **Métodos basados en apariencia:** A diferencia de la coincidencia de plantillas, los modelos (o plantillas) se aprenden a partir de un conjunto de imágenes de entrenamiento que deberían capturar la variabilidad representativa de la apariencia facial. Estos modelos aprendidos se utilizan luego para la detección. Estos métodos están diseñados principalmente para la detección de rostros (Yang y cols., 2002).

3.1.2. RECONOCIMIENTO DE ROSTROS

El reconocimiento facial es una manera de identificar o verificar la identidad de una persona mediante su rostro (Kaspersky, 2021).

Los sistemas de reconocimiento facial capturan una imagen entrante desde un dispositivo con cámara de forma bidimensional o tridimensional en función de las características del dispositivo y realiza el siguiente proceso para realizar el reconocimiento (Sotaquira, 2019):

- **Detección del rostro:** se realiza la tarea de detección de rostro explicado en el apartado anterior, indicando si hay un rostro o un conjunto de rostros pues para cada rostro detectado se obtiene el bounding box.
- **Alineación del rostro:** se realiza un análisis facial donde se localiza los componentes del rostro y, mediante transformaciones geométricas, la normaliza respecto propiedades geométricas, como el tamaño y la pose, y fotométricas, como la iluminación. Para normalizar las imágenes de rostros, se pueden seguir diferentes reglas, como la distancia entre las pupilas, la posición de la nariz, o la distancia entre las comisuras de los labios. También se debe definir el tamaño de las imágenes y la gama de colores. Normalmente, para disminuir la carga computacional del sistema, se acostumbra a utilizar imágenes pequeñas en escala de grises.
- **Extracción de características:** se realiza un proceso de conversión de la imagen en datos, transformando la información analógica en información digital. Este proceso proporciona información para distinguir entre las caras de diferentes personas según variaciones geométricas o fotométricas, como un vector de características comúnmente denominado embedding.
- **Reconocimiento:** se realiza la búsqueda de una coincidencia comparando con una base de datos de embedings de rostros conocidos. el vector de características extraído se compara con los vectores de características extraídos de las caras de la base de datos. Si encuentra uno con un porcentaje elevado de similitud, nos devuelve la identidad del rostro; si no, nos indica que es un rostro desconocido.

3.2. REDES NEURONALES

Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona trabajando de manera simultánea un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas (Haykin, 2009). Una red neuronal suele estar compuesta por tres tipos de capas:

- **Capa de entrada:** con unidades de procesamiento o neuronas, que se encargan de recibir los datos o la información de entrada a la red neuronal.

- **Capas ocultas:** con neuronas encargadas del procesado de la información, interconectadas y sin contacto con el exterior, es decir, tanto sus entradas como sus salidas pertenecen al sistema.
- **Capa de salida:** cuenta con neuronas encargadas de ofrecer la respuesta del sistema después de que las capas anteriores procesaran la información.

Los datos de entrada se presentan en la primera capa y los valores se propagan desde cada neurona de la capa actual hasta cada neurona de la capa siguiente. Al final, se envía un resultado desde la capa de salida, como se muestra en la Figura 3.2.

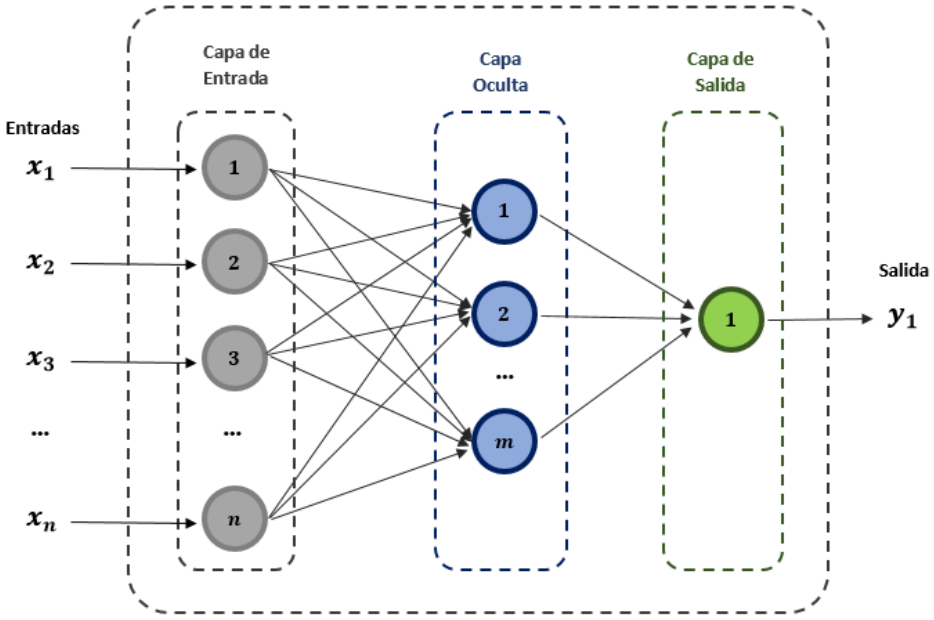


Figura 3.2: Red Neuronal Artificial con n neuronas de entrada, m neuronas en su capa oculta y una neurona en la capa de salida.

La red aprende examinando los registros personales, generando una predicción para cada registro y llevando a cabo ajustes a las ponderaciones una vez que ejecuta una predicción errónea. Este proceso se repite muchas veces, la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada.

Inicialmente, todas las ponderaciones son aleatorias y las respuestas que resultan de la red son, posiblemente, disparatadas. Continuamente se presentan a la red ejemplos para los

que se conoce el resultado, y las respuestas que da se comparan con los resultados conocidos. A medida que progresa el entrenamiento, la red se va llevando a cabo cada vez más precisa en la replicación de resultados conocidos.

3.2.1. PERCEPTRÓN

El Perceptrón es la red neuronal artificial más antigua, siendo capaz de aprender a reconocer patrones sencillos y realizar una clasificación binaria (Haykin, 2009). Es un modelo neuronal unidireccional, compuesto por dos capas de neuronas, una de entrada y otra de salida (ver Figura 3.3).

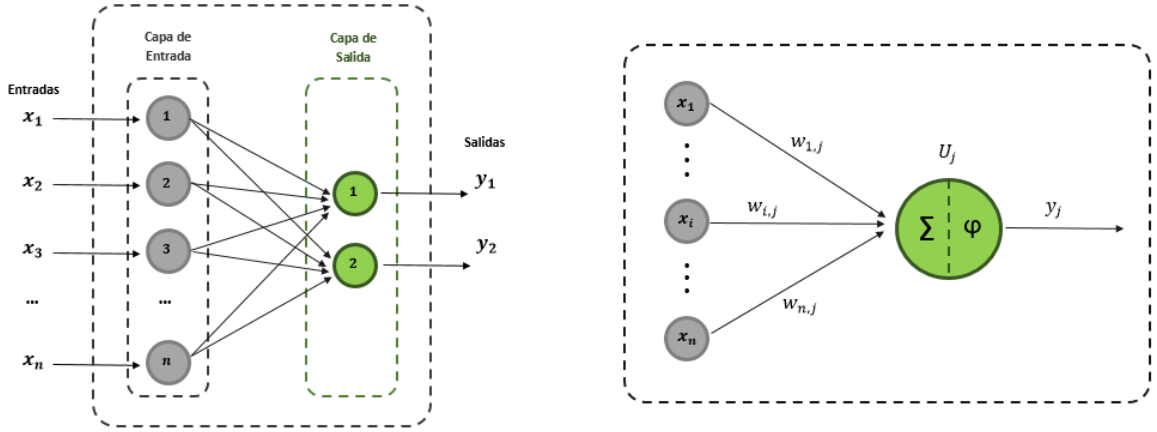


Figura 3.3: A la izquierda un Perceptrón simple, a la derecha los componentes de conexión entre neuronas de entrada y neuronas de salida.

La operación de una red de este tipo, con n neuronas de entrada y m neuronas de salida, se puede expresar con la ecuación 3.1, donde las neuronas de entrada no realizan ningún cómputo, únicamente envían la información a las neuronas de salida. Las neuronas de salida obtienen la suma ponderada de sus entradas con los pesos, calcula su diferencia con el umbral θ y le aplica una función H( ) (función escalón para activacion, ver Figura 3.4), como parte de la función de transferencia.

$$y_i(t) = H(\sum_{j=1}^{n} w_{i_j} x_j - \theta), \forall_i, 1 \leq i \leq m \tag{3.1}$$

Regla de apredizaje

El aprendizaje del Perceptrón es de tipo supervisado, por lo que cuando se evalúen los resultados obtenidos, se deben hacer unas modificaciones del sistema hasta que este se ajuste correctamente al problema. Como se ha visto, la salida del sistema depende de los pesos de las conexiones, por tanto, estos valores serán modificados para ajustar el sistema. Para ello, el algoritmo es el siguiente (González, 1994).

1. Inicialización de los pesos w_i y el umbral $\theta = -w_0$.
2. Presentación de los datos de entrada $X_p = (x_1, x_2, ..., x_N)$ con la salida esperada $d(t)$.
3. Cálculo de la salida del sistema según la Ecuación 3.1.
4. Adaptación de los pesos según la Ecuación 3.2.

$$w_i(t+1) = w_i(t) + \alpha[d(t) - y(t)]x_i(t), 0 \leq i \leq N \tag{3.2}$$

   Donde $\alpha[0,1]$ es un factor de aprendizaje.

5. Repetición desde el paso 2 hasta que el sistema sea capaz de clasificar sus entradas y se de por terminado el aprendizaje.

3.2.2. PERCEPTRÓN MULTI-CAPA

El Perceptrón simple no es capaz de realizar clasificaciones que no puedan separarse de forma lineal. De la necesidad de establecer regiones más complejas nace el perceptrón multi-capa (MLP). Como su propio nombre indica, el MLP es una red neuronal de tipo feedforward compuesta por una o más capas ocultas (Haykin, 2009) (González, 1994), cuya estructura se representa en la Figura 3.2.

El MLP presenta los tres tipos de capas descritas en el apartado de redes neuronales de este documento: la capa de entrada, las capas ocultas y las de salida.

Regla de apredizaje

El MLP es una red de tipo feedforward y su técnica de aprendizaje es una de las más populares, la back-propagation. El funcionamiento de este método es similar al algoritmo

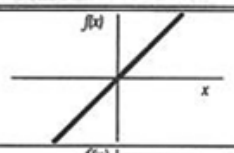
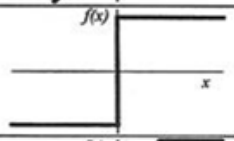
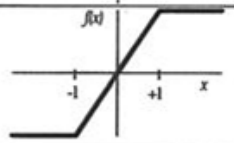
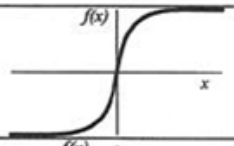
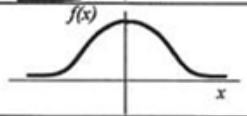
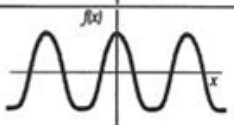
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \leq x \leq -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Figura 3.4: Ejemplos de función de activación habituales.
(Haykin, 2009)

de aprendizaje del Perceptrón, pero con el añadido de dividir la contribución del error para cada peso de la red. Así, su procedimiento es el siguiente (Josh Patterson, 2017):

1. Inicialización del valor de los pesos de la red.
2. Presentación de los datos de entrada $X_p = (x_1, x_2, ..., x_N)$ con la salida esperada $d(t)$.
3. Comparación de la salida esperada con la salida del sistema $y(t)$ y calcular el error: $\epsilon = d(t) - y(t)$.
4. Actualización de los pesos de la capa de salida:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha F(\sum_{i=0}^{n} w_{ij} y_i)\epsilon_j F^{'}(\sum_{i=0}^{n} w_{ij} y_i) \tag{3.3}$$

Donde $F^{'}$ representa la derivada de la función de activación F (que deberá ser una función derivable como la rampa, tangente hiperbólica, logística, etc.).

5. Definiendo el término de error de una capa j:

$$\Delta_j = \epsilon_j F^{'}(\sum_{i=0}^{n} w_{ij} y_i) \tag{3.4}$$

La regla de propagación de los valores de error será:

$$\Delta_i = F^{'}(\sum_{i=0}^{n} w_{ij} y_i) \sum_{i} w_{ij} \Delta_j \tag{3.5}$$

6. Cálculo de los nuevos pesos de la capa anterior k:

$$w_{ki}(t+1) = w_{ki}(t) + \alpha F(\sum_{i=0}^{n} w_{ik} y_i) \Delta_i \tag{3.6}$$

Repitiendo este proceso las veces que sea requerido, se pueden configurar todos los pesos de la red para ajustarla a una aplicación concreta.

3.3. REDES NEURONALES CONVOLUCIONALES (CNN)

Las redes neuronales convolucionales (CNN) son un tipo concreto de red neuronal artificial capaces de aprender a distinguir características en un conjunto de datos a través del cálculo de convoluciones. Es por ello que este tipo de redes sean muy utilizadas en el reconocimiento de objetos en imágenes (Josh Patterson, 2017).

El funcionamiento de una CNN consiste en transformar los datos que obtiene mediante la capa de entrada en un conjunto de valores calculados por capas ocultas completamente conectadas. De esta forma, la estructura más general de una CNN se divide en tres tipos de capas: una de entrada, una de extracción de características y otra de clasificación (ver Figura 3.5).

3.3.1. CAPA DE ENTRADA

La capa de entrada es aquella que recibe los datos provenientes del exterior de la CNN. En el caso de que estos datos sean imágenes, las entradas serán tridimensionales cuyas dimensiones son el ancho, el alto y los canales de color como, por ejemplo, los tres valores RGB de cada píxel (Josh Patterson, 2017).

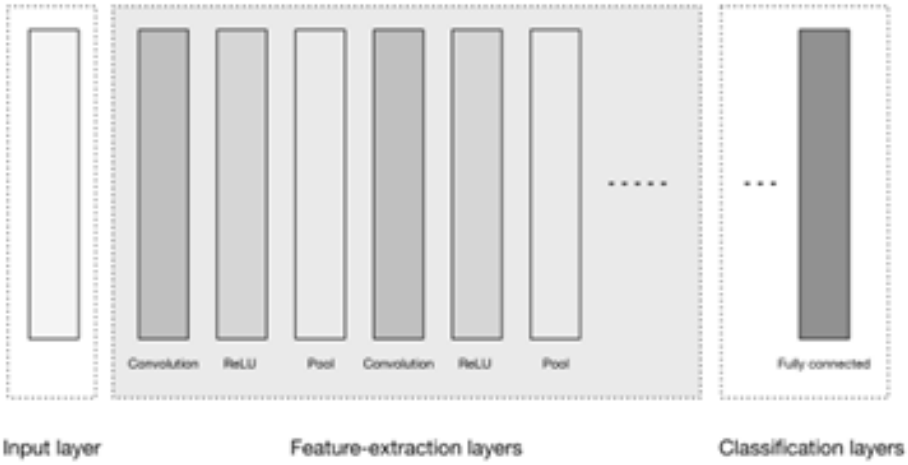


Figura 3.5: Representación general de una CNN (red neuronal convolucional). (Josh Patterson, 2017)

3.3.2. CAPAS DE EXTRACCIÓN DE CARACTERÍSTICAS

Son una serie de capas que se encargan de obtener información relevante para construir progresivamente características de orden superior. A su vez, las capas de extracción de características se pueden diferenciar en dos tipos: las capas de convolución, encargadas de calcular dicha función, y las capas de pooling que, a continuación, reúnen los resultados obtenidos por las anteriores capas (Josh Patterson, 2017).

3.3.2.1. Capas de convolución

Las capas convolucionales representan la parte más importante de una CNN. Estas tienen como objetivo transformar los datos de entrada mediante un conjunto de neuronas que son conectadas localmente desde la capa anterior. Esta capa calculará un producto de puntos entre una región concreta de la capa de entrada y los pesos que la conectan a la capa de salida. Normalmente, este proceso suele mantener las mismas dimensiones espaciales.

La detección de características de la CNN es realizada mediante la de convolución, operación que es realizada en este tipo de capas. La entrada de ésta pueden ser los propios datos brutos de entrada, o una salida proveniente de otra convolución anterior. En muchas ocasiones, esta operación se interpreta como un filtrado de los datos de entrada, donde un kernel encargado de filtrar se suele corresponder con el conjunto de pesos de la capa convolucional (Josh Patterson, 2017).

En la Figura 3.6 se muestra cómo un kernel se va desplazando a lo largo de unos datos de entrada. En cada paso, se multiplica el filtro por los valores de los datos de entrada, creando una nueva característica de salida.

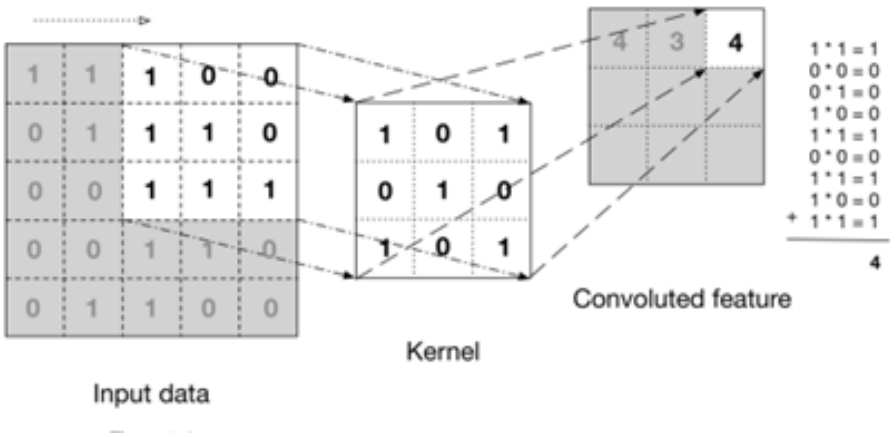


Figura 3.6: Representación de la operación de convolución con un kernel.
(Josh Patterson, 2017)

Como se muestra en la Figura 3.5, las capas convolucionales pueden ir seguidas de unas capas ReLU, las cuales tendrán el objetivo de aplicar una función de activación a modo de umbral en cero a sus datos de entrada, por lo que estas capas podrán cambiar algunos valores, pero no la dimensionalidad inicial de los datos.

3.3.2.2. Capas de pooling

Las capas de pooling son situadas entre capas convolucionales para reducir la dimensionalidad de los datos que pasan a través de las diferentes capas. Para ello, las capas de pooling suelen usar un filtro de, por ejemplo, 2×2. Este filtro es aplicado a los datos, de modo que los cuatro píxeles que sean filtrados serán sustituidos por un único valor, que se corresponderá con el máximo valor de los píxeles filtrados (Josh Patterson, 2017), según se muestra en la Figura VII.

3.3.3. CAPA DE CLASIFICACIÓN

La capa de clasificación representa la última capa de la CNN, aunque en algunas ocasiones puede haber más de una capa de este tipo. Este tipo de capa se encuentra completamente conectada a su capa adyacente y su objetivo es tomar las características finales calculadas

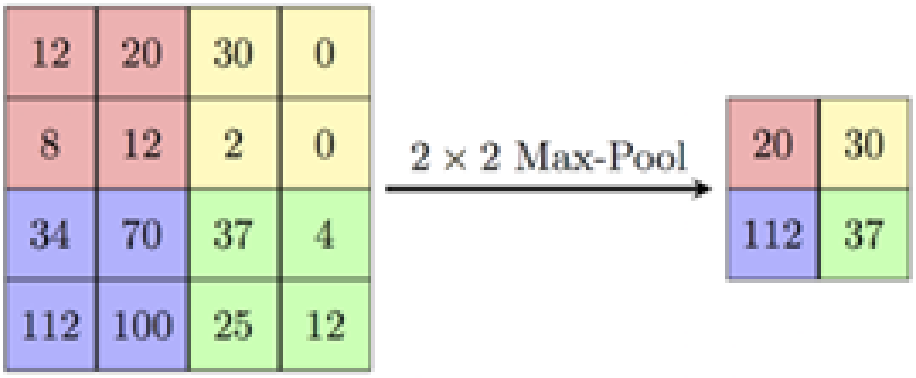


Figura 3.7: Funcionamiento de la capa de pooling.
(Migdał, 2017)

por el resto de la red y producir puntuaciones o probabilidades correspondientes a las clases entrenadas. El resultado de estas capas representará los datos de salida, los cuales tendrán unas dimensiones completamente menores, siendo únicamente un vector con tantos elementos como clases hayan sido analizadas (Josh Patterson, 2017).

3.3.4. FUNCIONES DE PÉRDIDA

Es un método para evaluar qué tan bien un algoritmo específico modela los datos otorgados, es decir, la función de perdida es la encargada de que la red neuronal aprenda. Cuando las predicciones se desvían demasiado de los resultados reales, la función de pérdida en Machine Learning arrojaría un número muy grande. La función de perdida ayuda a reducir el error en la predicción.

3.3.4.1. Triplet Loss

Es una función de pérdida para algoritmos de aprendizaje automático donde una entrada de referencia (llamada ancla) se compara con una entrada coincidente (llamada positiva) y una entrada no coincidente (llamada negativa). La distancia desde el ancla a la entrada positiva se minimiza y la distancia desde el ancla a la entrada negativa se maximiza (Chechik, Sharma, Shalit, y Bengio, 2010).

Este método de entrenamiento consiste en obtener un vector de características (embedding) a partir de la imagen de referencia i (con la expresión $f(i)$), que servirá para representar la imagen en un espacio de características R^d. Cada embedding calculado se encontrará en

una hiperesfera, de modo que su módulo se corresponda con la unidad: $||f(i)|| = 1$. En este espacio las imágenes coincidentes(pi) o de una misma persona se encontrarán próximas con la imagen de referencia (ai), mientras que las imágenes no coincidentes (ni) o de personas distintas presentarán distancias mayores, independientemente de las condiciones de iluminación de las imágenes.

Figura 3.8: Funcionamiento de tripet loss.

Al fijar una distancia mayor de las imágenes negativas respecto a la distancia de la referencia con imágenes positivas, se evita que se produzcan futuros falsos positivos, es decir, la red evita relacionar a una persona con la identidad de otra diferente. Matemáticamente, este proceso se representa con la siguiente restricción:

$$||a_i - p_i||^2 + \alpha < ||a_i - n_i||^2, \forall (a_i, p_i, n_i) \in \tau \tag{3.7}$$

Donde α representa un margen entre los pares de imágenes positiva y negativa para evitar que la red no optimice hacia la igualdad, y τ es el conjunto de todos los N triplets posibles para entrenar. Para realizar el entrenamiento de la CNN usando los triplets que optimicen los resultados se deberá calcular el mínimo valor de la llamada función Loss L que se representa en la Ecuación 3.8, en dependencia de los embeddings $f(i)$ de las tres imágenes del triplet.

$$L = \sum_{i}^{N} [||f(a_i) - f(p_i)||^2 - ||f(a_i) - f(n_i)||^2 + \alpha] \tag{3.8}$$

3.3.4.2. ArcFace Loss

Con la función de perdida Softmax no se logra optimizar explícitamente la incorporación de características para asignar una mayor similitud dentro de la misma clase (imágenes coincidentes de una misma identidad) y una mayor diversidad para las muestras interclase (imágenes no coincidentes de identidades diferentes), generando una brecha en el rendimiento para el reconocimiento facial profundo bajo grandes variaciones de apariencia intraclase, es decir, imágenes de un mismo individuo con variación de: pose, edad y grandes volúmenes de imágenes de prueba (Deng, Guo, Xue, y Zafeiriou, 2019). Arcface se propuso para optimizar el aprendizaje teniendo en consideración lo descrito anteriormente.

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N} log\frac{\epsilon^{s(cos(\theta_{y_i}+m))}}{\epsilon^{s(cos(\theta_{y_i}+m))}+\sum_{j=1,j=y_i}^{n}\epsilon^{s(cos(\theta_j))}} \tag{3.9}$$

La Ecuación 3.9 representa el método ArcFace, donde la normalización de características y pesos hace que las predicciones solo dependan del ángulo entre la característica y el peso. Los embeddings aprendidos se distribuyen en una hiperesfera de radio s (Deng, Guo, Xue, y Zafeiriou, 2019). A medida que los embeddings se distribuyen alrededor de cada centro de características en la hiperesfera, se agrega una penalización de margen angular aditiva m entre las características y los pesos para mejorar simultáneamente la compacidad intraclase y la discrepancia entre clases.

3.3.5. ONE-SHOT LEARNING

Para la tarea de reconocimiento facial no es factible que el modelo deba reentrenarse cada vez que se desee ingresar un nuevo individuo al sistema, este tipo de tareas se denominan one-shot learning o aprendizaje de una sola oportunidad, lo que indica que la red neuronal tendrá una sola oportunidad para aprender las características del rostro para poder reconocerlo en futuras ocasiones (Sotaquira, 2018). Una red de clasificación no logra aprender a un individuo con una sola imagen (Chanda y cols., 2019).

Al momento de construir un sistema de Machine Learning para procesar imágenes se suele usar una red convolucional, donde se tienen capas convolucionales, encargadas de extraer características relevantes de las imágenes, capas fully conected, las cuales obtienen la representación compacta de las características obtenidas en la imagen y una capa de

clasificación para determinar la predicción que dará el sistema para la imagen procesada (ver Figura V). Al eliminar la capa de clasificación tendremos a la salida de la red convolucional una representación compacta de las características de la imagen de entrada, es decir, en lugar de tener una predicción tendremos un vector con las características (embedding) del rostro, con la información esencial para lograr el reconocimiento al comparar el embedding resultante con los embeddings de los individuos almacenados en el sistema (Chanda y cols., 2019).

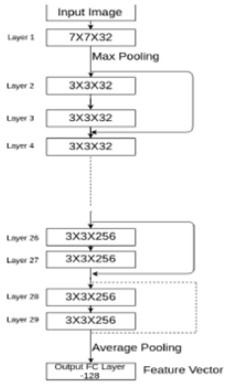


Figura 3.9: One-shot learning aplicado en arquitectura ResNet.
(Chanda y cols., 2019)

3.3.6. MEDIDA DE SIMILITUD

Cuando las imágenes son procesadas y se extraen las características de los rostros, se tiene un embedding (representación vectorial) por cada rostro. Con estos embedding se pueden realizar tareas de reconocimiento o de verificación. Para la tarea de reconocimiento, como se indicó en la sección anterior, se debe comparar el embedding del rostro procesado con los embedding almacenados para los individuos previamente registrados y así determinar la identidad. En la tarea de verificación, dos imágenes son procesadas por la red y sus embeddings se comparan para determinar si son de la misma persona. El proceso de comparación se realiza el cálculo de similitud empleando cualquiera de los métodos diseñados para este fin.

