Verificación de Identidad Mediante Inteligencia Artificial: Comparación de Reconocimiento Facial y Documentos Legales Capturados con Dispositivos Móviles.

Conceptos Avanzados de Inteligencia Artificial

Profesor: Gustavo Gutierrez

Estudiante: Jaime Andrés Mejía Osorio
Código: 1004776790

Universidad Tecnológica de Pereira
Marzo, 2025

Tabla de contenido

1. Introducción y Objetivos

1.1 Contexto y Necesidad

El reconocimiento facial se ha convertido en una herramienta fundamental en múltiples aplicaciones, especialmente en el registro masivo de personas y la verificación de identidad en entornos digitales. Su uso permite automatizar procesos que de otro modo requerirían una gran cantidad de recursos humanos y tiempo, garantizando mayor seguridad y eficiencia. En particular, este documento se enfoca en el desarrollo de un sistema de reconocimiento facial para verificar que una persona realmente exista y que su rostro tenga una alta similitud con el de su documento de identidad.

Esta necesidad se encuentra presente en aplicaciones de gobierno, instituciones bancarias, plataformas de educación en línea, control de acceso en empresas y aeropuertos, entre otros sectores. La correcta implementación de un sistema de reconocimiento facial no solo reduce los errores humanos, sino que también agiliza procesos y fortalece la seguridad en la validación de identidad. En nuestro caso nos encontramos tempranamente en la necesidad de implementar este sistema en una aplicación de seguridad ciudadana enfocada en prestar atención en mujeres que estén en riesgo y necesiten presencia policiaca, en la que a partir de datos de registro en la app (1 foto de su rostro y dos fotos de su documento de identidad) para no solo las mujeres sino también varias personas que hacen parte de su contacto de agenda, vemos la necesidad de poder abarcar validaciones de que se trate de una persona real y un documento legal de identidad válido.

En este documento exploramos dos enfoques principales:

Prueba piloto en entorno local: Un desarrollo controlado que nos permitirá comprender los fundamentos del reconocimiento facial desde la inteligencia artificial, evaluar tecnologías de código abierto y optimizar recursos computacionales para un sistema funcional sin depender de servicios en la nube.

Escalabilidad y producción: Una proyección a futuro donde consideramos cómo integrar servicios avanzados como AWS Rekognition o Google Vision para mejorar la eficiencia en producción y manejar cargas de trabajo a gran escala, tercerizando la necesidad para tener resultados más fiables sin tantas complicaciones.

1.2 Objetivos

Los objetivos de este documento se dividen en dos niveles:

1.2.1 Objetivos de la Prueba Piloto

El objetivo principal de la prueba piloto es comprender a fondo el funcionamiento de la inteligencia artificial aplicada al reconocimiento facial y evaluar su viabilidad en un entorno controlado antes de considerar una implementación a gran escala. Para ello, abordaremos los siguientes puntos:

Comprender el pipeline del reconocimiento facial: Desde la detección de rostros con sus características hasta comparar y encontrar un % de similitud con su documento de identidad suficiente para aprobar o no aprobar.

Seleccionar e implementar modelos de detección y extracción de características: Usaremos **RetinaFace** para la detección facial y **ArcFace** para la comparación de rostros, abarcando la razón de elegir estos modelos sobre otras alternativas como **MTCNN** o **FaceNet**.

Justificar el uso de modelos pre entrenados: Comprender las razones por las que entrenar un modelo propio no es viable en nuestro contexto, debido a la ausencia de grandes volúmenes de datos y poder computacional.

Evaluar el rendimiento y precisión del sistema en hardware específico: Implementaremos nuestro sistema en un equipo con Intel i5-12450HX, RTX 4050 Laptop (6GB DDR6) y 24GB RAM para medir su desempeño.

Comparar el desempeño del reconocimiento facial con alternativas en la nube: Realizar comparaciones entre el desempeño y resultados de nuestra prueba piloto contra servicios comerciales como **AWS Rekognition** y **Google Vision**.

1.2.2 Objetivos para el Sistema en Producción

El desarrollo de la prueba piloto servirá como base para proyectar la implementación de un sistema de reconocimiento facial en producción. En este nivel, los objetivos incluyen:

Escalabilidad: Plantear cómo un sistema local puede escalar a una plataforma en la nube con servicios que permitan manejar grandes volúmenes de datos y usuarios para conseguir resultados más fieles y seguros.

Integración con sistemas externos: Evaluar cómo los servicios en la nube pueden integrarse con nuestro flujo de trabajo o como poder aplicarlo y qué APIs pueden facilitar esta transición en un entorno web.

Optimización del reconocimiento facial en entornos reales: Considerar cómo mejorar el rendimiento del sistema en términos de velocidad y precisión sin comprometer la seguridad.

Análisis de costos: Comparar la inversión en infraestructura local con el costo de utilizar servicios de terceros.

1.3 Justificación de las Tecnologías Seleccionadas

En este documento hemos elegido tecnologías específicas en función de su precisión, rendimiento y aplicabilidad en nuestro caso de estudio. A continuación, justificamos estas elecciones en comparación con otras opciones disponibles:

1.3.1 Detección de Rostros

Hemos seleccionado **RetinaFace** debido a su precisión en la detección de rostros, especialmente en imágenes donde los rostros son pequeños o presentan oclusión parcial. Aunque **MTCNN** es una alternativa más ligera y rápida, su desempeño en condiciones desafiantes es inferior.

1.3.2 Extracción de Características y Comparación de Rostros

Para la extracción de características y la comparación de rostros, elegimos **ArcFace** debido a su alta precisión y adopción en entornos académicos e industriales. Alternativas como **FaceNet** y **DeepFace** fueron consideradas, pero **ArcFace** demostró mejores resultados en benchmarks.

1.3.3 Implementación Local vs. Servicios en la Nube

Optamos por una implementación local en la prueba piloto para mantener el control del proceso y evitar la dependencia de servicios en la nube. Sin embargo, en producción, **AWS Rekognition** o **Google Vision** podrían ser soluciones viables para mejorar la escalabilidad y reducir la carga computacional en nuestros servidores.

2. Introducción

El avance de la inteligencia artificial (IA) ha transformado radicalmente la manera en que se abordan problemas complejos en distintas industrias, particularmente en el ámbito de la identificación biométrica. Uno de los campos más relevantes dentro de esta área es el reconocimiento facial, una técnica que permite verificar la identidad de una persona a partir del análisis de características faciales extraídas de una imagen.

