Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

RECONOCIMIENTO FACIAL ONE-SHOT USANDO REDES SIAMESAS PARA CONTROL DE ACCESO

Autor: D.Jaime Rodríguez Cárave

Tutor: Dr. Sergio Luis Toral Marín

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

RECONOCIMIENTO FACIAL ONE-SHOT USANDO REDES SIAMESAS PARA CONTROL DE ACCESO

Autor:
D.Jaime Rodríguez Cárave

Tutor:
Dr. Sergio Luis Toral Marín
Catedrático de Universidad

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Máster: RECONOCIMIENTO FACIAL ONE-SHOT USANDO REDES SIAMESAS PARA CONTROL DE ACCESO

Autor: D.Jaime Rodríguez Cárave
Tutor: Dr. Sergio Luis Toral Marín

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

La primera página de este trabajo fin de máster merecía hacer mención a todas las personas que han creído en mí y hecho posible ser la persona que soy hoy en día.

Papá y mamá, por darme absolutamente todo a lo largo de mi vida, sin vosotros nada de esto sería posible.
Abuelos, por vuestros cuidados desde que era pequeño.
Juli, por ser el mejor hermano que se puede tener.
María, por ser compañera de vida, apoyo y estar en todas mis etapas.
Familia, por acompañarme durante mi camino.
Amigos, por ser diversión, desahogo y consejos en todo momento.
Sergio, por ser la ayuda para realizar este trabajo.

Gracias.

Resumen

El reconocimiento facial engloba multitud de aplicaciones de una importancia vital para el desarrollo de la vida diaria y entornos empresariales. Por esto, en este trabajo la premisa ha sido la elaboración de un control de acceso utilizando esta característica tan única en los seres humanos, el rostro. Para llevarlo a cabo, Python ha sido el lenguaje de programación que engloba todo el proyecto. Las redes neuronales fueron las encargadas de, mediante técnicas de *one-shot learning*, aprender a diferenciar entre sujetos desconocidos a través de sus caras. Así pues, las redes siamesas y la función de pérdidas *Triplet Loss*, fueron las elegidas para realizar esta labor haciendo uso de estructuras tales como ResNet-50, Xception e Inception-ResNet V1. El objetivo que se propuso fue el entrenamiento de estas redes, comparándolas con el rendimiento de una red preentrenada, junto a una posterior validación para el control de acceso en una Raspberry Pi 4 como ejemplo de sistema embebido. Los resultados obtenidos para ambas vertientes demostraron una correcta distinción entre diferentes sujetos, a la vez que se podía autenticar la identidad de una persona comparando su cara con las guardadas en una base de datos, todo esto probando que el reconocimiento facial es una herramienta práctica en la disciplina del control de acceso.

Abstract

In Facial recognition includes several applications with vital importance for the development of the daily and business life. Hence, the main goal in this paper has been to develop an access controller using the humansúnique characteristic, the face. To do so, Python has been the programming language used during the project. Neural networks, using one-shot learning techniques, were responsible of learning to differenciate between unknown subjects through their faces. Thereby, siamese networks and the Triplet Loss function were the chosen ones to do the aforementioned, using structures such us ResNet-50, Xception and Inception-ResNet V1. The goal was to train this networks comparing them with the performance of a already trained network and a later validation for access control in a Raspberry Pi 4 as an example of an embedded system. Results obtained in both cases showed a correct distinction between different subjects and, at the same time, that it was possible to authenticate the personal identity comparing their face with the ones stored in a database. All this, demonstrating that facial recognition is a practical tool in the access control field.

Índice Abreviado

Índice

1 Introducción

El objeto del presente proyecto, es la elaboración de una herramienta cuya función sea la de reconocer la identidad de una persona con la simple adquisición de una imagen de su cara haciendo uso de una cámara. La aplicación principal de esto se centrará en el ámbito de la problemática del control de acceso en lugares que requieran de esta seguridad. Previamente a desarrollar este trabajo, cabría preguntarse, ¿en qué consiste un sistema de control de acceso?

Un sistema de control de acceso se basa en la restricción selectiva de lugares o recursos a los usuarios que requieran la utilización de ellos. Para autorizarles, se crean unas reglas de acceso definidas previamente por el que elabora este sistema. Estas reglas se pueden implementar de forma diversa, dependiendo de la complejidad técnica y el coste económico que pueda afrontar la empresa. Teniendo en cuenta estas dos variables, nos encontramos con, ordenado de menor a mayor dificultad de desarrollar, los tres grandes grupos que forman actualmente el control de acceso:

- Sistemas con teclado externo. Es el más sencillo de todos porque hace uso de una contraseña la cual, al ser introducida correctamente, se autoriza el acceso.
- Sistemas de proximidad. Se basa en la utilización de objetos para una identificación positiva. La principal ventaja es la facilidad de implementarlo y de usarlo, pero el tener que llevar tarjetas u otra clase de elementos trae consigo la posibilidad de una vulnerabilidad ya sea por robo o simplemente por pérdida.
- Sistemas biométricos. Consiste en reconocer una característica física de la persona para garantizar su acceso. Dentro de este grupo nos podemos encontrar con reconocimiento de huella, de iris, de voz o de cara. Es en este último donde se engloba el proyecto a realizar, ya que para la persona es simple puesto que no requiere de hacer ninguna comprobación invasiva ni acción por parte del usuario. Este no sería el caso de la huella, que necesita del contacto con el lector (en periodos de pandemia esto es algo que se intentó evitar), ni el caso del iris, que necesita del acercamiento a la cámara para su correcta realización.

Conociendo las principales vertientes que se encuentran en el mercado para el control de acceso, cabe desarrollar los cuatro principios básicos que lo forman. Estos se pueden visualizar en la Figura 1.1.

1. Identificación. Primer paso que utiliza los métodos anteriores para la identificación del usuario con el fin de determinar si cumple con los requisitos de acceso.
2. Autenticación. Se verifica en la base de datos si la persona en cuestión se encuentra en ella.
3. Autorización. Es el paso donde se autoriza o deniega la entrada al recinto o recurso dependiendo de las reglas establecidas.
4. Trazabilidad. Todos los datos de acceso son almacenados para su posible posterior uso en cuestiones internas de la empresa.

Con la visión general de qué es un sistema de control de acceso, vamos a destacar las ventajas que este puede aportar a la sociedad. La virtud más destacable que ofrece es la seguridad, que se consigue a la hora de denegar la entrada a personas no deseadas en el lugar de aplicación. Además, cabe destacar que todo avance lleva consigo un incremento en la productividad, y este caso no es una excepción: implementando un sistema

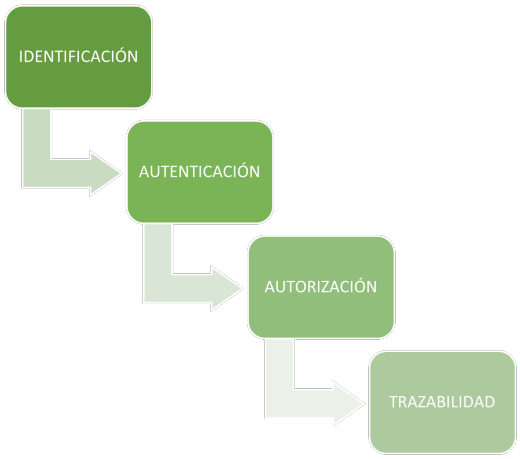


Figura 1.1 Principios básicos control de acceso. Elaboración propia.

de acceso de control electrónico se consigue facilitar la organización empresarial sabiendo quién accede al recinto, conociendo a qué lugares ha intentado entrar, limitando las horas de entrada y salida para personas específicas, cambiando las condiciones para las cuáles se tiene potestad para acceder o no...

Vivimos en un mundo donde la información se ha apoderado de nuestro día a día y nuestro tiempo se vuelve cada vez más valioso. Es por esto que estas técnicas se han hecho un hueco en la sociedad, buscamos protección y aligerar cualquier tarea que resulte tediosa. Aquí es donde entra la función del control de acceso.

Los campos de aplicación son múltiples y por ende, las disciplinas que se encuentran trabajando en él. Las premisas que condicionan la evolución natural de esta tecnología es buscar la facilidad de uso para el usuario siempre asegurando una excelente robustez. Entre las implicaciones más palpables en nuestro día a día que trae consigo esta herramienta podemos destacar:

- La autorización a la hora de entrar en nuestro lugar de trabajo, donde no necesitamos de ninguna contraseña o elemento para poder identificarnos, solo ser reconocidos por una cámara. Con esto se incrementa la celeridad para pasar ciertos controles y se organiza de manera eficiente los privilegios de ciertos usuarios para zonas concretas del edificio.
- El desbloqueo del móvil sin necesidad de utilizar un patrón o contraseña. La tecnología más madura y quizás más impresionante para los usuarios es el *Face ID* de la empresa Apple (Figura 1.2). Con una velocidad sorprendente se consigue desbloquear el teléfono e incluso autorizar pagos bancarios con la simple detección de la cara de la persona.

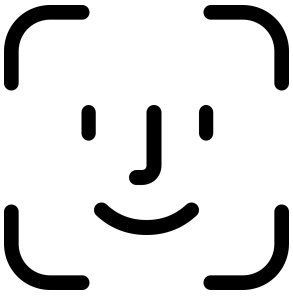


Figura 1.2 Face ID de Apple.(Apple, 2017).

Una vez desarrollado los aspectos generales y las bondades que trae consigo el control de acceso haciendo uso del reconocimiento facial, en los siguientes capítulos se ahondará en el estado del arte de esta tecnología y el trabajo a realizar para demostrar la valía de esta.

2 Estado del arte

Este trabajo se centra en poder reconocer la identidad de un sujeto a través de su cara. Aunque este sea el objetivo principal, cabe destacar que existen otras técnicas para conseguir tal propósito.

Como se comenta en (Venayagamoorthy y cols., 1998, p. 29), la biometría (del griego *bios* vida y *metron* medida) se utiliza para detectar rasgos fisiológicos o incluso de comportamiento únicos en el usuario para poder así identificarlo inequívocamente.

Tal y como expresa (Khosla, 2010, pp. 852-854), en toda seguridad biométrica dos son los procesos que tienen lugar: la adquisición de la característica del sujeto y la autenticación de esa información comparándola con una base de datos. En general, y para que quede claro el concepto genérico de cualquier seguridad de este tipo, se refleja el esquema a seguir en la Figura 2.1, donde queda constancia del proceso de adquisición de la información, su posterior tratamiento y el paso final de verificación y toma de decisión.

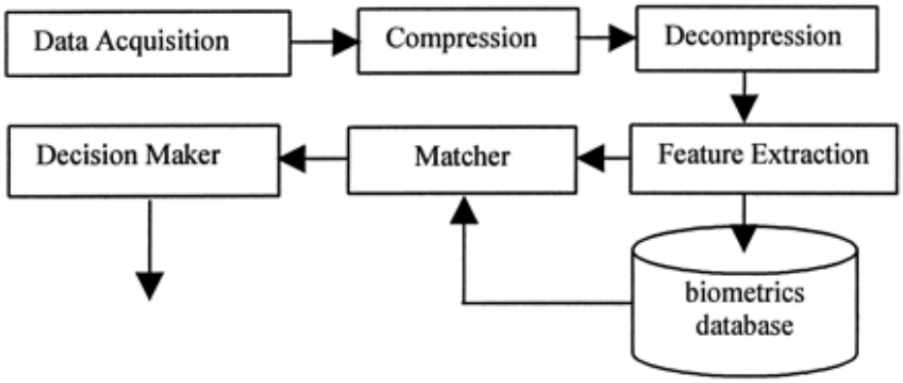


Figura 2.1 Esquema genérico sistema de seguridad biométrico. (Khosla, 2010).

Una de las numerosas maneras de abordar la identificación de un usuario utilizando biometría es mediante la huella dactilar. Numerosos han sido los esfuerzos en esta materia para innovar y aplicar en múltiples campos de aplicación. Entre ellos podemos destacar (Valdes-Ramirez y cols., 2019), realizando un estudio sobre las posibles maneras de extraer las características de las huellas latentes (las encontradas en el lugar donde un delito ha sido cometido) y sus posibles aplicaciones; el artículo de (Bawarith, Basuhail, Fattouh, y Gamalel-Din, 2017), cuyo esfuerzo se focaliza en evitar el fraude en exámenes online (algo muy necesario actualmente debido a los cambios sufridos en la educación por causa de la pandemia); el trabajo de (Yang, Wang, Hu, Zheng, y Valli, 2019) que se centra en las vulnerabilidades de seguridad que un sistema biométrico, en este caso de huella dactilar, pudiera tener, como sería una huella falsa que sea reconocida por el sensor o ataques cibernéticos a la base de datos o al sistema de comunicaciones.

Otra alternativa dentro del mundo de la biometría es el caso de la voz. En (Rashid, Mahalin, Sarijari, y Aziz, 2008), es esta el hilo conductor para llevar a cabo la labor del control de acceso. Gracias al efervescente crecimiento de tecnologías como procesamiento de señales y la inteligencia artificial, en este trabajo se plantea remplazar la huella dactilar como vehículo biométrico por la autenticación de la persona mediante los patrones de su voz.