Al determinar si dos rostros corresponden a la misma persona, se define un punto de corte o umbral, para comparar el valor obtenido al realizar el cálculo de la distancia. Por ejemplo, en el caso de FaceNet, se determinó un punto de corte en 1.242 (La distancia euclidiana para

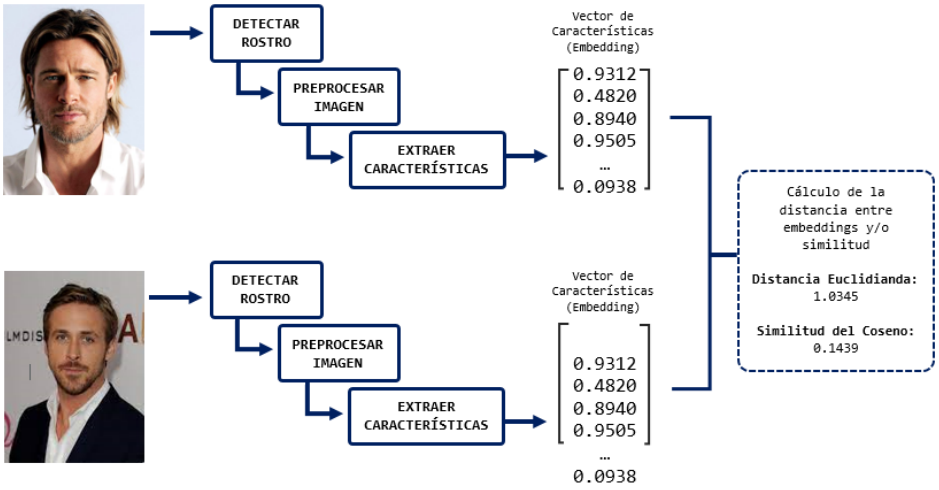


Figura 3.10: Ejemplo de cómo se genera los vectores de características y se calcula la distancia entre los vectores de caracteristicas.

FaceNet da un valor entre 0 y 4) (Schroff y cols., 2015). Este punto de corte establece que, si ambas imágenes tienen una distancia menor a 1.242, se concluye que ambas imágenes son de la misma persona y viceversa. Este punto de corte es variable, en el caso de OpenFace, el punto de corte fue establecido en 0.99. Por lo tanto, a partir de la comparación con el punto de corte definido, el sistema toma una decisión comprobando si es o no la misma persona (Amos, Ludwiczuk, y Satyanarayanan, 2016).

3.3.6.1. Similitud del coseno del espacio vectorial

La similitud del coseno usa el valor del coseno del ángulo entre dos vectores en el espacio vectorial como una medida de la diferencia entre dos individuos. En comparación con la medida de la distancia, la similitud del coseno presta más atención a la diferencia en la dirección de dos vectores, que a la distancia o la longitud. La fórmula es la siguiente (Gustavo Mendoza Olguín, 2019):

$$S_c(A,B) = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{3.10}$$

El rango de la similitud del coseno está entre -1 y 1. En el espacio del producto interno, esta es una medida de similitud en la dirección (y no en el tamaño) entre dos vectores, que no son el vector cero. La similitud del coseno es igual al coseno entre dos vectores y es lo mismo que un producto interno entre dos vectores normalizados:

- Si dos vectores están en la misma dirección, la similitud del coseno es 1.
- Si dos vectores están en un ángulo de 90 grados, la similitud del coseno es 0.
- Si son opuestos, entonces la similitud del coseno es -1.

3.3.6.2. Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson también se llama coeficiente de correlación de momento de producto de Pearson, que es una estadística utilizada para reflejar la similitud de dos variables. En otras palabras, se puede usar para calcular la similitud de dos vectores (aplicado en la clasificación de texto basada en modelos de espacio vectorial y sistemas de recomendación de preferencia del usuario) (Gustavo Mendoza Olguín, 2019).

Es el coeficiente de correlación r en el análisis de correlación. El ángulo del coseno del vector espacial se calcula en base a la normalización general de A y B respectivamente. La fórmula es la siguiente:

$$r(A,B) = \frac{N\sum AB - \sum A \sum B}{\sqrt{N\sum A^2 - (\sum A)^2} \cdot \sqrt{N\sum B^2 - (\sum B)^2}} \tag{3.11}$$

El coeficiente de correlación de Pearson oscila entre –1 y +1 (CIMEC, 2021):

- Un valor menor que 0 indica que existe una correlación negativa, es decir, que las dos variables están asociadas en sentido inverso. Cuánto más se acerca a -1, mayor es la fuerza de esa relación invertida (cuando el valor en una sea muy alto, el valor en la otra será muy bajo). Cuando es exactamente -1, eso significa que tienen una correlación negativa perfecta.
- Un valor mayor que 0 indica que existe una correlación positiva. En este caso las variables estarían asociadas en sentido directo. Cuanto más cerca de +1, más alta es su asociación. Un valor exacto de +1 indicaría una relación lineal positiva perfecta.
- Finalmente, una correlación de 0, o próxima a 0, indica que no hay relación lineal entre las dos variables.

3.3.6.3. Coeficiente de Jaccard

El coeficiente de Jaccard se utiliza principalmente para calcular la similitud entre individuos medida por valor simbólico o booleano. Debido a que los atributos característicos

de los individuos se identifican por valores simbólicos o booleanos, es imposible medir el valor específico de la diferencia y solo obtener este resultado, por lo que el coeficiente de Jaccard solo se ocupa de la cuestión de si las características compartidas por los individuos son consistentes. Si compara los coeficientes de similitud de Jaccard de A y B, solo compare el mismo número en A_i y B_i, la fórmula es la siguiente:

$$Jaccard(A,B) = \frac{A \cap B}{A \cup B} \tag{3.12}$$

3.4. TECNOLOGÍAS DE DESARROLLO DE SOFTWARE

3.4.1. METODOLOGÍAS DE DESARROLLO

Una metodología hace referencia a un conjunto de procedimientos genéricos y lógicos que se utilizan para alcanzar un objetivo particular usando un conjunto de habilidades y conocimientos (Maida E. G., 2015). Las metodologías de desarrollo de software siempre parten de un componente teórico y cuando son usadas por los equipos de trabajo conllevan a la utilización de un conjunto de técnicas y métodos que al final determinarán las tareas generales y específicas que se deberían realizar para alcanzar un objetivo.

Existen diferentes tipos de metodologías de desarrollo de software que fueron ideadas pensando en problemas particulares presentados en la industria en contextos específicos, por lo cual es importante conocer sus diferentes características y contrastarlas con las necesidades particulares a las que se enfrenta a la hora de desarrollar un producto y servicio. Es decir, cada una de estas tiene ventajas y enfoques que pueden ser reutilizados en diferentes momentos.

Existen dos grandes clasificaciones de metodologías de desarrollo de software que se agrupan generalmente como marcos de trabajo tradicionales o marcos de trabajo ágiles.

Los **marcos de trabajo tradicionales** o **metodologías tradicionales** se caracterizan por centrar la mayor parte de su esfuerzo en la planeación y control del proceso, lo que conlleva a una documentación exhaustiva y precisa de los artefactos que describen los requisitos y los modelos del sistema en las etapas iniciales del desarrollo del proyecto (Maida E. G., 2015). Este tipo de enfoques son óptimos en proyectos en los cuales los requisitos están plenamente identificados y delimitados, donde no se producirá ningún cambio en lo establecido mientras

el proyecto es finalizado.

Los **marcos de trabajo ágiles** o **metodologías ágiles** para el desarrollo de software nacen como otra opción para abordar proyectos donde no es posible tener un detalle completo de los requerimientos y sus estimaciones en la primera fase del proyecto o donde es necesario hacer procesos de adaptabilidad a lo largo del proceso de desarrollo de software (Maida E. G., 2015).

Las metodologías ágiles proveen un conjunto de pautas y principios que buscan facilitar y priorizar la entrega de producto sobre procesos de documentación exhaustiva, haciéndolos más simples, donde interactúa el cliente final desde las primeras etapas del proyecto. El inicio de las metodologías ágiles nació en el año 2001 a partir del manifiesto ágil de software donde se establecen cuatro valores fundamentales (Agilemanifesto.org, 2021):

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

3.4.1.1. Metodología de desarrollo cascada

Este es uno de los modelos genéricos más ampliamente conocido en la ingeniería de software y se deriva de procesos de ingeniería de sistemas más generales (Royce, 1987). Este modelo plantea un proceso lineal donde las actividades de desarrollo de un producto o servicios de software se agrupan en un conjunto de fases sucesivas donde éstas son desarrolladas una única vez y los resultados de cada fase son la entrada requerida para cada fase subsiguiente, además ninguna fase puede iniciar si la fase anterior no ha sido finalizada generalmente mediante un formalismo que puede ser un documento.

Según (Sommerville I., 2005) el modelo en cascada se compone de cinco (5) etapas principales que se asocian con actividades fundamentales en el proceso de desarrollo de software, las cuales son:

- **Análisis:** En esta fase se realiza una especificación muy detallada del sistema indicando los alcances, servicios a construir y restricciones. Esta fase se encarga principalmente en la especificación de los requisitos.

- **Diseño:** En esta fase se establece la arquitectura final del sistema. Se describen todas las abstracciones fundamentales del software y relaciones generalmente usando un lenguaje de modelado.
- **Implementación:** Esta fase involucra la construcción del software a partir de los diseños del sistema. Se hace una verificación inicial e individual del funcionamiento de cada uno de los módulos que se vayan construyendo.
- **Verificación:** Esta fase se encarga de hacer el proceso formal de pruebas. Se verifica fundamentalmente que el conjunto de módulos construidos funcione correctamente. Se verifica que cumpla con los requerimientos establecidos en la primera fase.
- **Funcionamiento y mantenimiento:** Esta es la fase en la que entra el proyecto una vez el producto o servicio de software es entregado para ser usado por el cliente. El mantenimiento hace referencia a la corrección de errores que no fueron detectados en las fases anteriores.

3.4.1.2. Metodología de desarrollo iterativo

La metodología de desarrollo iterativo se basa en el modelo tradicional de cascada (Mohammed, Munassar, & Govardhan, 2010). La diferencia principal consiste en que la metodología de desarrollo iterativo realiza el ciclo de vida en varias iteraciones. Cada iteración del proceso tiene como resultado un producto funcional. Además, esto permite obtener opiniones de los usuarios prontamente acerca de la situación del proyecto. Esto brinda más flexibilidad en el desarrollo de los requerimientos mediante desarrollos incrementales (Larman y Basili, 2003). Las fases son las mismas que en la metodología cascada, pero se regresa de forma iterativa a la etapa de análisis y para realizar entregables más pequeños.

Figura 3.11: Metodología de Desarrollo de Software Iterativo.

3.4.1.3. Metodología de desarrollo RUP (Proceso Racional Unificado)

RUP es una sigla en inglés equivalentes a Proceso Racional Unificado, el cual es un proceso de desarrollo de software tradicional basado en el modelo cascada y que fue desarrollado por la empresa Rational Software que es propiedad de IBM; esta metodología se centra en la arquitectura y es guía por casos de uso (requerimientos) (Kruchten, 2003). RUP divide el proceso de desarrollo en cuatro grandes fases, dentro de las que se realizan algunas iteraciones donde se desglosan, en mayor o menor intensidad, un conjunto de disciplinas según la fase que se está abordando. A continuación, se observan las fases y disciplinas propuestas por RUP.

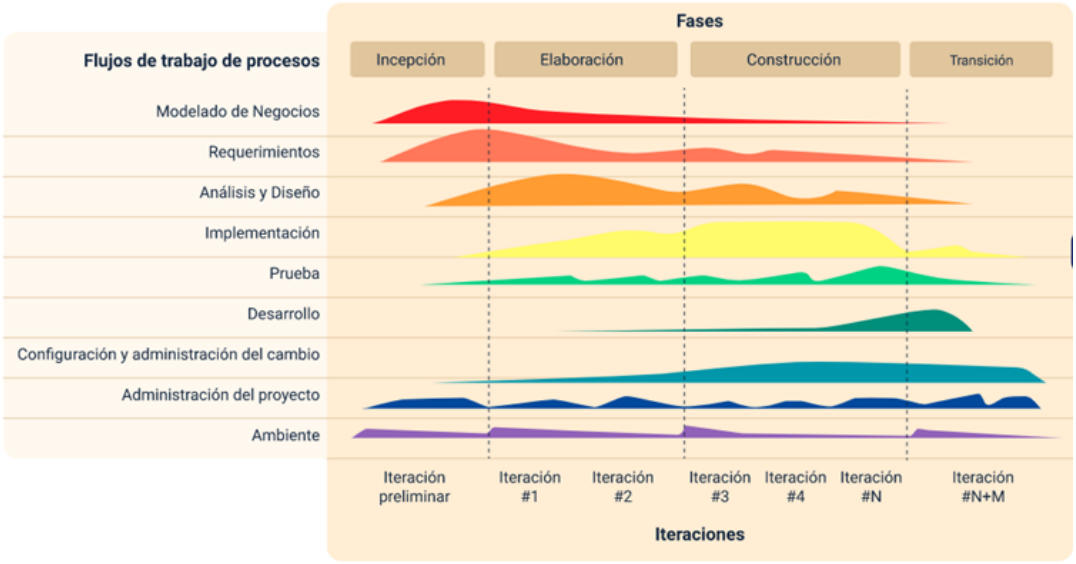


Figura 3.12: Fases y disciplinas de RUP.
(Kruchten, 2003)

Fases de RUP:

- **Fase de Incepción o Inicio:** se abordan actividades principalmente enfocadas en la comprensión del problema y el tipo de tecnología a utilizar, por lo que hay una gran carga en actividades relacionadas con la disciplina de modelado del negocio y especificación de requisitos.
- **Elaboración:** se centra la mayor parte del esfuerzo en la definición general de la arquitectura del sistema y el refinamiento de requisitos y modelado del negocio. RUP generalmente se apoya en el lenguaje de modelado UML para el modelado del negocio y descripción de la arquitectura.

- **Construcción:** se centran en las actividades relacionadas con la construcción del producto. Normalmente esta fase está constituida por varias iteraciones donde en cada una de ellas se desarrollan un subconjunto de requerimientos que normalmente están especificados como casos de uso. Cada una de estas pequeñas iteraciones se realiza utilizando el modelo cascada y el conjunto de todas las iteraciones producen al final una versión del producto.
- **Transición:** se desarrollan principalmente las disciplinas de pruebas y despliegue, es decir las actividades encaminadas a garantizar que el producto esté listo para entrega a sus usuarios.

3.4.2. PROTOCOLOS DE TRANSFERENCIA DE DATOS

3.4.2.1. Hypertext Transfer Protocol (HTTP)

Protocolo de Transferencia de Hipertexto, es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también. Sigue el clásico modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta del mismo. Se trata de un protocolo sin estado, lo que significa que el servidor no guarda ningún dato (estado) entre dos peticiones. Aunque en la mayoría de casos se basa en una conexión del tipo TCP/IP, puede ser usado sobre cualquier capa de transporte segura o de confianza, es decir, sobre cualquier protocolo que no pierda mensajes silenciosamente, tal como UDP (Mozilla Developer Network, s.f.).

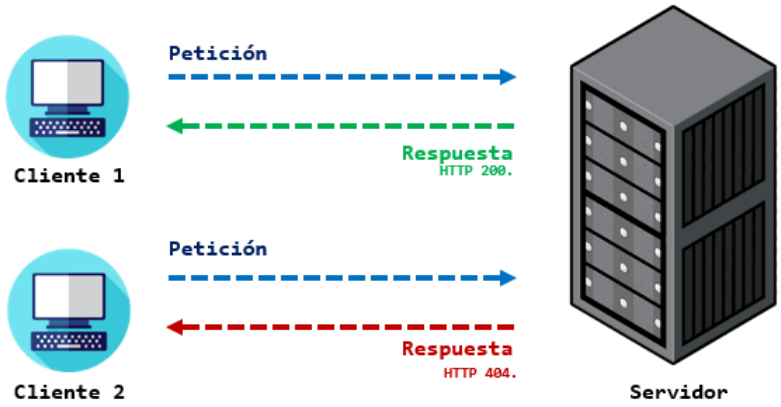


Figura 3.13: Ejemplo del funcionamiento de la Arquitectura HTTP.

3.4.2.2. Representational State Transfer (REST)

Es un conjunto de pautas de arquitectura para realizar comunicación entre cliente-servidor, apoyándose en el protocolo HTTP para comunicación al servidor. Los desarrolladores pueden implementarlo de distintas maneras.

Cuando el cliente envía una solicitud a través de una API de REST, se transfiere una representación del estado del recurso requerido a quien lo haya solicitado o al extremo. La información se entrega por medio de HTTP en uno de estos formatos: JSON (JavaScript Object Notation), XML o texto sin formato. JSON es el formato de texto para transferencia de datos más popular, ya que tanto las máquinas como las personas lo pueden comprender y no depende de ningún lenguaje.

Los encabezados y los parámetros también son importantes en los métodos HTTP de una solicitud HTTP de la API de REST, ya que contienen información de identificación importante con respecto a los metadatos, la autorización, el identificador uniforme de recursos (URI), el almacenamiento en caché, las cookies y otros elementos de la solicitud. Hay encabezados de solicitud y de respuesta, pero cada uno tiene sus propios códigos de estado e información de conexión HTTP.

Para que una API se considere de REST, debe cumplir los siguientes criterios:

- Arquitectura cliente-servidor compuesta de clientes, servidores y recursos, con la gestión de solicitudes a través de HTTP.
- Comunicación entre el cliente y el servidor sin estado, lo cual implica que no se almacena la información del cliente entre las solicitudes de GET y que cada una de ellas es independiente y está desconectada del resto.
- Datos que pueden almacenarse en caché y optimizan las interacciones entre el cliente y el servidor.
- Una interfaz uniforme entre los elementos, para que la información se transfiera de forma estandarizada. Para ello deben cumplirse las siguientes condiciones:
    - Los recursos solicitados deben ser identificables e independientes de las representaciones enviadas al cliente.

- El cliente debe poder manipular los recursos a través de la representación que recibe, ya que esta contiene suficiente información para permitirlo.
- Los mensajes autodescriptivos que se envíen al cliente deben contener la información necesaria para describir cómo debe procesarla.
- Debe contener hipertexto o hipermedios, lo cual significa que cuando el cliente acceda a algún recurso, debe poder utilizar hipervínculos para buscar las demás acciones que se encuentren disponibles en ese momento.

- Un sistema en capas que organiza en jerarquías invisibles para el cliente cada uno de los servidores (los encargados de la seguridad, del equilibrio de carga, etc.) que participan en la recuperación de la información solicitada.
- Código disponible según se solicite (opcional), es decir, la capacidad para enviar códigos ejecutables del servidor al cliente cuando se requiera, lo cual amplía las funciones del cliente.

3.4.3. INFRAESTRUCTURA DE ALOJAMIENTO WEB

El alojamiento web es el servicio que hace referencia al lugar donde se alojarán los archivos que forman tu página web para que estén permanentemente en línea y cualquiera que quiera visitar tu web pueda acceder a ella.

Dado que no todos los sitios web necesitan los mismos recursos para funcionar, ni reciben el mismo número de visitas cada día, existen distintos planes de alojamiento web que pueden adaptarse a las necesidades de tu web y a las exigencias de tu economía.

3.4.3.1. Tipos de Alojamiento

Existen varios tipos de hosting con diferentes características para adaptarse mejor a las necesidades de tu página web.

- **Hosting compartido:** En este tipo de hospedaje, tu página web se almacena junto a otras páginas web y se comparten entre ellas tanto el espacio disponible en el disco duro, como los recursos del servidor. De ese modo se comparte el coste del servidor, pero a cambio se obtiene un rendimiento reducido.

  Este sistema puede resultar idóneo para webs personales o con pocas visitas. Como punto negativo cabe destacar que, si la dirección IP de una de las webs entra en una

"lista negra" por enviar spam o contener contenido malicioso, todas las páginas que compartan ese servidor correrán la misma suerte ya que también comparten dirección IP.

- **Servidores Privados Virtuales - VPS:** Esta es la opción más común entre los servicios de alojamiento web. Consiste en crear mediante virtualización varios servidores web sobre un servidor real y asignar una determinada cantidad de los recursos reales disponibles entre los servidores virtuales. La página web se alojará en uno de esos servidores virtuales.

  Su mayor ventaja es que, a diferencia del hosting compartido, puedes contratar previamente la cantidad de recursos que tendrá el servidor web para que tu web siempre funcione correctamente. Otra ventaja de este tipo de alojamiento es su escalabilidad, ya que puedes contratar ampliaciones en el servidor si el proyecto web crece y recibes muchas visitas cada día.

- **Servidores Dedicados:** Este es tal vez uno de los sistemas menos económicos, pero a cambio, como su propio nombre indica, tienes a tu disposición toda la potencia que pueda ofrecer un servidor exclusivamente para hacer funcionar la página web. Este tipo de hosting está recomendado para empresas con una gran página web corporativa y que, además, centralizan otros servicios al mismo servidor como tienda online, servidor de correo, etc.

- **Cloud Hosting dinámico o alojamiento en la nube:** Cuentan con una gran infraestructura técnica que permite combinar varios servidores web en uno solo creando un "macro servidor virtual" (o nube).

  Este tipo de alojamiento destaca por su flexibilidad, permite establecer las necesidades reales de rendimiento hora a hora, para que una página web funcione a pleno rendimiento, sin pagar un sobrecoste por ello.

3.4.3.2. Virtualización de entornos

Hay algunas situaciones problemáticas derivadas del entorno de desarrollo, que pueden generar fallos, así como de consumir tiempo ajustando los parámetros para que la aplicación funcione en otros dispositivos ajenos al dispositivo usado en el desarrollo.

Para el desarrollo en local no se tiene exactamente el mismo entorno de ejecución que en el servidor remoto. Por ejemplo, puede ocurrir que estemos desarrollando en una estación

de trabajo con sistema operativo Windows y luego el servidor tenga sistema operativo Linux. También puede ocurrir que las versiones del lenguaje de programación local y producción no sean las mismas, o quizás que se tengan instaladas las mismas librerías o versiones.

Todas estas situaciones problemáticas se resuelven de la misma manera: a través de un entorno de desarrollo basado en virtualización. Básicamente, consiste en la instalación de una máquina virtual en el ordenador del desarrollador, haciendo que nos sirva de plataforma de ejecución, en lugar de correr el código directamente en la máquina host.

Esta situación permite que todos los desarrolladores del equipo de trabajo compartan una misma virtualización como plataforma de ejecución del proyecto que están desarrollando en local. De este modo, lo que funciona para un integrante del equipo estamos seguros de que funcionará también para el resto de los compañeros, ya que en el fondo todos están ejecutando el proyecto bajo el mismo sistema operativo, las mismas versiones de lenguajes y librerías, etc. Además, gracias a la virtualización del entorno de desarrollo se puede conseguir que se tengan los mismos recursos locales que se tendrían en el servidor en remoto, allí donde el código va a desplegarse en producción. Lo ideal es por tanto que las virtualizaciones de los entornos de desarrollo sean una copia exacta del servidor que se va a disponer en producción; de este modo estaremos seguros de que, lo que funciona en local, funcionará también en remoto.

Máquina Virtual (MV)

Las VM se ejecutan en un entorno de hipervisor en el que cada máquina virtual debe incluir su propio sistema operativo invitado dentro del mismo, junto con sus archivos binarios, bibliotecas y archivos de aplicaciones correspondientes. Esto consume una gran cantidad de recursos y genera mucha sobrecarga, especialmente cuando se ejecutan varias VM en el mismo servidor físico, cada una con su propio sistema operativo invitado [XC].

Una máquina virtual es aquella que emula a un ordenador completo. Es decir, es un software que puede hacerse pasar por otro dispositivo, como un PC, de tal modo que puedes ejecutar otro sistema operativo en su interior. Tiene su propio disco duro, memoria, tarjeta gráfica y demás componentes de hardware, aunque todos ellos son virtuales. Aunque los componentes sean virtuales no quiere decir necesariamente que no existan. Una máquina virtual puede tener unos recursos reservados de 8 GB de RAM y 25 GB de disco duro, estos recursos de ubican de manera física en el dispositivo donde está instalada la máquina virtual,

también llamado hipervisor, host o dispositivo anfitrión.

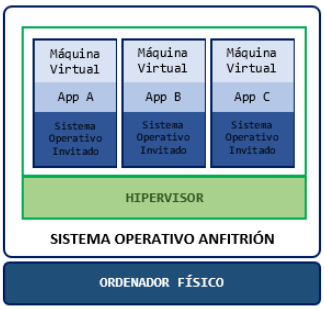


Figura 3.14: Arquitectura Máquinas Virtuales.

Contenedores

Los contenedores son una forma de virtualización del sistema operativo. Un solo contenedor se puede usar para ejecutar cualquier cosa, desde un microservicio o un proceso de software a una aplicación de mayor tamaño. Dentro de un contenedor se encuentran todos los ejecutables, el código binario, las bibliotecas y los archivos de configuración necesarios. Sin embargo, en comparación con los métodos de virtualización de máquinas o servidores, los contenedores no contienen imágenes del sistema operativo. Esto los hace más ligeros y portátiles, con una sobrecarga significativamente menor. En implementaciones de aplicaciones de mayor tamaño, se pueden poner en marcha varios contenedores como uno o varios clústeres de contenedores. Estos clústeres se pueden gestionar mediante un orquestador de contenedores, como Docker Swarm y Kubernetes [XC].

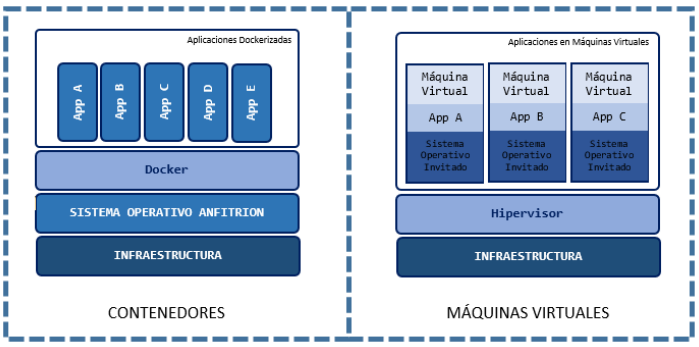


Figura 3.15: Contenedores vs Máquinas Virtuales.

Capítulo 4

MATERIALES Y MÉTODOS

Este capítulo inicia con la metodología para el análisis, diseño y desarrollo de FR-ARCA y la descripción de las diferentes herramientas utilizadas para construir cada uno de los módulos que componen el sistema FR-ARCA. En la sección de preparación de datos se detallan las características que tiene el set de datos recolectado, el cual se usa en la sección de experimentos, en esta sección describe la metodología con la cual se realizó cada uno de los experimentos para realizar la evaluación del sistema y finalmente se mencionan las métricas utilizadas.

4.1. METODOLOGÍA PARA EL ANÁLISIS, DISEÑO Y DESARROLLO DE FR-ARCA

La metodología de desarrollo de software utilizada en este proyecto fue RUP (Proceso Racional Unificado). Se dividió el proceso de desarrollo en cuatro fases que se describen a continuación:

- **Fase I Incepción.** En la cual se abordaron las actividades principalmente enfocadas en la comprensión del problema y el tipo de tecnología a utilizar. Se realizó el análisis del problema y se listaron las principales consideraciones a tener en cuenta (Ver tabla 5.1). Se realizó el estudio de los conceptos, las tecnologías y los antecedentes que intervienen en el contexto de detección y reconocimiento de rostros, lo cual permitió proponer soluciones a las consideraciones anteriores y elegir las diferentes tecnologías utilizadas en el sistema FR-ARCA.
- **Fase II Elaboración.** Durante esta fase se construyó el modelo propuesto para FR-

ARCA el cual se describe a nivel general en la figura 5.1. Se define la metodología para reconocimiento de rostros a través de cálculo de similitud (ver figura 5.3). A partir del refinamiento de los requerimientos especiales del sistema y las consideraciones para implementar y desplegar modelos de Machine Learning en un producto de software, se realizó la arquitectura tecnológica del sistema desarrollado, FR-ARCA (ver figura 5.4). En esta fase se realizó la búsqueda y selección de los modelos preentrenados de detección y reconocimiento de rostros implementados en este proyecto (ver tabla 5.2).