Este documento presenta el desarrollo de un sistema de reconocimiento facial enfocado en la verificación de identidad mediante la comparación entre el rostro de un ciudadano y la imagen contenida en su documento legal de identidad colombiano. Esta solución se orienta inicialmente hacia una prueba piloto en entorno local, y posteriormente a su escalado hacia un sistema productivo, confiable y automatizado para entornos reales con alto volumen de registros.

2.1 Problemática

Actualmente, existen procesos manuales en distintas plataformas donde operadores humanos deben validar que las fotografías proporcionadas por los usuarios durante el registro correspondan efectivamente a personas reales, y que coincidan con los datos biométricos impresos en documentos de identidad. Este procedimiento:

- Es propenso a errores humanos.
- Se vuelve ineficiente en escenarios con grandes volúmenes de registros.

En el caso particular de este proyecto, el sistema será aplicado a una aplicación de seguridad ciudadana donde mujeres en riesgo pueden registrar contactos de confianza (familiares, amigos, etc.; donde también deben registrarse y pasar un filtro de identificación) y dejar asociada su identidad mediante el registro facial y documental. Automatizar esta verificación es clave para agilizar el proceso y reducir fallos.

2.2 Justificación Académica y Técnica

Desde un enfoque académico, este trabajo se enmarca como parte de un proyecto en la materia de maestría de Conceptos Avanzados de Inteligencia Artificial, y tiene como finalidad:

- Profundizar en la comprensión técnica de los algoritmos de detección, extracción y comparación facial (Como elección del presente alumno).
- Evaluar en entorno local las capacidades reales de modelos de código abierto sin depender de soluciones de terceros.
- Identificar oportunidades de mejora en las etapas del pipeline para su futura escalabilidad.

En el ámbito técnico, se busca construir una base funcional capaz de:

- Filtrar imágenes no aptas desde el punto de vista visual (borrosas, mal iluminadas, sin rostro).
- Detectar con precisión los rostros en la imagen del ciudadano y de su documento.
- Extraer vectores de características (embeddings) y calcular su similitud para emitir un veredicto de coincidencia.

La elección de tecnologías como **RetinaFace** y **ArcFace** (Ambas de InsightFace) se fundamenta en su precisión y uso en la comunidad investigativa, con acceso a documentación para poder formular una solución técnicamente robusta con capacidad de análisis para comprender la aplicabilidad de la inteligencia artificial en el medio en base a una necesidad expuesta.

2.3 Alcance del Documento

Este documento cubre todo el ciclo de diseño e implementación de la solución en su fase inicial (prueba piloto), incluyendo:

- Revisión de tecnologías existentes y justificación de elecciones.
- Indagación técnica de cómo funcionan estas inteligencias.
- Diseño del pipeline de reconocimiento facial.
- Configuración del entorno local y hardware utilizado.
- Evaluación del rendimiento y precisión de los modelos seleccionados.
- Análisis comparativo frente a soluciones comerciales en la nube.
- Posibles consideraciones para su escalabilidad, sostenibilidad y aplicación en campo real.

3. Estado del Arte y Comparación de Tecnologías

3.1 Modelos de Detección de Rostros

La detección de rostros es la etapa inicial en nuestro sistema de reconocimiento facial. Su precisión y robustez condicionan directamente la calidad del resto del pipeline, ya que un fallo en esta fase imposibilita o degrada la extracción de características y la comparación de rostros.

Para nuestro caso consideramos dos modelos, el **MTCNN** (Multi-task Cascaded Convultional Networks) y **RetinaFace**.

3.1.1 MTCNN (Multi-task Cascaded Convolutional Networks)

MTCNN es un detector de rostros basado en cascadas de redes neuronales convolucionales (CNN) entrenadas de manera conjunta para tres tareas simultáneas: detección del rostro, estimación de bounding box y localización de cinco landmarks faciales. Su diseño en tres etapas (P-Net, R-Net, O-Net) permite filtrar progresivamente las regiones candidatas, mejorando la precisión con cada paso.

Ventajas técnicas:

- Eficiente en dispositivos con recursos limitados.
- Buen rendimiento en condiciones estándar de iluminación y pose.
- Soporta landmarks faciales, útiles para alineamiento posterior.
- Fácil de implementar e integrar en aplicaciones ligeras.

Desventajas técnicas:

- Presenta dificultades con rostros pequeños o parcialmente obstruidos.
- Menor precisión frente a métodos más recientes como RetinaFace.
- No incorpora mecanismos explícitos de 3D o contexto multiescala.

Contexto de uso ideal:
Aplicaciones móviles, sistemas en tiempo real, prototipos rápidos con bajo costo computacional.

3.1.2 RetinaFace

RetinaFace es un detector de rostros avanzado desarrollado por **InsightFace**, basado en un enfoque single-shot y multiescala que extiende el concepto de detección de objetos al plano facial. Incorpora de forma unificada tres tareas: detección de rostro, alineación de landmarks 2D y reconstrucción 3D mediante regresión de vértices.

Principales aportes técnicos (Según lo expuesto en el paper de **RetinaFace**):

- Utiliza Feature Pyramid Networks (FPN) para capturar información en múltiples escalas, detectando rostros desde muy pequeños hasta grandes.
- Integra “deformable convolutions” para modelar el contexto local, mejorando la detección en rostros con pose, iluminación o expresión compleja.
- Implementa regresión simultánea de landmarks 2D y 3D, mejorando la precisión del bounding box al aprender puntos semánticos adicionales.
- Arquitectura “fully convolutional”, lo que permite inferencia eficiente y entrenamiento en una sola etapa.

Ventajas técnicas:

- Alta precisión en rostros difíciles: pequeños, ocluidos, con pose extrema.
- Localización precisa de landmarks, útil para el alineamiento facial posterior.
- Diseño robusto para detección en condiciones reales (in-the-wild).
- Compatible con modelos de extracción como **ArcFace** y **FaceNet**.

Desventajas técnicas:

- Mayor carga computacional, requiere GPU para funcionar en tiempo razonable.
- Implementación algo más compleja que **MTCNN**.

Contexto de uso ideal:
Procesamiento por lotes, aplicaciones donde la precisión es crítica (e.g., documentos escaneados, seguridad).

3.1.3 Comparación Directa y Justificación de Elección

Criterio	MTCNN	RetinaFace
Precisión en rostros pequeños	Media	Alta
Manejo de oclusiones y ángulos	Limitado	Robusto
Landmarks 2D	Sí (5 puntos)	Sí (5 puntos + opcional malla 3D)
Velocidad	Alta (CPU)	Media-Baja (requiere GPU)
Escalabilidad	Limitada	Alta (mejor rendimiento en grandes datasets)
Nivel de detalle	Básico	Avanzado (detección + alineación + reconstrucción 3D)

Dado que la prueba piloto contempla imágenes capturadas por múltiples dispositivos y fotografías de cédulas que contienen rostros pequeños y con posibles oclusiones (reflejos, calidad baja, sombras), **RetinaFace** representa una mejor opción. Su capacidad para detectar rostros de tamaño reducido y mejorar la estabilidad mediante landmarks 2D y proyecciones 3D garantiza una base más sólida para las etapas posteriores de alineación y reconocimiento.