Volviendo al punto central de este proyecto, una de las más recientes propuestas en el ámbito del control de acceso es mediante el reconocimiento facial. En este caso dejamos de lado las características personales como pueden ser la huella dactilar y la voz anteriormente mencionadas, para concentrarnos en la cara del usuario. En (Baron, 1981) se encuentra un extenso estudio de la conexión existente del cerebro humano y los mecanismos que posee para poder interactuar con su entorno, con vista a poder reconocer sujetos viendo sus caras. Complejidad a nivel del cerebro y neuronas son tratados en éste.

Por otra parte, enfocando a una esfera más práctica, en (del Rio, Moctezuma, Conde, de Diego, y Cabello, 2016) se plantea emplear esta herramienta para la seguridad en los aeropuertos haciendo uso de los Automated border control (ABC). Para abordarlo, se necesita de imágenes en alta calidad y la utilización de algoritmos que no sean sensibles a condiciones no ideales como la postura de la persona, la iluminación o el uso de objetos que tapen parcialmente la cara. En (Carlos-Roca, Torres, y Tena, 2018) el proyecto diseña una herramienta que facilite la labor a los guardias del control de seguridad llamada *iBorderCtrl* cuya estructura se refleja en la Figura 2.2. El fundamento básico es un registro previo del usuario donde se toman imágenes de su cara y luego una comprobación triple cuando se intenta pasar el control. Primero, se comprueba que el usuario que quiere pasar el control, tomando fotos en el momento, concuerda con el mostrado en el pasaporte; segundo, se verifica que la persona del pasaporte se corresponda con la que hizo el registro previo; tercero, se vuelve a comprobar que el usuario es el mismo al que se tomó las imágenes en el registro. Todas estas comprobaciones en cascada evalúan el riesgo de llegar a permitir a una persona equivocada pasar la seguridad.

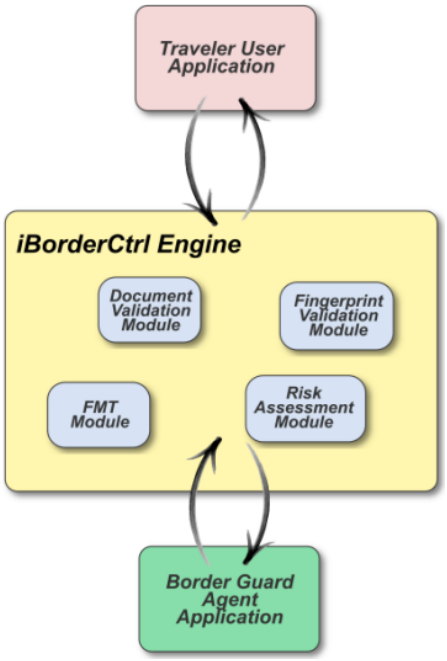


Figura 2.2 Estructura de iBorderCtrl. (Carlos-Roca y cols., 2018).

Todo lo aprendido en el tratamiento de las características de la cara han servido para aplicarlo en temas de salud. Como expresa (Hurst, 2018), se podría usar dentro del campo de la dismorfología (estudio de malformaciones humanas o defectos congénitos) para complementar el juicio clínico dictado por el profesional sanitario en cuestión. Calcular las distancias existentes entre rasgos faciales del paciente podría reflejar ciertos patrones que se traducirían en la probabilidad de desarrollar una enfermedad. Además, otro ejemplo de la gran utilidad que esta herramienta puede ofrecer se encuentra en (Jayanthy, Anishkka, Deepthi, y Janani, 2019). Mediante la identificación facial el acceder al registro médico se facilita enormemente. La mayor ventaja de esto no es sólo por temas de comodidad y reducción de tiempos de espera, sino que podría llegar a ser crucial en situaciones de emergencia donde demorar el acceso a los antecedentes médicos del paciente se vuelve cuestión de vida o muerte, por ejemplo, en un accidente.

2.1 Detección de caras

Hasta este punto, hemos hecho un recorrido por el campo de la biometría, destacando algunas técnicas y sus aplicaciones. Sabiendo que nuestro cometido es el de detectar la cara de una persona y, posteriormente, identificar el sujeto, cabe plantearse cuáles son las herramientas con las que contamos para acometer el primer paso. Para realizar esto, numerosos han sido los avances en esta materia, todos englobadas bajo el título inglés de *Facial detection*. Como preámbulo, se refleja en la Figura 2.3 un resumen esquematizado de los métodos existentes. Las dos grandes categorías en las que se divide esta disciplina son *Feature base* e *Image base*. Como se describe en (Kumar, Kaur, y Kumar, 2019), la diferencia entre estas reside en la forma de dar solución al problema de la localización y detección de una cara.

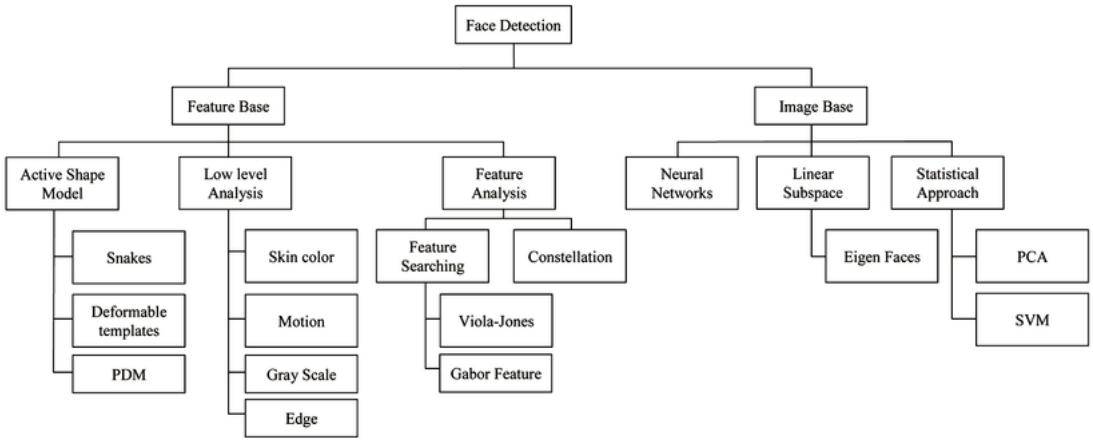


Figura 2.3 Esquema métodos de detección de caras. (Hatem y cols., 2015).

- *Feature base*. Se extraen y analizan características faciales del ser humano tales como color de piel, nariz, orejas..., comparándolos con conocimiento teórico de la distribución de estos en la cara de una persona. Se busca discernir entre rasgos faciales y rasgos que no lo son, el problema es que los algoritmos son sensibles a la iluminación y al ruido.
- *Image base*. Haciendo uso de imágenes de entrenamiento y test, se busca con técnicas de análisis estadístico y *machine learning* (ML), encontrar características relevantes para optimizar la detección. Es aquí donde nos encontramos métodos como *principal component analysis* (PCA) y las Neural network (NN).

En las siguientes líneas se presentarán los algoritmos más relevantes y utilizados dentro de este campo en la actualidad por su desempeño tanto por su precisión como su ligereza computacional.

2.1.1 Principal component analysis

Principal component analysis (PCA) deriva del trabajo realizado en (Turk y Pentland, 1991), donde se presentaron los *eigenfaces*, que eran características (que no tenían que ser rasgos faciales) relevantes para el problema analizado. PCA evolucionó en una técnica que involucraba una serie de operaciones matemáticas que buscaban transformar variables correlacionadas en un conjunto menor de variables no correlacionadas para facilitar la carga de trabajo (Barnouti, Al-dabbagh, y Al-bamarni, 2016). Esto se consigue proyectando en un espacio de dimensión menor que el de la propia imagen la información considerada importante.

En general, este acercamiento en la detección de caras resulta de una mayor celeridad y de no buscar conceptos tan triviales como puede llegar a ser la distancia entre diferentes rasgos de la cara de una persona.

2.1.2 Viola Jones (Haar-Cascade)

Uno de los trabajos más revolucionarios fue llevado a cabo en (Viola y Jones, 2001). Hoy en día se sigue utilizando debido a su bajo coste computacional y su buena precisión, método conocido como clasificación Haar-Cascade.

> La técnica cascada se basa en la concatenación de varios clasificadores débiles, cada uno analizando una porción diferente de una imagen o frame en el caso de vídeo. Se consideran débiles porque tienen alta probabilidad de dar falso positivo, pero cuando se combinan los resultados, en conjunto, por el contrario, son muy potentes. (Ángel y Ambrogio, 2020, p.66)

Pero, ¿cómo se analizan cada región de una imagen con vista a detectar caras? Esto se consigue haciendo uso de los llamados *Haar-like features*. Estos son una evolución de la alternativa que propusieron (Papageorgiou, Oren, y Poggio, 1998) sobre los *Haar wavelets* (Mallat, 1989). Primero el algoritmo transforma la imagen a la escala de grises para aligerar los cálculos. Luego aplica los ya mencionados *Haar-like features*, que son regiones rectangulares que se van a aplicando a las regiones de la imagen para comparar la relación entre los píxeles, especialmente la diferencia de contraste entre estos. Existen tres tipos dependiendo de la característica que se busque en el rostro: de borde, de línea o de centro (Figura 2.4).

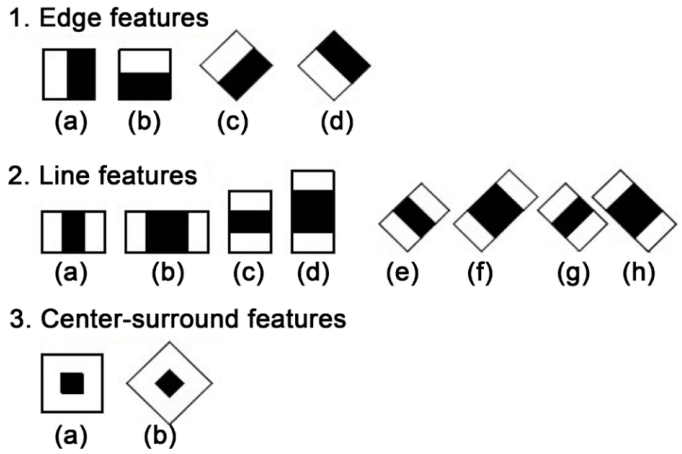


Figura 2.4 Haar-like Features. (X. Zhang y cols., 2017).

Con todo esto, cabría recorrer la imagen aplicando esta serie de *features* con el objetivo de detectar rasgos faciales reconocibles. Por ejemplo, para detectar los ojos, bastaría con aplicar el *Haar-like features* 1.b, colocando el lado oscuro sobre los ojos y el lado claro sobre el párpado inferior (Figura 2.5). Con aplicar una sola vez no sería adecuado porque no se conseguiría un algoritmo fiable. Para solventar esto de forma óptima, se aplica lo que se comentó al principio de esta sección sobre este método, un clasificador en cascada. Lo que se hace es pasar diferentes etapas por cada región de la imagen: si se diera el caso que fuera satisfactoria cada una de estas etapas, se podría afirmar que la región analizada es una cara. El flujo de trabajo de este clasificador se puede ver en la Figura 2.6.

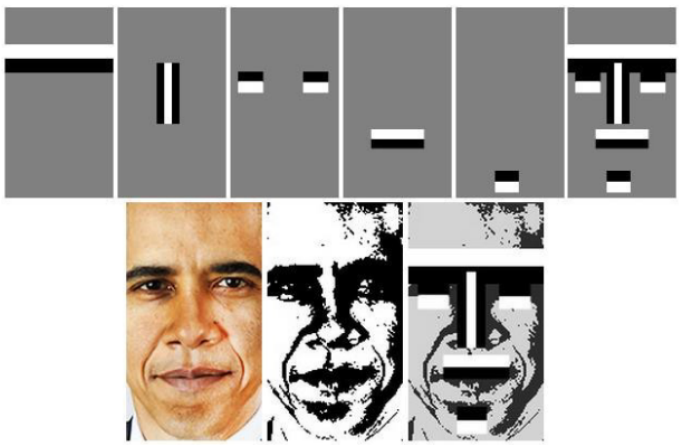


Figura 2.5 Aplicación Haar-like Features. (Kadir y cols., 2015).

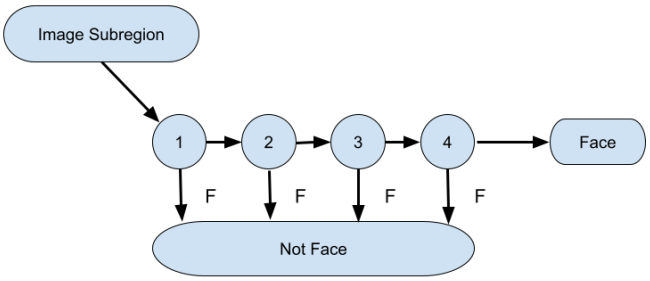


Figura 2.6 Clasificador en cascada. (Tyagi, 2021).