- **Fase III La Construcción.** En la cual se ejecutaron las actividades relacionadas con el desarrollo del sistema FR-ARCA. Se realizaron varias iteraciones en donde se refinaron progresivamente cada una de las funcionalidades propuestas. Estas iteraciones se realizaron utilizando el modelo en cascada y el conjunto de todas las iteraciones dio como resultado el sistema final FR-ARCA.

- **Fase IV Transición.** En esta fase se realizó el despliegue del sistema FR-ARCA en un entorno local y en un entorno remoto, sobre los cuales se realizaron los experimentos propuestos en este capítulo; de esta manera evaluamos todos los módulos que componen el sistema FR-ARCA y se realizaron las configuraciones finales. Los resultados obtenidos demuestran un excelente cumplimiento de los objetivos de este proyecto.

4.2. CAJA DE HERRAMIENTAS

En esta sección daremos la descripción de las herramientas tecnológicas usadas en la construcción de cada uno de los módulos descritos en la arquitectura (ver figura 5.4).

4.2.1. Dashboard UI

Es el módulo que funciona como una interfaz para el usuario, el cual le permitirá acceder a todos los servicios que ofrece el sistema FR-ARCA. Para realizar esta interfaz gráfica de usuario, se usó: HTML-5, CSS-3, Bootstrap 4 y JavaScript ES6.

- *HTML:* El Lenguaje de Etiquetas de Hipertexto (HyperText Markup Language) es un lenguaje simple de etiquetas usado para crear documentos de hipertexto que son portables de una plataforma a otra. Los documentos HTML son documentos SGML con una semántica genérica que es apropiada para representar información de un amplio rango de aplicaciones (Mozilla Developer Network, 2021c).

- *CSS:* Hojas de Estilo en Cascada (del inglés Cascading Style Sheets) es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML. CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios. CSS es uno de los lenguajes base de la Open Web y posee una especificación estandarizada por parte del W3C (Mozilla Developer Network, 2021b).

- *Bootstrap:* Es el conjunto de herramientas de código abierto de front-end más popular del mundo; en su versión 5 incluye variables y mixins de Sass, un sistema de cuadrícula receptivo, amplios componentes preconstruidos y potentes complementos de JavaScript (Otto, Thornton, y Bootstrap contributors, s.f.).

- *JavaScript:* A menudo abreviado como JS, es un lenguaje ligero, interpretado y orientado a objetos con funciones de primera clase, y mejor conocido como el lenguaje de programación para las páginas Web, pero también se utiliza en muchos entornos que no son de navegador. Es un lenguaje de scripts que es dinámico, multiparadigma, basado en prototipos y admite estilos de programación orientados a objetos, imperativos y funcionales (Mozilla Developer Network, 2021a).

4.2.2. API Gestión de Datos

Fue desarrollada en el lenguaje de programación Python versión 3.8 acompañado del framework Django versión 3.2, una combinación muy usada para desarrollar aplicaciones web con interfaz de usuario.

Django es un marco web Python de alto nivel que fomenta un desarrollo rápido y un diseño limpio y pragmático. Creado por desarrolladores experimentados, se ocupa de gran parte de las molestias del desarrollo web, por lo que puede concentrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto (Django Software Foundation, s.f.).

4.2.3. API de Detección y Reconocimiento de Rostros

Debido a la complejidad de los algoritmos implementados en este módulo, se usaron las tecnologías descritas en esta sección, puestas en ejecución en una instancia EC2 de Amazon Web Services (AWS).

- **Python 3.7:** Lenguaje de programación de propósito general, orientado a objetos y fácilmente adaptable a arquitecturas de microservicios. Se ha seleccionado este lenguaje dada la naturaleza del proyecto. Este lenguaje ofrece un amplio catálogo de herramientas y librerías de Deep Learning.

- **FastAPI:** Es un marco web moderno, de alto rendimiento para crear API con Python 3.6 o superior, rápido para codificar, listo para producción y seguro. La integración en los proyectos con python consiste en la carga de la librería y el empleo de la estructura propuesta en la documentación de la misma. Posee sentencias para la validación y serialización de datos, además de ofrecer una interfaz de usuario generada automáticamente para la documentación de la API construida (Ramírez, s.f.).

- **Uvicorn:** Es una implementación de servidor de aplicaciones ASGI (Asynchronous Server Gateway Interface), utiliza uvloop y httptools. Con este software desplegamos las APIS construidas con FastAPI, asignando fácilmente el host y puerto donde estará expuesta la API (Uvicorn.org, s.f.).

- **TensorFlow:** TensorFlow es una plataforma de aprendizaje automático de código abierto de extremo a extremo. Para este proyecto se utilizó la versión 2. Con esta plataforma se pueden construir y entrenar redes neuronales, con el fin de detectar patrones y correlaciones, análogo al aprendizaje y razonamiento usado por los humanos (Google, 2015). TensorFlow combina cuatro habilidades clave:

  - Ejecución eficiente de operaciones de tensor de bajo nivel en CPU, GPU o TPU.
  - Cálculo del gradiente de expresiones diferenciables arbitrarias.
  - Escalamiento del cómputo a muchos dispositivos, como clústeres de cientos de GPU.
  - Exportación de programas gráficos a tiempos de ejecución externos, como servidores, navegadores, dispositivos móviles e integrados.

- **Keras:** es una biblioteca de redes neuronales de código abierto escrita en Python, capaz de ejecutarse sobre TensorFlow. Keras es una interfaz accesible y altamente productiva para resolver problemas de aprendizaje automático, con un enfoque en el aprendizaje profundo moderno. Proporciona abstracciones esenciales y bloques de construcción para desarrollar y enviar soluciones de aprendizaje automático con alta velocidad de iteración (Keras Team, s.f.).

Keras permite a los ingenieros e investigadores aprovechar al máximo la escalabilidad y las capacidades multiplataforma de TensorFlow 2: puede ejecutar Keras en TPU o en grandes grupos de GPU, y puede exportar sus modelos de Keras para ejecutarlos en el navegador o en un dispositivo móvil. La versión usada en este proyecto es 2.6.0.

- **NumPy:** es el paquete fundamental para la computación científica en Python. Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas) y una variedad de rutinas para operaciones rápidas en matrices, que incluyen manipulación matemática, lógica, de formas, clasificación, selección, E/S., transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas y simulación aleatoria (NumPy Developers, s.f.). La versión usada en este proyecto es 1.19.5.

- **OpenCV-Python:** es una biblioteca de enlaces de Python diseñada para resolver problemas de visión por computadora originalmente implementada en C++ (OpenCV, s.f.). Con esta biblioteca se puede realizar preprocesado de imágenes y gestión de archivos de imagen, para dejarlas listas para alimentar los modelos de detección y reconocimiento. La versión usada en este proyecto es 4.5.3.56.

- **Deepface:** Es un marco híbrido de reconocimiento facial que envuelve modelos de última generación: VGG-Face, Google FaceNet, OpenFace, Facebook Deepface, DeepID, ArcFace y Dlib (Serengil, 2022). La versión usada en este proyecto es 0.0.68.

- **Urllib3:** Es una biblioteca HTTP, para Python con agrupación de conexiones seguras para subprocesos y publicación de archivos (Petrov, 2022). La versión usada en este proyecto es 1.26.7.

- **Base64:** Este módulo proporciona funciones para codificar datos binarios en caracteres ASCII imprimibles y decodificar dichas codificaciones en datos binarios. Proporciona funciones de codificación y decodificación especificadas en RFC 4648, que define los algoritmos Base16, Base32 y Base64, y para las codificaciones estándar de facto Ascii85 y Base85.

  Las codificaciones RFC 4648 son adecuadas para codificar datos binarios para que puedan enviarse de forma segura por correo electrónico, usarse como parte de las URL o incluirse como parte de una solicitud HTTP POST (Python Software Foundation, 2021).

- **InsightFace:** Es una biblioteca de Python integrada para el análisis de rostros en 2D y 3D. InsightFace implementa de manera eficiente una rica variedad de algoritmos de reconocimiento facial, detección facial y alineación facial, que se optimizan tanto para el entrenamiento como para el despliegue (InsightFace, s.f.). La versión usada en este proyecto es 0.5.

- **Apache MXNet:** Es un marco DL diseñado tanto para la eficiencia como para la flexibilidad (Apache Software Foundation, s.f.). Está desarrollado por Pedro Domingos y un equipo de investigadores de la Universidad de Washington, también es parte del DMLC (DMLC 2018). MXNet es portátil y liviano, y se escala de manera efectiva a múltiples GPU y múltiples máquinas. También admite una implementación eficiente de modelos entrenados en dispositivos de gama baja para inferencia, como dispositivos móviles (usando Amalgamation), dispositivos IoT (usando AWS Greengrass), Serverless (usando AWS Lambda) o contenedores.

  MXNet tiene una licencia Apache-2.0 y tiene un amplio soporte de lenguaje API para R, Python, Julia y otros lenguajes (Chen et al. 2015). MXNet es compatible con los principales proveedores de nube pública (Chen y cols., 2015). La versión usada en este proyecto es 1.8.0.

- **ONNX:** Es un formato abierto creado para representar modelos de aprendizaje automático. ONNX define un conjunto común de operadores, los componentes básicos de los modelos de aprendizaje automático y aprendizaje profundo, y un formato de archivo común para permitir que los desarrolladores de IA utilicen modelos con una variedad de marcos, herramientas, tiempos de ejecución y compiladores (ONNX, s.f.-b). La versión usada en este proyecto es 1.10.1.

- **ONNX Runtime:** Es un acelerador para modelos de aprendizaje automático con soporte multiplataforma y una interfaz flexible para integrarse con bibliotecas específicas de hardware. ONNX Runtime se puede usar con modelos de PyTorch, Tensorflow/Keras, TFLite, scikit-learn y otros marcos (ONNX, s.f.-a). La versión usada en este proyecto es 1.9.0.

4.2.4. Aplicación de Captura de Imagen

Para el desarrollo de este módulo se usaron las tecnologías de: HTML-5, CSS-3, Bootstrap 4 y JavaScript ES6. Mismas tecnologías usadas para el desarrollo del módulo Dashboard UI. Como parte del proceso de inicio de captura, se establece que los instructores pueden iniciar

la grabación de las cámaras por medio de la aplicación móvil, para realizar la comunicación entre el dispositivo móvil y la interfaz web de este módulo, se usaron los siguientes protocolos de comunicación:

- **HTTP:** Hypertext Transfer Protocol (HTTP) (o Protocolo de Transferencia de Hipertexto en español) es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también. Sigue el clásico modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta del mismo. Se trata de un protocolo sin estado, lo que significa que el servidor no guarda ningún dato (estado) entre dos peticiones. Aunque en la mayoría de los casos se basa en una conexión del tipo TCP/IP, puede ser usado sobre cualquier capa de transporte segura o de confianza, es decir, sobre cualquier protocolo que no pierda mensajes silenciosamente, tal como UDP.
- **MQTT:** es un protocolo de red ligero de publicación-suscripción que transporta mensajes entre dispositivos. El protocolo generalmente se ejecuta sobre TCP / IP; sin embargo, cualquier protocolo de red que proporcione conexiones bidireccionales ordenadas y sin pérdidas puede admitir MQTT (Oasis Open, 2019).

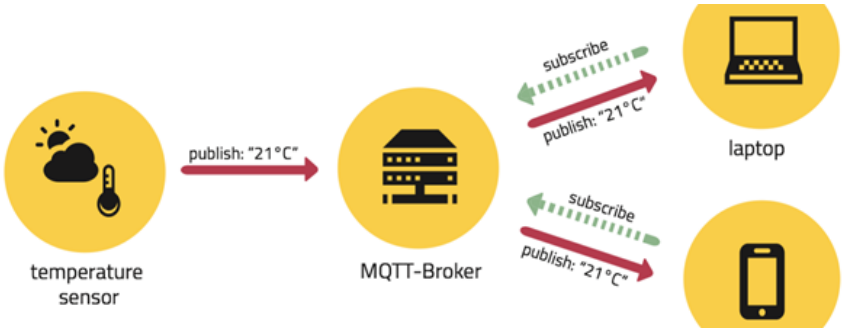


Figura 4.1: Arquitectura funcionamiento protocolo MQTT.
(Stanford-Clark y Truong, 2013)

4.2.5. Gestión de Datos y Embeddings

Para el almacenamiento de datos relacionado con la gestión de información y los embeddings, seleccionamos el SGDB PostgreSQL, el cual es un poderoso sistema de base de datos relacional de código abierto con más de 30 años de desarrollo activo que le ha valido una sólida reputación por su confiabilidad, robustez de funciones y rendimiento (The PostgreSQL Global Development Group, s.f.).

4.2.6. Aplicación Móvil (Captura imagen Offline)

La aplicación móvil para dispositivos Android es una alternativa para dar cobertura en los ambientes de formación que no cuentan con conectividad de internet. Los instructores cargaran la información de los programas de formación donde imparten las clases, cuando se encuentren en ambientes sin conectividad la aplicación permite almacenar de manera local las imágenes del aula para sincronizarlas en el momento que tengan internet.

- **Android Studio:** Es el IDE oficial de Android que se creó exclusivamente a fin de acelerar el desarrollo y ayudar a compilar apps de la más alta calidad para todos los dispositivos Android, basado en IntelliJ IDEA, proporciona el menor tiempo de respuesta en tu flujo de trabajo de codificación y ejecución. Para este proyecto se usó la versión Android Studio Bumblebee 2021.1.1.

- **JAVA:** Es un lenguaje de programación de propósito general y orientado a objetos; dentro de su versatilidad es empleado para el desarrollo de aplicaciones móviles para Android.

- **SDK:** Software development kit, abreviado SDK, es un paquete de herramientas y datos que facilita e incluso permite a los programadores desarrollar programas en un lenguaje concreto o para una plataforma o aplicación específica. Para el desarrollo de aplicaciones móviles existe un conjunto amplio de SDK los cuales nos indican para cual sistema operativo se desarrollará la aplicación (Desarrollador Android, 2015). Para este proyecto se usó el SDK de la API 21 (Android 5.0 Lollipop).

- **Volley 1.1.2:** Volley es una biblioteca HTTP que facilita y agiliza el uso de redes en apps para Android. Se integra fácilmente con cualquier protocolo y, además, incluye compatibilidad con strings sin procesar, imágenes y JSON. Volley elimina la necesidad de escribir código estándar y te permite concentrarte en la lógica que es específica de tu app (Revelo, 2015).

4.3. PREPARACIÓN DE DATOS

Para la experimentación se utiliza FR-ARCA Dataset. Debido que el objetivo es el desarrollo de un sistema piloto el cual se usará en los ambientes de formación del centro de atención sector agropecuario, consideramos que los experimentos realizados en el Capítulo 4, deben aproximarse a escenario real, por lo tanto, fue necesario la construcción del dataset antes mencionado.

4.3.1. Software de recolección de rostros

Es una aplicación web desarrollada con el fin de extender la participación de los aprendices en este proyecto, la aplicación cuenta con la base de datos de aprendices matriculados en el programa de formación Tecnólogo en Análisis y Desarrollo de Sistemas de Información.

Los aprendices ingresaron a la aplicación web, digitando el documento de identidad, aceptaron el consentimiento informado donde se explica el tratamiento de datos y el objetivo por el que se recolectaron; e iniciaron el proceso de captura de las imágenes del rostro.

Con ayuda de esta herramienta, inicialmente se lograron recolectar 290 imágenes de rostro de 58 aprendices distintos.

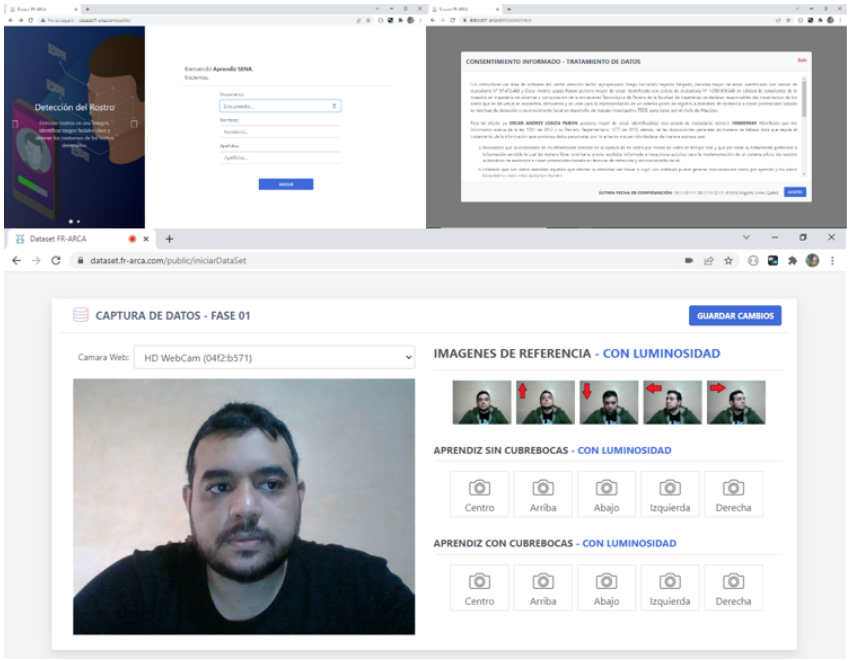


Figura 4.2: Interfaz de la aplicación para recolección de imágenes de aprendices

4.3.2. FR-ARCA Dataset

El conjunto de datos FR-ARCA, fue construido para este proyecto, con el fin de realizar pruebas de validación sobre los modelos seleccionados, para este proceso participaron los aprendices del SENA del Centro de Atención Sector Agropecuario en Risaralda.

El dataset construido durante el proceso de recolección de datos tiene 290 imágenes.

A este set de imágenes se le realizó una revisión en la cual se eliminaron: las imágenes en las cuales no se encontraron rostros, imágenes con el rostro recortado, imágenes con el rostro difuminado o pixelado, es decir, se eliminaron las imágenes que no cumplen con las condiciones adecuadas para realizar las evaluaciones del sistema. Finalmente, y con el fin de tener datos balanceados únicamente fueron seleccionados los aprendices con las 5 imágenes completas de rostro en diferentes poses.

Luego del anterior proceso de selección de datos, las imágenes del dataset resultante se encuentran distribuidas bajo las siguientes condiciones:

- ***Grupos:***
  Grupo 1 código 1564696, 5 aprendices.
  Grupo 2 código 1624722, 6 aprendices.
  Grupo 3 código 1821662, 8 aprendices.
  Grupo 4 código 1908104, 8 aprendices.
  Grupo 5 código 2055000, 9 aprendices.
  Grupo 6 código 2204685, 6 aprendices.
- ***Número de Aprendices:*** 42
- ***Poses:*** Cada aprendiz tiene 5 tipos de pose, las cuales consistían en dirigir el rostro viendo hacia el frente, parte superior, parte inferior, izquierda y derecha de la cámara.

Las imágenes están etiquetadas con el documento de identificación de cada aprendiz, el directorio de cada aprendiz se encuentra en la carpeta del grupo al cual pertenece, que están etiquetadas con el código de grupo.

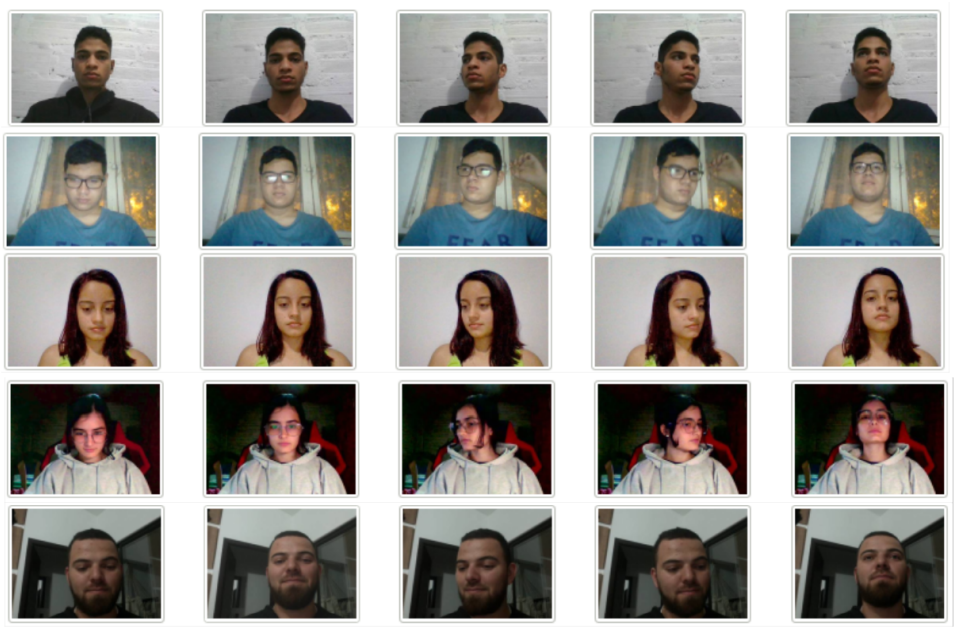


Figura 4.3: Muestra de imágenes recolectadas para el dataset

En la figura 4.3 se muestran ejemplos de las imágenes recolectadas con el software de recolección de rostros presentado anteriormente. Por tal motivo las imágenes tienen variedad de resoluciones, iluminación, accesorios en el rostro, expresión facial, sexo biológico y tono de la piel; datos que ayudan a probar el sistema con imágenes de rostro en diferentes condiciones.

4.4. EXPERIMENTOS

Esta sección presenta la explicación de los experimentos realizados para la evaluación al sistema FR-ARCA la cual se enfoca en medir la precisión para registrar asistencia a clases. FR-ARCA tiene implementado diferentes métodos de detección y reconocimiento de rostros (ver tabla 4.1) por lo cual se ponen a prueba todos los métodos de detección y reconocimiento implementados, realizando una serie de experimentos con el conjunto de datos propio del proyecto FR-ARCA Dataset.

ID Método	Detection	Recognition
1	Retinaface	Deepface
2	RetinaFace	OpenFace
3	RetinaFace	Facenet
4	RetinaFace	VGG-Face
5	RetinaFace	ArcFace Keras
6	MTCNN	ArcFace Keras
7	MTCNN	DeepFace
8	MTCNN	OpenFace
9	MTCNN	Facenet
10	MTCNN	VGG-Face
11	SCRFD-10GF	ArcFace - ResNet100
12	SCRFD-10GF	ArcFace - ResNet50
13	SCRFD-2.5GF	ArcFace - ResNet50
14	SCRFD-500MF	MobileFaceNets
15	SCRFD-500MF Align False	MobileFaceNets

Tabla 4.1: Asociación de modelos de detección y reconocimiento, usados en los experimentos 1 y 2

4.4.1. Experimento 1. Reconocimiento de rostros con búsqueda de máxima similitud segmentada por grupos, para el cálculo de *accuracy* de diferentes métodos de detección y reconocimiento de rostros.

Se tiene el dataset con 5 imágenes por cada aprendiz, se toma una sola imagen para generar el vector de características principales (embedding) y se almacena en base de datos. Posteriormente se envían las 4 imágenes distintas a la anterior, el sistema realiza el proceso y extrae el embedding de cada imagen y luego selecciona los embedding del grupo al cual corresponden esos aprendices y compara calculando la distancia del coseno entre dos embeddings buscando la mejor similitud; para cada imagen el sistema retorna los datos del aprendiz con quien tenga mayor similitud (ver figura 5.3).

Con el resultado se construye un dataframe que contiene la identificación real de cada imagen enviada, el valor de similitud calculado y la identificación predicha por el sistema. Con estos datos se construye la matriz de confusión multiclase y se calcula *accuracy*.

4.4.2. Experimento 2. Reconocimiento de rostros con búsqueda de máxima similitud en el conjunto completo de embeddings, para el cálculo de *accuracy* de diferentes métodos de detección y reconocimiento de rostros.

Este experimento es similar al experimento 1; el cambio radica en que el embedding de cada imagen obtenido se compara con todos los embeddings almacenados en base de datos, calculando la distancia del coseno entre ellos y buscando la mejor similitud. Con el resultado se construye un dataframe que contiene la identificación real de cada imagen enviada, el valor de similitud calculado y le identificación predicha por el sistema. Con estos resultados se construye la matriz de confusión multiclase y se calcula *accuracy*.

4.4.3. Experimento 3. Tasa de verificación de pares de imágenes.

Para este experimento formulamos el problema con dos clases: si asiste (1), no asiste (0). Tomamos la combinación posible de todas las imágenes de un aprendiz, como se muestra en la figura 4.4.

Figura 4.4: Imágenes de rostros de un aprendiz

Se realiza todas las combinaciones posibles de esas imágenes: F1-F2, F1-F3, F1-F4, F1-F5, F2-F3, F2-F4, F2-F5, F3-F4, F3-F5, F4-F5. Se envía cada pareja de imágenes al sistema y como resultado se obtiene el valor de similitud de cada pareja, estas se etiquetan con la clase 1. Se repite este proceso para todas las imágenes de los aprendices del dataset FR-ARCA. La clase 1 significa que cada pareja pertenece a la misma identidad. También se deben generar parejas pertenecientes a distintas identidades, procesarlas en el sistema y etiquetarlas en la clase 0, para esto se asocian las imágenes de los aprendices a imágenes de distintas identidades. Como el número de parejas de clase 1 debe ser igual al número de parejas de clase 0 entonces se balancean los datos igualando el número de parejas de las dos clases.

Con los datos de similitud que retorna el sistema FR-ARCA y teniendo en cuenta la clase real, se realizaron las curvas ROC y AUC, probando diferentes umbrales.

4.4.4. Experimento 4. Prueba de comparación de FR-ARCA, CompreFace, Azure Face Recognition y Amazon Rekognition con respecto a la precisión en el test de tasa de verificación de pares de imágenes.

Este experimento es similar al experimento 3, aplicando la tasa de verificación de imágenes en los siguientes servicios de reconocimiento existentes en el mercado:

- ***Exadel CompreFace:*** es un servicio de reconocimiento facial gratuito y de código abierto que se puede integrar fácilmente en cualquier sistema sin conocimientos previos de aprendizaje automático. CompreFace proporciona REST API para reconocimiento facial, verificación facial, detección de rostros, detección de puntos de referencia, reconocimiento de edad y género y se implementa fácilmente con Docker.
- ***Azure Face:*** ofrece algoritmos de IA que detectan, reconocen y analizan caras humanas en las imágenes. El software de reconocimiento facial es importante en muchos escenarios diferentes, como la verificación de identidad, el control de acceso sin contacto y el desenfoque facial para la privacidad.
- ***Amazon Rekognition:*** proporciona funciones de análisis facial, comparación de rostros y búsqueda de rostros de alta precisión. Puede detectar, analizar y comparar rostros para una amplia variedad de casos de uso, incluida la verificación de usuarios, la catalogación, el conteo de personas y la seguridad pública.

Con los datos de similitud que retornan cada uno de los servicios y teniendo en cuenta su clase real, podemos realizar las curvas ROC y AUC, probando diferentes umbrales y obtener resultados.