Además, al formar parte del ecosistema **InsightFace**, **RetinaFace** se integra fácilmente con modelos como **ArcFace**, lo que simplifica la compatibilidad entre módulos.

3.1.4. ¿Cómo funciona RetinaFace?

- **Reconstrucción Facial 3D**

Principalmente por medio de la innovación frente a otros detectores es la reconstrucción tridimensional (3D) a partir de una imagen 2D por medio de una malla 3D del rostro que usa un número fijo de vértices en una topología triangular, haciendo que cada vértice representa un punto específico del rostro con coordenadas (x, y, z).

Según nos explica el paper se usa un $N = 1103$ vértices ($1k + 68\,puntos$) seleccionados de una malla de referencia más grande (Mesh53k) (La malla facial Mesh53k tiene aproximadamente 53 mil vértices en la superficie de un rostro humano en 3D), permitiendo conservar suficiente detalle de la forma facial.

Estos puntos se “predicen” directamente desde la imagen por medio de una **red neuronal** que aprende a mapear regiones del rostro a esta forma 3D

- **Vertex Loss (L_{vert})**

Posteriormente a esto lo que hace es evaluar que tan bien la red predijo cada uno de los $N = 1103$ vértices en comparación con los valores reales (ground-truth) usando la fórmula:

$$L_{vert} = \frac{1}{N}\sum_{i=1}^{M}\left\|V_i(x.\ y,\ z) - V_i^*(x,\ y,\ z)\right\|$$

¿De qué sirve esto? Forzar al modelo a colocar los puntos de la malla mas cerca posible de la posición correcta en 3D

- **Edge Loss (L_{edge})**

En conjunto con la posición de los puntos se toma en cuenta la longitud de los bordes entre puntos conectados por triángulos. Para así poder mantener la estructura geométrica de la cara por medio de la fórmula:

$$L_{edge} = \frac{1}{3M}\sum_{i=1}^{M}\left\|E_i - E_i^*\right\|$$

Siendo $M = 2110$ Es el número de bordes (triangulares) en la malla.

- **Pérdida total**

Ambas pérdidas se combinan en una unica funcion de perdida de regresión 3D

$$L_{mesh} = L_{vert} + \lambda_0 L_{edge}$$

Donde λ_0 es un parámetro de peso que normalmente es 1

- **Localización Multinivel del Rostro (Multi-level Face Localisation)**

RetinaFace incorpora múltiples tareas de localización dentro de un mismo modelo, prediciendo tanto el “bounding box”, “landmarks faciales 2D” y “coordenadas 3D” de vértices de la cara en un solo paso denominado **single-shot**

Landmarks: Puntos clave específicos en un rostro humano (e.g. Esquinas de los ojos, la punta de la nariz o las comisuras de la boca), para un contexto 2D se definen por coordenadas (x, y) en una imagen; usandose para analizar la forma y las características faciales siendo esenciales para tareas como el reconocimiento facial y la detección de expresiones.

Bounding box (Caja delimitadora): Es un rectángulo que encierra un objeto detectado en una imagen, en nuestro caso, un rostro. Se define por coordenadas de las esquinas indicando la ubicación del rostro en la imagen y se usa para localizar y aislar el rostro antes de realizar análisis faciales más detallados.

Anchor: Es una región candidata (pequeño recuadro) que propone la red para contener un rostro, cada uno se prueba con regresiones para determinar su posición y forma exacta.

Lo que realiza RetinaFace durante el entrenamiento en este punto es optimizar una pérdida conjunta (multi-task loss) que combina varias tareas:

$$L = L_{cls} + \lambda_1 p_i^* L_{box} + \lambda_2 p_i^* L_{pts} + \lambda_3 p_i^* L_{mesh}$$

Donde:

- L_{cls}: perdida de clasificación (rostro/no rostro)
- L_{box}: Error en la predicción del bounding box
- L_{pts}: Error en la predicción de landmarks faciales 2D
- L_{mesh}: Pérdida de reconstrucción 3D (Explicado anteriormente)
- p_i^*: Indica si el “anchor” de entrenamiento es positivo (1) o negativo (0)
- λ_1, λ_2, λ_3: Coeficientes de ponderación (todos igual a 1 según el paper)

    - ¿Que nos retorna?

Para cada anchor positivo (Es decir, aquel que contenga un rostro), el modelo predice:

- Posición del rostro por medio de regresión logarítmica del ancho y alto de la caja
- Ubicación de 5 landmarks faciales 2D: Ojos, nariz, comisuras de la boca
- Coordenadas 3D de vértices: Regresión para 1103 puntos considerando x, y, z

Se normalizan los valores por la escala del anchor para estabilizar el entrenamiento, y las coordenadas z se traducen de forma que el punto de la nariz sea $z = 0$, facilitando la reconstrucción coherente de la malla 3D

    - Es multinivel dado que…

Al dividirse por tareas (detection box, landmarks 2D y malla 3D) tenemos diferentes niveles de detalles:

Tarea	Nivel de precisión	Finalidad
Bounding box	Bajo	Localizar el rostro de forma general.
Landmarks 2D (ojos, boca)	Medio	Alineación del rostro y mejor centrado del box.
Malla 3D (1103 vértices)	Alto	Detalle anatómico del rostro, robusto a poses/extremos.

De esta manera la **red** aprende mejor la detección, ubicación y comprensión de un rostro.