2.1.3 Neural network

La gran revolución llegó en la ciencia de la detección de caras con la aparición de las NN. Un primer acercamiento en relación con esta materia se presentó en (Rowley, Baluja, y Kanade, 1998) en la que, gracias a las bondades que estas redes ofrecen, se consiguió discernir con gran precisión caras en una imagen. Esto se pudo llevar a cabo con una estructura dividida en dos etapas. Una primera que se centraba en aplicar filtros a regiones de la imagen (región de 20x20) cuya salida era una evaluación de lo probable que era que hubiera una cara. Una segunda etapa consiste en observar las regiones que se solapan y unirlas. Cuantas más regiones se solapen mayor es la probabilidad de que sea una cara la región unificada. Para evitar falsos positivos y conseguir una red robusta, se establece un umbral por el cual se admite el resultado. En la investigación de (H. Li, Lin, Shen, Brandt, y Hua, 2015) se utilizan varias Convolutional neural network (CNN) en cascada para obtener un detector de caras que, al trabajar en una primera etapa con las imágenes en baja resolución, de manera rápida rechaza las regiones que estima que no existen caras, mientras que en las regiones donde sí las evalúa en resolución mayor para trabajar de manera óptima con ellas.

Multitask Cascaded Convolutional Networks

Multitask Cascaded Convolutional Networks (MTCNN) fue una propuesta llevada a cabo en (K. Zhang, Zhang, Li, y Qiao, 2016) donde se hacía uso de CNN en cascada divididas en tres etapas diferentes. Tal y como se comenta en (Ma, Fan, Lu, y Tian, 2020), las tres etapas llevan el nombre de P-Net, R-Net and O-Net (Figura 2.7). La primera es una red completamente convolucional, donde se extrae las regiones de caras candidatas, mientras que las dos siguientes son CNN clásicas que refinan la selección rechazando falsos candidatos y devolviendo el marco donde la cara se encuentra y la localización de cinco puntos de refencia faciales (ojos, nariz y extremos de la boca).

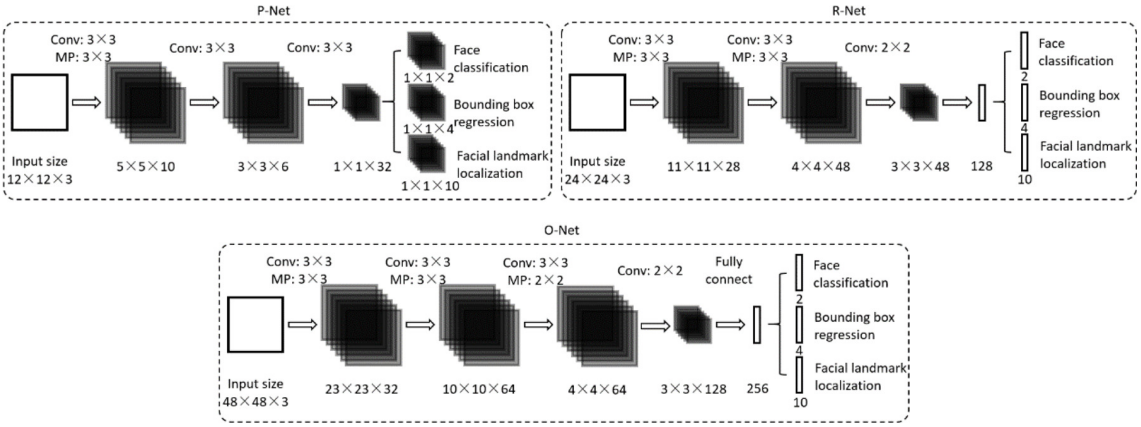


Figura 2.7 Etapas de MTCNN. (Ma y cols., 2020).

Por tanto, con MTCNN nos encontramos con una solución adicional a la propuesta por Viola Jones en anteriores líneas. Cabría preguntarse cuáles son las diferencias más palpables entre ambas técnicas, pudiendo destacar las ilustradas en la Tabla 2.1

Tabla 2.1 Comparación entre Viola Jones y MTCNN. (Adamczyk, 2021).

Característica	Viola Jones	MTCNN
Velocidad	Muy rápida (>30 FPS)	Rápida (>10 FPS)
Precisión	Buena	Muy buena
Robustez	Mala	Muy buena
Uso GPU	En algunas aplicaciones (no OpenCV)	Sí
Uso de color	No	Sí

YOLOv3

La base de la red neuronal YOLOv3 fue realizada por (Redmon, Divvala, Girshick, y Farhadi, 2015) con el nombre de You Only Look Once. Mientras que otras redes como MTCNN se basaban en entregar a la red región por región de una misma fotografía (es decir, la red no veía la imagen al completo), en YOLO la misma red ve toda la imagen, y es ella misma la que crea un cuadrado delimitador en los objetos de interés con el nombre de su clase y la confianza estimada. Cabe destacar que YOLO es capaz de detectar numerosos elementos diferentes. El método para conseguir esto se ilustra en la Figura 2.8. YOLO coge la imagen y la divide en un mallado de medida YxY (número de cuadrículas), en el que cada una de estas cuadrículas contiene x cuadrados limitadores. Cuadrados próximos que tengan una confianza alta y señalen a la misma clase, se juntan formando la detección final del objeto. Esto consigue que YOLO en velocidad permita órdenes de magnitud superiores a las anteriores soluciones que se basaban en regiones.

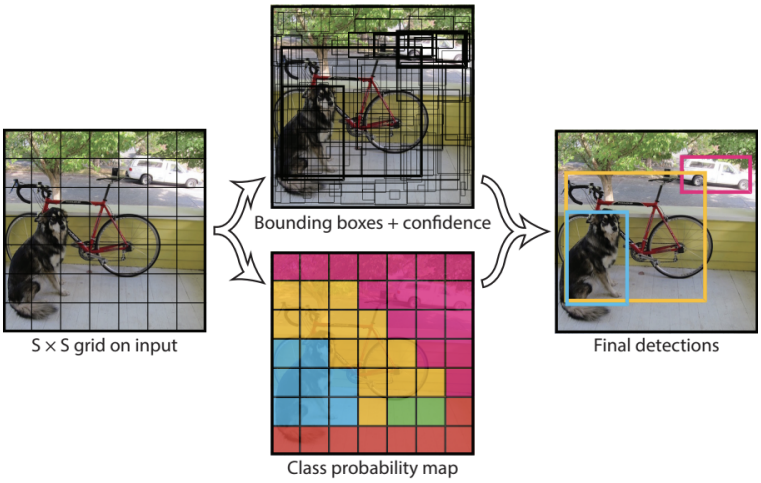


Figura 2.8 Flujo detección por parte de YOLO. (Redmon y cols., 2015).

La evolución que tuvo YOLO surgió con el trabajo de (Redmon y Farhadi, 2018). Aquí fue donde surgió la alternativa de YOLOv3 cuya diferencia principal residía en resolver problemas que tenía la red original para detectar objetos que estuvieran muy próximos unos de otros. Además, esta versión mejorada se comporta mejor en cuánto a velocidad y tiene una precisión equiparable a YOLO.

En (C. Li, Wang, Li, y Fei, 2020) podemos ver una aplicación de YOLOv3 a la detección de caras. En esta reentrenaron YOLOv3 implementado ligeros cambios para que su objetivo principal fuera discernir las caras en una imagen. Se consiguió aumentar en un 3.7 % la precisión que había mostrado MTCNN en la base de datos FDBB (Jain y Learned-Miller, 2010). Finalmente, en la Tabla 2.2 se van a ilustrar las ventajas y desventajas de los algoritmos mencionados.

Tabla 2.2 Comparativa diferentes algoritmos detección cara. Elaboración propia.

Algoritmos	Ventajas	Desventajas
Haar-Cascade	Rapidez	Mal rendimiento con poca luz o poses diferentes
MTCNN	Mayor precisión que Haar-Cascade	Alto coste computacional
YOLOv3	Gran balance entre velocidad y precisión	Red neuronal muy profunda

2.2 Reconocimiento facial

Una vez estudiado las herramientas que se tienen para poder detectar caras en una imagen, cabe avanzar al siguiente paso que nos concierne, cómo identificar sujetos con esas caras obtenidas. La estructura que se planteó anteriormente en la Figura 2.1 recibe ligeros cambios en esta casuística, siendo reflejados en la Figura 2.9. La novedad es que la información que adquirimos y procesamos en nuestro caso es el rostro del sujeto.

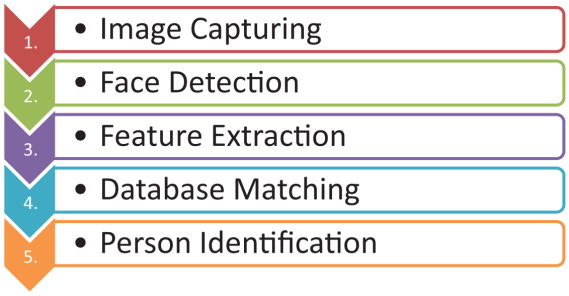


Figura 2.9 Flujo de trabajo en reconocimiento facial. (Kaur y cols., 2020).

2.2.1 Eigenfaces

Como se describió anteriormente, los *Eigenfaces* derivan del PCA. Se busca economizar los datos, es decir, reducir la dimensionalidad de la información proyectando en un espacio menor. Esto se consigue calculando los autovectores (*Eigenvectors*, llamados en reconocimiento facial *Eigenfaces*) de la matriz de covarianza. Por tanto, se toma un número de imágenes, calculando sus *Eigenfaces* y, tomando como referencia de espacio los que tengan los autovalores mayores (*Eigenvalues*), se proyectan el resto de caras y se guardan sus coeficientes. Para comparar caras de sujetos, se mide la distancia entre los coeficientes resultantes de la proyección de sus rostros según los *Eigenfaces* elegidos (Imran, Rahman, y Karmaker, 2015). Estos últimos son los que hacen posible pasar de un espacio de tamaño M (dimensión de la imagen de entrada) a uno menor P (dimensión de los *Eigenfaces*)

2.2.2 Fisherfaces

Un problema que presenta el anterior método es la sensibilidad ante cambios de iluminación y expresión facial. Es por esto que apareció el trabajo de (Belhumeur, Hespanha, y Kriegman, 1997) donde se mostraron los *Fisherfaces*. Esta técnica también se basa en la reducción de dimensiones, pero se le añade el uso de Linear Discriminant Analysis (LDA) para complementar su desempeño. Esta técnica que trata de maximizar la separación de clases diferentes a la vez que minimiza la distancia entre componentes de una misma clase. El problema que trae consigo PCA es que busca maximizar la dispersión de todas las imágenes en su conjunto, y esto trae consigo retener transformaciones que incluyen situaciones lumínicas y poses muy diferentes. Es una técnica supervisada ya que se requiere de aportar las clases de las imágenes.

2.2.3 Laplacianfaces

Es un trabajo que sigue avanzando el estado de arte en comparación con las dos técnicas anteriores (X. He, Yan, Hu, Niyogi, y Zhang, 2005). Utilizando Locality Preserving Projections (LPP), las imágenes son mapeadas en un subespacio que retiene la información local extraída de las fotografías de entrenamiento y preserva la estructura compleja de la cara. Aun siendo un método no supervisado, gracias a la inclusión de matriz de adyacencia se consigue grandes resultados al proyectar los nuevos rostros en el subespacio construido. Los resultados de la aplicación de los tres métodos descritos se reflejan en la Figura 2.10.

2.2.4 Modelos basados en redes neuronales

Para llevar a cabo esta compleja tarea, aparecerán las NN para poder facilitar su realización. Es por esto por lo que a lo largo de las siguientes líneas se mencionarán los modelos más destacados en reconocimiento de objetos que pueden extrapolarse a detección de caras.

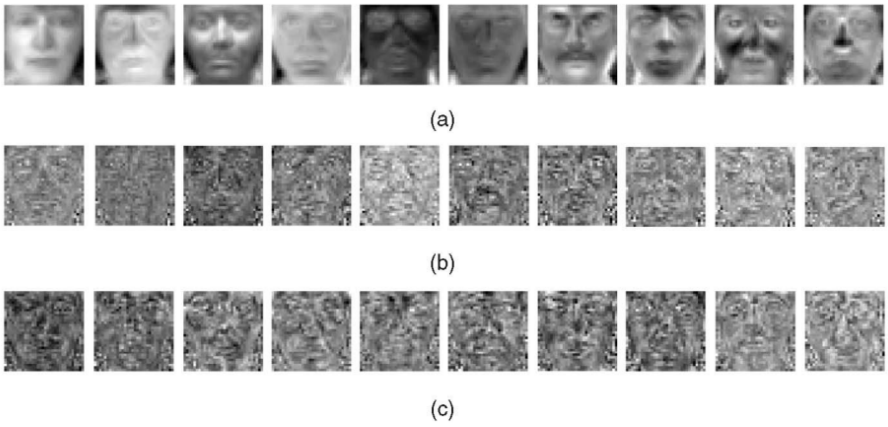


Figura 2.10 a) Eigenfaces b) Fisherfaces c) Laplacianfaces. (X. He y cols., 2005).