4.4.5. Experimento 5. Prueba de registro de asistencia a clases con FR-ARCA en ambientes de aprendizaje internos.

Con este experimento se realiza la prueba del sistema FR-ARCA en un entorno real. Se instala un dispositivo de captura de imágenes en dos ambientes de formación en los cuales están programados 3 grupos.

Ambiente de Aprendizaje	Grupo	Horario	Número de Aprendices
01 - Sede Calle 20	2055005	13:00 a 19:00	22
01 - Sede Calle 20	2055000	7:00 a 13:00	13
02 - Sede Calle 20	2204685	7:00 a 13:00	17

Tabla 4.2: Grupos programados para el experimento en entorno real

Durante 10 días de lunes a viernes se tomó asistencia a clases de los tres grupos con el procedimiento manual (llamando a lista) y además se usa el sistema FR-ARCA para realizar el registro automático de asistencia a clases. El dispositivo de captura de imagen se instala dentro del ambiente de aprendizaje, los aprendices se dirigen al dispositivo durante unos segundos y el sistema les indicara su respectivo registro de asistencia.

4.4.6. Experimento 6. Prueba de registro de asistencia a clases con FR-ARCA en ambientes de aprendizaje externos, usando la aplicación móvil.

Para este experimento se usa la población del experimento 5 mencionados en la Tabla 4.2. Durante 3 prácticas por cada grupo en ambientes externos, se tomó asistencia a clases con el procedimiento manual (llamando a lista) y además se usa el sistema FR-ARCA para realizar el registro automático de asistencia a clases de manera offline utilizando el dispositivo móvil. El instructor captura las imágenes del grupo usando la aplicación móvil de FR-ARCA, cuando el dispositivo tenga disponible una conexión a internet, el instructor puede realizar la sincronización con la API de detección y reconocimiento la cual le indicara su respectivo registro de asistencia para las imágenes capturadas anteriormente.

4.5. MÉTRICAS DE EVALUACIÓN

Hoy en día se utilizan varias medidas para evaluar el rendimiento del sistema de reconocimiento facial. Una de las herramientas más intuitivas y sencillas para evaluar la precisión y exactitud de los modelos, es la matriz de confusión, a partir de sus valores se obtienen las diferentes métricas.

En el caso de reconocimiento binario o reconocimiento de dos clases, el sistema tiene que diferenciar entre criterios faciales y no faciales. El verdadero positivo significa la porción

de imágenes de rostros que el sistema detectará, mientras que el falso positivo significa la porción de imágenes que no son de rostros que se detectarán como rostros. El término verdadero positivo aquí tiene el mismo significado que la tasa de detección y recuperación. Los falsos positivos implican hacer coincidir incorrectamente a las personas con las fotos en la base de datos, y los falsos negativos significan no identificar a las personas incluso cuando su foto está en la base de datos. Hay dos gráficos de evaluación principales: la curva de características operativas del receptor (ROC) y la curva de precisión y recuperación (PR). La curva ROC examina la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos, mientras que la curva PR extrae la relación entre la tasa de detección (recuperación) y la precisión de detección.

4.5.1. Matriz de Confusión

En el caso de un problema binario como el problema de registro a clases, un modelo clasifica como 0 o 1 un conjunto de datos, y a partir de ello se crea la matriz de confusión.

	Positive	**Negative**
True	True Positive (TP)	False Negative (FN)
False	False Positive (TP)	True Negative (FN)

Tabla 4.3: Estructura de la matriz de confusión

- ***TP (True Positive):*** son los valores que el algoritmo clasifica como aprendiz que asiste y es registrado como asistente.
- ***TN (True Negative):*** son los valores que el algoritmo clasifica como aprendiz que no asiste y no es registrado como asistente.
- ***FP (False Positive):*** son los valores que el algoritmo clasifica como aprendiz que no asiste y es registrado como asistente.
- ***FN (False Negative):*** son los valores que el algoritmo clasifica aprendiz que asiste y no es registrado como asistente.

Un registro de asistencia básicamente es un clasificador que trata de distinguir a los aprendices que fueron a clases y los que no. Sin embargo, como cualquier sistema, es susceptible a errores. En este sentido se puede decir que es posible que se presenten cuatro escenarios para cada aprendiz.

a. Aprendiz que asiste y es registrado como asistente. (Verdadero positivo).

b. Aprendiz que no asiste y no es registrado como asistente. (Verdadero negativo).

c. Aprendiz que asiste y no es registrado como asistente. (Falso negativo).

d. Aprendiz que no asiste y es registrado como asistente. (Falso positivo).

A continuación, se describen las métricas asociadas al cálculo de la matriz de confusión, estas métricas las podemos obtener de manera automática usando las librerías de Python.

- ***Accuracy:*** Representa el porcentaje total de valores correctamente clasificados, tanto positivos como negativos, por lo que corresponde a las predicciones correctas realizadas por el modelo sobre el número total de ejemplos. También se le conoce como exactitud. Es recomendable utilizar esta métrica en problemas en los que los datos están balanceados, es decir, que exista la misma cantidad de valores de cada etiqueta (Catal, 2012). Esta métrica se define a partir de la relación dada en la Ecuación 4.1.

$$Accuracy = \frac{TP + TN}{TP + TN - FP + FN} \tag{4.1}$$

- **Precisión:** Es la métrica utilizada para saber qué porcentaje de valores que se han clasificado como positivos son realmente positivos, es decir que evalúa los datos por el rendimiento de predicciones positivas (Catal, 2012). La forma de calcularla se expresa en la Ecuación 4.2.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

- **Recall:** es la proporción de verdaderos positivos clasificados correctamente, es utilizada para saber cuántos valores positivos cumplieron con esta condición. También se le denomina como sensibilidad (Catal, 2012).

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

- **Especificidad:** Mide el número de predicciones negativas correctas dividido sobre el número total de negativos, es la tasa negativa verdadera. Es lo opuesto a la sensibilidad (1 - sensibilidad) (Catal, 2012).

$$Especificidad = \frac{TN}{TN + FP} \tag{4.4}$$

- **F1 Score:** El puntaje F1, es el promedio ponderado de precisión y sensibilidad considerando tanto los FP como los FN. Esta es una métrica muy utilizada en problemas en los que el conjunto de datos a analizar está desbalanceado. Esta métrica combina la precisión y la sensibilidad, para obtener un valor mucho más objetivo (Catal, 2012).

$$F1 = \frac{2 * recall * precision}{recall + precision} \tag{4.5}$$

4.5.2. Curvas ROC y AUC

Las características operativas del receptor (ROC) son un gráfico que se utiliza para organizar y visualizar el rendimiento de un sistema. Es una opción distinta para las curvas de recuperación de precisión (Fawcett, 2006). Los gráficos ROC se utilizan normalmente en la toma de decisiones médicas y, en los últimos años, se utilizan cada vez más en la investigación de procesamiento de datos y aprendizaje automático. Es una representación gráfica para mostrar la transición entre TPR y FPR. TPR indica valores positivos totales o correctamente clasificados y representados en el eje y, mientras que FPR indica valores negativos totales o clasificados incorrectamente representados en el eje x.

La curva ROC como la de Figura 8 es una gráfica que enfrenta la tasa de falsos positivos (FPR, False Positive Rate) sobre el eje X contra el porcentaje de verdaderos positivos (TPR, True Positive Rate) o Recall sobre el eje Y, es decir que, enfrenta la falsa alarma (1-especificidad) versus la tasa de éxito (sensibilidad) del modelo. El mejor modelo es aquel cuya curva tenga el mejor balance entre sensibilidad y 1-especificidad buscando tener la mayor FPR y la menor TPR.

A partir de la curva ROC, se puede obtener una métrica sólida para problemas de clasificación binaria conocida como el área bajo la curva, AUC, y su valor se encuentra en un rango entre 0 y 1. (Ling, Huang, y Zhang, 2003) propuso el uso de AUC como parámetro de evaluación del rendimiento de clasificadores demostrando que es mucho más apropiado que la precisión para conjuntos de datos tanto equilibrados como desequilibrados.

Cuando AUC es aproximadamente 0, el modelo predice la clase negativa como una clase positiva y viceversa. Cuando el AUC es aproximadamente 0.5, el modelo no tiene capacidad de discriminación para distinguir entre la clase positiva y clase negativa, por tanto es equivalente a tener un modelo aleatorio. Cuando AUC es 1, se tiene un resultado óptimo que indica que el modelo generaliza muy bien, esta es una situación ideal ya que las dos curvas

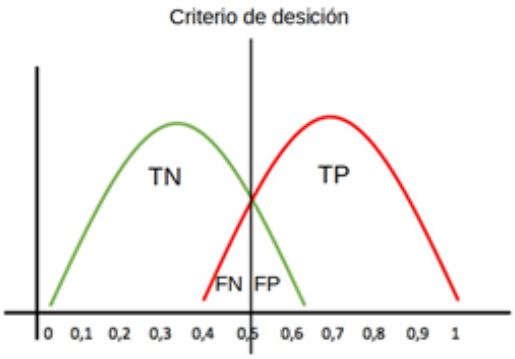


Figura 4.5: Curva de decisión binaria

de decisión no se superponen en absoluto y el modelo tiene una medida ideal de separación perfectamente capaz de distinguir entre clase positiva y clase negativa.

Otra medida de evaluación es la Tasa de Verificación (Ahlgren y Grönqvist, 2006), la cual se basa en una lista de pares de imágenes, donde se comparan pares iguales y pares con identidades diferentes. Dadas las listas de similitudes de tipos, se puede calcular el gráfico ROC y, finalmente, la tasa de verificación. Hay algunas medidas más, como Half Total Error Rate y similares, que se basan en conjuntos de desarrollo y evaluación independientes. La prueba de validación es un tipo de prueba que se utiliza para identificar rostros. El sistema de verificación utiliza algunas medidas (es decir, la misma tasa de error), mientras que otras suelen adoptarse para los sistemas de reconocimiento (es decir, la tasa de reconocimiento).

En la tarea de reconocimiento facial, para determinar si dos imágenes provienen de la misma persona, ROC primero calcula la medida de la distancia o la similitud entre las imágenes y luego completa el reconocimiento de acuerdo con el umbral. La abscisa de la curva ROC representa la tasa de falsos positivos (FPR) y la ordenada representa la tasa de recuperación o la tasa de verdaderos positivos (TPR) (Ahlgren y Grönqvist, 2006). Las definiciones de FPR y TPR son las siguientes:

$$TPR = \frac{TP}{TP + FN} \tag{4.6}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.7}$$

TP se refiere al par de muestras positivas pronosticado correctamente por el modelo, FN se refiere incorrectamente al par de muestras positivas predicho por el modelo, TN se refiere a la muestra negativa par predicho correctamente por el modelo, y FP se refiere a el par de muestras negativas mal predicho por el modelo.

Capítulo 5

DESARROLLO DE LA SOLUCIÓN

Este capítulo inicialmente presenta el modelo realizado para el sistema FR-ARCA el cual muestra a nivel general los procesos para el registro automático de asistencia a clases presenciales basado en técnicas de detección y reconocimiento facial, utilizando modelos de aprendizaje profundo.

Luego se describen las principales consideraciones y las propuestas de solución que se tuvieron en cuenta para el diseño y desarrollo del sistema FR-ARCA, como también las técnicas implementadas para realizar el sistema de reconocimiento facial sin reentrenamiento de redes neuronales convolucionales (CNN) para nuevos grupos de aprendices. En la selección de métodos de detección y reconocimiento de rostros se presenta la descripción de cada modelo y sus principales características, como también las técnicas de alineación de rostros utilizadas en este proyecto.

Para implementar modelos de Deep Learning al software FR-ARCA se contemplaron algunos retos de la ingeniería de software (SE) aplicada al Machine Learning (ML). Teniendo en cuenta estos retos, con los fundamentos de la ingeniería de software se definió la arquitectura para desarrollar el sistema FR-ARCA el cual presenta el diseño de cada uno de los módulos propuestos.

5.1. DISEÑO DEL MODELO DEL SISTEMA

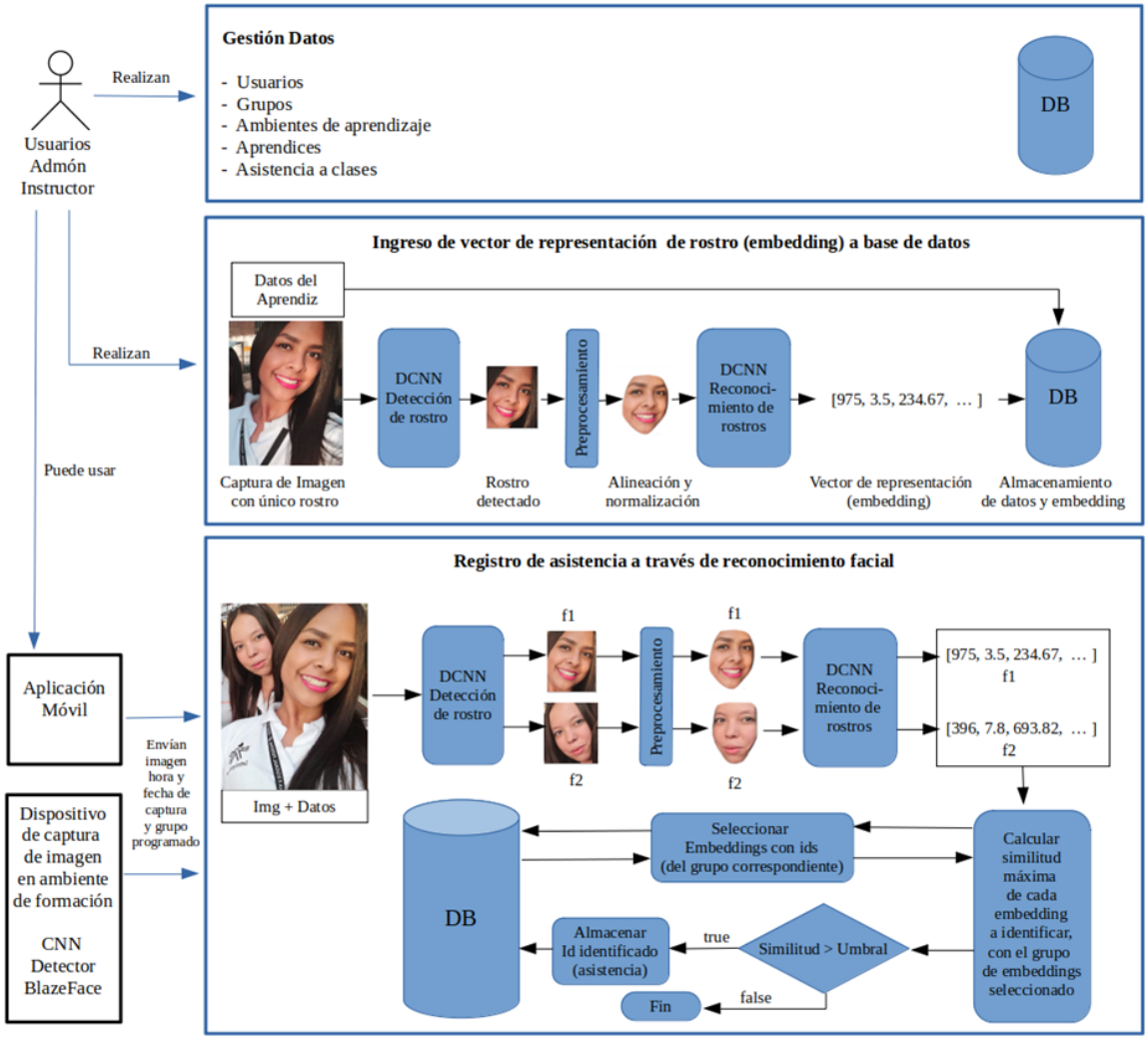


Figura 5.1: Modelo de procesos de FR-ARCA.

En la Figura 5.1 se describe el modelo propuesto para FR-ARCA. Para realizar un correcto registro de asistencia a clases, primero se gestiona la información relevante de este proceso como lo es: la gestión usuarios, programas de formación, ambientes de aprendizaje, grupos, aprendices y asistencia a clases. Luego es necesario ingresar los datos de aprendices a base de datos con la representación vectorial de su rostro, esto con el fin de posteriormente ser identificado. La detección de rostros tiene como objetivo localizar la región del mismo en la imagen de entrada; luego, se procede a la alineación y normalización del dicho rostro y por último, la extracción del vector de representación del rostro (embedding) el cual es almacenado en base de datos.

En el proceso de registro de asistencia a clases a través de reconocimiento facial, hay un primer actor que es el dispositivo de captura de imagen. Este puede ser un dispositivo móvil o dispositivo instalado en el ambiente de aprendizaje; dichos dispositivos se sincronizan con el módulo de gestión de información para almacenar correctamente la asistencia a clases de los grupos de aprendices programados con un instructor especifico, en el respectivo ambiente de formación. La activación y la configuración del dispositivo es realizada mediante el protocolo de transferencia de datos MQTT (Message Queueing Telemetry Transport). A través de estos se ingresan imágenes de aprendices para que el sistema extraiga el embedding de cada rostro, para que sean comparados con los embeddings almacenados en base de datos que pertenezcan al grupo correspondiente, con el fin de encontrar la similitud; Si la similitud supera el umbral configurado, se almacena la asistencia del aprendiz identificado en base de datos con los datos referentes a la programación de clases que envía el dispositivo.

5.2. PRINCIPALES CONSIDERACIONES

La tabla 5.1 presenta las principales consideraciones sobre la solución propuesta a la luz de su respectivo análisis realizado en la fase I incepción, de la metodología RUP (Proceso Racional Unificado).

CONSIDERACIÓN	PROPUESTA DE LA SOLUCIÓN
Gestión de información almacenada en el sistema, se categorizan de la siguiente manera: - Usuarios - Grupos - Ambientes de aprendizaje - Aprendices - Asistencia a clases	Se diseñó la aplicación de interfaz de usuario tipo dashboard o panel de control que permite registrar, editar, eliminar y consultar toda la información que requiere el registro de asistencia a clases. Esta aplicación se construye con una arquitectura API-REST de manera que permita comunicarse con otras aplicaciones o módulos encargados de realizar las tareas de visión por computadora.
La asistencia debe ser tomada en los ambientes de aprendizaje urbanos y rurales de las diferentes subsedes del Centro Atención Sector Agropecuario SENA – Risaralda.	Se diseñó el módulo para las tareas de detección y reconocimiento de rostros, bajo la arquitectura API-REST para que estos servicios puedan ser consumidos a través de peticiones http (hipertext transfer protocol).

CONSIDERACIÓN	PROPUESTA DE LA SOLUCIÓN
Garantizar el funcionamiento del sistema en los ambientes de aprendizaje que se encuentran en zonas rurales, donde se carece de conexiones eléctricas y redes de datos.	Se desarrolla una aplicación para dispositivos móviles, la cual permite capturar de manera offline las imágenes de los aprendices en formación, hace persistencia de datos. Cuando el dispositivo tenga conectividad a internet se sincroniza con el módulo de detección y reconocimiento API-REST para realizar el registro de asistencia.
Ingreso de nuevos grupos de aprendices de manera frecuente, debido que cada trimestre ingresa aprendices de formación titulada según el calendario de oferta de programas de formación.	No es viable que los modelos de detección y reconocimiento de rostros tengan que reentrenarse cada vez que ingresen nuevos aprendices. El Sistema fue diseñado con técnicas de one-shot-learning evitando el reentrenamiento de las redes neuronales para nuevos grupos de aprendices.
Detección y reconocimiento de múltiples aprendices en una sola imagen. En los procesos de formación las clases se imparten a un grupo de aprendices, por tal motivo las imágenes capturadas pueden tener uno o varios rostros.	El módulo API-REST de detección y reconocimiento, está diseñado para que pueda detectar varios rostros en una sola imagen, preprocesarlos, reconocer la identidad de cada uno, almacenar asistencia y retornar resultados.
Para el caso de ambientes de aprendizajes internos en sedes urbanas, el sistema debe tomar las imágenes de los aprendices en el ambiente de formación y realizar el registro de asistencia a clases teniendo en cuenta la programación del grupo.	Se diseñó una aplicación que estará instalada en los dispositivos de captura de imagen ubicados en los ambientes de formación. Esta aplicación es la que realizará el envío automatizado de las imágenes de las aulas al módulo de API-REST de detección y reconocimiento. Estos dispositivos se sincronizan con el módulo de gestión de información para almacenar correctamente la asistencia a clases de los grupos de aprendices programados con un instructor especifico, en el respectivo ambiente de formación. Cuando la aplicación detecta un rostro, se captura la imagen y envía al API-REST de detección y reconocimiento. La activación y la configuración del dispositivo es realizada mediante el protocolo de transferencia de datos MQTT (Message Queueing Telemetry Transport).

CONSIDERACIÓN	PROPUESTA DE LA SOLUCIÓN
Optimizar captura de imagen automática. Solo usar los servicios de reconocimiento cuando detecte rostros, con el fin de reducir el tráfico de datos.	El módulo de captura de imagen fue diseñado con una DCNN de detección liviana de rostros, con el fin de enviar imágenes con rostros y no saturar el sistema con peticiones innecesarias. El sistema de captura se desactiva automáticamente cuando termine de listar a todo el grupo o puede ser desactivado manualmente a través del dispositivo móvil.
Garantizar la precisión del reconocimiento del aprendiz para el registro de asistencia.	Una de las tareas más importantes del sistema es lograr la máxima precisión al reconocer el rostro de un aprendiz para ello en el sistema FR-ARCA se ha implementado con varios modelos de detección y reconocimiento de rostros de última tecnología. Se realiza la evaluación de varios modelos con diferentes experimentos y se presentan resultados. De acuerdo a los resultados se seleccionan los mejores modelos de detección y reconocimientos, además se define el umbral que mejor precisión otorga al sistema.
Garantizar el funcionamiento y el rendimiento del sistema.	Dentro del diseño del sistema, se establecen los diferentes módulos que son empaquetados en contenedores Docker. La orquestación se realiza con docker compose, lo cual permitirá un mejor desempeño y despliegues de microservicios en cloud computing para uso a gran escala.

Tabla 5.1: Consideraciones y análisis de la solución propuesta

5.3. SISTEMA DE RECONOCIMIENTO FACIAL SIN REENTRENAMIENTO DE DCNN PARA NUEVOS GRUPOS DE APRENDICES

El método de elección para el reconocimiento de rostros usando redes neuronales convolucionales profundas (DCNN) es la representación de rostros mediante un vector de características principales (embeddings) (Sun, Wang, y Tang, 2014a) (Schroff y cols., 2015).

Muchas tareas prácticas se pueden resolver construyendo buenas representaciones. Al estudiar otras áreas de aprendizaje automático, como el procesamiento del lenguaje natural y los sistemas de recomendación, es posible encontrar embeddings (Mikolov, Chen, Corrado, y Dean, 2013).

El embedding es un vector de números reales que contiene toda la información extraída. Se supone que los embeddings de objetos semánticamente similares están cerca en términos de alguna métrica. Entonces, si se tiene dos embeddings se puede compararlos entre sí y calcular su similitud. El reconocimiento facial está utilizando el mismo enfoque. Por lo general, se supone que la similitud de un par de rostros se puede obtener directamente calculando la similitud de sus embeddings.

Figura 5.2: Ejemplo de distribución de rostros.

Para comprender mejor cómo se distribuyen los embeddings en el espacio, podemos

suponer que cada rostro de la Figura 5.2 es un embedding reducido a un espacio de dos dimensiones. Se espera que las representaciones pertenecientes a una misma identidad sean más cercanas; mientras que las representaciones de diferentes identidades estén dispersas.

5.3.1. Metodología para reconocimiento de rostros a través de cálculo de similitud

Podemos seleccionar modelos de reconocimiento de rostros que permitan la extracción del vector de representación del rostro que contiene las características principales (embeddings) y almacenarlos en base de datos, para posteriormente realizar búsquedas de similitud de rostros. La siguiente figura explica el proceso propuesto.

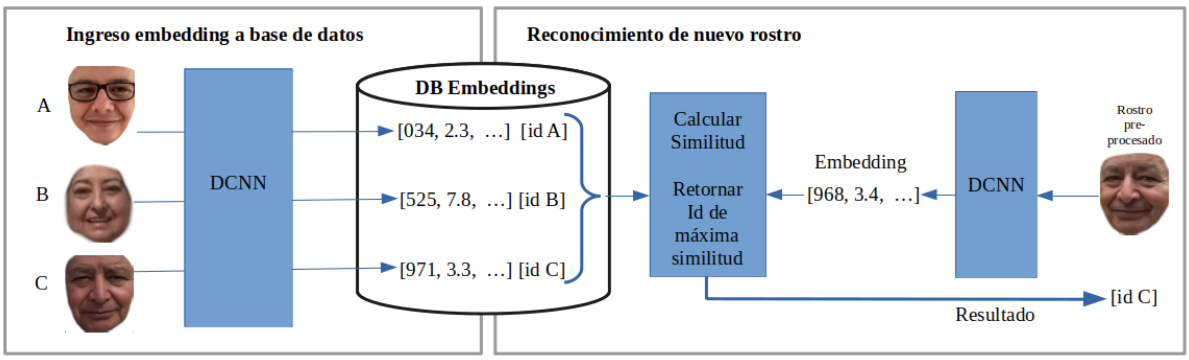


Figura 5.3: Proceso de ingreso y proceso de reconocimiento de rostros a través similitud de embeddings.

Este es un proceso de aprendizaje *One Shot Learning*, puesto que no se necesita que la persona a reconocer este en el set de datos de entrenamiento de la CNN. Por lo tanto, cada vez que ingresa un grupo de aprendices nuevo, simplemente se ingresa el o los embeddings del rostro de cada aprendiz a una base de datos de embeddings. Para realizar el registro de asistencia a clases a través del reconocimiento de rostros, primero se captura la imagen del aprendiz a identificar, se extrae el embedding del nuevo rostro y se calcula la máxima similitud con los embeddings en base de datos; este resultado entrega la identidad del rostro. De esta manera no se tendría que reentrenar las CNN's cuando ingresan nuevos grupos de aprendices.

5.3.2. Cálculo de similitud entre embeddings

Similitud de coseno: El rango de la similitud del coseno está entre -1 y 1. En el espacio del producto interno, esta es una medida de similitud en la dirección (y no en el tamaño) entre dos vectores, que no son el vector cero. La similitud del coseno es igual al coseno entre dos vectores y es lo mismo que un producto interno entre dos vectores normalizados:

- Si dos vectores están en la misma dirección, la similitud del coseno es 1.
- Si dos vectores están en un ángulo de 90 grados, la similitud del coseno es 0.
- Si son opuestos, entonces la similitud del coseno es -1.

La similitud del coseno de dos vectores se obtiene mediante la Fórmula 5.1.

$$cosinesimilarity = S_c(A,B) := cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.1}$$

5.4. SELECCIÓN DE MODELOS

Los métodos de detección y reconocimiento de rostros implementados en el proyecto se seleccionaron realizando el estudio del estado del arte, filtrando por: alta precisión en los resultados de evaluación publicados, implementación en el lenguaje de programación Python, que sus licencias permitan la implementación libre y que proporcionen modelos pre entrenados con el fin de evaluar varios en el contexto del proyecto.

Cabe anotar que todos los modelos seleccionados fueron implementados en el sistema FR-ARCA de este proyecto, y estos se pueden seleccionar configurándolos en el archivo correspondiente.