- **Localización Multinivel en un Sola Pasada (Single-shot Multi-level Face Localisation)**

RetinaFace realiza todos los procesos necesarios para obtener la información facial en un solo barrido (single-shot), permitiendo detección eficiente y de alta precisión, componiendo de esta forma una arquitectura de cuatro bloques fundamentales trabajando de forma conjunta

- ***Feature Pyramid Network (FPN)***

Se usa una pirámide de características construida a partir de una red base (Como ResNet). Genera cinco mapas de características (P2 a P6), cada uno capturando información a diferentes escalas sin necesidad de redimensionar o reprocesar:

- En las etapas C2 a C5 del backbone se genera cinco niveles de pirámide
- P2 detecta rostros pequeños
- P6 cubre rostros grandes o muy cercanos
- Cada nivel genera mapas de características de tamaño reducido pero ricos en semántica visual (Forma, textura, bordes, etc.)
- El paper nos indica que P2 a P5 se obtienen de ResNet (pre entrenada en ImageNet-11k)
- P6 se genera con una convulsión 3x3 sobre C5 para extender el rango de detección
- Las capas se conectan con conexiones top-down y laterales para evitar perder información jerárquica
- Con la información combinada de capas bajas y altas: Capas bajas tienen más detalles (bordes), las altas más abstracción (Forma general)

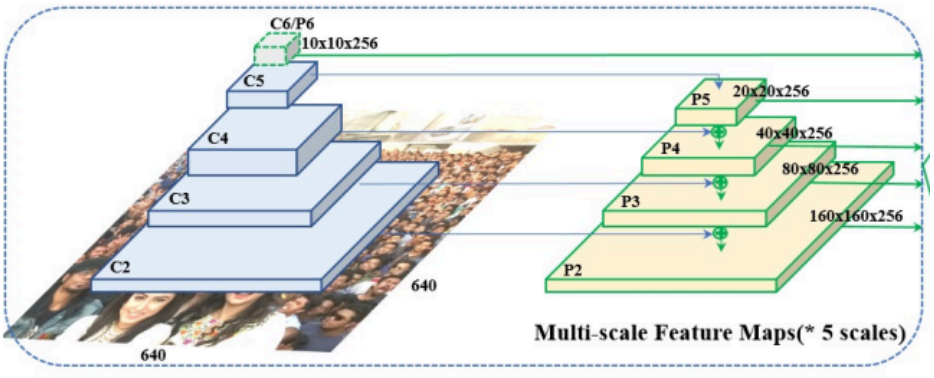


Figura 1. Ilustración de cómo RetinaFace construye una pirámide de características a partir de una imagen con entrada 640x640 px. En la izquierda se muestran las salidas de distintas etapas convolucionales (C2 a C5) del backbone de ResNet, capturando información jerárquica a diferentes niveles de abstracción. A la derecha se ven las salidas en cinco mapas de características (P2 a P6), cada uno con una resolución espacial.

FPN es clave en nuestro desarrollo dado que garantiza una detección robusta de rostros pequeños, medianos y grandes sin reescalar las imágenes de entrada. Cada escala se utiliza posteriormente por los módulos de contexto y regresión para predecir bounding boxes, landmarks y mallas 3D.

- ***Módulo de contexto (Context Module)***

Esta parte se diseñó para mejorar la capacidad de detección al entender mejor el entorno o contexto visual del rostro. Analizando no solo la región central del rostro sino también áreas circundantes para poder hacer detecciones más robustas. Esto lo hace de esta manera:

- Aplica un módulo de contexto a cada uno de los cinco niveles de la pirámide de características (P2 a P6)

- El módulo reemplaza las convulsiones estándar 3x3 por convulsiones deformables, una técnica más flexible que hace que el kernel aprenda en donde enfocarse dinámicamente, haciendo que en lugar de filtrar sobre una cuadrícula rígida las convoluciones deformables pueden “doblarse” para seguir la forma de la cara o de los ojos.
- Se aumenta así el campo receptivo, expandiendo lo que puede ver alrededor del punto central, mejorando la capacidad del modelo para adaptarse a rostros no rígidos como expresiones faciales o inclinaciones de cabeza.

Se inspira de SSH (Single Stage Headless), que usa módulos de contexto con distintas ramas de convolución para captar detalles a varias escalas; también se inspira de PyramidBox que explora el uso del contexto circundante para detectar rostros difíciles.

El usar un módulo de contexto ayuda a comprender ángulos con oclusiones (manos, gafas) o expresiones faciales fuertes, lo que nos podría ayudar significativamente en la comparación de la foto de la persona contra su documento legal de identificación, esto dado que el documento puede presentar oclusiones significativas a la hora de extraer información.

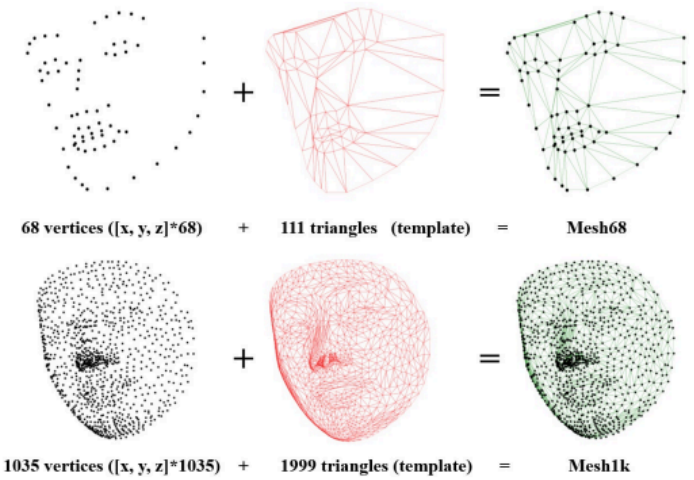


Figura 1. Una malla consta de vértices y triángulos. Mesh68 es una versión básica utilizada para la evaluación cuantitativa, mientras que Mesh1k es una versión más elaborada que incluye detalles faciales.

- ***Pérdida Multitarea en Cascada (Cascade Multi-task Loss)***

Es una técnica para mejorar la precisión del sistema mediante una estrategia de refinamiento progresivo, basándose en usar dos etapas consecutivas de predicción (o “cabezas”) que calculas distintas salidas (bounding box, landmarks, malla 3D), refinando los resultados paso a paso en en single-shot

Context Head 1:

- Conecta directamente los mapas de características (FPN)
- Predice las primeras versiones de:

    - Bounding box
    - Landmarks faciales
    - Malla 3D (1k vértices)
    - Clasificación (rostro/no rostro)
- Usa convolución 1x1 sobre cada uno de los 5 mapas (P2 a P6) para producir predicciones

Context Head 2:

- Usa resultados del primer Context Head como una nueva base (Regressed anchors)
- Vuelve a predecir todo pero de forma más precisa
- Actúa como refinación final

Ambas cabezas se ligan a la misma función de pérdida multitarea que se describió anteriormente

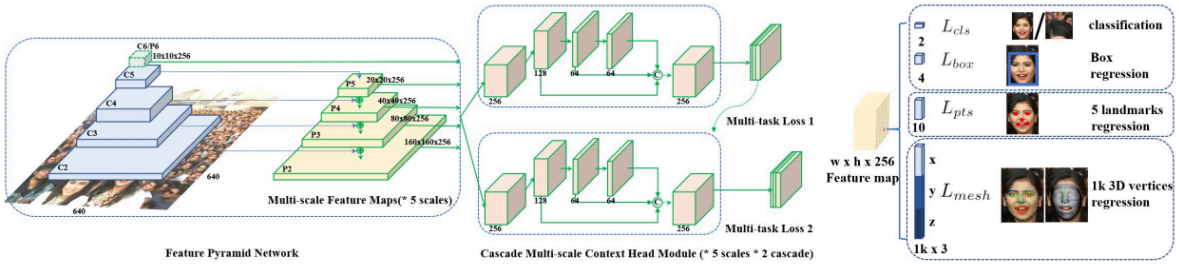


Figura 2. Arquitectura general de RetinaFace

- ***Configuración de anclas y Estrategia de Emparejamiento (Anchor Settings and Marching Strategy)***

Primero debemos definir “anchors”, en nuestro caso para detectar objetos (Rostros), podemos definirlo como cajas predefinidas de distintos tamaños y proporciones que se colocan a lo largo de la imagen para predecir si contienen un rostro. En palabras más simples, son “candidatos” iniciales que luego son refinados por el modelo.