VGGNet, propuesto en (Simonyan y Zisserman, 2014), se caracteriza por su simpleza. En cada capa va utilizando capas convolucionales de tamaño 3x3 seguido de un *max pooling*. En las últimas capas lo que se encuentra son capas enteramente conectadas para dar paso a la clasificación final. Hay diferentes variantes de esta red como pueden ser la VGG16 y VGG19 que se diferencian en la cantidad de capas (teniendo la VGG19 más capas convolucionales). En la Figura 2.11 se puede ver la estructura definida. Para el entrenamiento original lo que se optó fue por entrenar los modelos más pequeños para que luego sirvieran de iniciadores a redes más complejas.

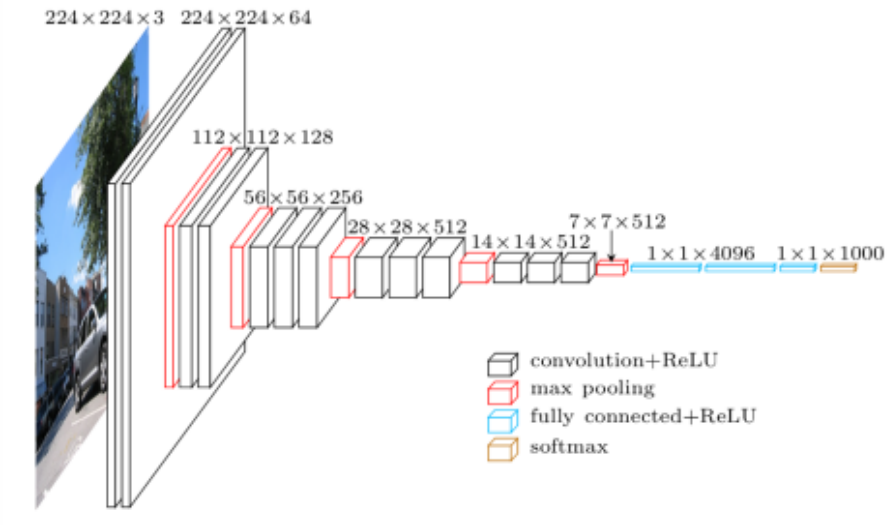


Figura 2.11 Estructura de VGG. (Frossard, 2016).

A diferencia de las redes que se han ido presentando en este trabajo, ResNet (K. He, Zhang, Ren, y Sun, 2015) introdujo una novedad que la hacía especial dentro del mundo del Deep Learning. Dejó de lado el concepto secuencial que estaba impuesto y añadió pequeños módulos que se iban juntando para formar la red. Esto vino impulsado por las redes profundas con muchas capas: llega un punto que se saturan y ya no aprenden más por muchos parámetros que tengan a su disposición para dar solución al problema. Es esta la finalidad de los módulos residuales (Figura 2.12), saltarse algunas capas que considere poco útiles para cambiar la forma en la que se calcula el gradiente y optimizar el aprendizaje, siendo en definitiva como atajos.

Inception (Szegedy y cols., 2015) surgió para aligerar la carga computacional que traían consigo redes muy profundas. Se pensó que una opción sería, en vez de hacerlas cada vez más profundas, podían hacerlas más amplias. Es aquí donde surge el concepto de *Inception module* (Figura 2.13). Lo que se busca es hacer uso de convolucionales de tamaño 1x1, 3x3 y 5x5 para limitar el número de canales de entrada, asegurando el aligerar la red. Una versión alternativa para reducir aún más y limitar el tamaño se propuso aplicando antes

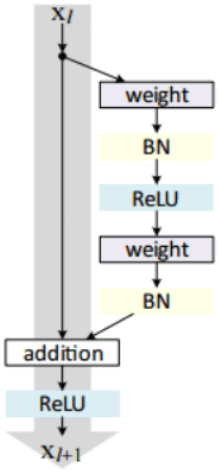
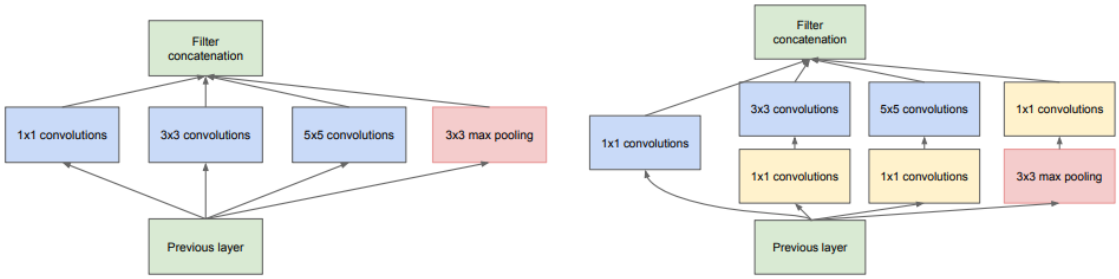


Figura 2.12 Módulo residual en Resnet. (Rosebrock, 2017).

de cada convolucional otra igual, pero de tamaño 1x1 que son los más sencillos de calcular. Por ejemplo, usando 30 filtros 1x1 sobre una entrada 80x80x60, se compactaría a 80x80x30. Procediendo con los otros diferentes filtros se obtendría una información igualmente profunda pero mucho más amplia apilada.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figura 2.13 Inception module. (Szegedy y cols., 2015).

Xception (Chollet, 2017) apareció mejorando el rendimiento de Inception cambiando los convolucionales por unas formas de cálculo llamadas *depthwise separable convolution*. Estos se componían de dos etapas:

1. *Depthwise Convolution*. Se recibe la matriz de entrada con el número de canales y lo que se busca es aplicar un filtro a cada dimensión para no alterar la dimensión original. Si tuviéramos una entrada 12x12x3, aplicando 3 filtros de 5x5 a cada dimensión independientemente daría lugar a una salida de 8x8x3 (Figura 2.14)

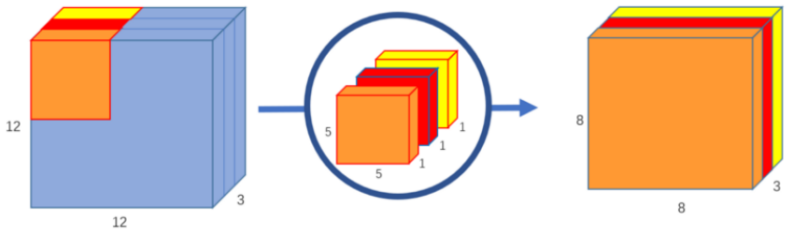


Figura 2.14 Depthwise Convolution. (C.-F. Wang, 2018).

2. *Pointwise Convolution*. Lo que se busca es aplicar un filtro 1x1 a través de todos los canales de la imagen teniendo la profundidad igual a la dimensión de la entrada. Por tanto, volviendo al caso anterior donde nuestro resultado fue 8x8x3, el resultado sería de 8x8x1 iterando a través de todas los canales.

Si quisiéramos expandir esta matriz de salida, sólo habría que utilizar el número de filtros deseados y compactarlos en una única salida (Figura 2.15).

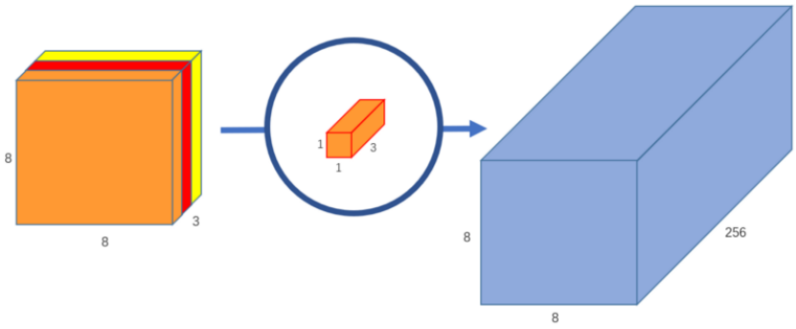


Figura 2.15 Pointwise Convolution. (C.-F. Wang, 2018).

DenseNet (G. Huang, Liu, van der Maaten, y Weinberger, 2017) toma lo aprendido en ResNet y lo lleva más lejos. Cuando hablábamos sobre los módulos residuales, se asemejaban a atajos que se hacían entre capas, pero era únicamente de una capa a otra. En DenseNet se adquiere este mismo concepto, pero lo expande a todas las capas, cada capa recibe la entrada de las capas predecesoras como se puede ver en la Figura 2.16. Por tanto, al recibir toda esta información ya no es necesario hacer redes tan profundas y pueden ser más compactas.

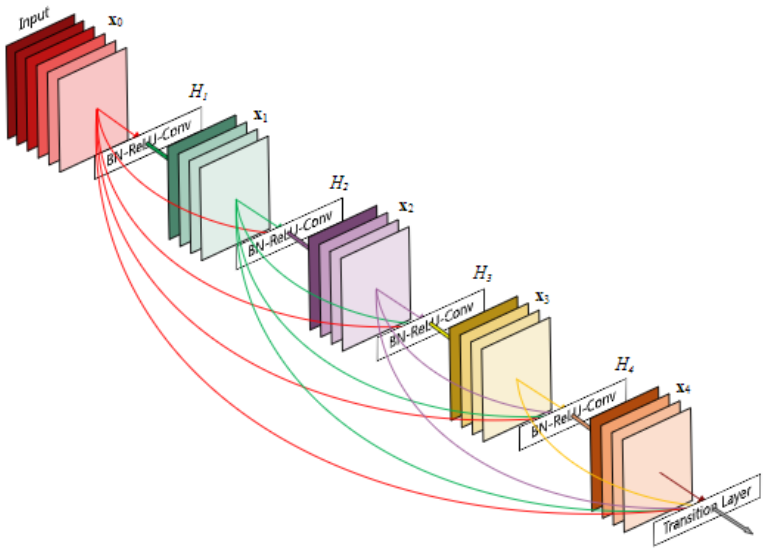


Figura 2.16 Estructura DenseNet . (G. Huang y cols., 2017).

One-shot learning

Hemos realizado una revisión general de los procesos que se pueden utilizar para realizar diferentes tareas, como podía ser discernir si en una foto había cara o no, localizarla e incluso llegar a predecir el sujeto que es cuando nuestra red está entrenada con imágenes de esa persona. Pero, ¿qué pasa cuando aparece un individuo que nuestra red nunca ha visto anteriormente?

Esto para el ser humano es una tarea relativamente sencilla, teniendo a una persona delante y una foto de ella en otro momento de su vida podemos determinar con cierta precisión si es el mismo individuo. Es una tarea que se realiza diariamente en situaciones como controles de seguridad (aeropuertos, eventos especiales, controles fronterizos...) o simplemente demostrando la identidad a la hora de recibir un envío importante.

Sin embargo, este proceso mental que al ser humano le puede parecer simple, a las NN no ya que son simples algoritmos entrenados con una base de datos específica: si el algoritmo está especializado en la

detección de *n* clases, son esas *n* con las que puede trabajar con mejor o menor desempeño. Como se comenta en (Chanda y cols., 2019), añadir un sujeto más al abanico de clases requeriría el reentreno de toda la red, con el gasto en tiempo y en procesamiento que conllevaría. Esta es la razón de ser del concepto de *one-shot learning*.

Está nueva forma de atajar el problema del reconocimiento facial se basa en dejar de lado el acercamiento de clasificar por uno nuevo encaminado a evaluar las diferencias. Si la red está entrenada para detectar diferencias entre caras y no para predecir la identidad de la persona que ve la NN, no es necesario que esta hubiera visto en su fase de entreno a este nuevo individuo, únicamente necesitaría algo con lo que comparar esta nueva cara (Figura 2.17). Por esto aparecieron las redes siamesas.

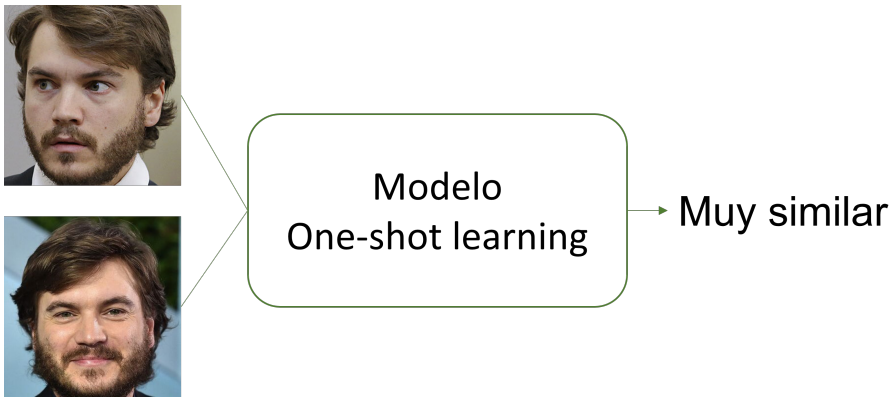


Figura 2.17 Ejemplo One-shot learning . Elaboración propia.

Redes Siamesas

Una red siamesa (Siamese Neural Network (SNN)) consiste en una NN cuya estructura está replicada idénticamente, es decir, está formado por el mismo modelo con idénticos pesos. En la fase de entrenamiento, ambas redes comparten los pesos que van actualizando y son afectados por la misma *backpropagation*. La salida que sacan son los *embeddings* (la representación vectorial) de las entradas para finalmente compararlas entre ellas y sacar la similitud que tienen (Chicco, 2021). Para clarificar el concepto descrito, se refleja en la Figura 2.18 la estructura que este tipo de red posee.