5.4.1. Modelos de detección de rostros

- **SCRFD (Sample and Computation Redistribution for Efficient Face Detection) (J. Guo y cols., 2021)** Es un enfoque eficiente de detección de rostros de alta precisión, en particular, SCRFD-34GF supera al mejor competidor, TinaFace (Deepinsight, 2021), en un 3,86 % (AP en configuración fija) y es 3 veces más rápido en GPU con imágenes de resolución VGA.

En este proyecto se usó SCRFD-10GF con CNN ResNet34x0.25, SCRFD-2.5GF con CNN ResNet34x0.25 y SCRFD-500MF con CNN Depth-wise Conv. GF significa que el modelo cuesta xG FLOP mientras la imagen de entrada tiene una resolución VGA (640x480) (Deepinsight, 2021).

- **RetinaFace (Deng, Guo, Zhou, y cols., 2019):** La implementación original se basa en mxnet y fue reimplementado en Tensorflow 2.0, en una red troncal Resnet50 por Stanislas Bertrand. La reimplementación proporciona el modelo preentrenado que en su validación en modo fácil solo está 1 % por debajo del original y en modo difícil 2 % por debajo de la implementación original (Bertrand, 2020). La versión usada en este proyecto es 0.0.5.

- **MTCNN:** Librería para detección de rostros originalmente implementada en el marco de aprendizaje profundo Caffe (Berkeleyvision, s.f.) existe una reimplementación de Iván de Paz Centeno, o ipazc (Centeno, 2021), disponible bajo una licencia permisiva de código abierto del MIT que proporciona una implementación de la arquitectura MTCNN usando TensorFlow y OpenCV y un modelo preentrenado de alto rendimiento. La versión usada en este proyecto es 0.1.1

Con el fin de optimizar las peticiones que realiza el módulo de captura de imágenes y evitar saturar la API de detección y reconocimiento con peticiones innecesarias, desde este módulo se filtran las imágenes que solo posean rostros usando el detector BlazeFace. En su artículo se describe como un detector de rostros liviano y de buen rendimiento diseñado para la inferencia de GPU móvil. Funciona a una velocidad de 200–1000+ FPS en dispositivos insignia. Incluyen una red de extracción de características liviana inspirada en MobileNetV1/V2, pero distinta de él, un esquema de anclaje compatible con GPU modificado de Single Shot MultiBox Detector (SSD) y una alternativa de estrategia de resolución de vínculo mejorada a la supresión no máxima (Bazarevsky y cols., 2019).

Los modelos preentrenados de detección de rostros utilizados en este proyecto, tienen licencia que permite usarlos con fines de investigación.

5.4.2. Modelos de reconocimiento de rostros

Los métodos descritos en la tabla 5.2 fueron implementados en FR-ARCA, con sus respectivos modelos preentrenados, los cuales tienen licencia que permite usarlos con fines de

investigación.

ID	Método.	Accuracy LFW	Dataset entrenamiento	Arquitectura CNN	Tamaño de Entrada	Dim. Vector salida	Fuente modelo preentrenado
1	FaceNet	99,65	VGGFace2	Inception ResNet v1	160, 160, 3	128	(Sandberg, 2018)
2	VGG-Face	97,27	desconocido	VGG-Very-Deep-16	224, 224, 3	2622	(Serengil, 2019)
3	OpenFace	92,10	CASIA-WebFace.	nn4.small1	96, 96, 3	128	(Victor Iwantoox-xoox, 2021)
4	DeepFace	97,35	VGGFace2	AlexNet	152, 152, 3	4096	(Ghosh, 2019)
5	ArcFace en Keras	99.4	CASIA, E40	ResNet34	112, 112, 3	512	(Leondgarse, 2022)
6	ArcFace	99,83	Glint360K	ResNet100	112, 112, 3	512	(Deepinsight, 2022c)
7	ArcFace	99,83	WebFace600K	ResNet50	112, 112, 3	512	(Deepinsight, 2022c)
8	ArcFace	99,83	WebFace600K	ResNet50	112, 112, 3	512	(Deepinsight, 2022c)
9	ArcFace	99,70	WebFace600K	MobileFaceNets	112, 112, 3	512	(Deepinsight, 2022c)
10	ArcFace	99,70	WebFace600K	MobileFaceNets	112, 112, 3	512	(Deepinsight, 2022c)

Tabla 5.2: Datos técnicos de los métodos de reconocimiento facial seleccionados.

Los datos muestran la precisión obtenida en evaluaciones sobre el dataset LFW, como también el dataset y la arquitectura de CNN con los cuales se entrenaron los modelos; además presentamos el tamaño de entrada de la imagen (bounding boxes) en forma ancho, alto y los 3 canales de colores RGB; finalmente se encuentra el tamaño de la dimensión del embedding que retorna cada método y la fuente del modelo preentrenado.

5.4.3. Alineación y normalización de rostros

FaceNet de Google en su artículo (Schroff y cols., 2015) menciona que la alineación facial incrementa la precisión de su modelo en casi un 1 %, del 98.87 % al 99.63 %. Este proyecto se

realiza la alineación de rostros de dos maneras:

- Los métodos de la Tabla 5.2 con identificadores 1,2,3,4,5 utilizan alineación dada en (Serengil y Ozpinar, 2020), la cual requiere aplicar algunos trucos trigonométricos para alinear las caras correctamente. Si ya se conoce la ubicación de los ojos, entonces se puede dibujar un triángulo rectángulo. Un lado del triángulo debe ser horizontal. De esta forma se puede conocer la longitud de los 3 lados. Por lo tanto, el ángulo entre la hipotenusa y el lado horizontal se puede calcular aplicando la regla del coseno en la fórmula 5.2 y luego la función trigonométrica inversa utilizada para recuperar el ángulo A; ya que la regla del coseno devuelve el valor del coseno.

$$cos(A) = (b^2 + c^2 + a^2)/(2bc) \tag{5.2}$$

- Los métodos de la Tabla 5.2 con identificadores 6, 7, 8, 9 y 10 usan Regresión de coordenadas del proyecto InsightFace (Deepinsight, 2022a). Este proyecto proporciona modelos de puntos de referencia faciales ligeros con regresión de coordenadas rápidas. La entrada de estos modelos es una imagen de cara recortada suelta de 192x192, mientras que la salida son las coordenadas directas del punto de referencia. La implementación en Python tiene como arquitectura CNN MobileNet0.5.

5.5. ARQUITECTURA

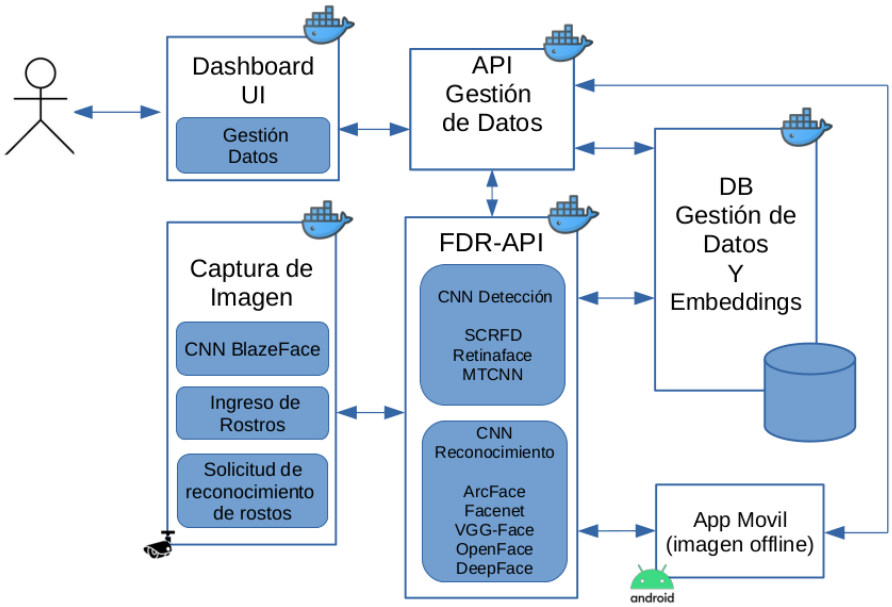


Figura 5.4: Arquitectura FR-ARCA.

A partir de los requerimientos especiales del sistema y las consideraciones para implementar y desplegar modelos de Machine Learning en un producto de software, presentamos la arquitectura tecnológica del sistema desarrollado, FR-ARCA (Facial Recognition for Automatic Registration of Class Attendance) (ver figura 5.4).

La arquitectura fue realizada en diferentes módulos, cada uno de ellos cumple una tarea específica, y se describe a continuación:

Dashboard UI: Interfaz gráfica de usuario, contiene dos componentes: Gestión de datos e Ingreso de rostros.

- Gestión de datos: Interfaz para gestión de usuarios, programas de formación, ambientes de aprendizaje, grupos, aprendices y asistencia a clases.
- Ingreso de rostros: Interfaz que realiza la captura de rostros de una persona, esta interfaz envía la imagen al módulo FDR-API para ingresar las características principales de ese rostro (Embedding) a la base de datos.

API Gestión de datos: Permite la interacción del cliente (Dashboard UI) con el servidor a

través de la transferencia de estado representacional (REST) y la base de datos de gestión de usuarios, con el fin de gestionar el flujo de información de los datos usuarios, programas de formación, ambientes de aprendizaje, grupos, aprendices y asistencia a clases.

DB Gestión de datos y Embedding: Contenedor encargado de almacenamiento de datos necesarios para la gestión de usuarios, programas de formación, ambientes de aprendizaje, grupos, aprendices y asistencia a clases. Igualmente se almacenan el vector con las características principales de los rostros (embeddings) usados para calcular la similitud en el proceso de reconocimiento.

API de detección y reconocimiento de rostros. (FDR-API): Módulo principal API-REST con modelos de detección y reconocimiento de rostros. Contiene los servicios para recibir imágenes, pre-procesar imágenes, detectar rostros, reconocer rostros, extraer características de rostros y almacenar en base de datos.

Captura de Imagen: Módulo para configuración y captura de imágenes a través de las cámaras que se instalan en los ambientes de formación. Dentro del módulo se implemente una CNN de detección liviana BlazeFace, con el fin de filtrar las imágenes que se envían a FDR-API para realizar el proceso de reconocimiento facial y registro de asistencia.

App Móvil: Aplicación móvil para consumir los servicios del sistema FDR-ARCA en casos donde los ambientes de formación estén ubicados en zonas rurales o donde no sea posible la instalación de cámaras.

La infraestructura tecnológica que soportará los productos del sistema está distribuida en contenedores docker para dar escalabilidad y de ser necesario distribuir los recursos en diferentes servidores, con el fin de lograr una alta disponibilidad cumpliendo los requerimientos. Todo lo anterior se diseñó teniendo en cuenta las consideraciones para implementar y desplegar modelos de Machine Learning en un producto de software con un enfoque coherente y basado en microservicios.

5.6. DISEÑO

5.6.1. Usuarios del Sistema

Se identifican los siguientes usuarios del sistema:

1. **Administrador:** Es el tipo de usuario que tiene la autorización de crear o desactivar cuentas de administración del sistema. Además, tiene todos los permisos del usuario coordinador de formación.

2. **Coordinador de formación:** está encargado de administrar la información contenida en el sistema, tiene los permisos de insertar, modificar, eliminar y consultar. Tiene acceso al panel de control (Dashboard UI) para gestionar toda la información. Puede realizar ingreso de rostros a la base de datos embeddings, solo puede crear roles de instructor.

3. **Instructor:** este tipo de usuario solo puede ser creado por el coordinador de formación, el cual asignará nombre de usuario y contraseña para el ingreso. Este tipo de usuario puede realizar consultas de grupos, programación de grupos, listas de aprendices y listas de asistencia.

5.6.2. Ingreso de rostros al sistema

Dentro del proceso de formación del SENA, el ingreso de rostros se debe realizar cuando ingrese un nuevo programa de formación a etapa lectiva, este proceso se realiza una sola vez por programa de formación al igual que el proceso de carnetización que lleva a cabo la entidad.

Para llevar a cabo el anterior proceso, se define el diagrama de la figura 5.5; El proceso inicia consultando la identificación del aprendiz. El sistema valida el token de la solicitud y si tiene los permisos de acceso recibirá la información correspondiente al aprendiz seleccionado, con el dispositivo de captura de imagen se realizan las fotos del aprendiz y se envían al módulo FDR-API para que las redes neuronales de detección y reconocimiento de rostros puedan extraer las características principales del rostro y almacenar el vector o los vectores resultantes en la base de datos de embeddings.

5.6.3. Reconocimiento de rostros

Dentro de este proceso se realizan dos tareas importantes del sistema, la detección del rostro del aprendiz a partir de las imágenes en el ambiente de formación y el reconocimiento de los aprendices a partir de los rostros detectados.

Para registrar la asistencia a clases, se realiza el siguiente proceso (ver figura 5.6): El módulo de captura de imagen se encarga de realizar la solicitud de acceso y datos de programación del grupo a la API Gestión de Datos. Después de recibir los datos anteriores, el módulo empieza a capturar las fotos del ambiente de formación y las envía al módulo API Detección y Reconocimiento (FRD-API), el envío contiene la foto y el token de acceso. FRD-API inicia detectando rostros en la imagen, si los encuentra pasa a pre-procesar cada rostro, extrae las características principales de cada rostro en un vector y compara siguiendo el proceso que se presenta en la figura 5.3.

Este proceso puede repetirse varias veces durante una misma clase, el sistema está configurado para enviar una foto que contenga rostros de manera periódica, con el fin de tomar varias imágenes y así incrementar la posibilidad de reconocer todos los aprendices asistentes.

Para realizar el mismo procedimiento en ambientes de aprendizaje rurales o que no tengan la posibilidad de instalar dispositivos de captura de imágenes, el sistema posee una aplicación móvil en la cual el instructor que está en esa clase puede iniciar sesión, ingresar los datos del grupo correspondiente, tomar las fotos y enviar al sistema, en caso de que en el lugar no tenga acceso a datos la aplicación hace persistencia de datos y sincroniza cuando tenga la conexión. Aunque el proceso de captura de imagen no es automático el proceso de registro de asistencia a través de reconocimiento facial si lo es. El flujo de datos es igual al presentado en la Figura 5.6 pero el primer actor es la aplicación móvil.

5.6.4. Gestión de datos

En el proceso de registro de asistencia a clases en el SENA se debe tener en cuenta otros aspectos involucrados como son: Gestión de usuarios, programas de formación, ambientes de aprendizaje, grupos, aprendices y finalmente la asistencia a clases. El módulo gestión de datos es el encargado de realizar estas tareas se conecta mediante el protocolo de comunicación REST a la interfaz web (ver figura 6.11).

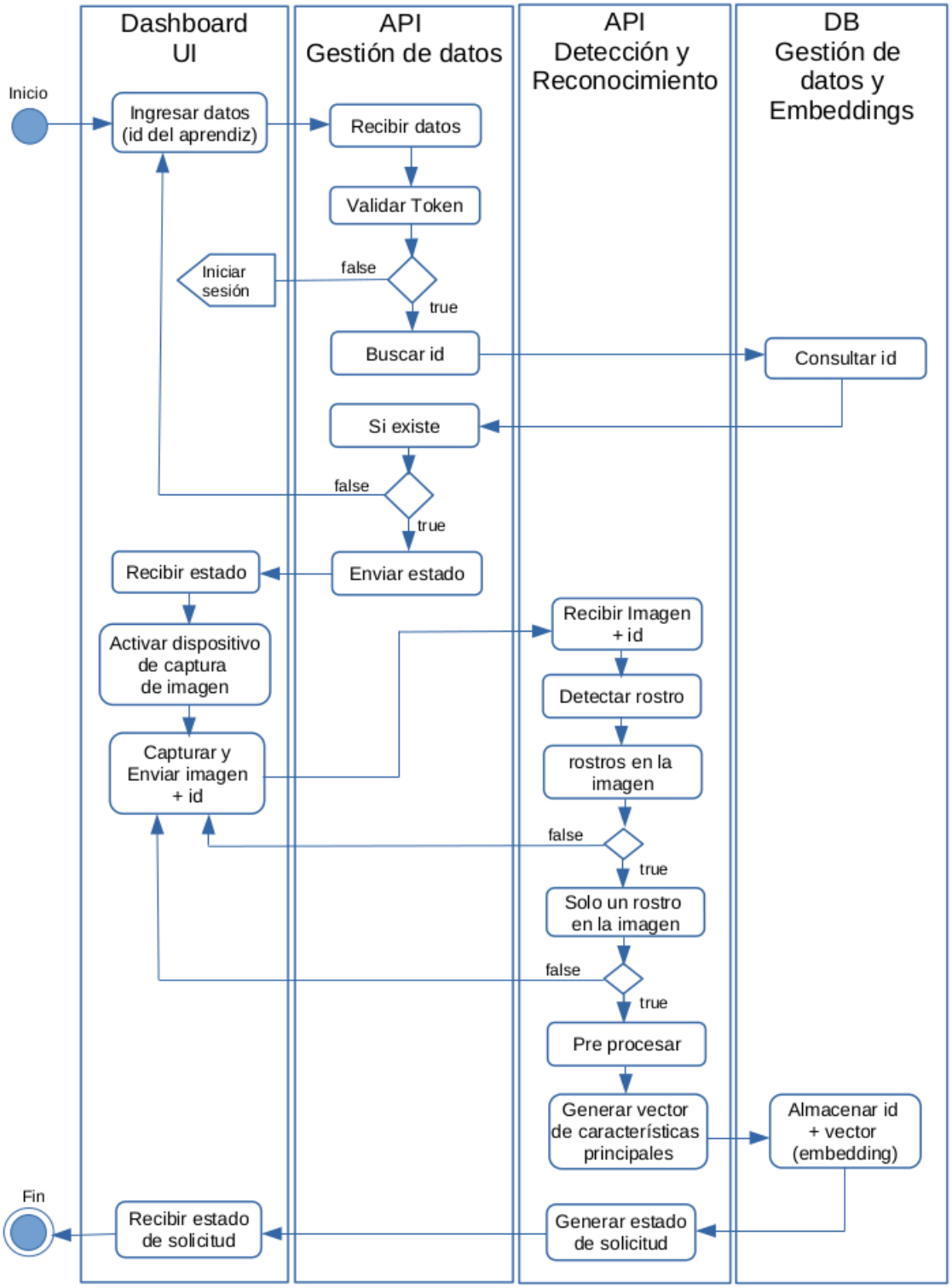


Figura 5.5: Diagrama de flujo ingreso de rostro del aprendiz al sistema.

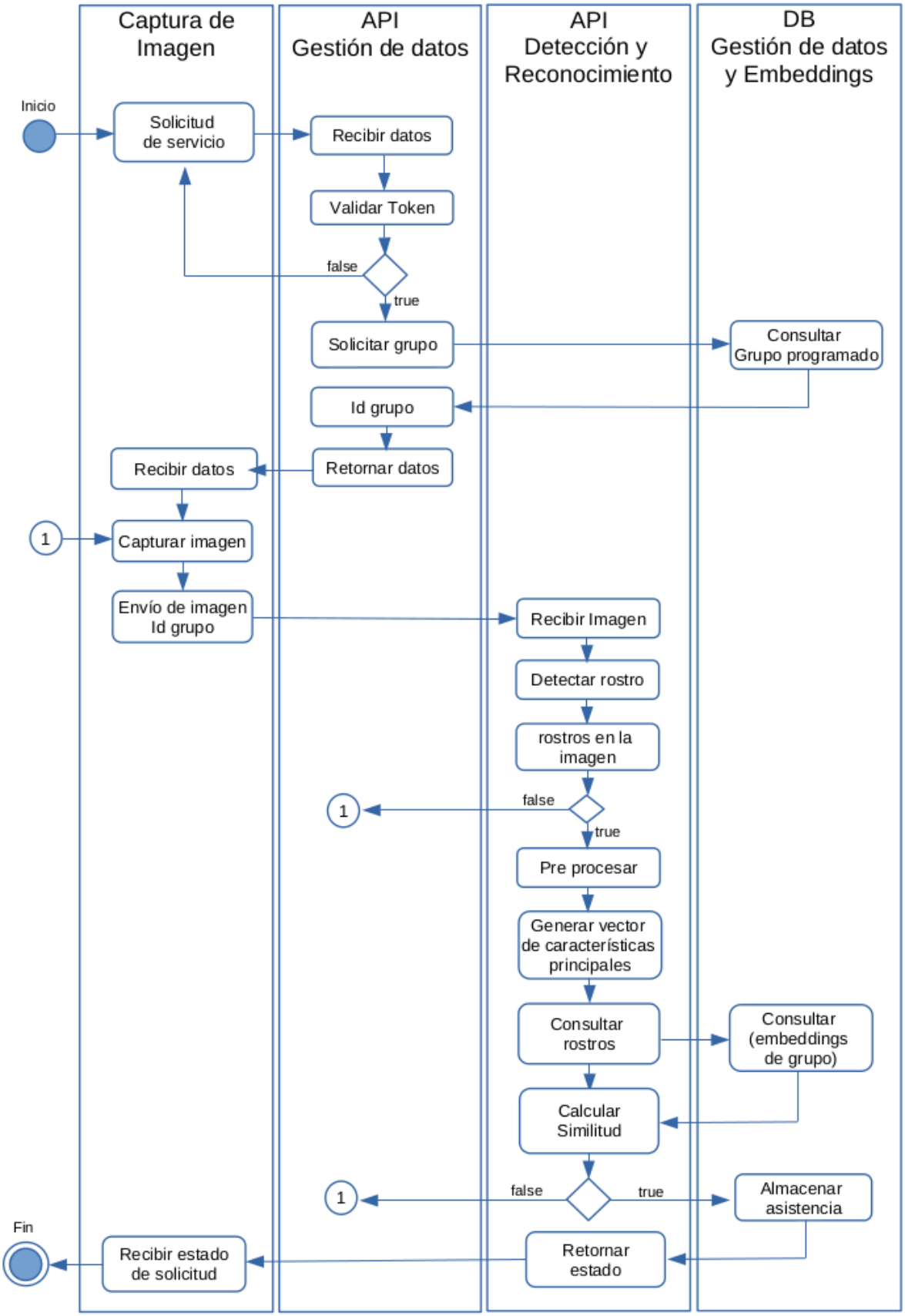


Figura 5.6: Diagrama de flujo reconocimiento de rostros de aprendices.

Para garantizar la seguridad de la gestión de datos se usa el método JSON Web Token (JWT) es un estándar abierto Request for Comments (RFC) 7519 basado en JSON propuesto por Internet Engineering Task Force (IETF)(*RFC 7519 - JSON Web Token (JWT)*, s.f.). La figura 5.7 muestra el proceso para obtener el token JWT de acceso y así usar los servicios de todas las APIs del proyecto.

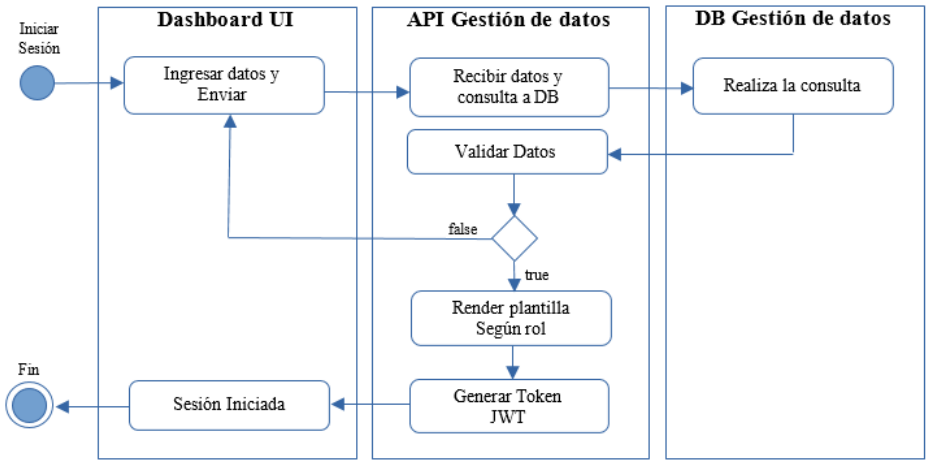


Figura 5.7: Diagrama flujo inicio sesión en Dashboard UI.

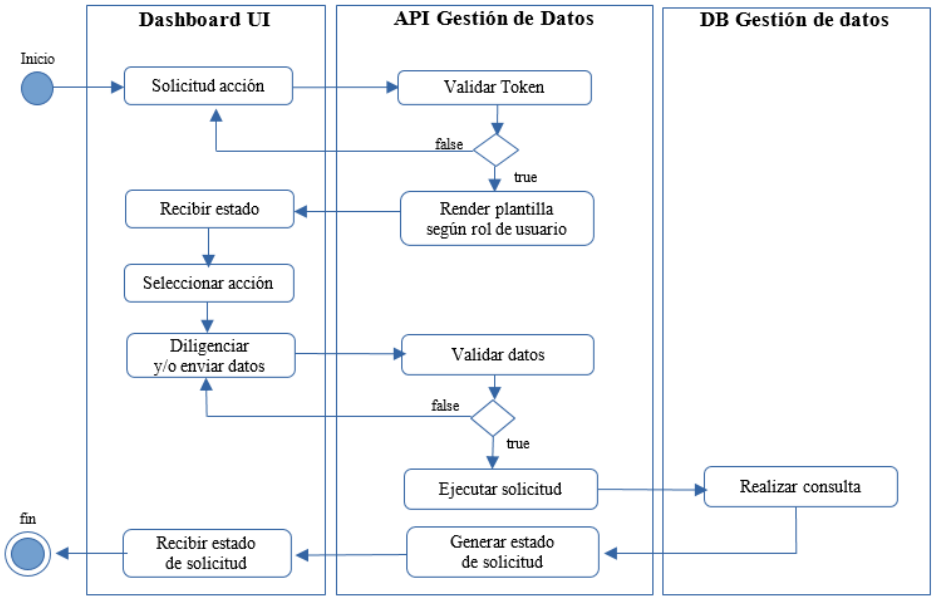


Figura 5.8: Diagrama flujo consumo servicios Dashboard UI.

Según el rol de usuario, en el Dashboard UI se puede ejecutar las acciones de ingresar,

modificar, eliminar y consultar los datos de usuarios, programas de formación, ambientes de aprendizaje, grupos, aprendices y finalmente la asistencia a clases, antes se debe iniciar sesión. En la figura 5.8 se presenta el flujo de información a nivel general teniendo en cuenta que el usuario puede realizar una de las acciones nombradas.

Capítulo 6

RESULTADOS

Esta sección presenta los resultados de los experimentos realizados para la evaluación al sistema FR-ARCA la cual se enfoca en medir la precisión para registrar asistencia a clases con diferentes métodos (ver tabla 4.1 Asociación de modelos de detección y reconocimiento).

Los experimentos 1, 2 y 3 se realizaron con el conjunto de datos propio del proyecto FR-ARCA Dataset (ver 4.4 Preparación de Datos), para estos experimentos el Sistema FR-ARCA fue instalado en un entorno local con las siguientes características:

- Procesador, Intel(R) Xeon(R) W-2145 CPU @ 3.70GHz
- Memoria, 32604MB (4960MB usados)
- Disco Duro SSD ATA KINGSTON SA400S3 448GB
- Sistema operativo, Ubuntu 20.04.4 LTS

Para el experimento 4 el sistema CompreFace se instaló en el entorno local anteriormente mencionado, para Amazon Rekognition y Azure Face se realizó el consumo a través de las APIs de cada servicio.