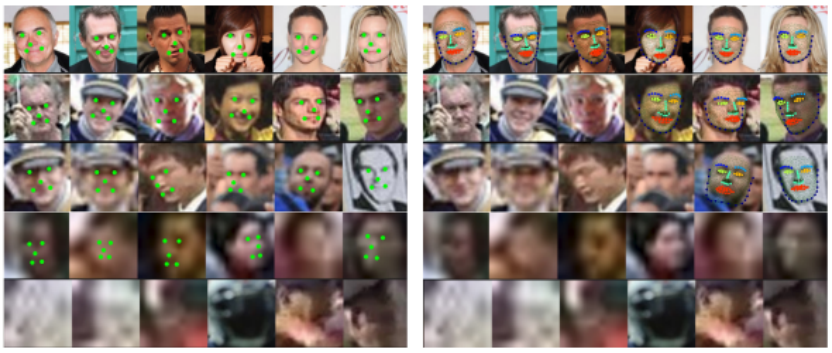
Con esto en mente, podemos explicar cómo se abordan en RetinaFace:

- Se usan anchors específicos por escala en cada nivel de FPN
- P2 se enfoca especialmente en rostros pequeños por lo que se usan anchors más pequeños, esto incrementa la carga computacional, sin embargo, mejora la detección en casos difíciles.
- Usa una escala base, se avanza en $2^{\frac{1}{3}}$ para generar anclas de diferentes tamaños
- Relación de aspecto fija en 1:1
- Para una imagen de entrada 640x640 pixeles
    - Los anchors cubren desde 16x16 hasta 406x406
    - Se general 102300 anchors en total
    - 75% de ellos provenientes de P2

Para la Estrategia de Emparejamiento, cada anchor se compara con las cajas reales (ground-truth), decidiendo:

- Positivo: Cuando IoU (Intersección sobre Unión) con una caja real es mayor a:
    - 0.7 para la primera cabeza de regresión
    - 0.5 para la segunda cabeza
- Negativo (Background): Cuando IoU es menor a:
    - 0.3 para la primera cabeza
    - 0.4 para la segunda cabeza
- Anchors no emparejados se descartan

Se usa OHEM (Online Hard Example Mining) para equilibrar ejemplos positivos y negativos durante el entrenamiento, dando así más peso a los ejemplos difíciles de clasificar.



(a) Five Landmarks Annotation (b) 1k 3D Vertices Annotation

Figura 3. a) Anotación de los 5 landmarks clásicos. b) 1k vértices 3D utilizados en la malla facial

- **Entrenamiento de RetinaFace**
    - ***Dataset***

RetinaFace fue entrenado utilizando principalmente el dataset WIDER FACE, con anotaciones adicionales de AFLW y FDDB para tareas específicas

- WIDERFACE
    - 32203 imagenes con 393703 rostros anotados
    - Tiene variaciones extremas de escala, pose, expresión, oclusión e iluminación
    - Para entrenar landmarks:
        - Se anotan manualmente cinco puntos claves (ojos, nariz, boca) en:
            - 84.6k rostros del conjunto de entrenamiento
            - 18.5k rostros del conjunto de validación
    - Entrenamiento de malla 3D:
        - Pipeline semiautomatico:
            - Se generan 68 puntos faciales
            - Se ajusta un modelo 3DMM con 53k vértices
            - Si la reconstrucción no es precisa, el anotador lo corrige automáticamente
- AFLW y FDDB
    - Se añaden tareas de reconstrucción 3D
    - Se incluyen más de 67k anotaciones 3D adicionales

- ***Implementación del Dataset***

Para mejorar la robustez del modelo y aumentar la cantidad de datos útiles:

- Se aplicaron recortes aleatorios sobre las imágenes.
- Las imágenes se redimensionan a 640 × 640.
- Espejado horizontal aleatorio.
- Distorsiones fotométricas (color, brillo, etc.).

Optimizaciones:

- Se usó SGD (Stochastic Gradient Descent) con:
  - Momentum = 0.9
  - Weight decay = 0.0005
  - Bath size = 32 (8x4 GPUs)
- Aprendizaje con tasa variable
  - 10^{-3} durante 5 épocas
  - Luego 10^{-2} hasta la época 55
  - Reducción en las épocas 55 y 68
  - Finaliza en la epoca 80

Infraestructura:

- 4 GPUs NVIDIA Tesla P40 de 24GB
- Implementación en MXNet
- Modelo base: ResNet-50, con tamaño total de 155 MB
- Velocidad de inferencia: 22.3ms por imagen en una P40

SGD (Stochastic Gradient Descent)
Es un algoritmo de optimización para entrenar redes neuronales profundas, teniendo como objetivo minimizar la función de pérdida ajustado pesos de la red.

Tiene como diferencia con el descenso de gradiente tradicional (Este calcula el gradiente usando todo el conjunto de datos), lo hace calculando el gradiente usando un subconjunto aleatorio de datos (batch), así permitiendo actualizaciones más frecuentes y rápidas aunque más ruidosas.

$$\theta_{t+1} = \theta_t - \eta \cdot \Delta_0 L(\theta_t)$$

Donde:

- θ_t: Representa los pesos del modelo en el paso t
- η: tasa de aprendizaje (Learning rate)
- $\Delta_0 L(\theta_t)$: Gradiente de la función de pérdida L respecto a los pesos, calculado sobre un minibatch de datos

Dado que RetinaFace usa momentum, implica añadir un termino de velocidad acomulada para suavizar actualizaciones:

$$v_t + 1 = \mu v_t - \eta \cdot \Delta_0 L(\theta_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$

Donde:

- v_t: Es la velocidad (momentum) en el paso t, acomulando gradientes pasados
- μ: Es el coeficiente de momentum, usualmente cercano a 1 (En RetinaFace se usa 0.9)

Learning Rate (Tasa de aprendizaje variable)
Esta controla el tamaño del paso que da el optimizador al ajustar los pesos, si es alta el modelo puede saltar una solución óptima; si es muy baja, el entrenamiento es muy lento o puede atascarse en mínimos.