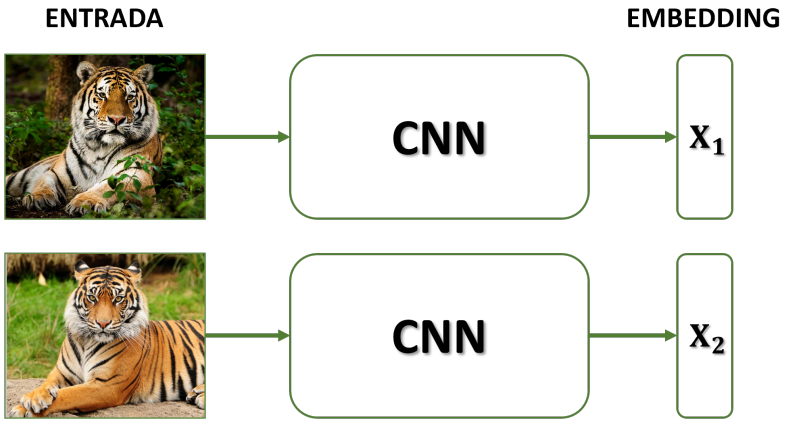


Figura 2.18 Estructura de SNN. Elaboración propia.

Buscamos que imágenes del mismo objeto presenten *embeddings* similares mientras que clases diferentes se alejen lo máximo posible. Este parámetro comparador se puede calcular mediante:

- Distancia euclídea. Es la longitud del segmento que une dos puntos en el espacio euclídeo.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{2.1}$$

- Similitud coseno. El intervalo en el que trabaja es [-1,1]: dos vectores proporcionales darían como resultado 1, dos ortogonales 0 y dos apuntado al sentido contrario -1.

$$\cos(\theta) = \frac{A \cdot B}{\|A\|_2 \|B\|_2} \tag{2.2}$$

Pero, ¿cómo entrenamos la red para que estos vectores se alejen entre sí o se acerquen dependiendo de lo que requiera la situación? La respuesta se encuentra en la función de pérdidas establecida en el modelo para optimizar la red, la cual es vital en las SNN. Una propuesta de función es la llamada *Triplet Loss* que se va a explicar en las siguientes líneas.

Triplet Loss

Triplet Loss fue introducido por primera vez en (Schroff, Kalenichenko, y Philbin, 2015). Necesita de la generación de un triplete de imágenes (γ) donde dos fotografías sean de la misma persona (ancla y positivo) y otra de un sujeto diferente (negativo). Con este lote de imágenes se intenta maximizar la distancia entre el ancla y el negativo y minimizar entre el ancla y el positivo en el entrenamiento de la red (Figura 2.19).

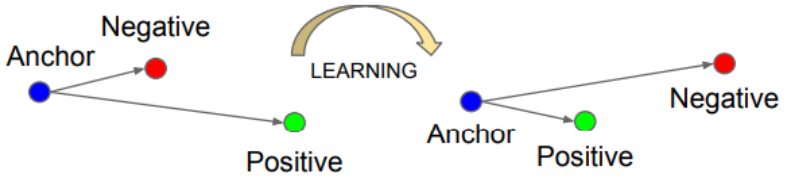


Figura 2.19 Concepto de Triplet Loss. (Schroff y cols., 2015).

Esto matemáticamente se traduce en la Fórmula 2.3. Se busca que la distancia positiva (entendida como la dada entre el ancla y el positivo) sea menor a la negativa (imagen ancla y negativa).

$$\left\|x_i^a - x_i^p\right\|_2^2 < \|x_i^a - x_i^n\|_2^2, \forall (x_i^a, x_i^p, x_i^n) \in \gamma \tag{2.3}$$

A esta expresión se le añade un término, α (Fórmula 2.3), llamado margen, cuyo propósito es el de intensificar aún más esa desigualdad y conseguir alejar las clases diferentes (un valor típico suele ser 0.2).

$$\left\|x_i^a - x_i^p\right\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2, \forall (x_i^a, x_i^p, x_i^n) \in \gamma \tag{2.4}$$

Por tanto, solamente queda definir la función de pérdidas que la SNN adoptará para conseguir discernir mediante los *embeddings* caras de un mismo sujeto. Esta función queda definida en la Fórmula 2.5.

$$Loss = \sum_i^N [\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha] \tag{2.5}$$

Un problema que trae consigo *Triplet Loss* es la convergencia de la red, es díficil de entrenar. Esto es debido a que entrenamos con tripletes de imágenes, algo muy costoso computacionalmente, y, además, tal y como hemos definido la función de pérdidas, puede llegar un momento que la imagen negativa sea tan lejana a la clase ancla que la función se vuelva 0 y la red deje de aprender. Esto ocurre si los tripletes se eligen de manera trivial, pero recientes trabajos en esta disciplina han apostado por estrategias para evitar esto como en (Moindrot, 2018).

Este enfoque se llama *triplet mining* y su principio es el de no coger tripletes aleatorios, sino construirlos siguiendo un principio. Se definen tres tipos de tripletes (Figura 2.20):

1. Tripletes fáciles. Son los que la función de pérdida es igual a 0, $d(a,p) + \alpha < d(a,n)$
2. Tripletes difíciles. Se da cuando la imagen negativa está más cerca a la ancla que la positiva, $d(a,n) < d(a,p)$
3. Tripletes semi-difíciles. Es una mezcla de ambos, y el que se ha demostrado que da mejores resultados. La imagen no está más cerca del ancla que la positiva como pasaba en el triplete difícil, pero aun así tiene un valor de pérdida positivo, $d(a,p) < d(a,n) < d(a,p) + \alpha$

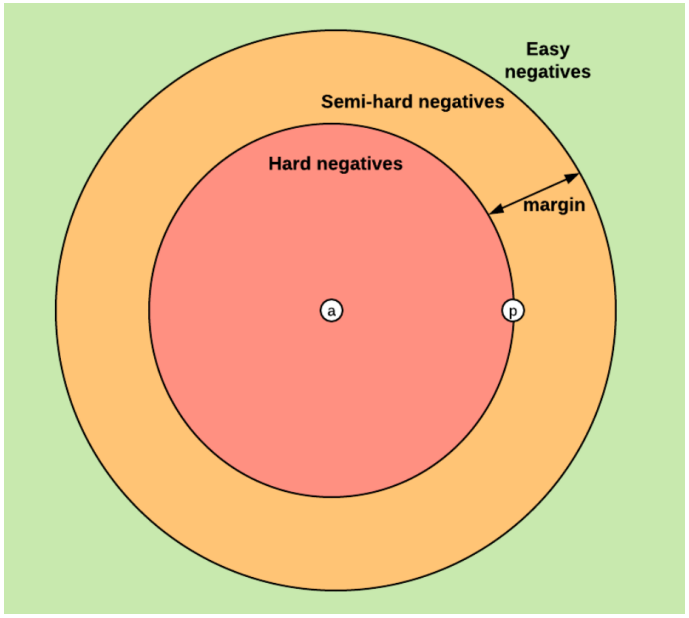


Figura 2.20 Los tres tipos de negativos según el triplete. (Moindrot, 2018).

3 Objetivos

En este trabajo fin de máster, el objetivo fijado ha sido el de crear una aplicación que sirva de control de acceso mediante el reconocimiento facial de los sujetos. Para esto, en el anterior capítulo se ha realizado una extensa recapitulación de las técnicas que han ido evolucionando dentro de esta temática. En especial, vamos a apoyarnos en la ciencia del Deep Learning (DL) para poder llevar a cabo nuestro fin ya que es lo más novedoso que se aplica dentro del ámbito.

El trabajo se va a dividir en dos campos claramente diferenciados. En la Figura 3.1 se refleja los componentes que a grandes rasgos forman el proyecto. En primer lugar, se busca un algoritmo que consiga distinguir identidades de diferentes personas: esto será trabajo de las redes neuronales. En segundo lugar, una etapa de validación, donde se compruebe la adecuada ejecución de esta herramienta en un control de acceso.

3.1 Entrenamiento red neuronal

Se necesitará un modelo capaz de poder discernir la identidad del sujeto dándole como entrada una foto de su cara. Aquí se englobará todo lo realizado para que la red CNN aprenda y lleve a cabo su labor.

Para la parte de entrenamiento, nos centraremos en ciertos aspectos. En primer lugar, será la búsqueda de las bases de datos. Ésta se centrará en priorizar las que tengan mayor variabilidad tanto de poses, como de sujetos como de situación lumínicas. En el siguiente capítulo se destacarán las más usadas en la disciplina del reconocimiento facial y las elegidas en este trabajo.

Respecto al entrenamiento de la red, tres estructuras serán elegidas para este fin: ResNet-50, Xception e Inception-ResNet V1. Aparte del entrenamiento desde cero que se hará sobre estas, se comparará con el rendimiento que tendría una red como la Inception-ResNet V1 con unos pesos preentrenados sobre una base de datos con más imágenes que las bases que se utilizarán en este proyecto. Además, se probará a realizar *transfer learning* para ver si el rendimiento mejora sobre fotografías de personas que no sean estrictamente famosas, ya que todo el estado del arte de los *databases* se centran en estas.

3.2 Validación

Una vez obtenido el modelo, se hará una comprobación de, alimentándolo con fotografías de una *webcam*, un hipotético caso de control de acceso donde distinga la persona que quiere acceder y la que no esté en la base de datos sea rechazada su entrada por cuestión de seguridad.

Para llevar a cabo la validación, se comprobará el rendimiento ofrecido por cada red neuronal en diferentes bases de datos, dos de famosos y una última propia elaborada con sujetos cercanos a mi persona. Asimismo, para la parte más práctica de adquirir imagen del rostro y compararla con la base de datos para la posterior verificación, la Raspberry Pi 4 será utilizada como ejemplo de sistema embebido de bajo coste. En esta se ejecutará la aplicación del control de acceso y se comprobará su correcto funcionamiento.

En resumen, el trabajo se centrará en un primer entrenamiento del algoritmo que nos permitirá discernir entre sujetos. Esto se llevará a cabo utilizando diferentes estructuras de redes neuronales, usando la función de pérdidas *Triplet Loss*, ya que nos centramos en técnicas de *one-shot learning*. Una vez obtenido esta herramienta, el siguiente paso será validar el desempeño. Se capturarán imágenes de sujetos cercanos y se comprobará su correcto funcionamiento para autentificar el acceso a estos. Esto se implementará tanto en

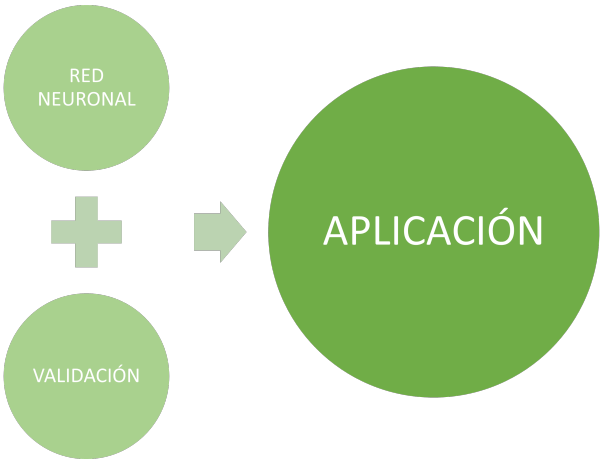


Figura 3.1 Componentes del trabajo. Elaboración propia.

ordenador como en Raspberry para hacer la comparativa entre dos sistemas con potencia computacional muy opuesta.

En el siguiente capítulo, se presentarán los materiales y la metodología a seguir de manera más detallada y extensa para conseguir el objetivo propuesto.

4 Metodología

En este capítulo, se va a describir los materiales y métodos que han hecho posible este trabajo, enumerando el Software (SW) y el Hardware (HW) utilizado y el flujo de trabajo a realizar.

4.1 Materiales

En recursos físicos se ha hecho uso de un ordenador personal de sobremesa compuesto por un procesador AMD Ryzen 5 1600, 16 GB de RAM, tarjeta gráfica NVIDIA GeForce GTX 1070 y una webcam como accesorio para la parte de comprobación. Esta última ha sido la encargada de tomar las imágenes que permitían verificar el control de acceso. Aparte, para la parte de implementación e inferencia, una Raspberry Pi será usada para complementar el trabajo realizado en el ordenador junto a la Pi Camera.

4.2 Métodos

Los métodos utilizados han sido centrados en el SW, especialmente en la programación. Python (Rossum y cols., 2009) ha sido la piedra angular que ha sustentado el proyecto. Python es un lenguaje orientado a objetos con una sintaxis clara y una gran potencia debido a los numerosos paquetes que ofrece. Estos se encargarán de ir abordando con cierta solvencia los problemas que vayan apareciendo durante la realización de nuestra tarea.