Los experimentos 5 y 6 se realizaron en entorno real con 3 grupos de aprendices del Centro Atención Sector Agropecuario Regional – Risaralda (ver tabla 4.2) por lo cual para estos experimentos el sistema FR-ARCA fue desplegado en instancias EC2 de Amazon Web Services (AWS), con las siguientes características:

- Imagen: Ubuntu Server 20.04 LTS (HVM) de 64 bits (x86).

- Tipo de instancia: t2.xlarge la cual cuenta con 4 v CPU Intel 2.300Mhz.
- 16 GiB de memoria y desempeño de bajo a moderado de la red.
- Tamaño de almacenamiento: 12 GiB de SSD de uso general (gp2).

Para obtener el *accuracy* de los experimentos 1 y 2 utilizamos la matriz de confusión multiclase. Adicionalmente se mide el tiempo dentro del servicio de detección y reconocimiento facial desde que la imagen enviada es procesada hasta obtener la máxima similitud, e involucra los siguientes procesos.

- La imagen enviada se convierte de archivo a imagen con tres canales RGB.
- Detectar rostros en la imagen y construir boundig box.
- Obtención del vector extracción de características principales del rostro o rostros de la imagen.
- Consultar los embeddings de base de datos según los criterios (por grupo o todos).
- Calcular la máxima similitud de los embeddings consultados en base de datos con el o los nuevos embeddings (match).

Para los resultados del experimento 3 primero, se configuraron los diferentes metodos de la Tabla 4.1 en FR-ARCA, para cada configuración se obtuvo el valor de similitud calculado para 840 pares de imagenes, 50 % de clase 1 y 50 % de clase 0, con el valor de similitud y la clase se graficaron las curvas ROC y el valor de AUC.

6.1. RESULTADOS DE LOS EXPERIMENTOS REALIZADOS

6.1.1. Resultados Experimento 1. Reconocimiento de rostros con búsqueda de máxima similitud segmentada por grupos, para el cálculo de *accuracy* de diferentes métodos de detección y reconocimiento de rostros.

En este experimento son notorias las diferencias entre algunos métodos probados. La Figura 6.1 muestra un *accuracy* por debajo de 0.67 en los métodos identificados 1, 2, 7, 8 los cuales pertenecen a los modelos de reconocimiento Deepface y OpenFace, aunque el

resultado con el detector MTCNN es mejor con respecto al detector Retinaface están a una diferencia aproximada de 0.30 puntos respecto a los demás métodos.

Id Método	Detection	Recognition	Accuracy	Tiempo total (s)	Promedio de tiempo por imagen (s)
1	Retinaface	Deepface	0,5893	165,15	0,98
2	RetinaFace	OpenFace	0,5714	150,44	0,90
3	RetinaFace	Facenet	0,9583	158,32	0,94
4	RetinaFace	VGG-Face	0,9940	171,52	1,02
5	RetinaFace	ArcFace Keras	0,9762	159,87	0,95
6	MTCNN	ArcFace Keras	0,9702	94,12	0,56
7	MTCNN	DeepFace	0,6667	103,01	0,61
8	MTCNN	OpenFace	0,6250	84,99	0,51
9	MTCNN	Facenet	0,9048	89,86	0,53
10	MTCNN	VGG-Face	0,9821	106,02	0,63
11	SCRFD-10GF	ArcFace - ResNet100	1,0000	89,01	0,53
12	SCRFD-10GF	ArcFace - ResNet50	1,0000	61,33	0,37
13	SCRFD-2.5GF	ArcFace - ResNet50	1,0000	52,89	0,31
14	SCRFD-500MF	MobileFaceNets	1,0000	22,52	0,13
15	SCRFD-500MF Align False	MobileFaceNets	1,0000	9,81	0,06

Tabla 6.1: Resultados *accuracy*, tiempo total en proceso de reconocimiento y promedio de tiempo por imagen, en búsqueda de similitud segmentada por grupos

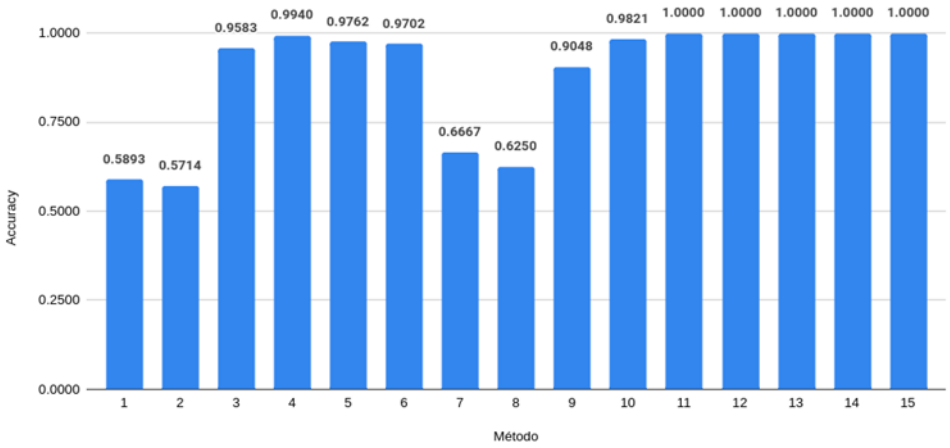


Figura 6.1: *Accuracy* de los métodos identificados en la Tabla 6.1. Búsqueda de similitud segmentada por grupos.

Las asociaciones del modelo de reconocimiento VGG-Face retornaron muy buenos resultados en el *accuracy*, 0.9940 con el detector RetinaFace y 0.9821 con el detector MTCNN. Cabe anotar que VGG-Face es un modelo que fue presentado por Visual Geometry Group at Oxford en el año 2015, es el modelo más antiguo que hemos usado en este proyecto. El modelo preentrenado usado fue reimplementado del modelo original MatConvNet a Keras (Serengil, 2019).

Para los métodos que tienen modelo de reconocimiento ArcFace reimplementado para Keras (Leondgarse, 2022), los resultados de *accuracy* con el detector RetinaFace fue de 0.9762 y con el detector MTCNN de 0.9702.

Los métodos 11, 12, 13, 14 y 15 que tiene el modelo de reconocimiento ArcFace en su implementación original ONNX (Deepinsight, 2022c), obtienen el mayor valor de *accuracy* 1.0 en este experimento.

6.1.2. Resultados Experimento 2. Reconocimiento de rostros con búsqueda de máxima similitud en el conjunto completo de embeddings, para el cálculo de *accuracy* de diferentes métodos de detección y reconocimiento de rostros.

En este experimento se realiza el mismo procedimiento del experimento 1, pero se busca la máxima similitud, comparando el vector de características de cada rostro con todos

los embeddings almacenados en la base de datos; este único cambio reduce el valor del *accuracy* resultante en las asociaciones identificadas del 1 al 9, debido a que entre mayor sea la cantidad de vectores a comparar, mayor será la probabilidad de que los vectores de características principales tengan una corta distancia entre ellos afectando el resultado del cálculo de similitud.

Id Método	Detection	Recognition	Accuracy	Tiempo total (s)	Promedio de tiempo por imagen (s)
1	Retinaface	Deepface	0,4286	178,85	1,06
2	RetinaFace	OpenFace	0,3631	153,46	0,91
3	RetinaFace	Facenet	0,8929	161,72	0,96
4	RetinaFace	VGG-Face	0,9464	192,32	1,14
5	RetinaFace	ArcFace Keras	0,9345	164,58	0,97
6	MTCNN	ArcFace Keras	0,8988	101,62	0,60
7	MTCNN	DeepFace	0,4405	117,90	0,70
8	MTCNN	OpenFace	0,3869	88,95	0,53
9	MTCNN	Facenet	0,8095	91,10	0,54
10	MTCNN	VGG-Face	0,9048	121,60	0,72
11	SCRFD-10GF	ArcFace - ResNet100	1,0000	89,66	0,53
12	SCRFD-10GF	ArcFace - ResNet50	1,0000	62,05	0,37
13	SCRFD-2.5GF	ArcFace - ResNet50	1,0000	54,94	0,33
14	SCRFD-500MF	MobileFaceNets	1,0000	25,69	0,15
15	SCRFD-500MF Align False	MobileFaceNets	1,0000	12,19	0,07

Tabla 6.2: Resultados *accuracy*, tiempo total en proceso de reconocimiento y promedio de tiempo por imagen, en búsqueda de similitud en el conjunto completo de embeddings.

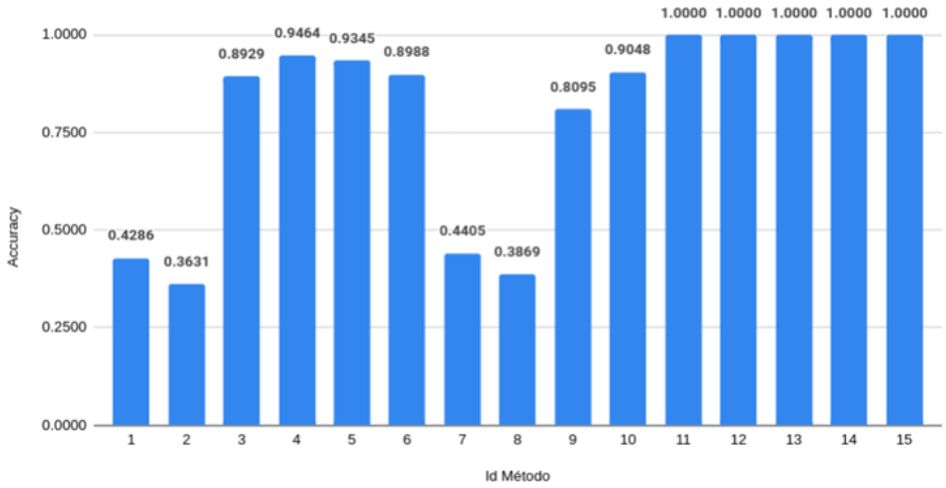


Figura 6.2: *Accuracy* de los métodos identificados en la Tabla 6.2. Búsqueda de similitud en el conjunto completo de embeddings.

Los métodos del 11 hasta el 15 siguen con el mayor valor para *accuracy* de 1.0. Estos métodos tienen la capacidad de generar mayores distancias entre vectores de características principales del rostro, aunque su precisión puede disminuir al calcular la similitud comparando gran cantidad de embeddings.

Las exactitudes que brindan los métodos que tienen como modelo de reconocimiento Deepface y OpenFace, disminuyen por debajo de 0.45, por esta razón los métodos identificados como 1, 2, 7 y 8 son descartados para los siguientes experimentos.

6.1.3. Resultados Experimento 3. Tasa de verificación de pares de imágenes.

En el dataset FR-ARCA existen imágenes de 42 aprendices donde cada aprendiz tiene 5 imágenes, por lo cual el número de combinaciones de parejas con la misma identidad para cada uno es de 10, entonces se tienen 420 combinaciones posibles etiquetadas en la clase 1. Generamos la misma cantidad de parejas con diferente identidad y las etiquetamos con la clase 0 para un total de 840 combinaciones. Estas combinaciones de pares de imágenes se prueban en el sistema FR-ARCA, del cual se obtiene el valor de similitud de cada par de imágenes. Se repite el experimento en los diferentes métodos y se obtienen los resultados que se muestran en la figura 6.3.

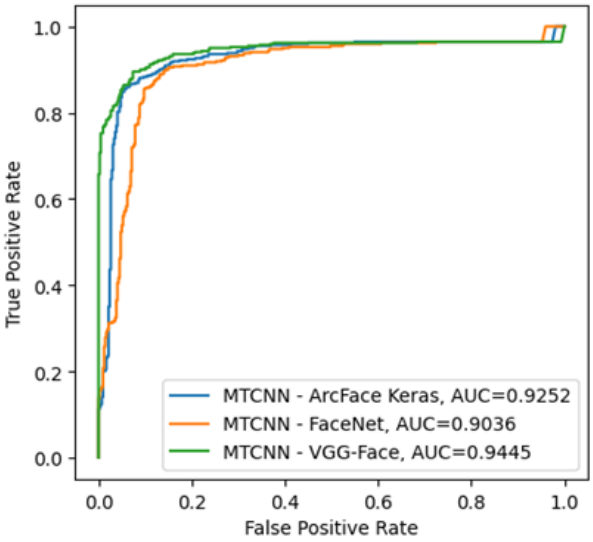


Figura 6.3: Curvas ROC y valor del AUC al combinar el detector MTCNN con los modelos de reconocimiento VGG-Face, Facenet y ArcFace Keras .

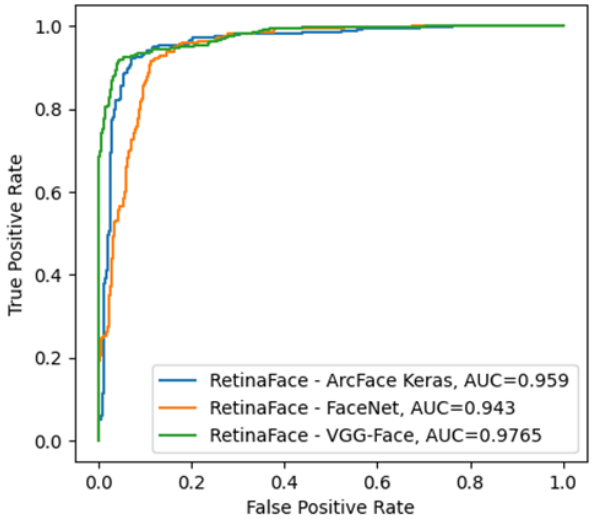


Figura 6.4: Curvas ROC y valor del AUC al combinar el detector RetinaFace con los modelos de reconocimiento VGG-Face, Facenet y ArcFace reimplementación en Keras.

Las Figuras 6.3 y 6.4 muestran que para la red neuronal de reconocimiento VGG-Face con el detector MTCNN y el detector RetinaFace, se obtiene el mejor valor de AUC 0.9445 y 0.9765 respectivamente. Los resultados son muy buenos, aunque sigue existiendo un porcentaje pequeño de posibilidad de ingresar al sistema datos erróneos, ya que el valor de AUC da información para distinguir qué tanto dos clases se superponen y con ello la probabilidad que el sistema pueda distinguir entre dos clases. Para el mejor resultado de AUC hay un 97,56 %

de probabilidad de que el modelo pueda distinguir entre clase positiva y clase negativa.

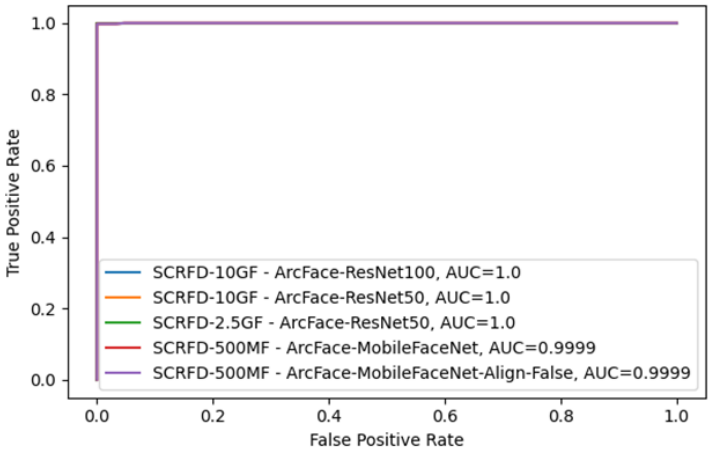


Figura 6.5: Curvas ROC y valor del AUC. Al combinar el detector SCRFD de diferentes GF con los modelos de reconocimiento ArcFace implementación oficial del proyecto insightface.

En los métodos identificados con 11, 12, 13, 14 y 15 de la Tabla 4.1 se obtiene un resultado perfecto en el valor de AUC, lo que quiere decir que el sistema FR-ARCA configurado con alguno de estos métodos estaría retornando predicciones de identidades con 100 % de probabilidad de acierto para el dataset de prueba. Se debe tener en cuenta que a medida que los datos de grupos se incrementen y se usen diferentes variables en las imágenes enviadas, estos resultados pueden cambiar.

En la figura 6.6 se muestran las curvas del histograma de distancias entre pares de imágenes con la misma identidad (clase 1) y pares de imágenes con distinta identidad (clase 0), construidas con los resultados de máxima similitud que retornaron los diferentes métodos probados en FR-ARCA con el set de datos FR-ARCA dataset.

En los primeros 6 gráficos de curvas de distribución de la figura 6.6 se observa que las dos clases se superponen, es decir, no hay umbral que pueda distinguir con una probabilidad del 100 % entre ellas. Pero en las figuras de los métodos que usan el detector SCRFD con los diferentes modelos de reconocimiento de ArcFace en su implementación original ONNX, se observa una buena separación de clases en la distribución de densidades en el eje de distancia, es decir, una mayor posibilidad de tener umbrales que permitan obtener 100 % de probabilidad de acierto al identificar rostros por cálculo de máxima similitud.

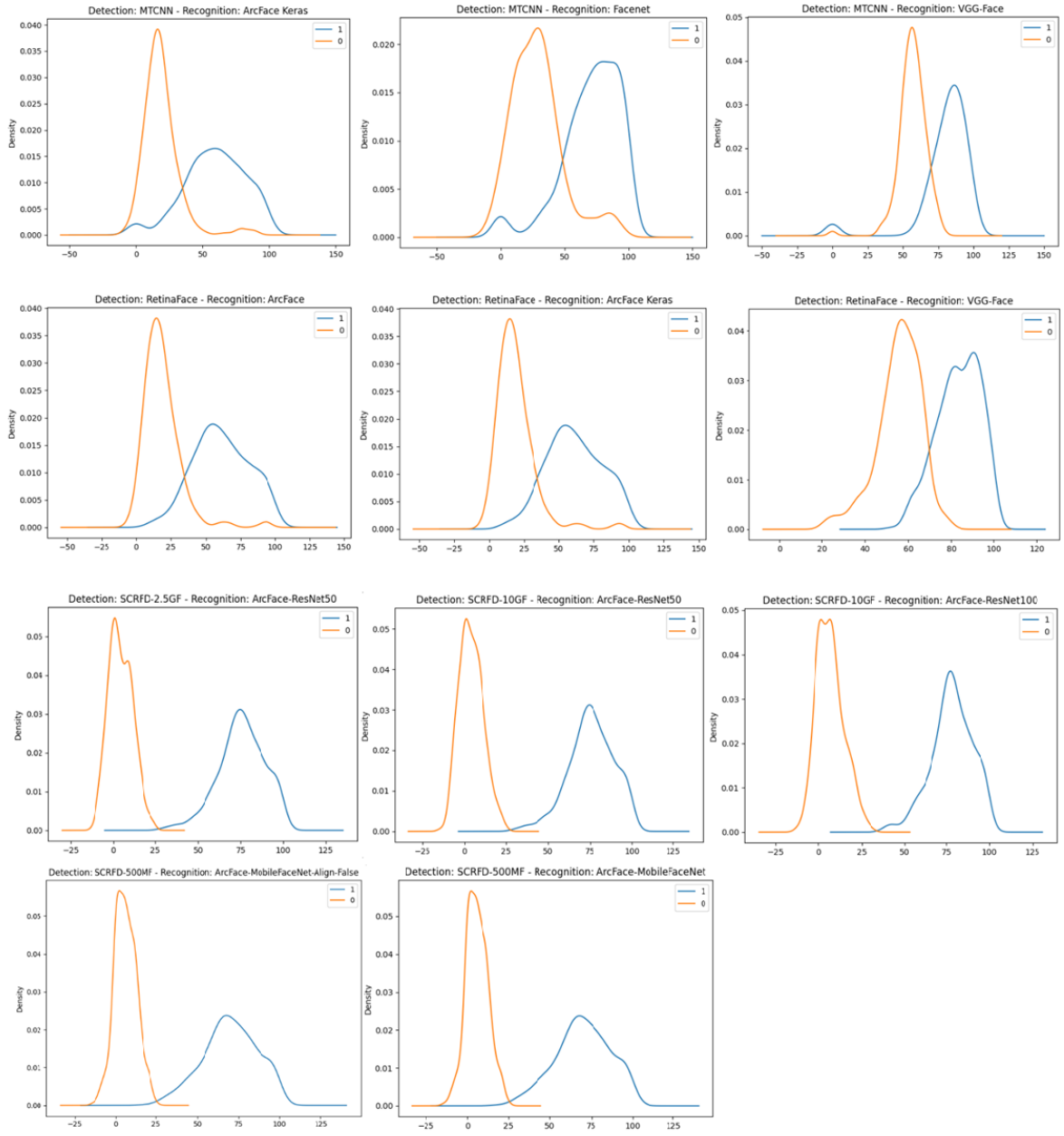


Figura 6.6: Curvas de distribución de clases. Clase 1 Pares de imágenes con la misma identidad, clase 0 pares de imágenes con distinta identidad.

Teniendo en cuenta los anteriores resultados, definimos el método con el detector SCRFD-2.5GF y el modelo de reconocimiento ArcFace con arquitectura CNN ResNet50 como el método por defecto en FR-ARCA con el cual se realizaron los experimentos 4, 5 y 6.

6.1.4. Resultados Experimento 4. Prueba de comparación de FR-ARCA, CompreFace, Azure Face Recognition y Amazon Rekognition con respecto a la precisión en el test de tasa de verificación de pares de imágenes.

Para este experimento se tienen 420 combinaciones de parejas de imágenes con la misma identidad etiquetadas con la clase 1. Se genera la misma cantidad de parejas con diferente identidad y las etiquetamos con la clase 0 para un total de 840 combinaciones. Con estas combinaciones de pares de imágenes se prueban los servicios de reconocimiento facial que proporcionan Amazon Rekognition (Amazon Inc., s.f.-c), Azure Face de Microsoft (Amazon Inc., s.f.-c), el software libre de reconocimiento facial CompreFace de Exadel (Exadel, s.f.) en su configuración por defecto y FR-ARCA configurado con el método con el detector SCRFD-2.5GF y el modelo de reconocimiento ArcFace con arquitectura CNN ResNet50.

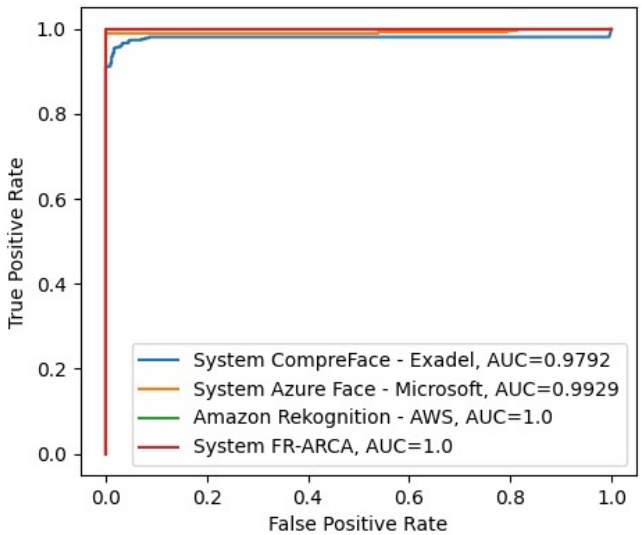


Figura 6.7: Curvas ROC y valor del AUC. Resultados de pruebas en los sistemas CompreFace, Azure Face Recognition, Amazon Rekognition y FR-ARCA.

FR-ARCAR y Amazon Rekognition estarían retornando predicciones de identidades con 100 % de probabilidad de acierto para el dataset de prueba.

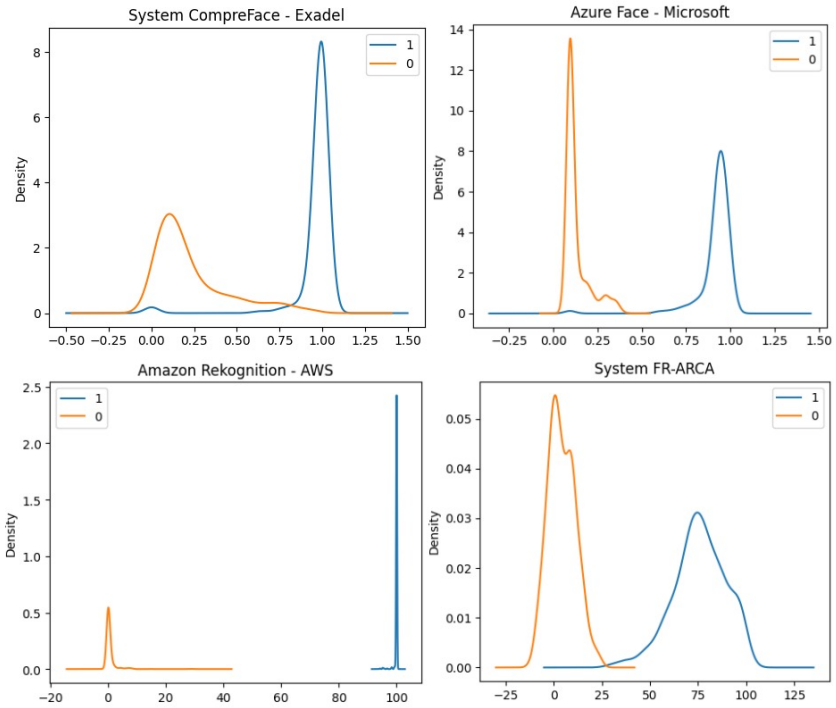


Figura 6.8: Curvas de distribución de clases. Clase 1 Pares de imágenes con la misma identidad, clase 0 pares de imágenes con distinta identidad. Sistemas CompreFace, Azure Face Recognition, Amazon Rekognition y FR-ARCA.

En la Figura 6.8 se muestran las curvas del histograma de distancias entre pares de imágenes con la misma identidad (clase 1) y pares de imágenes con distinta identidad (clase 0). Como se puede observar Amazon Rekognition tiene la mejor distribución de clases, aunque los demás sistemas también logran una separación en las cuales se puede lograr seleccionar umbrales que disminuyan al máximo la posibilidad de faltos positivos o verdaderos negativos.

Cabe notar que FR-ARCA está a la altura de software libre reconocidos como CompreFace de Exadel y de servicios cloud de grandes empresas como Amazon y Microsoft.

6.1.5. Resultados Experimento 5. Prueba de registro de asistencia a clases con FR-ARCA en ambientes de aprendizaje internos.

Para este experimento se procede a configurar el sistema con el detector SCRFD-2.5GF y como red de reconocimiento ArcFace–ResNet50, con esta combinación se obtuvo un *accuracy* 1, tiempo promedio de imagen de 0,31 segundos y la mejor separación de clases en la distribución de densidades en el eje de distancia.

En las tablas 6.3, 6.4 y 6.5 se presentan los resultados obtenidos de registrar la asistencia a clases de tres grupos distintos durante 10 días, ver tabla 4.2. La letra M es la columna que representa el procedimiento tradicional, llamado a lista y la letra S el registro de asistencia con el sistema FR-ARCA. La clase 1 representa la asistencia registrada y la clase 0 es cuando no se registró asistencia.

	Día 1		Día 2		Día 3		Día 4		Día 5		Día 6		Día 7		Día 8		Día 9		Día 10	
Aprendiz	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S
1004995253	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
1088353689	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1010137902	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1
1004778708	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1088033722	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1087998992	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1004778076	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1004669161	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1004628338	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
1002854262	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1055000428	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1	1
1192743075	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1004794595	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
1004739575	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1001772962	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1005092069	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0
1004529543	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1193247052	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1092916055	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
1004794819	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	1	1
1004767554	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1
1004683334	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabla 6.3: Resultados de prueba de registro de asistencia a clases con FR-ARCA del grupo 2055005 en ambientes de aprendizaje internos.