RetinaFace emplea una estrategia de tasa variable que tiene como enfoque aprender rápido al inicio, redefiniendo cuidadosamente el modelo reduciendo la tasa, mejorando la estabilidad y precisión final.

Etapa	Tasa de aprendizaje	Propósito
Épocas 1–5	10−310^{-3}10−3	Inicio suave, evita inestabilidad.
Épocas 6–54	10−210^{-2}10−2	Aprendizaje rápido en fase media.
Época 55↓	tasa reducida gradualmente	Refinamiento final, mayor precisión.

3.2. Modelos de Extracción de Características y Comparación de Rostros

En este apartado nos encontramos en el análisis de modelos encargados de generar representaciones numéricas (embeddings) de un rostro para compararlos entre sí. Dada nuestra necesidad de identificar a la persona por la foto de su rostro contra la de su documento legal necesitamos determinar un grado de similitud para la verificación de identidad.

Embedding facial

Vector de características de alta dimensión (e.g. 512 valores) que encapsula los rasgos más representativos de un rostro. Los vectores se comparan posteriormente usando distancias (coseno, euclidiana) para determinar si dos rostros pertenecen a una misma persona.

3.2.1. FaceNet

Es un modelo de reconocimiento facial propuesto por Google en 2015, su innovación fue representar un rostro directamente como un vector de características (embedding) en un espacio euclidiano. FaceNet entrena su propia red para minimizar una función de pérdida tripleta (triplet loss), logrando que las imágenes de una misma persona estén más cerca entre sí que aquellas de diferentes personas.

Ventajas técnicas:

- Es compacto y eficiente, los embeddings pueden tener solo 128 dimensiones
- Apto para tareas de verificación, identificación y clustering facial
- Tiene buen rendimiento
- Tiene buena documentación y ejemplos dentro de la comunidad

Desventajas técnicas:

- Tiene una precisión inferior a ArcFace en benchmarks recientes
- La selección de triplets para entrenamiento es crítica y puede generar sobreajuste si no se hace correctamente
- No incorpora pérdida angular, limitando la separación entre clases en el espacio latente

Contexto de uso ideal:
Aplicaciones donde el almacenamiento y la eficiencia sean prioritarios, como dispositivos móviles o bases de datos distribuidas con pocos ejemplos por identidad.

3.2.2. ArcFace

Es un modelo de reconocimiento facial basado en aprendizaje profundo, propuesto por el equipo de InsightFace. Este incorpora el uso de una función de pérdida angular aditiva (Additive Angular Margin Loss), mejorando significativamente la iscriminación entre entidades al forzar que los embeddings de distintas personas estén más separados entre sí en el espacio angular. En lugar de solo minimizar la distancia entre ellos, maximiza la separación angular entre clases, mejorando la precisión en tareas de verificación y reconocimiento facial.

Ventajas técnicas:

- Alta precisión en benchmarks (LFW, MegaFace, IJB-B), superando a modelos como FaceNet y SphereFace
- Mejor discriminación entre clases gracias al margen angular
- Fácil integración con frameworks existentes y compatible con embeddings de dimensión fija
- Robustez ante variaciones de iluminación, pose y expresión.

Desventajas técnicas:

- Requiere mayor potencia de cómputo en entrenamiento
- El margen angular puede necesitar ajustes finos para diferentes datasets
- Al trabajar con clasificación angular, requiere entrenamiento supervisado con identidades etiquetadas

Contexto de uso ideal:
Sistemas donde la presión y robustez son críticas como la verificación de identidad en ambientes no controlador, comparación de rostros en condiciones distintas (e.g. Selfie contra documento) y aplicaciones de seguridad, control de acceso y monitoreo biométrico.

Integración con RetinaFace:

- RetinaFace realiza detección y alineamiento facial con alta precisión
- ArcFace se encarga de convertir el rostro alineado en un vector de características discriminativo
- Ambos forman parte del ecosistema InsightFace, facilitando la integración técnica.

3.2.3 Comparación Directa y Justificación de Elección

Criterio	FaceNet	ArcFace
Tipo de pérdida	Triplet Loss (basado en distancia euclidiana entre tripletas de imágenes)	Additive Angular Margin Loss (basado en separación angular entre clases)
Precisión en benchmarks	Alta (LFW ~99.6%)	Muy alta (LFW ~99.8%, mejor en MegaFace e IJB-B)
Separación entre identidades	Basado en distancia euclidiana	Más robusta: separación angular en la hiperesfera
Requisitos de entrenamiento	Conjunto bien estructurado en tripletas (ancor, positivo, negativo)	Necesita identidades etiquetadas (clasificación)
Robustez ante variaciones (luz, expresión, pose)	Alta	Muy alta
Complejidad de implementación	Moderada, ampliamente adoptado	Ligeramente más compleja, pero bien documentada
Facilidad de uso	Hay muchas implementaciones disponibles (Keras, TensorFlow, PyTorch)	También existen implementaciones en PyTorch, especialmente vía InsightFace
Integración con detectores modernos	Compatible con RetinaFace, MTCNN	Altamente compatible con RetinaFace y entorno InsightFace

Dado que nuestro sistema busca validar la identidad de una persona comparando su rostro con el de un documento de identidad (en condiciones variadas de iluminación, calidad y resolución), necesitamos una solución con:

- Alta discriminación entre rostros de distintas personas.
- Robustez ante capturas no controladas.
- Facilidad para obtener un valor de similitud que permita validar si hay match o no.

ArcFace, al operar sobre una hiperesfera angular, ofrece una mejor separación entre clases y una mayor capacidad de generalización que FaceNet. Además, forma parte del

ecosistema InsightFace, junto con RetinaFace, lo cual garantiza compatibilidad, consistencia en el pipeline y rendimiento optimizado.

Por estas razones, ArcFace ha sido seleccionado como nuestro modelo para extracción de características faciales y comparación de similitud, mientras que FaceNet se mantiene como una referencia académica y comparativa en este estudio.

3.2.4. ¿Cómo funciona ArcFace?

- **Contexto general**

El reconocimiento facial basado en redes neuronales profundas se enfrenta a diferentes desafíos como separar correctamente clases (identidades) diferentes, y a su vez agrupar ejemplos de una misma clase en el espacio de representación. Técnicamente hablamos de aumentar la intra-class compactness y la inter-class separability.