Con vistas a mantener todo estructurado, un entorno de desarrollo integrado es la mejor opción por la que apostar. PyCharm ha sido el entorno de desarrollo elegido para esto. Además, mantener los cambios realizados en los códigos registrados es vital para poder revertirlos o trabajar en alternativas en todo momento sin perjudicar al núcleo central. Git es la solución a esta disyuntiva, un control de versiones para mantener todo el código, lo que se ha expandido vinculando con un repositorio virtual ofrecido por GitHub.

Por tanto, un esquema general que englobe los métodos utilizados se puede visualizar en la Figura 4.1.

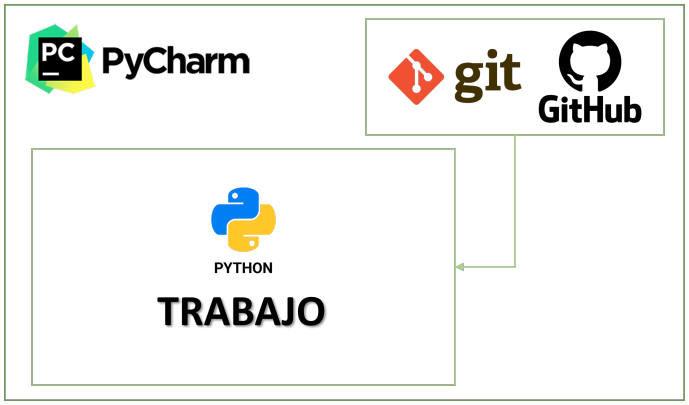


Figura 4.1 Esquema general de los métodos utilizados. Elaboración propia.

Como se comentó en el anterior capítulo, se requiere poder trabajar con fotografías y con NN, específicamente CNN al tratarse de un problema basado en imágenes. Con esta premisa en mente, se va a hacer uso de paquetes en Python orientados a este fin. Se necesitarán paquetes que sean capaces manipular redes neuronales, paquetes que puedan abrir imágenes y realizar operaciones con ellas y paquetes para poder visualizar los resultados. Cada una de las librerías más importantes que se han utilizado en este proyecto se mencionan en la Tabla 4.1 cuyos logos principales se pueden visualizar en la Figura 4.2.

Tabla 4.1 Paquetes utilizados. Elaboración propia.

Paquete	Descripción	Referencia
TensorFlow	Biblioteca creada por Google que trabaja con las redes neuronales. Tensorflow implementa actualmente a Keras, su Application Programming Interface (API) de más alto nivel	(Abadi y cols., 2019)
Scikit-learn	Biblioteca focalizada en trabajar con algoritmos de ML y el preprocesamiento de la información entre otras cosas.	(Pedregosa y cols., 2011)
NumPy	Paquete para trabajar con vectores y matrices multidimensionales.	(Harris y cols., 2020)
Matplotlib	Paquete orientado a la visualización de los datos con gráficos.	(Hunter, 2007)
Seaborn	Librería basada en Matplotlib que ofrece una gráficos avanzados con una sintaxis más sencilla.	(Waskom, 2021)
OpenCV	Librería de código abierto de visión artificial.	(Bradski, 2000)

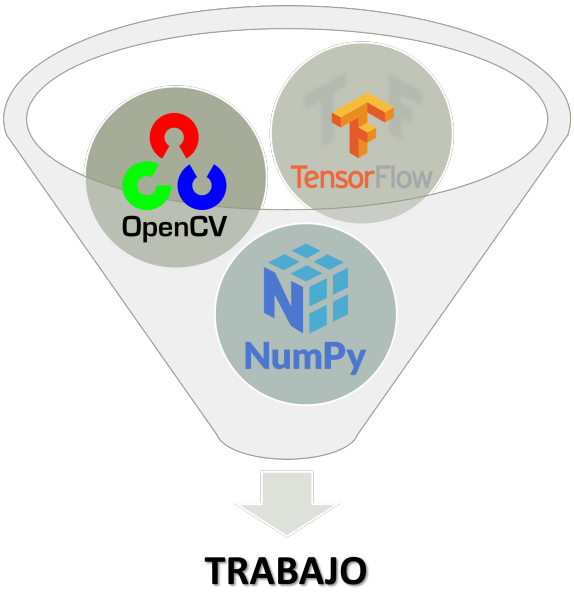


Figura 4.2 Logos de los paquetes principales. Elaboración propia.

4.3 Raspberry Pi

La Raspberry Pi es un pequeño ordenador establecido en una única placa. Lo asombroso que trae consigo este HW es la relación calidad/precio que nos ofrece. Se ha elegido esta opción porque se busca probar en un sistema embebido sencillo la aplicación de un control de acceso mediante reconocimiento facial.

En especial, el modelo de Raspberry elegido ha sido el Pi 4 modelo B (Figura 4.3) cuyas características más destacables son la inclusión de un procesador 64 bits Quad-Core Cortex-A72 y 4GB de RAM para mover

cualquier aplicación cómodamente. Aprovechando el procesador, se ha elegido la instalación del sistema operativo Raspberry Pi OS (Figura 4.4), anteriormente llamado Raspbian, en su versión de 64 bits nombrada como *bullseye*.

Figura 4.3 Raspberry Pi 4 Modelo B. Elaboración propia.

Figura 4.4 Raspberry Pi OS. (Pi, 2020).

Para realizar la adquisición de imágenes, se ha optado por acompañar a la Raspberry la Pi Camera v2 (Figura 4.5), cámara de la misma compañía y que posee un sensor de 8 megapíxeles de la marca Sony. Esta junto a las herramientas que ofrece OpenCV serán las que permitan adquirir y procesar la información en forma de imagen para poder detectar la identidad del sujeto que se muestre en el control de acceso.

Figura 4.5 Pi Camera v2. Elaboración propia.

4.4 Base de datos

La estructura de una red neuronal es su esencia, pero esta necesita de algo importante para que pueda funcionar: los datos de entrada. Para poder entrenar una red, es necesario alimentarla con información para

que, realizando sus operaciones y aprendiendo del error que comete en la tarea encomendada, pueda aprender y actualizar los pesos de sus capas.

Es por esto que en esta sección se va a hacer un inciso en las bases de datos más utilizadas en el ámbito de cara de personas, detallando el número de imágenes y sujetos que las componen, reflejándose en la Tabla 4.2.

Tabla 4.2 Base de datos. (M. Wang y Deng, 2021).

Base de datos	Imágenes	Sujetos	Referencia
MS-Celeb-1M	10M	100k	(Guo, Zhang, Hu, He, y Gao, 2016)
MegaFace	4.7M	672.057	(Nech y Kemelmacher-Shlizerman, 2017)
VGGFace2	3.31M	9.131	(Cao, Shen, Xie, Parkhi, y Zisserman, 2018)
CASIAWebFace	494.414	10.575	(Yi, Lei, Liao, y Li, 2014)
MillionCelebs	18.8M	636K	(Y. Zhang y cols., 2020)
IMDB-Face	1.7M	59K	(F. Wang y cols., 2018)
UMDFaces-Videos	22.075	3.107	(Bansal, Nanduri, Castillo, Ranjan, y Chellappa, 2018)
VGGFace	2.6M	2.622	(Parkhi, Vedaldi, y Zisserman, 2015)
CelebA	202.599	10.177	(Liu, Luo, Wang, y Tang, 2015)
LFW	13.233	5.749	(G. B. Huang, Mattar, Berg, Learned-Miller, y Learned-Miller, 2008)
Google	>500M	>10M	(Schroff y cols., 2015)
Facebook	4.4M	4K	(Taigman, Yang, Ranzato, y Wolf, 2014)

En este trabajo se van a hacer uso de tres bases de datos: CelebA, LFW y una base propia. Esta última fue elaborada con imágenes de personas cercanas mías, realizadas con sus propios móviles y en diferentes días, haciendo un total de 186 imágenes distribuidas en 15 sujetos diferentes. Ejemplos de imágenes tanto de CelebA y LFW se pueden visualizar en las Figuras 4.6 y 4.7 respectivamente, ambas caracterizadas por tener fotografías de sujetos con diferentes poses y condiciones lumínicas.

Figura 4.6 Ejemplo imágenes CelebA. Elaboración propia.

Figura 4.7 Ejemplo imágenes LFW. Elaboración propia.

4.5 Red Neuronal

El primer paso para la realización de la aplicación es contar con un modelo que, alimentado con imágenes, sepa discernir entre sujetos. Para esto, como se comentó en capítulos anteriores, la función de pérdidas establecida para la red será la Triplet Loss. Esta se basa en entregar un triplete que contenga una imagen ancla, una imagen del mismo sujeto que el ancla (positiva) y otra de una persona diferente (negativa). Lo que busca esta función es acercar los *embeddings* (que son los vectores que produce la red a la salida según la información suministrada) de las fotografías de la misma persona, mientras que al mismo tiempo alejar estos *embeddings* del de la imagen negativa. Para optimizar estos vectores asociados a cada imagen, el funcionamiento de Triplet Loss necesita que las capas finales de la red no presenten ninguna activación y, con vistas a controlar el funcionamiento del margen en la fórmula del Triplet Loss, la última capa sea una normalización L2 para localizar la salida en una hiperesfera de módulo unidad.

Por tanto, en este trabajo se van a emplear diferentes tipos de redes neuronales para compararlas. La base de datos elegida para el entreno de estas es la CelebA, debido a su variabilidad en las fotos y al contener un número aceptable de imágenes que pueden ser procesadas por un ordenador personal. Respecto a la estructura de las redes a utilizar, se va a hacer uso de las siguientes:

- ResNet-50 entrenada desde cero, llamada así al tener 50 capas de profundidad.
- Xception entrenada desde cero, una extensión de la Inception.
- Inception-ResNet V1 modificada con una capa final formada por 512 neuronas y una capa de normalización L2 (Uri, 2020), también entrenada desde cero.

El flujo de trabajo que se va a seguir en ambas estructuras se refleja en la Figura 4.8, donde se alimenta con la base de datos, la red minimiza la pérdida Triplet Loss y calcula los *embedding* de cada imagen.

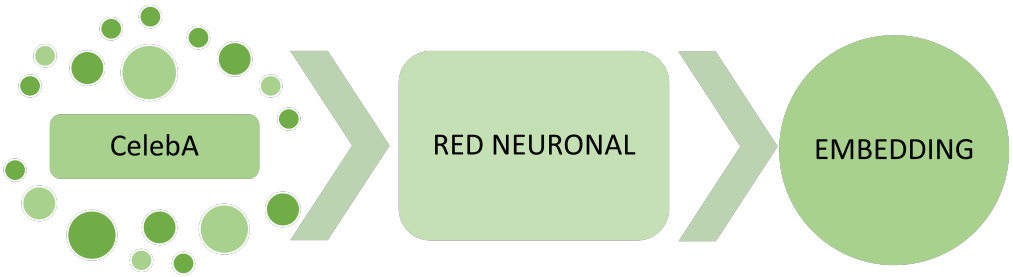


Figura 4.8 Flujo de trabajo entreno red. Elaboración propia.

Aparte de estas redes, como modo comparativo, se va a tomar la misma estructura Inception modificada, pero aplicando unos pesos entrenados en la base de datos MS-Celeb-1M (Taniai, 2018). Además, se va a realizar *transfer learning* sobre esta misma red con la base de datos personal por si hubiera alguna mejora reentrenando la red con un caso práctico real.

4.6 Control Acceso

En el último apartado del trabajo, una vez obtenida la herramienta para extraer la información de las imágenes, quedaría comprobar el correcto funcionamiento. Para esto, se va a seguir el flujo de trabajo reflejado en la Figura 4.9.

En primer lugar, se parte de una toma de imágenes de los sujetos a los que se va a permitir el acceso (en el caso del trabajo, se toman doce imágenes). Una vez obtenidas, se extraen las caras y se sacan sus *embeddings*. El paso siguiente sería calcular un prototipo de toda la información obtenida de los sujetos, esto significa calcular la mediana entre todos los cálculos para aligerar el peso que los valores atípicos pueden aportar. Obtenido un prototipo para cada sujeto, se guarda en la base de datos para su posterior uso. Todo el proceso mencionado se hace de manera *offline* para poder utilizarlo cuando se requiera.

Por otra parte, quedaría el proceso en tiempo real del control de acceso. Aquí se supondrá un caso en el que una persona requiere acceder a un lugar y es aquí donde nuestra aplicación actuaría. En primera instancia, detectaría la cara del sujeto y alimentaría la red neuronal para producir el *embedding* correspondiente. Una vez realizado este paso, se recorrería todos los prototipos guardados en la base de datos y se compararían con el vector obtenido: si la distancia, que puede medirse según la similitud coseno o distancia euclídea, del vector y algún prototipo es menor a un límite establecido, el sujeto es reconocido, en caso contrario, se rechaza la entrada.