	Día 1		Día 2		Día 3		Día 4		Día 5		Día 6		Día 7		Día 8		Día 9		Día 10	
Aprendiz	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S
1114401958	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1004683387	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1002961706	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1004685039	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1004738353	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1006029135	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1
1118237121	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1004737494	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1002944265	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1007605570	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1
1004719255	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1111818489	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
1193113161	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabla 6.4: Resultados de prueba de registro de asistencia a clases con FR-ARCA del grupo 2055000 en ambientes de aprendizaje internos.

	Día 1		Día 2		Día 3		Día 4		Día 5		Día 6		Día 7		Día 8		Día 9		Día 10	
Aprendiz	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S
1004574849	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1088354524	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1110530710	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
1088348320	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1	1	1	1	1	1
1088825027	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
1086279492	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1004752357	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
1088264027	1	1	0	0	1	1	1	1	1	1	0	0	1	1	0	0	0	0	1	1
1088321073	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
1004701137	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1	1
1004755858	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1193569087	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
1005090087	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	1	1
1004736719	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1088344208	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1093222947	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
1004736363	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0

Tabla 6.5: Resultados de prueba de registro de asistencia a clases con FR-ARCA del grupo 2204685 en ambientes de aprendizaje internos.

La tabla 6.6 muestra el resumen de los resultados del experimento 5, la columna total de registros indica la cantidad de asistencias registradas. Se observa en la columna aciertos la cantidad de veces que el sistema FR-ARCA registró correctamente la asistencia y en la columna fallos las veces que el sistema registró mal la asistencia. Con estos datos se puede

afirmar que el sistema FR-ARCA en este experimento, registró asistencia a clases con un porcentaje de acierto del 99.81 % y solo un 0.19 % de error.

Grupo	No. Aprendices	Total Registros	Aciertos	Fallos
2055005	22	220	219	1
2055000	13	130	130	0
2204685	17	170	170	0
Totales		520	519	1

Tabla 6.6: Resumen de resultados de prueba de registro de asistencia a clases con FR-ARCA en ambientes de aprendizaje internos.

6.1.6. Resultados Experimento 6. Prueba de registro de asistencia a clases con FR-ARCA en ambientes de aprendizaje externos, usando la aplicación móvil.

Para este experimento se procede a configurar el sistema con el detector SCRFD-2.5GF y como red de reconocimiento ArcFace-ResNet50, con esta asociación se obtuvo *accuracy* de 1.0, tiempo promedio de imagen de 0,31 segundos y muy buena separación de clases.

En las tablas 6.7, 6.8 y 6.9 se encuentran los resultados del registro de asistencia a clases con el procedimiento manual (llamando a lista) y el sistema FR-ARCA usando la aplicación móvil como dispositivo de captura y persistencia de datos. El experimento se realiza durante 3 días para cada grupo.

	Día 1		Día 2		Día 3	
Aprendiz	M	S	M	S	M	S
1004995253	1	1	1	1	1	1
1088353689	1	1	1	1	1	1
1010137902	1	1	1	1	1	1
1004778708	1	1	1	1	1	1
1088033722	1	1	1	1	1	1
1087998992	1	1	0	0	0	0
1004778076	1	1	1	1	1	1
1004669161	1	1	1	1	1	1
1004628338	1	1	1	1	1	1
1002854262	1	1	1	1	1	1
1055000428	1	1	1	1	1	1
1192743075	1	1	1	1	1	1
1004794595	1	1	1	1	1	1
1004739575	1	1	1	1	1	1
1001772962	1	1	1	1	1	1
1005092069	1	1	1	1	1	1
1004529543	1	1	0	0	1	1
1193247052	1	1	1	1	1	1
1092916055	1	1	1	1	1	1
1004794819	1	1	0	0	0	0
1004767554	1	1	1	1	0	0
1004683334	1	1	1	1	1	1

Tabla 6.7: Resultados de prueba de registro de asistencia a clases con FR-ARCA del grupo 2055005 en ambientes de aprendizaje externos.

	Día 1		Día 2		Día 3	
Aprendiz	M	S	M	S	M	S
1114401958	1	1	1	1	1	1
1004683387	1	1	1	1	0	0
1002961706	1	1	1	1	1	1
1004685039	1	1	1	1	1	1
1004738353	1	1	1	1	1	1
1006029135	1	1	1	1	1	1
1118237121	1	1	1	1	1	1
1004737494	1	1	1	1	1	1
1002944265	1	1	1	1	1	1
1007605570	1	1	0	0	1	1
1004719255	1	1	0	0	1	1
1111818489	1	1	1	1	1	1
1193113161	1	1	1	1	1	1

Tabla 6.8: Resultados de prueba de registro de asistencia a clases con FR-ARCA del grupo 2055000 en ambientes de aprendizaje externos.

	Día 1		Día 2		Día 3	
Aprendiz	M	S	M	S	M	S
1004574849	1	1	1	1	1	1
1088354524	0	0	1	1	1	1
1110530710	1	1	1	1	1	1
1088348320	1	1	1	0	1	1
1088825027	1	1	1	1	1	1
1086279492	1	1	1	1	0	0
1004752357	1	1	1	1	1	1
1088264027	1	1	0	0	1	1
1088321073	1	1	1	1	1	1
1004701137	0	0	0	0	1	1
1004755858	1	1	1	1	1	1
1193569087	1	1	1	1	1	1
1005090087	1	1	1	1	1	1
1004736719	1	1	1	1	1	1
1088344208	1	1	1	1	1	1
1093222947	1	1	1	1	0	0
1004736363	1	1	1	1	1	1

Tabla 6.9: Resultados de prueba de registro de asistencia a clases con FR-ARCA del grupo 2204685 en ambientes de aprendizaje externos.

La tabla 6.10 muestra el resumen de los resultados del experimento 6. La columna total de registros indica la cantidad de asistencias registradas. Se observa en la columna aciertos la cantidad de veces que el sistema registró correctamente la asistencia y en la columna rotulada como fallos, las veces que el sistema registró mal la asistencia. Con estos datos se puede afirmar que el sistema FR-ARCA en este experimento registró asistencia a clases con un porcentaje de acierto del 99.36 % y solo un 0.16 % de error.

Grupo	No. Aprendices	Total Registros	Aciertos	Fallos
2055005	22	66	66	0
2055000	13	39	39	0
2204685	17	51	50	1
Totales		156	155	1

Tabla 6.10: Resumen de resultados de prueba de registro de asistencia a clases con FR-ARCA en ambientes de aprendizaje externos.

6.2. PRODUCTO FINAL FR-ARCA

Como producto final se tiene un sistema de registro automático de asistencia a clases presenciales basado en técnicas de detección y reconocimiento facial, utilizando modelos de aprendizaje profundo para el Centro Atención Sector Agropecuario SENA – Risaralda, al cual llamamos FR-ARCA por sus siglas en inglés Facial Recognition for Automatic Registration of Class Attendance.

Para lograr los objetivos propuestos fue necesario el desarrollo y la integración de 6 módulos que conforman el sistema FR-ARCA:

- **Módulo Dashboard UI**, como aplicación web.
- **Módulo API Gestión de Datos**, como una API REST en contenedor docker.
- **Módulo DB de Gestión de Datos y Embeddings**, como base de datos postgres en contenedor docker.
- **Módulo FDR-API**, como API REST con modelos de detección y reconocimiento de rostros, en contenedor docker.
- **Módulo Captura de Imagen**, como aplicación web.
- **Módulo Aplicación Móvil Android**, con el fin de usar el dispositivo móvil para captura de imágenes off line.

El sistema cuenta con varias interfaces gráficas que permiten una fácil interacción del usuario con el sistema:

6.2.1. Interfaz gráfica para gestión de datos.

En el módulo **Dashboard UI** se encuentra la Interfaz gráfica de usuario que permite la gestión de datos en sincronía con el módulo **API Gestión de Datos**. La figura 6.9 muestra el formulario de inicio de sesión, ya que este módulo está protegido a través de JSON Web Token (JWT), se debe iniciar sesión para obtener el token JWT de acceso y así usar los servicios las APIs del proyecto. Las figuras 6.10 y 6.11 muestran la interfaz del dashboard que permite realizar las tareas de la programación de la formación y consultar los registros de asistencia a

clases realizados con el sistema FR-ARCA.

Figura 6.9: Interfaz gráfica para inicio de sesión

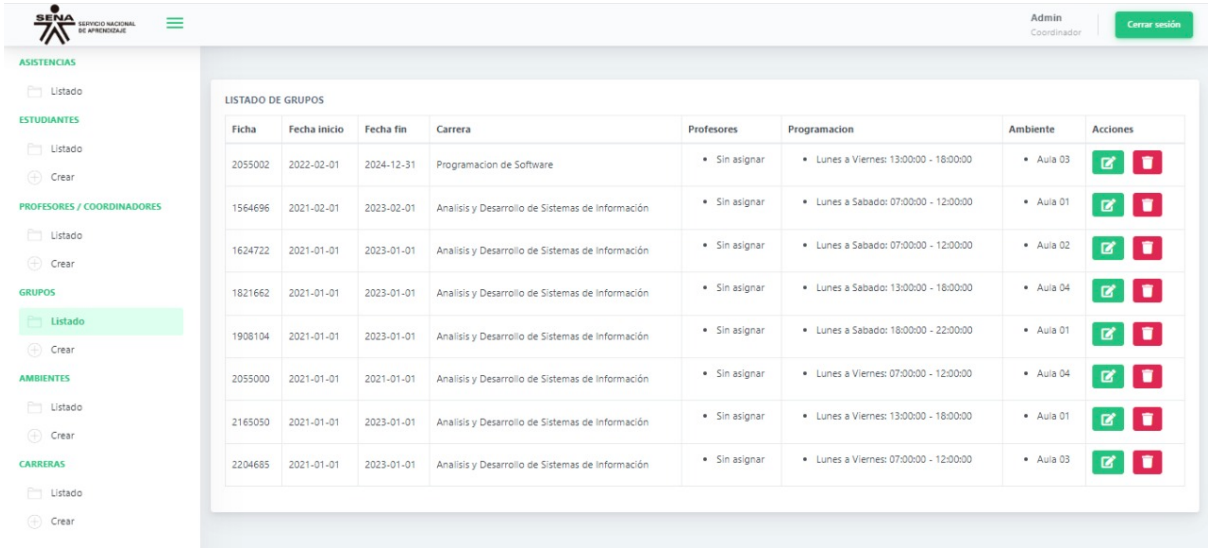


Figura 6.10: Dashboard, sección programación de grupos

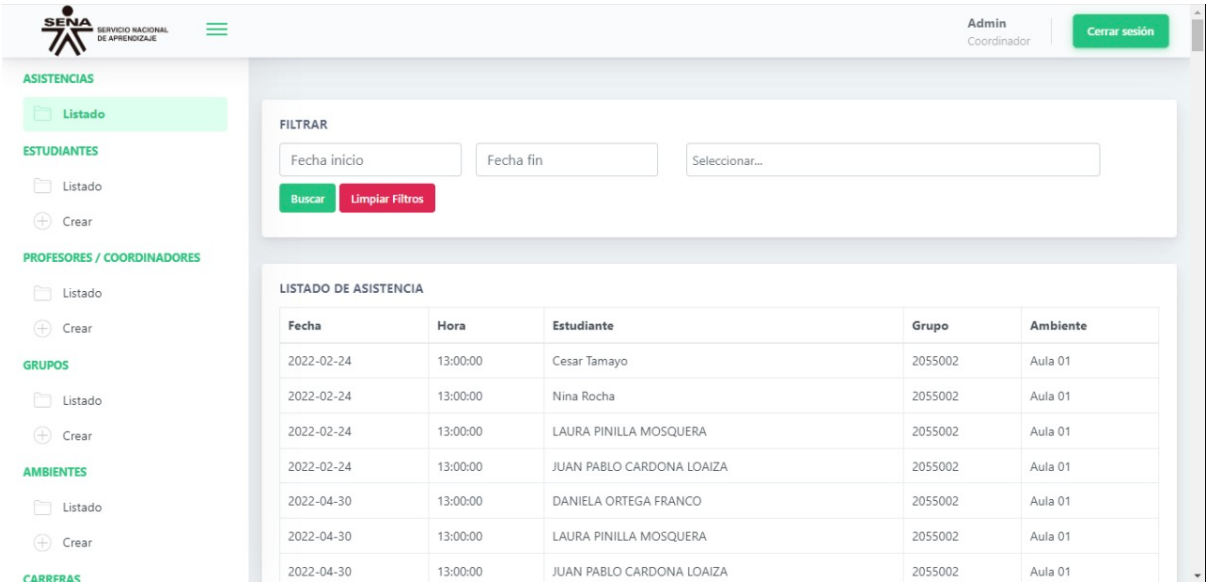


Figura 6.11: Dashboard, sección lista de asistencia a clases

6.2.2. Interfaz gráfica para ingreso de características principales de rostros.

La figura 6.12 muestra la interfaz que facilita al usuario permitido realizar el ingreso de rostros al sistema y por cada aprendiz que ingresa a la institución, se realiza una sola vez. En la interfaz, primero se debe seleccionar el grupo de formación y el aprendiz, luego capturar una o varias imágenes del rostro de cada aprendiz y al hacer clic en registrar, el sistema FR-ARCA almacena el vector de características principales de cada rostro etiquetado. La interfaz también informa cuantos embeddings de rostros ya se tienen almacenados para el aprendiz seleccionado.

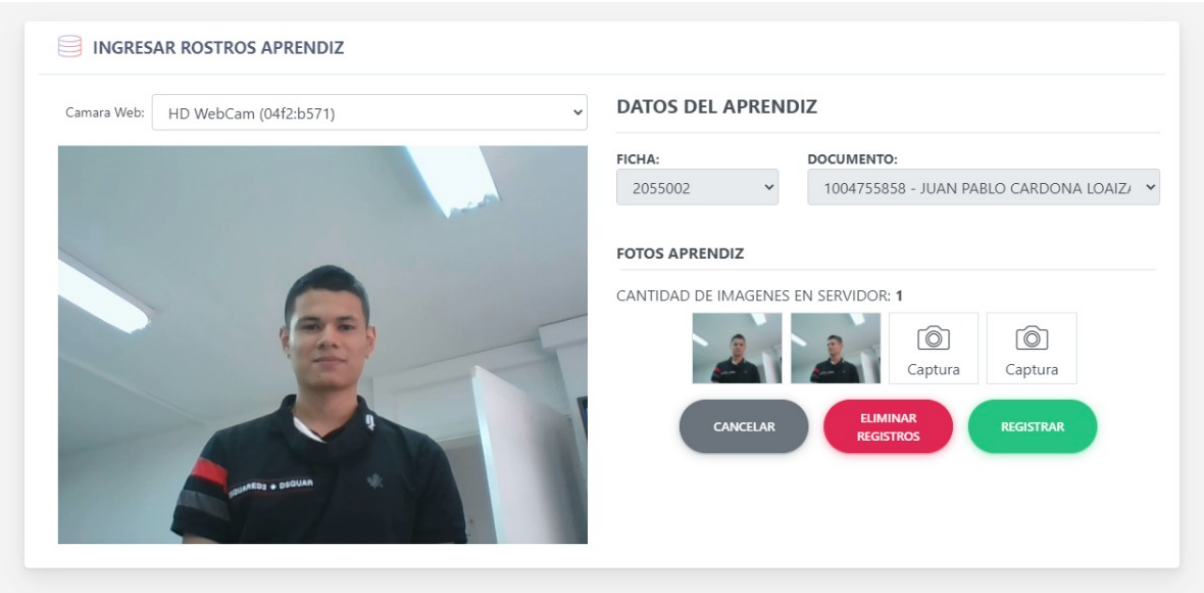


Figura 6.12: Interfaz gráfica para ingreso de rostros al sistema FR-ARCA

6.2.3. Interfaz gráfica del dispositivo de captura de imágenes.

El dispositivo de captura de imágenes puede ser cualquier dispositivo que permita ingresar a un sitio web a través de un navegador y tenga una cámara. Primero hay que activar el dispositivo para que inicie el proceso de captura de rostros, la figura 6.13 muestra el inicio de la interfaz en espera de ser activada desde la aplicación móvil del instructor que inicia su clase. Una vez activado el dispositivo inicia el proceso de captura de rostros para realizar el registro automático de asistencia a clases como se muestra en la figura 6.14. Este módulo trabaja con el detector de rostros BlazeFace ya que solo envía imágenes cuando detecta un rostro, el cual es marcado con un cuadro de color azul trasparente. En esta interfaz también se puede observar el listado de aprendices que ya fueron detectados.

Figura 6.13: Interfaz gráfica del dispositivo de captura de rostros, en espera de activación

Figura 6.14: Interfaz gráfica del dispositivo de captura de rostros, capturando rostros para registro de asistencia a clases.

6.2.4. Interfaz gráfica de la aplicación móvil.

La aplicación móvil para dispositivos Android es una alternativa para dar cobertura en los ambientes de formación externos, donde no se cuenta con conectividad a internet. La figura 6.15 muestra la interfaz de inicio de sesión; una vez se inicie sesión en la parte inferior se encuentra el menú de funcionalidades de la aplicación. En esta figura se encuentra selec-

cionada la sección de ambientes, esta funcionalidad permite activar el inicio de captura de rostros en los dispositivos de captura de imágenes instalados en los ambientes de formación internos ver figuras 6.13 y 6.14.

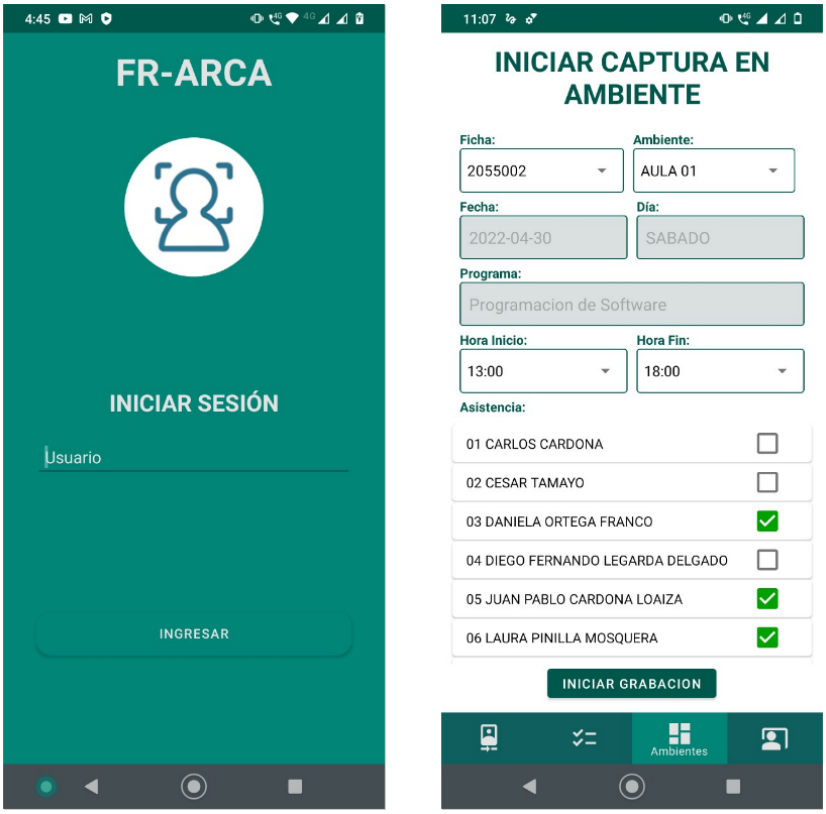


Figura 6.15: Interfaz gráfica de inicio de sesión y activación del dispositivo de captura de rostros.

Los dispositivos móviles permiten hacer persistencia de datos y sincronizar imágenes con el módulo API de detección y reconocimiento de rostros cuando el dispositivo tenga conexión a internet. La figura 6.16 muestra la interfaz que permite realizar la captura de imágenes de manera off line, una vez tenga el dispositivo recupera la conexión a Internet, se envían las imágenes y retorna el listado de asistencia para los rostros reconocidos. En el menú de la interfaz gráfica también se pueden seleccionar el histórico de los reportes de asistencia a clase.

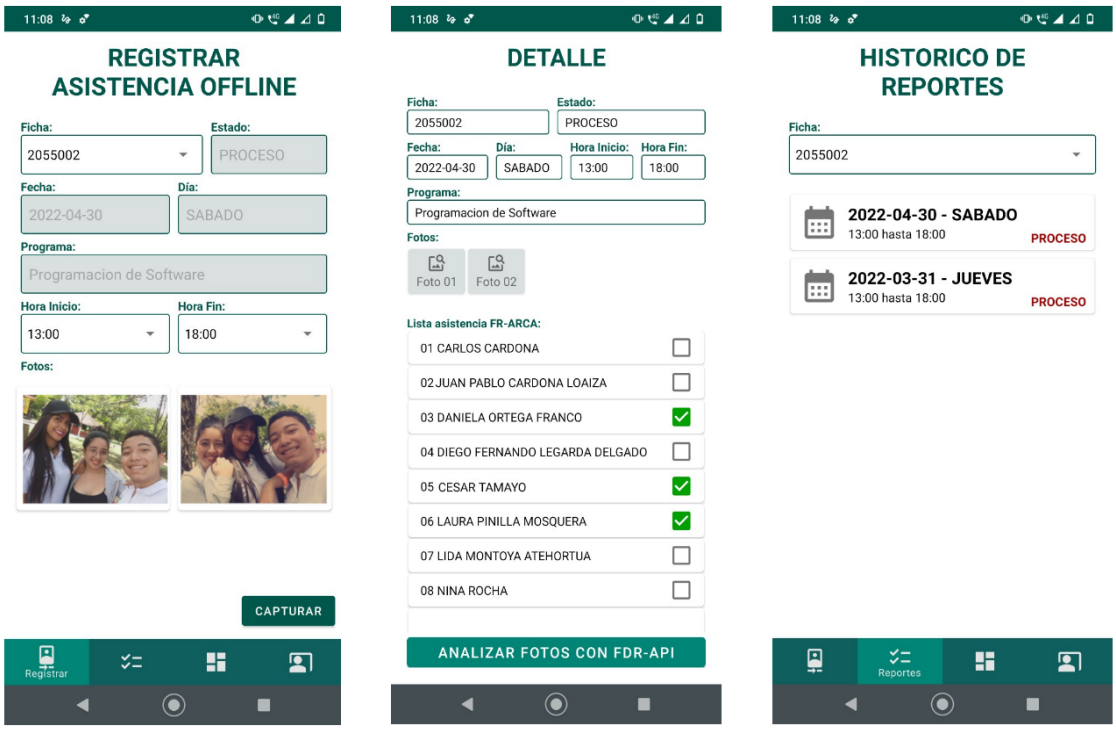


Figura 6.16: Interfaz gráfica del registro de asistencia off line y reportes de asistencia a clase .

Capítulo 7

CONCLUSIONES Y TRABAJOS FUTUROS

CONCLUSIONES

Se desarrolló un sistema piloto de registro automático de asistencia a clases presenciales basado en técnicas de detección y reconocimiento facial, utilizando modelos de aprendizaje profundo para el Centro Atención Sector Agropecuario SENA – Risaralda, al cual llamamos FR-ARCA por sus siglas en inglés *Facial Recognition for Automatic Registration of Class Attendance*.

Para dar solución a todas las consideraciones planteadas en la tabla 5.1 fue necesario la integración de múltiples herramientas de software, por lo cual se desarrollaron 6 módulos: El módulo **Dashboard UI**, como aplicación web que permite al usuario gestionar la información del proceso de programación de la formación y el registro de asistencia a clases, este módulo consume los servicios del módulo **API gestión de datos** quien a su vez consulta los datos con el SGBD del módulo **DB de gestión de datos y embeddings**. El módulo de **captura de imagen** es una aplicación web que usa la cámara del dispositivo cuando este sea activado, este módulo envía imágenes al módulo de **detección y reconocimiento de rostros (FDR-API)**, con el fin de registrar nuevos rostros en base de datos de embeddings o realizar la identificación de rostros que permiten registra la asistencia a clases. Finalmente, el módulo de **Aplicación Móvil** Android, que permite usar el dispositivo móvil para captura de imágenes offline y posteriormente ser enviadas para registro de asistencia a clases.

En el módulo de detección y reconocimiento de rostros FDR-API, se implementaron varios modelos de aprendizaje profundo para la detección y reconocimiento de rostros, que permiten realizar la identificación del aprendiz a través del cálculo de similitud entre vectores

de características principales.

Se evaluó el desempeño del sistema FR-ARCA utilizando casos de pruebas simulados y entornos reales, los resultados nos permitieron configurar el sistema para obtener la mejor exactitud en el reconocimiento de identidad, disminuyendo los casos de falsos positivos y verdaderos negativos derivados de reconocimientos erróneos. Se utilizaron los indicadores más comunes como la matriz de confusión, las curva ROC y AUC.

La implementación de los modelos de deep learning en entornos reales de producción y fuera de ambientes controlados, fueron un reto enorme a nivel de ingeniería, en gran medida por las dependencias de las diferentes librerías, arquitecturas y modelos de las redes neuronales convolucionales. El uso y la integración de contenedores, el lenguaje de programación Python con los frameworks FastAPI y Django, el uso de frameworks para Machine Learning como Keras, TensorFlow, PyTorch, OpenCV y MXNet, los protocolos de comunicación REST y MQTT, además de las estrategias algorítmicas y de almacenamiento de datos implementadas en esta investigación, permitieron obtener un excelente desempeño del sistema FR-ARCA.

Usando la estrategia de identificación de identidades a través del cálculo de similitud entre los vectores de características principales (embeddings), no se necesita que las redes neuronales convolucionales sean reentrenadas cuando ingresan nuevos grupos de aprendices a la institución, solo es necesario almacenar el vector de características principales de estas nuevas personas para posteriormente identificarlas.

El umbral de discriminación entre identidades iguales y distintas es dependiente de los métodos de detección y reconocimiento que se deseen utilizar, lo que hace necesario realizar experimentos y selección de métricas que sea coherentes con el objetivo que se persiga. Teniendo en cuenta los resultados, se define para la configuración por defecto en el sistema FR-ARCA el método con el detector SCRFD-2.5GF y el modelo de reconocimiento ArcFace con arquitectura CNN ResNet50. Para este método, los umbrales que mejor pueden clasificar identidades van desde el umbral de similitud de 0.25 al 0.30 y para disminuir la posibilidad de generar verdaderos negativos se recomienda incrementar el umbral de similitud.

Los resultados presentados en esta investigación demuestran que el desempeño del sistema FR-ARCA desarrollado, puede estar aproximado al nivel de desempeño de software libre reconocidos como CompreFace de Exadel y de servicios cloud de grandes empresas

como Amazon y Microsoft.

TRABAJOS FUTUROS

Algunas vías interesantes para el trabajo futuro implican enfoques para aumentar las prestaciones del sistema FR-ARCA, en temas relacionados con la asistencia a clases presenciales. A continuación, se proponen algunos trabajos futuros sobre los cuales se puede orientar la investigación abordada en este proyecto:

- Adaptar el sistema FR-ARCA para medir la permanencia de los aprendices en las clases presenciales, repitiendo el actual proceso de registro de asistencia durante las sesiones activas de las clases presenciales y registrar el momento de ausencia de los aprendices. Estudiar cómo las variables identificadas dentro del proceso de permanencia afectan el desempeño de los aprendices.