ArcFace usa un gran enfoque:

- Pérdidas tipo softmax con penalizaciones geométricas, también conocidas como margin-based softmax methods (SphereFace, CosFace, etc.), que realizan comparaciones muestra-a-clase (sample-to-class) y tienden a ser más estables y eficientes. Estas técnicas han demostrado buen desempeño al introducir márgenes geométricos (angulares o cosenoidal) en la función softmax para maximizar la separación entre clases.

ArcFace propone un margen angular aditivo directamente interpretado en términos de distancia geodésica sobre una hiperesfera, dándole una interpretación geométrica más intuitiva y robusta.

- **Formulación matemática básica**

El reconocimiento facial se puede abordar como un problema de clasificación multiclase, donde cada clase representa una identidad diferente. Dado un embedding (vector de características) extraído de una imagen facial, el objetivo es clasificarlo correctamente en su clase correspondiente (es decir, la identidad correcta). Para ello, los métodos tradicionales suelen usar una función de pérdida basada en softmax combinada con entropía cruzada:

$$L_1 = -\ log\left(\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{N} e^{W_j^T x_i + b_j}}\right)$$

Donde:

- $x_i \in R^d$ es el embedding del rostro de entrada (imagen)
- y_i es la clase (identidad) verdadera
- $W_j \in R^d$ es el vector de pesos asociado a la clase j

- $b_j \in R$ es el sesgo (bias) de la clase j
- N es el número total de clases

- **Limitación del Softmax tradicional**

La formulación no impone restricciones explícitas sobre las propiedades geométricas del espacio de embeddings, pudiendo resultar en:

- Alta variabilidad intra-clase: vectores del mismo sujeto pueden estar muy dispersos
- Separación poco clara entre clases: lo que debilita la discmirniacion entre identidades

ArcFace introduce modificaciones clave que restringen el espacio y mejoran la discriminación para mejorarlo.

- Normalización de embeddings y pesos

Para reducir impacto de magnitudes arbitrarias y centrar la atención en la dirección angular de los vectores, se normalizan tanto los embeddings como los pesos de las clases:

$$\left|\left|W_j\right|\right| = 1 \text{ y } \left|\left|x_i\right|\right| = 1$$

Significa que todos los vectores viven en una hiperesfera unitaria. Adicionalmente, se aplica un factor de escala s para estabilizar el entrenamiento y controlar el rango de las actividades

$$logit\colon\ s \cdot cos(\theta_j)$$

Donde θ_j es el ángulo entre x_i y W_j. El producto entre estos vectores normalizados se convierte en el coseno del ángulo entre ellos.

- **Distribución sobre la hiperesfera**

Después de la normalización y escalado, todos los embeddings están proyectados sobre una hiperesfera de radio s, es decir:

$$\left|\left|x_i\right|\right| = \left|\left|W_j\right|\right| = 1 \Rightarrow x_i, W_j \in S^{d-1}$$

Esto significa que:

- La predicción depende únicamente del ángulo entre el embedding y el vector de clase
- Se elimina el sesgo por norma (magnitud), lo que ayuda a hacer la comparación más justa y robusta

Esta transformación prepara el terreno para introducir un margen angular aditivo, el cual es el core de la función de pérdida propuesta por ArcFace, este margen separa explícitamente las clases en el espacio angular, lo que se traduce en mejores resultados en las verificaciones y reconocimiento facial.

- **Additive Angular Margin Loss (ArcFace)**

ArcFace incorpora un margen angular aditivo dentro de la función softmax, permitiendo la compactación intra-clase y la separación inter-clase en el espacio de embeddings. Esto para garantizar la separación entre clases esté relacionada directamente con la distancia

geodésica en la hiperesfera unitaria (por medir de introducir el margen angular aditivo constante sobre el ángulo entre el embedding y el vector de clase correcta), con una medida de distancia más natural para este tipo de espacio.

- **Formulación matemática de la pérdida ArcFace**

Partiendo del softmax normalizado:

$$L_2 = -\ log\left(\frac{e^{s\cdot cos(\theta_{y_i})}}{e^{s\cdot cos(\theta_{y_i})} + \sum_{j\neq y_i} e^{s\cdot cos(\theta_j)}}\right)$$

ArcFace modifica esto agregando un margen angular m en el ángulo θ_{y_i} correspondiente a la clase verdadera:

$$L_3 = -\ log\left(\frac{e^{s\cdot cos(\theta_{y_i}+m)}}{e^{s\cdot cos(\theta_{y_i}+m)} + \sum_{j\neq y_i} e^{s\cdot cos(\theta_j)}}\right)$$

Donde:

- θ_{y_i} es el ángulo entre el embedding x_i y el vector de la clase verdadera W_{y_i}
- m es el margen angular aditivo (hiperparámetro), por ejemplo, $m = 0.5$
- s es el factor de escala (por ejemplo, $s = 64$), que amplifica los logits

- **Evidencia:**

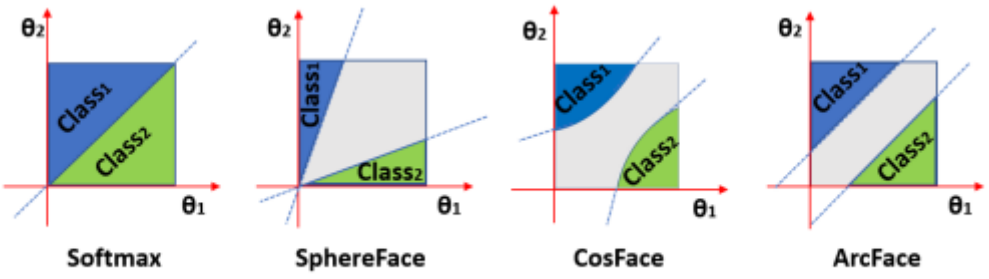
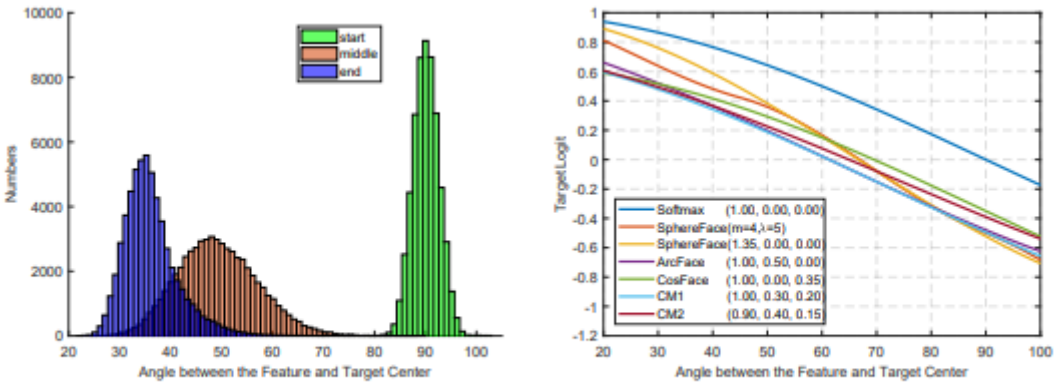


Figura 4. Las curvas de decisión muestran cómo ArcFace produce fronteras más claras y separadas



(a) θ_j Distributions (b) Target Logits Curves

Figura 5. Muestra una disminución progresiva de los ángulos intra.clase con el entrenamiento

La incorporación del margen genera mejores resultados en benchmarks de verificación facial como LFW, MegaFace, etc.