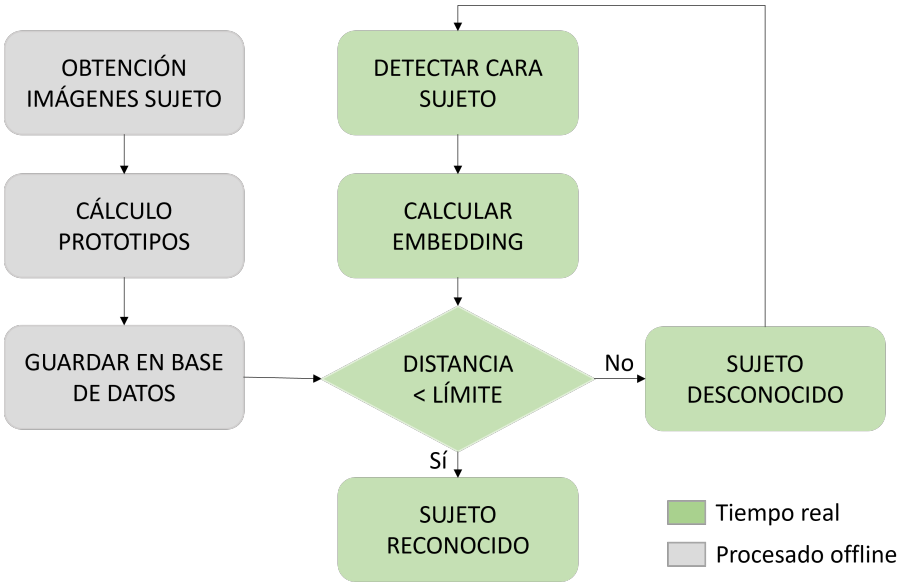


Figura 4.9 Flujo de trabajo control acceso. Elaboración propia.

Concluyendo este capítulo, hemos realizado un recorrido por la metodología que se ha seguido durante el trabajo. En las siguientes páginas se mostrarán los diferentes resultados que se han logrado aplicando lo expuesto anteriormente.

5 Resultados

En este capítulo serán expuestos los resultados obtenidos al aplicar la metodología comentada en páginas anteriores. Comenzaremos viendo las diferentes redes neuronales y su comportamiento, seguido de la aplicación más directa de control de acceso, incluyendo en esta el uso de la Raspberry.

5.1 Red Neuronal

Respecto a las redes neuronales, vamos a dividir esta sección en las diferentes estructuras que se han ido utilizando, desglosando su rendimiento frente a la base de datos propia, una pequeña colección de 20 sujetos de la base de datos LFW y la base original de entrenamiento, CelebA.

En primer lugar, un punto común a todas las redes que han sido entrenadas es la utilización del *data augmentation*. Esto se basa en aplicar técnicas a la base de datos para mejorar el entreno puesto que la información suministrada es crucial en el entrenamiento. Al tener una base de datos y capacidad computacional limitada, la mejor opción es hacer uso de esta utilidad implantanda en Tensorflow.

Para poner en práctica este recurso, lo que se hace es añadir a la entrada de los modelos de redes una serie de capas que realizan ciertas operaciones, en este caso, a las imágenes proporcionadas. Las utilizadas en este trabajo se pueden ver en la Tabla 5.1.

Tabla 5.1 Técnicas de Augmentation utilizadas. Elaboración propia.

Técnica	Descripción
Rescaling	Transforma los píxeles de la imagen de [0,255] a [0,1]
RandomFlip	Volteo horizontal a la imagen
RandomRotation	Rotación de la imagen de 36° en ambos sentidos
RandomZoom	Zoom a la imagen de un 10 %

CelebA es la base de datos elegida para el entrenamiento. Un ejemplo del *data augementation* aplicada a esta se refleja en la Figura 5.1. Como se observa, la imagen es rotada, volteada y aplicada un cierto zoom de manera aleatoria. Todas estas acciones son realizadas durante la fase de entreno, pero no en el modo de inferencia, es decir, cuando se realizan predicciones con la red. La única capa que persiste en inferencia sería el reescalado de la imagen, ya que si no fuera así la red recibiría valores con los que nunca ha trabajado.

Recapitulando, en las siguientes líneas las diferentes redes se irán presentando. Todas comparten las mismas capas finales, una capa densa con una dimensión de 518 y una capa de normalización L2. La función de pérdida utilizada es Triplet Loss, con una estrategia de tripletes semi-difíciles y un margen de 0.3. El optimizador elegido es el Adam con un *learning rate* inicial de 0.01. Para supervisar el entrenamiento, se han aplicado los recursos descritos en la Tabla 5.2. Todos estos se enfocan a controlar el aprendizaje y evitar que el modelo se detenga en un mínimo local, sufriendo de *overfitting* o *underfitting*, es decir, enfocarse a buscar el entreno óptimo.

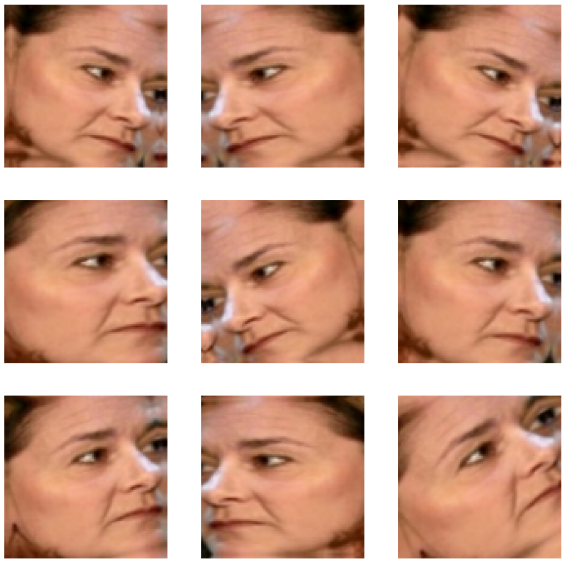


Figura 5.1 Data Augmentation sobre CelebA. Elaboración propia.

Tabla 5.2 Recursos para supervisar el entreno. Elaboración propia.

Recurso	Descripción
CSVLogger	Vuelca los datos del entreno a un csv al finalizar cada *epoch*
EarlyStopping	Finaliza el entreno cuando *validation loss* deja de decrecer. La condición es que transcurran 4 *epochs* con esta premisa
ReduceLROnPlateau	Reduce el *learning rate* cuando *validation loss* deja de decrecer. La condición es que transcurran 2 *epochs* con esta premisa
Backup	Guarda todo el proceso cuando finaliza cada *epoch*

5.1.1 ResNet-50

La elección de la ResNet-50 se debe a ser una red tradicionalmente utilizada por su buen rendimiento y ser relativamente ligera (25.6 millones parámetros). Las características de su entreno son las comentadas anteriormente, destacando que para completarlo se requirió de 19 horas en 52 *epochs*. La función de pérdida a lo largo de estos se ve reflejada en la Figura 5.2. Como se observa, el modelo va parejo en cuanto al valor de entreno y de validación, por lo que nos indica que no sufre ni de *overfitting* ni de *underfitting*.

Para medir la eficiencia de la red, se va a hacer uso de:

- La curva Receiver Operating Characteristic (ROC), destacando los valores de sensibilidad (relación entre los positivos verdaderos y falsos negativos) y el límite a imponer para que exista un False positive rate (FPR) de 0.1 %.
- Matriz de confusión representando los valores de True positive (TP), False negative (FN), False positive (FP) y True negative (TN).
- Gráfico con la distancia existente entre las clases de las bases de datos. Se toma una imagen de un sujeto y se compara con una de las otras clases para medir la distancia existente.
- Representación T-distributed Stochastic Neighbor Embedding (TSNE) para visualizar la agrupación de los diferentes sujetos de la base personal.

Para la base de datos reducida de LFW, los resultados son los mostrados en la Figura 5.3, destacando que se consigue una sensibilidad de 59.3 % . En el caso de la base de datos personal (Figura 5.4), la sensibilidad se reduce ligeramente a 55.3 % , aun así se logra unos resultados bastante aceptables para la detección de los sujetos reales. Si comprobamos el rendimiento para la parte de test de CelebA, en la Figura 5.5 se refleja una sensibilidad del 44.9 %, aunque un gran porcentaje de falsos positivos al existir un gran número de iteraciones.

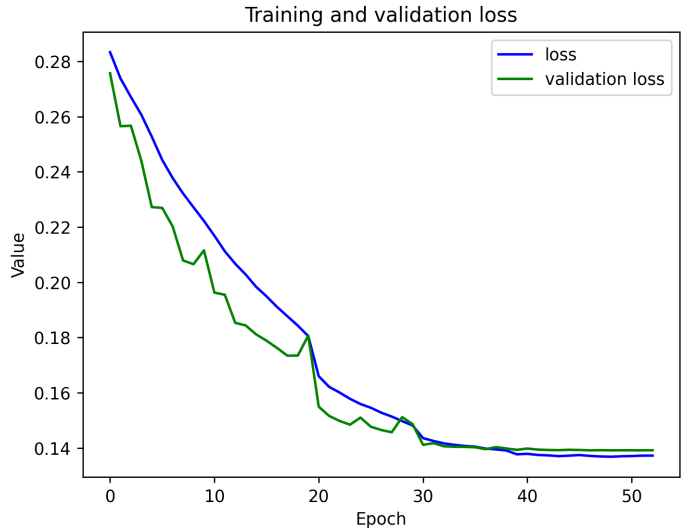


Figura 5.2 Función de pérdidas entrenamiento ResNet-50. Elaboración propia.

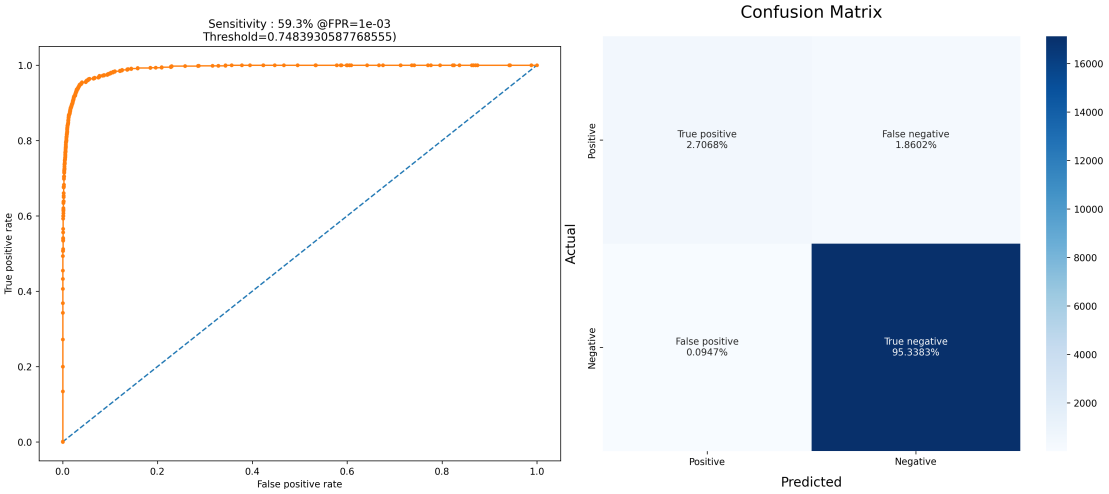


Figura 5.3 ROC y matriz de confusión ResNet-50 para LFW. Elaboración propia.

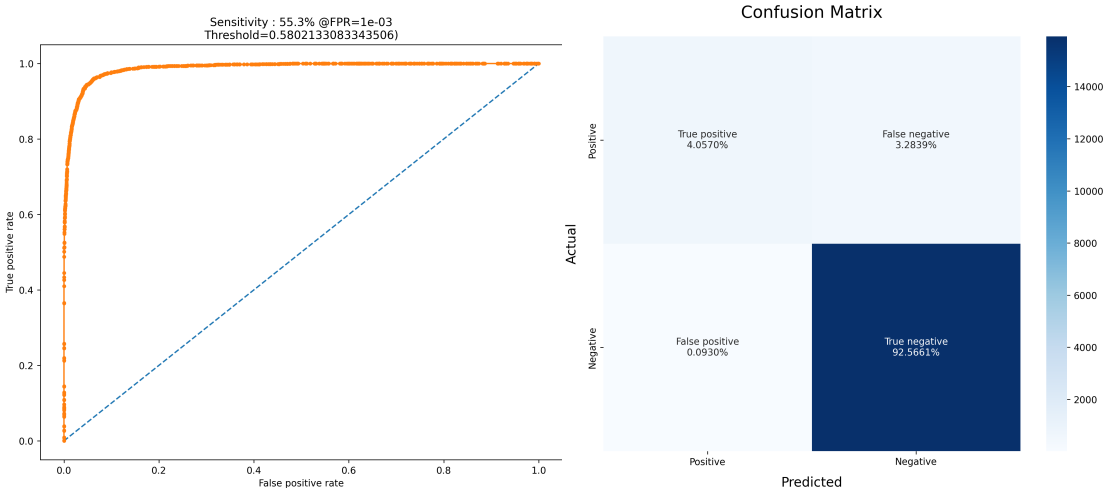


Figura 5.4 ROC y matriz de confusión ResNet-50 para base de datos propia. Elaboración propia.