- Medir individualmente el nivel de atención y motivación de los aprendices en las clases presenciales, de manera que sea posible identificar los factores que intervienen en estos dos aspectos y correlacionar estos factores con la asistencia y permanencia en clases.

- Estudiar el comportamiento del sistema frente a un mayor volumen de estudiantes, este nuevo conjunto de datos debería tener una mayor variabilidad en pose, iluminación, oclusión y rangos de edad. Identificar las variables que inciden en el proceso de reconocimiento, para un correcto registro de asistencia a clases presenciales frente este aumento de la variabilidad en los datos.

Capítulo 8

PUBLICACIONES

Durante el desarrollo de este proceso de investigación se realizaron diferentes productos de software que conforman el sistema FR-ARCA, donde se destaca el código fuente de la aplicación final, aplicaciones intermedias como la construida para la realización del conjunto de datos propios, scripts para la ejecución de cada uno de los experimentos y la documentación de instalación del software. Actualmente se han realizado las siguientes tareas de publicación:

- Documentación del proceso de desarrollo del sistema, en el artículo titulado: *"Development and Validation of the Automatic Registration for Classroom Attendance System, Based on Facial Detection and Recognition Techniques, Using Deep Learning Models"*. Este artículo contiene la descripción del desarrollo del sistema FR-ARCA para el registro automático de asistencia a clases presenciales, basado en técnicas de detección y reconocimiento facial, utilizando modelos de aprendizaje profundo para el Centro Atención Sector Agropecuario de Risaralda. Se espera que sea publicado en revista indexada.
- Solicitud de inscripción del soporte lógico, ante la oficina de registro de software del ministerio del interior y de justicia dirección nacional de derecho de autor (UAE), para la obra titulada: *"FR-ARCA (Facial Recognition for Automatic Registration of Class Attendance)"*.

Referencias

Adith, N. T. (2021, septiembre). *Single shot detector (SSD) + architecture of SSD.* https://iq.opengenus.org/single-shot-detector/.

Agilemanifesto.org. (2021). *Manifiesto por el desarrollo ágil de software.* https://agilemanifesto.org/.

Ahlgren, P., y Grönqvist, L. (2006). Retrieval evaluation with incomplete relevance data: A comparative study of three measures. En *[proceedings of the 15th acm international conference on information and knowledge management]*. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1183614.1183773.

Alshangiti, M., Sapkota, H., Murukannaiah, P. K., Liu, X., y Yu, Q. (2019). Why is developing machine learning applications challenging? a study on stack overflow posts. En *2019 acm/ieee international symposium on empirical software engineering and measurement (esem)* (p. 1-11). doi: 10.1109/ESEM.2019.8870187

Alvarez-Rodríguez, J., Zuñiga, R., Moreno, V., y Llorens, J. (2019, julio). Challenges and opportunities in the integration of the systems engineering process and the ai/ml model lifecycle. *INCOSE International Symposium, 29*, 560-575. doi: 10.1002/j.2334-5837.2019.00621.x

Amazon Inc. (s.f.-a). *Amazon rekognition image.* https://aws.amazon.com/es/rekognition/image-features/.

Amazon Inc. (s.f.-b). *Preguntas frecuentes sobre amazon rekognition.* https://aws.amazon.com/es/rekognition/faqs/.

Amazon Inc. (s.f.-c). *Rekognition.* https://aws.amazon.com/es/rekognition/?nc=sn&loc=0.

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... Zimmermann, T. (2019). Software engineering for machine learning: A case study. En *2019 ieee/acm 41st international conference on software engineering: Software engineering in practice (icse-seip)* (p. 291-300). doi: 10.1109/ICSE-SEIP.2019.00042

Amos, B., Ludwiczuk, B., y Satyanarayanan, M. (2016). Openface: A general-purpose face

recognition library with mobile applications..

An, X., Zhu, X., Gao, Y., Xiao, Y., Zhao, Y., Feng, Z., ... Fu, Y. (2021, octubre). Partial fc: Training 10 million identities on a single machine. En *[ICCV es el principal evento internacional de visión por computadora que comprende la conferencia principal y varios talleres y tutoriales ubicados en el mismo lugar.]*. https://iccv2021.thecvf.com/.

Anderson, K. M. (2015). Embrace the challenges: Software engineering in a big data world. En *2015 ieee/acm 1st international workshop on big data software engineering* (p. 19-25). doi: 10.1109/BIGDSE.2015.12

Apache Software Foundation. (s.f.). *Apache MXNet*. https://mxnet.apache.org/versions/1.9.0/.

Avance Jurídico Casa Editorial Ltda. (2022, abril). *Normograma del SENA Acuerdo 0007 2012*. https://normograma.sena.edu.co/normograma/docs/acuerdo_sena_0007_2012.htm.

Baier, L., Jöhren, F., y Seebacher, S. (2019, mayo). Challenges in the deployment and operation of machine learning in practice. En *[ECIS 2019 - 27th European Conference on Information Systems]*. Kista, Sweden. http://ecis2019.eu/.

Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., y Grundmann, M. (2019). Blazeface: Sub-millisecond neural face detection on mobile gpus. *arXiv*. https://arxiv.org/abs/1907.05047. doi: 10.48550/ARXIV.1907.05047

Belani, H., Vukovic, M., y Car, Z. (2019). *Requirements engineering challenges in building ai-based complex systems*. https://arxiv.org/abs/1908.11791. arXiv. doi: 10.48550/ARXIV.1908.11791

Berkeleyvision. (s.f.). *Caffe*. https://caffe.berkeleyvision.org/.

Bertrand, S. (2020). *Retinaface-tf2*. https://github.com/StanislasBertrand/RetinaFace-tf2.

Bukar, A., y Ugail, H. (2017, julio). Convnet features for age estimation. En *[Presented at: The 11th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing]*. Lisbon, Portugal. https://bradscholars.brad.ac.uk/handle/10454/12860.

Catal, C. (2012, enero). Performance evaluation metrics for software fault prediction studies. *Acta Polytechnica Hungarica, 9*. http://hdl.handle.net/11413/1662.

Centeno, I. D. P. (2021). *MTCNN face detection implementation for TensorFlow, as a pip package*. https://github.com/ipazc/mtcnn.

Chanda, S., Gv, A. C., Brun, A., Hast, A., Pal, U., y Doermann, D. (2019). Face recognition - a one-shot learning perspective. En *2019 15th international conference on Signal-Image*

technology & Internet-Based systems (SITIS). IEEE.

Chechik, G., Sharma, V., Shalit, U., y Bengio, S. (2010, 03). Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, *11*, 1109-1135.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., ... Zhang, Z. (2015). MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv [cs.DC]*.

CIMEC. (2021, febrero). *Coeficiente correlación de pearson*. https://www.cimec.es/coeficiente-correlacion-pearson/.

Congreso de la república de Colombia. (1994, febrero). *Ley 119*. http://www.secretariasenado.gov.co/senado/basedoc/ley_0119_1994.html.

Dalal, N., y Triggs, B. (2005). Histograms of oriented gradients for human detection. En *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. IEEE.

Danilo Sato, C. W., Arif Wider. (2019). *Continuous delivery for machine learning*. https://martinfowler.com/articles/cd4ml.html.

Deepinsight. (2021). *Scrfd an efficient high accuracy face detection*. https://github.com/deepinsight/insightface/tree/master/detection/scrfd.

Deepinsight. (2022a). *Insightface: 2d and 3d face analysis project*. https://github.com/deepinsight/insightface.

Deepinsight. (2022b, mayo). *Insightface: State-of-the-art 2D and 3D face analysis project*. https://github.com/deepinsight/insightface.

Deepinsight. (2022c). *Mode zoo insightface*. https://github.com/deepinsight/insightface/tree/master/model_zoo.

Deng, J., Guo, J., Ververas, E., Kotsia, I., y Zafeiriou, S. (2020). Retinaface: Single-shot multi-level face localisation in the wild. En *2020 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 5202-5211). doi: 10.1109/CVPR42600.2020.00525

Deng, J., Guo, J., Xue, N., y Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. En *2019 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 4685-4694). doi: 10.1109/CVPR.2019.00482

Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., y Zafeiriou, S. (2019). Retinaface: Single-stage dense face localisation in the wild. *arXiv*. https://arxiv.org/abs/1905.00641. doi: 10.48550/ARXIV.1905.00641

Deriche, M. (2008). Trends and challenges in mono and multi biometrics. En *2008 first workshops on image processing theory, tools and applications* (p. 1-9). doi: 10.1109/IPTA.2008.4743801

Desarrollador Android. (2015, marzo). *Instalar el SDK*. https://desarrollador-android

.com/desarrollo/herramientas/descargas-desarrollo/instalar-el-sdk.

Django Software Foundation. (s.f.). *The web framework for perfectionists with deadlines*. https://www.djangoproject.com.

Du, H., Shi, H., Zeng, D., y Mei, T. (2020). The elements of end-to-end deep face recognition: A survey of recent advances. *CoRR, abs/2009.13290*. https://arxiv.org/abs/2009.13290.

Dubuisson, M.-P., y Jain, A. (1994). A modified hausdorff distance for object matching. En *Proceedings of 12th International Conference on Pattern Recognition* (Vol. 1, p. 566-568 vol.1). doi: 10.1109/ICPR.1994.576361

Exadel. (s.f.). *CompreFace - face recognition service*. https://exadel.com/solutions/compreface/.

Face-benchmark.org. (s.f.). *WebFace260M*. https://www.face-benchmark.org/challenge.html.

Fawcett, T. (2006, junio). Introduction to roc analysis. *Pattern Recognition Letters, 27*, 861-874. doi: 10.1016/j.patrec.2005.10.010

García, N., Castro, V. G., Gutiérrez, E. A., y Fernández, E. F. (2017). Comparación de métodos de detección de rostros en imágenes digitales. En *[Actas de las XXXVIII Jornadas de Automática]*. https://dialnet.unirioja.es/servlet/articulo?codigo=6591683.

Geitgey, A. (s.f.). *face_recognition: The world's simplest facial recognition api for python and the command line*.

Ghosh, S. (2019, septiembre). *DeepFace*. https://github.com/swghosh/DeepFace/releases.

Giray, G. (2020). A software engineering perspective on engineering machine learning systems: State of the art and challenges. *CoRR, abs/2012.07919*. https://arxiv.org/abs/2012.07919.

González, J. R. H. (1994). *Redes neuronales artificiales. fundamentos, modelos y aplicaciones*. RA-MA S.A. Editorial y Publicaciones.

Google. (s.f.). *Mediapipe*. https://mediapipe.dev/.

Google. (2015, noviembre). *TensorFlow: Open source machine learning*. https://www.tensorflow.org/.

Guo, J., Deng, J., Lattas, A., y Zafeiriou, S. (2021). Sample and computation redistribution for efficient face detection. *arXiv*. https://arxiv.org/abs/2105.04714. doi: 10.48550/ARXIV.2105.04714

Guo, Q., Chen, S., Xie, X., Ma, L., Hu, Q., Liu, H., ... Li, X. (2019). An empirical study towards characterizing deep learning development and deployment across different frameworks

and platforms. *arXiv*. https://arxiv.org/abs/1909.06727. doi: 10.48550/ARXIV.1909.06727

Gustavo Mendoza Olguín, M. P. d. C. H., Yadira Laureano de Jesús. (2019). Métricas de similaridad y evaluación para sistemas de recomendación de filtrado colaborativo. *Revista de Investigación en Tecnologías de la Información*, *7*(14), 224–240. Descargado de https://www.riti.es/ojs2018/inicio/index.php/riti/article/view/175 doi: 10.36825/RITI.07.14.019

Haykin, S. (2009). *Neural Networks and Learning Machines*. (3rd ed.). Pearson Education. https://lps.ufrj.br/~caloba/Livros/Haykin2009.pdf.

Heikkilä, M., Pietikäinen, M., y Schmid, C. (2009). Description of interest regions with local binary patterns. *Pattern Recognition*, *42*(3), 425-436. Descargado de https://www.sciencedirect.com/science/article/pii/S0031320308003282 doi: https://doi.org/10.1016/j.patcog.2008.08.014

Huang, G. B., Mattar, M., Berg, T., y Learned-Miller, E. (2008, octubre). Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments. En *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*. Marseille, France. https://hal.inria.fr/inria-00321923.

InsightFace. (s.f.). *State of the art deep face analysis library*. https://insightface.ai/.

Jolliffe, I., y Springer-Verlag. (2002). *Principal component analysis*. (2a ed.). Springer. https://books.google.com.co/books?id=TtVF-ao4fI8C.

Josh Patterson, A. G. (2017). *Deep learning: The definitive guide*. O'Reilly UK Ltd. https://www.oreilly.com/library/view/deep-learning/9781491924570/.

Kaspersky. (2021, agosto). *Reconocimiento facial: definición y explicación*. https://latam.kaspersky.com/resource-center/definitions/what-is-facial-recognition.

Kaur, N., Nazir, N., y Manik. (2021). A review of local binary pattern based texture feature extraction. En *2021 9th international conference on reliability, infocom technologies and optimization (trends and future directions) (icrito)* (p. 1-4). doi: 10.1109/ICRITO51393.2021.9596485

Keras Team. (s.f.). *Keras: the python deep learning API*. https://keras.io/.

Kim, M., Zimmermann, T., DeLine, R., y Begel, A. (2018). Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering*, *44*(11), 1024-1038. doi: 10.1109/TSE.2017.2754374

Kirchberg, K. J., Jesorsky, O., y Frischholz, R. (2002). Genetic model optimization for hausdorff distance-based face localization. En M. Tistarelli, J. Bigün, y A. K. Jain (Eds.), *Biometric*

authentication, international ECCV 2002 workshop copenhagen, denmark, june 1, 2002, proceedings (Vol. 2359, p. 103-111). Springer. Descargado de https://doi.org/10.1007/3-540-47917-1_11 doi: 10.1007/3-540-47917-1_11

Kruchten, P. (2003). *The rational unified process: An introduction* (3.ª ed.). USA: Addison-Wesley Longman Publishing Co., Inc.

Larman, C., y Basili, V. R. (2003, jun). Iterative and incremental development: A brief history. *Computer*, *36*(6), 47–56. Descargado de https://doi.org/10.1109/MC.2003.1204375 doi: 10.1109/MC.2003.1204375

Larman, C., y Kruchten, P. (2002). *Applying uml and patterns: An introduction to object-oriented analysis and design and the unified process*. Prentice Hall PTR. https://books.google.com.co/books?id=r8i-4En_aa4C.

Legarda, D. Loaiza, O. (2020, abril). *Encuesta a instructores del Centro Atención Sector Agropecuario SENA Risaralda. Registro de asistencia a clases presenciales*. https://docs.google.com/forms/d/1cAyj98V24LQ0MpuawVZ3kKNQTINFoz3v8Yxp9VVD3TM/viewanalytics.

Leondgarse. (2022). *Insightface keras*. https://github.com/leondgarse/Keras_insightface#current-accuracy.

Lin, S.-H., Kung, S.-Y., y Lin, L.-J. (1997). Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Networks*, *8*(1), 114-132. doi: 10.1109/72.554196

Ling, C. X., Huang, J., y Zhang, H. (2003). Auc: A better measure than accuracy in comparing learning algorithms. En Y. Xiang y B. Chaib-draa (Eds.), *Advances in artificial intelligence* (pp. 329–341). Berlin, Heidelberg: Springer Berlin Heidelberg.

Lwakatare, L. E., Raj, A., Bosch, J., Olsson, H. H., y Crnkovic, I. (2019). A taxonomy of software engineering challenges for machine learning systems: An empirical investigation. En P. Kruchten, S. Fraser, y F. Coallier (Eds.), *Agile processes in software engineering and extreme programming* (pp. 227–243). Cham: Springer International Publishing.

Mahto, S., y Yadav, Y. (2014). A survey on various facial expressionrecognition techniques. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Energy*, *3*.

Maida E. G., P. J. (2015). *Metodologías de desarrollo de software*. [Facultad de Química e Ingeniería “Fray Rogelio Bacon”, Universidad Católica Argentina]. https://repositorio.uca.edu.ar/handle/123456789/522.

Microsoft. (s.f.). *Reconocimiento facial azure*. https://azure.microsoft.com/es-es/services/cognitive-services/face/.

Migdał, P. (2017, abril). *Learning deep learning with keras*. https://p.migdal.pl/2017/04/30/teaching-deep-learning.html.

Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv*. https://arxiv.org/abs/1301.3781. doi: 10.48550/ARXIV.1301.3781

Moschoglou, S., Papaioannou, A., Sagonas, C., Deng, J., Kotsia, I., y Zafeiriou, S. (2017, julio). Agedb: The first manually collected, in-the-wild age database. En (p. 1997-2005). doi: 10.1109/CVPRW.2017.250

Mozilla Developer Network. (s.f.). *Http: Hypertext transfer protocol*. https://developer.mozilla.org/es/docs/Web/HTTP.

Mozilla Developer Network. (2021a, julio). *Acerca de JavaScript*. https://developer.mozilla.org/es/docs/Web/JavaScript/About_JavaScript.

Mozilla Developer Network. (2021b, julio). *CSS*. https://developer.mozilla.org/es/docs/Web/CSS.

Mozilla Developer Network. (2021c, abril). *Html: Lenguaje de etiquetas de hipertexto*. https://developer.mozilla.org/es/docs/Web/HTML.

Nascimento, E. d. S., Ahmed, I., Oliveira, E., Palheta, M. P., Steinmacher, I., y Conte, T. (2019). Understanding development process of machine learning systems: Challenges and solutions. En *2019 acm/ieee international symposium on empirical software engineering and measurement (esem)* (p. 1-6). doi: 10.1109/ESEM.2019.8870157

NumPy Developers. (s.f.). *NumPy documentation - v1.22*. https://numpy.org/doc/stable.

Oasis Open. (2019, marzo). *MQTT Version 5.0*. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

Ojala, T., Pietikainen, M., y Harwood, D. (1994). Performance evaluation of texture measures with classification based on kullback discrimination of distributions. En *Proceedings of 12th international conference on pattern recognition* (Vol. 1, p. 582-585).

ONNX. (s.f.-a). *ONNX runtime (ORT)*. https://onnxruntime.ai/docs/.

ONNX. (s.f.-b). *Open Neural Network Exchange*. https://onnx.ai/.

OpenCV. (s.f.). *Introduction to OpenCV-Python tutorials*. https://docs.opencv.org/4.x/d0/de3/tutorial_py_intro.html.

Ordieres-Meré, J., Limas, M., Ascacibar, F. J., Alba-Elías, F., González-Marcos, A., Pernía-Espinoza, A., y Vergara, E. (2006). *Técnicas y algoritmos básicos de visión artificial recurso electrónico - en línea*.

Ortiz, N., Hernandez Beleño, R., Moreno, R., Mauledoux, M., y Avilés Sánchez, O. (2018, enero).

Survey of biometric pattern recognition via machine learning techniques. *Contemporary Engineering Sciences, 11*, 1677-1694. doi: 10.12988/ces.2018.84166

Otto, M., Thornton, J., y Bootstrap contributors. (s.f.). *Bootstrap*. https://getbootstrap.com/.

Parkhi, O. M., Vedaldi, A., y Zisserman, A. (2015). Deep face recognition. En *British machine vision conference*.

Petrov, A. (2022). *Urllib3 1.26.9 documentation*. https://urllib3.readthedocs.io/en/stable/.

Pham, H. V., Qian, S., Wang, J., Lutellier, T., Rosenthal, J., Tan, L., ... Nagappan, N. (2020). Problems and opportunities in training deep learning software systems: An analysis of variance. En *[Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering]*. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3324884.3416545.

Python Software Foundation. (2021, mayo). *base64 – codificaciones de datos base16, base32, base64, y base85*. https://docs.python.org/es/3/library/base64.html.

Ramírez, S. (s.f.). *FastAPI*. https://fastapi.tiangolo.com.

Rath, S. R. (2021, junio). *Face detection with dlib using HOG and linear SVM*. https://debuggercafe.com/face-detection-with-dlib-using-hog-and-linear-svm/.

Ravichandran, T., y Rothenberger, M. A. (2003, agosto). Software reuse strategies and component markets. *Commun. ACM*, *46*(8), 109–114. https://doi.org/10.1145/859670.859678. doi: 10.1145/859670.859678

Revelo, J. (2015, febrero). *Volley: Librería de android para realizar peticiones http*. https://www.develou.com/android-volley-peticiones-http/.

RFC 7519 - JSON web token (JWT). (s.f.). https://datatracker.ietf.org/doc/html/rfc7519.

Royce, W. W. (1987). Managing the development of large software systems: Concepts and techniques. En *Proceedings of the 9th international conference on software engineering* (p. 328–338). Washington, DC, USA: IEEE Computer Society Press.

Sandberg, D. (2018, abril). *Facenet: Face recognition using TensorFlow*. https://github.com/davidsandberg/facenet.

Sapp, C. E. (2017). *Preparing and architecting for machine learning*. https://dl.icdst.org/pdfs/files3/c8d6cb6df517f1d7d89f4eb475389530.pdf.

Schroff, F., Kalenichenko, D., y Philbin, J. (2015, junio). FaceNet: A unified embedding for face recognition and clustering. *IEEE*. https://doi.org/10.1109%2Fcvpr.2015.7298682. doi: 10.1109/cvpr.2015.7298682

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... Dennison, D. (2015). Hidden technical debt in machine learning systems. En C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, y R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf.

Sengupta, S., Chen, J.-C., Castillo, C., Patel, V. M., Chellappa, R., y Jacobs, D. W. (2016). Frontal to profile face verification in the wild. En *2016 ieee winter conference on applications of computer vision (wacv)* (p. 1-9). doi: 10.1109/WACV.2016.7477558

Serengil, S. I. (2019, julio). *Matlab Model to Keras*. https://github.com/serengil/tensorflow-101/blob/master/python/Matlab-Model-to-Keras.ipynb.

Serengil, S. I. (2022). *Deepface*. https://github.com/serengil/deepface.

Serengil, S. I., y Ozpinar, A. (2020). Lightface: A hybrid deep face recognition framework. En *2020 innovations in intelligent systems and applications conference (asyu)* (p. 23-27). https://doi.org/10.1109/ASYU50717.2020.9259802. doi: 10.1109/ASYU50717.2020.9259802

Silva, I. J. (2018). *Reconocimiento facial basado en redes neuronales convolucionales*. [Ingeniería de las Tecnologías de Telecomunicación, Universidad de Sevilla]. https://idus.us.es/bitstream/handle/11441/85086/TFG-1975-JIMENEZ.pdf.

Sommerville I., M. A. B., Galipienso M. I. A. (2005). *Ingenieria del software*. (7a ed.). Pearson Addison Wesley.

Sotaquira, M. (s.f.). *Detección de rostros con machine learning*. https://www.codificandobits.com/blog/deteccion-de-rostros-machine-learning/.

Sotaquira, M. (2018, junio). *Reconocimiento facial con machine learning: FaceNet y one-shot learning*. https://www.codificandobits.com/blog/reconocimiento-facial-machine-learning/.

Sotaquira, M. (2019). *¿qué son las redes convolucionales?* https://www.codificandobits.com/blog/redes-convolucionales-introduccion/.

Stanford-Clark, A., y Truong, H. L. (2013). *MQTT for sensor networks (MQTT-SN) protocol specification version 1.2*. https://www.oasis-open.org/committees/download.php/66091/MQTT-SN_spec_v1.2.pdf.

SthPhoenix. (2021). *InsightFace-REST*. https://github.com/SthPhoenix/InsightFace-REST.

Sufyanu, Z., Mohamad, F., Yusuf, A., y Nuhu, A. (2016, octubre). Feature extraction methods for face recognition. *International journal of applied engineering research (IRAER), 5*, 5658-5668.

Sun, Y., Wang, X., y Tang, X. (2014a). Deep learning face representation by joint identification-verification. *arXiv*. https://arxiv.org/abs/1406.4773. doi: 10.48550/ARXIV.1406.4773

Sun, Y., Wang, X., y Tang, X. (2014b). Deep learning face representation from predicting 10,000 classes. En *2014 IEEE Conference on Computer Vision and Pattern Recognition* (p. 1891-1898). doi: 10.1109/CVPR.2014.244

Taigman, Y., Yang, M., Ranzato, M., y Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. En *2014 ieee conference on computer vision and pattern recognition* (p. 1701-1708). doi: 10.1109/CVPR.2014.220

The PostgreSQL Global Development Group. (s.f.). *Postgresql*. https://www.postgresql.org/.

Uvicorn.org. (s.f.). *Uvicorn: An asgi web server, for python*. https://www.uvicorn.org.

Victor Iwantooxxoox. (2021, marzo). *Keras OpenFace*. https://github.com/iwantooxxoox/Keras-OpenFace.

Viola, P., y Jones, M. (2005). Rapid object detection using a boosted cascade of simple features. En *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. IEEE Comput. Soc.

Wan, Z., Xia, X., Lo, D., y Murphy, G. C. (2021). How does machine learning change software development practices? *IEEE Transactions on Software Engineering*, *47*(9), 1857-1871. doi: 10.1109/TSE.2019.2937083

Wang, M., y Deng, W. (2021, marzo). Deep face recognition: A survey. *Neurocomputing*, *429*, 215–244. https://doi.org/10.1016%2Fj.neucom.2020.10.081. doi: 10.1016/j.neucom.2020.10.081

Wu, X., He, R., Sun, Z., y Tan, T. (2015). *A light cnn for deep face representation with noisy labels*. https://arxiv.org/abs/1511.02683. arXiv. doi: 10.48550/ARXIV.1511.02683

Yang, M.-H., Kriegman, D., y Ahuja, N. (2002). Detecting faces in images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(1), 34-58. doi: 10.1109/34.982883

Yi, D., Lei, Z., Liao, S., y Li, S. Z. (2014). Learning face representation from scratch. *arXiv*. https://arxiv.org/abs/1411.7923. doi: 10.48550/ARXIV.1411.7923

Zhang, K., Zhang, Z., Li, Z., y Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, *23*(10), 1499-1503. doi: 10.1109/LSP.2016.2603342

Zhang, R., Xiao, W., Zhang, H., Liu, Y., Lin, H., y Yang, M. (2020, junio). An empirical study on program failures of deep learning jobs. En *[Proceedings of the ACM/IEEE 42nd Inter-*

national Conference on Software Engineering, Association for Computing Machinery]. Seoul South Korea. https://doi.org/10.1145/3377811.3380362.

Zhang, T., Gao, C., Ma, L., Lyu, M., y Kim, M. (2019, octubre). An empirical study of common challenges in developing deep learning applications. En *[2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)]*. Berlin, Germany. https://ieeexplore.ieee.org/document/8987482.

Zheng, T., y Deng, W. (2018, febrero). *Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments* (Inf. Téc. n.º 18-01). Beijing University of Posts and Telecommunications. doi: null

Zheng, T., Deng, W., y Hu, J. (2017). Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments. *arXiv*. https://arxiv.org/abs/1708.08197. doi: 10.48550/ARXIV.1708.08197

Zhu, Z., Huang, G., Deng, J., Ye, Y., Huang, J., Chen, X., ... Zhou, J. (2021). Webface260m: A benchmark unveiling the power of million-scale deep face recognition. *arXiv*. https://arxiv.org/abs/2103.04098. doi: 10.48550/ARXIV.2103.04098