- **Implementación Técnica del Entrenamiento**

Para el entrenamiento de ArcFace esta sigue una estructura estándar de aprendizaje profundo con algunas consideraciones clave para garantizar estabilidad y rendimiento

- **Backbone: ResNet-100**

Este modelo base es usado para extraer los embeddings faciales es una variante profunda de ResNet

- ResNet-100: Red residual con 100 capas, usada por su capacidad para aprender representaciones ricas y profundas sin sufrir del problema del desvanecimiento del gradiente
- Entrenado desde 0 sobre los datasets seleccionados, sin necesidad de transfer learning previo

Esta arquitectura entrega como salida un vector de 512 dimensiones por imagen del rostro, que es el embedding sobre el cual se aplicará la función de pérdida de ArcFace

- **Framework**

- Implementación realizada sobre MXNet, un framework de deep learning optimizado para entrenamiento distribuido.
- El modelo fue entrenado utilizando GPU NVIDIA Tesla P40 (24 GB), con un batch size de 512 imágenes (64 × 8 GPUs en configuración distribuida).

- **Hiperparámetros de ArcFace**

Dos hiperparámetros fundamentales definen el comportamiento de la función de pérdida:

Parámetro	Significado	Valor típico
s (scale)	Factor de escala aplicado a los logits normalizados. Aumenta la separación entre clases.	64
m (margen angular)	Ángulo extra que debe superar el embedding para ser clasificado correctamente.	0.5

Estos valores fueron seleccionados **empíricamente** tras experimentar con distintos márgenes. En implementaciones posteriores, valores como $m = 0.5$ y $s = 64$ se han vuelto estándares por su buen desempeño.

- **Datasets usados para el entrenamiento**

ArcFace se entrenó sobre MS1MV2, una versión balanceada del dataset MS-Celeb-1M, compuesta por:

- >5.8 millones de imágenes
- >85000 identidades únicas

Este dataset fue procesado para eliminar ruido y duplicados, usando estrategias de limpieza semi-automáticas y clustering facial

4. Implementación del sistema

4.1. Lenguaje y entorno de desarrollo

Para la implementación se contempla Python por su amplio ecosistema de librerías para visión por computadora, aprendizaje profundo y manipulación de imágenes. La modularidad y legibilidad del lenguaje facilitan futuras adaptaciones del código (Consideraciones posibles para el sistema en producción).

4.2. Pipeline general del sistema

Se sigue el flujo secuencial:

1. **Carga y validación de entrada:** Se recibe una imagen del rostro y una foto del documento de identidad de la persona.
2. **Detección de rostros:** Se usará RetinaFace para localizar de forma precisa el rostro en cada imagen
3. **Alineación facial:** Los landmarks detectados por RetinaFace permiten el recorte y alineación de rostros antes de la extracción de características.
4. **Extracción de embeddings faciales:** Se usará ArcFace para generar vectores de características normalizadas a partir de las imágenes alineadas.
5. **Cálculo de similitud:** Se compara el embedding del rostro con los embeddings de la imagen del documento usando la distancia coseno

4.3. Tecnologías utilizadas

- **RetinaFace (InsightFace)**: detección de rostros, landmarks 2D y reconstrucción 3D.
- **ArcFace (InsightFace)**: extracción de embeddings con margen angular aditivo.
- **OpenCV**: lectura, transformación y visualización de imágenes.
- **NumPy / SciPy**: operaciones matemáticas y métricas de similitud.
- **PyTorch**: backend utilizado por los modelos preentrenados de InsightFace.

4.4. Hardware y configuración

En nuestro entorno local para la prueba piloto se tienen las siguientes características:

- **CPU:** Intel Core i5-12450HX
- **GPU:** NVIDIA RTX 4050 Laptop (6GB DDR6)
- **RAM:** 24 GB DDR5
- **SO:** Windows 11
- **Lenguaje**: Python 3.~

4.5. Umbral de aceptación

Aún en espera de investigación para definir si se acepta o no se acepta

5. Resultados

Pruebas no realizadas aún

6. Aclaraciones

Las descripciones técnicas, tales como definiciones, fórmulas, imágenes y contextos fueron extraídos de los papers correspondientes a cada modelo (RetinaFace y ArcFace)

Bibliografía

Deng, J., Guo, J., Yang, J., Xue, N., Kotsia, I., & Zafeiriou, S. (2019). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. arXiv preprint arXiv:1801.07698. https://arxiv.org/abs/1801.07698

Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., & Zafeiriou, S. (2020). RetinaFace: Single-stage Dense Face Localisation in the Wild. arXiv preprint arXiv:1905.00641. https://arxiv.org/abs/1905.00641

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. arXiv preprint arXiv:1503.03832. https://arxiv.org/abs/1503.03832

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks (MTCNN). arXiv preprint arXiv:1604.02878. https://arxiv.org/abs/1604.02878

MTCNN Documentation. Introduction to MTCNN. https://mtcnn.readthedocs.io/en/latest/introduction/

InsightFace. Official Project Page for ArcFace and RetinaFace. https://insightface.ai/

Serengil, S. (2022). DeepFace Benchmarks. GitHub Repository. https://github.com/serengil/deepface/tree/master/benchmarks

LFW Dataset Benchmark. Face Identification on LFW. Papers With Code. https://paperswithcode.com/dataset/lfw

MegaFace Benchmark. Face Identification on MegaFace. Papers With Code. https://paperswithcode.com/sota/face-identification-on-megaface

IJB-B Benchmark. Face Recognition on IJB-B. Papers With Code. https://paperswithcode.com/sota/face-recognition-on-ijb-b

ChatGPT (2025). Asistencia en redacción técnica y análisis académico del reconocimiento facial. OpenAI. Participación directa en la redacción, análisis de papers y síntesis del contenido técnico del presente trabajo.