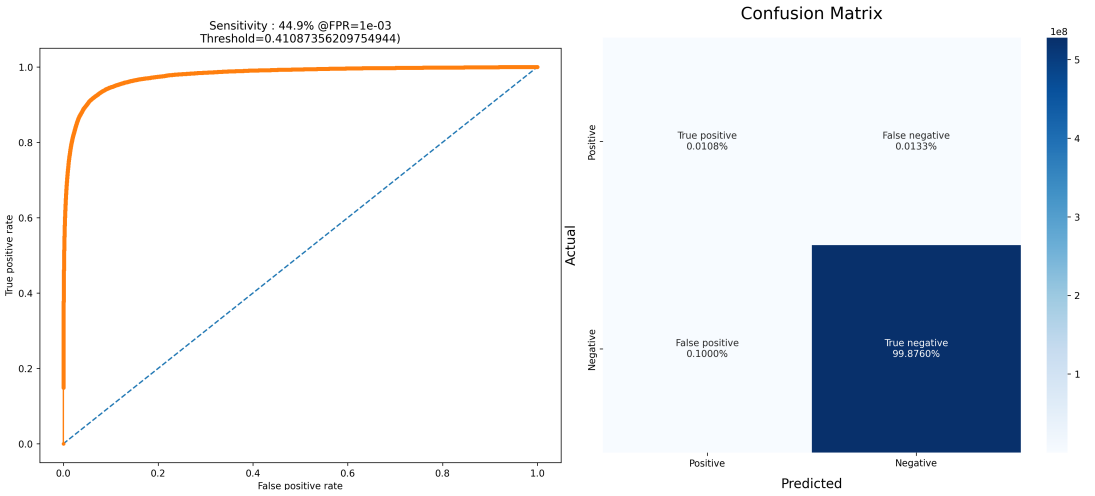
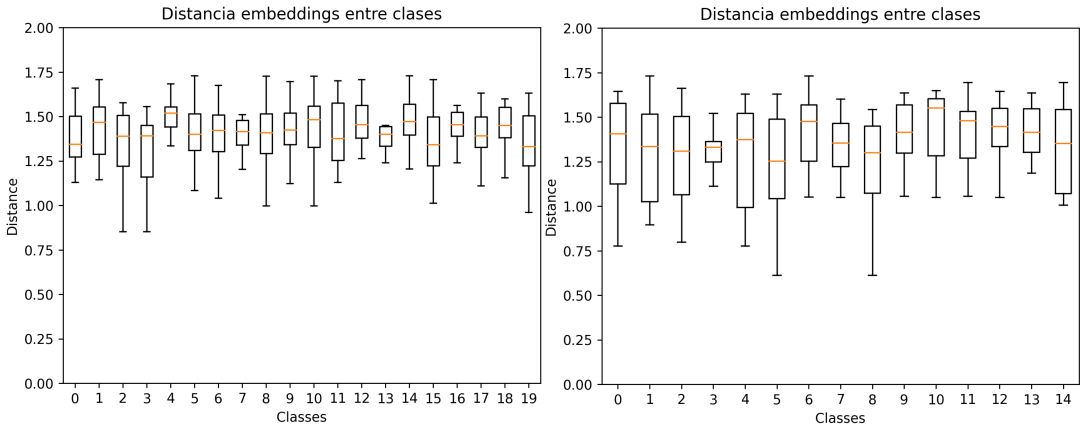


Figura 5.5 ROC y matriz de confusión ResNet-50 para CelebA. Elaboración propia.

La distancia entre clases se visualiza en la Figura 5.6. En ambos casos se observa una distancia media entre clases de 1.5.



(a) LFW. **(b)** Base propia.

Figura 5.6 Distancia entre clases de ResNet-50. Elaboración propia.

Para ver el comportamiento que tiene la red para agrupar imágenes de una misma persona, tal y como se comentó anteriormente, se utiliza el gráfico TSNE (Figura 5.7). En este se representan los 15 sujetos con su correcta asociación.

En la Figura 5.8 se representan ejemplos de LFW comparándose entre ellos para medir su distancia. En ambos casos las distancias entre imágenes del mismo sujeto se sitúan por debajo del límite calculado en el ROC anterior.

5.1.2 Xception

Xception ha sido otra estructura elegida por su ligereza (22.9 millones de parámetros) y con el fin de probar otra de las redes presentadas en el estado del arte. La función de pérdida se muestra en la Figura 5.9, destacando su entrenamiento en 49 epochs y una duración de 18 horas.

Los resultados obtenidos en relación con la sensibilidad tanto en LFW (Figura 5.10), como en la base de datos propia (Figura 5.11) como en CelebA (Figura 5.12) reflejan estar un escalón por debajo en cuanto a rendimiento frente a la ResNet-50.

Respecto a la distancia entre clases, se observa un buen comportamiento en la Figura 5.13, manteniendo los valores atípicos contenidos y una media cercana a 1.25.

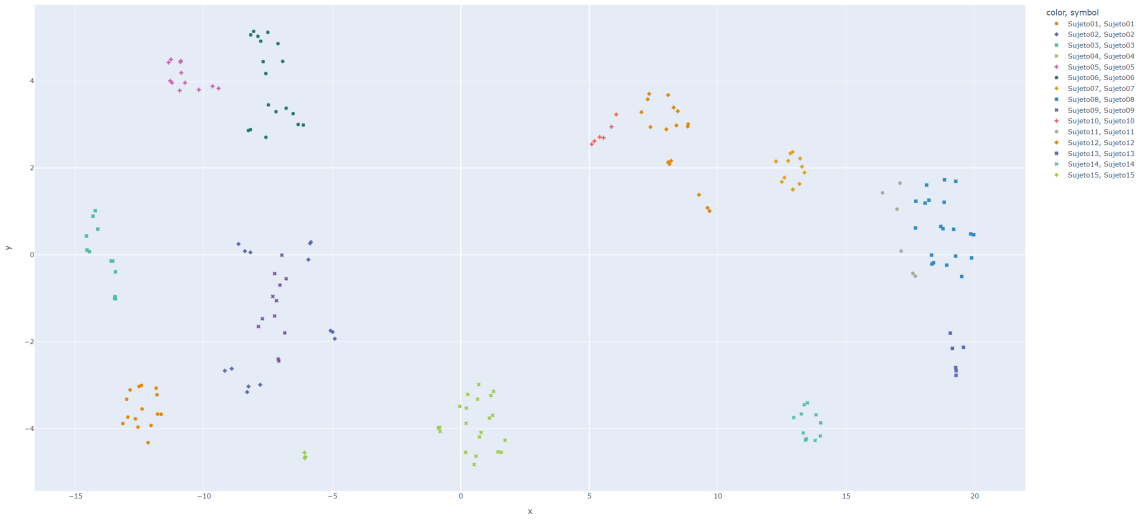


Figura 5.7 TSNE ResNet-50 a base de datos propia. Elaboración propia.

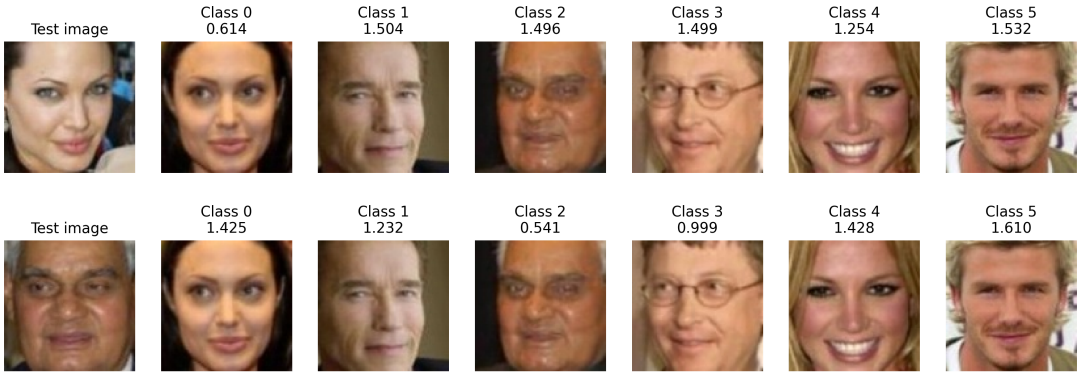


Figura 5.8 Ejemplos de distancia en LFW por ResNet-50. Elaboración propia.

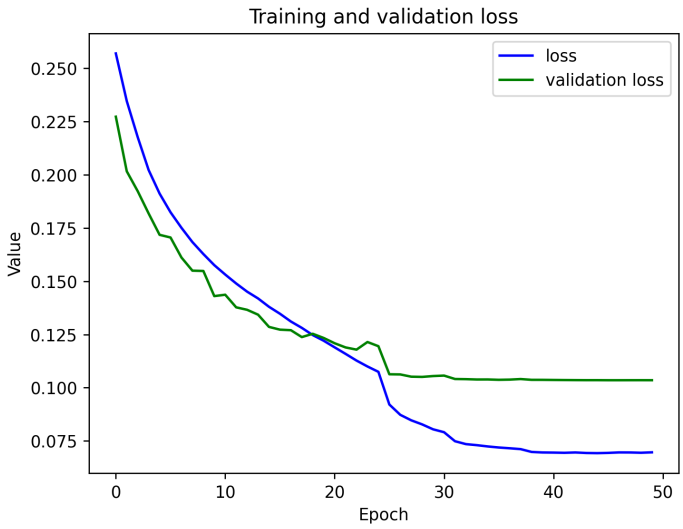


Figura 5.9 Función de pérdidas entrenamiento Xception. Elaboración propia.

La agrupación por clases se consigue con relativa solvencia, permitiendo discernir entre imágenes de distintos sujetos (Figura 5.14).

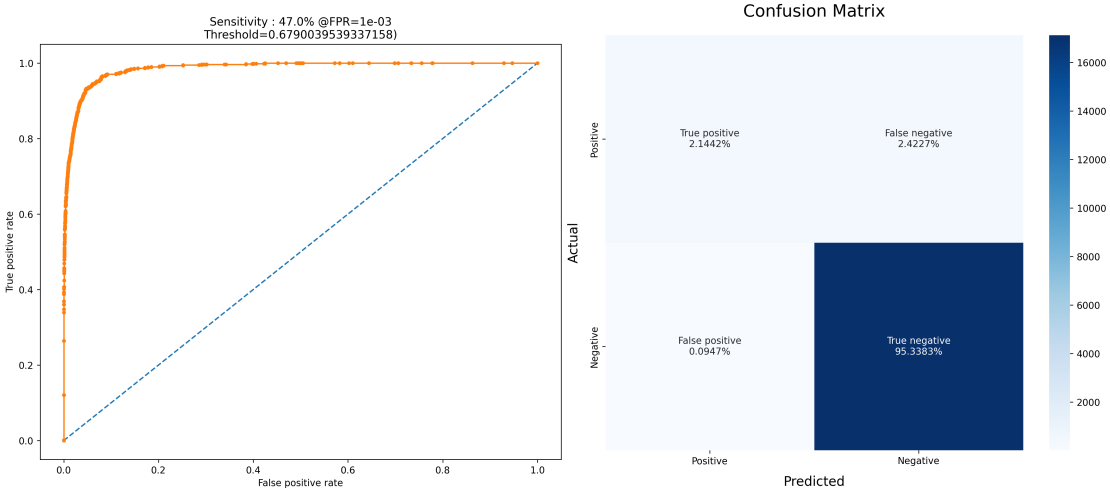


Figura 5.10 ROC y matriz de confusión Xception para LFW. Elaboración propia.

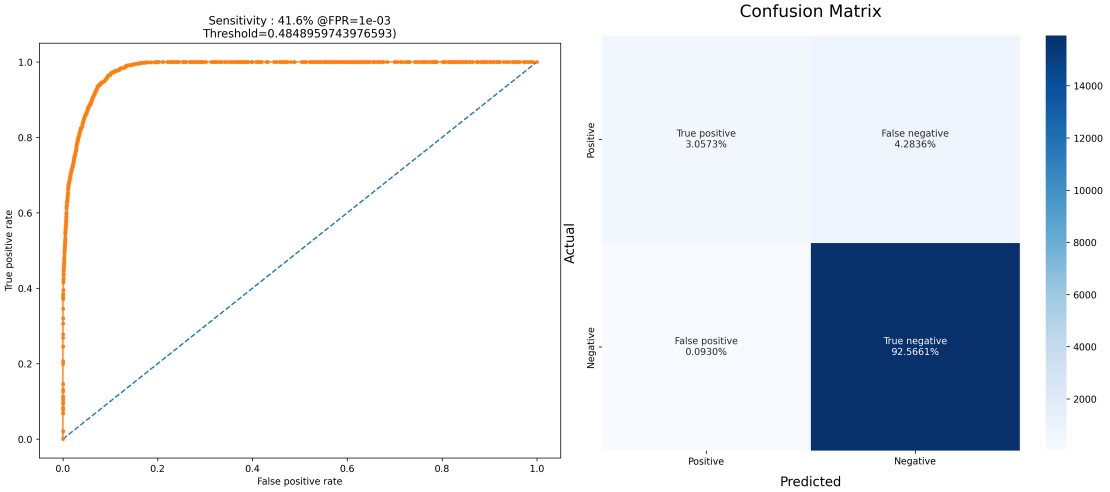


Figura 5.11 ROC y matriz de confusión Xception para base de datos propia. Elaboración propia.

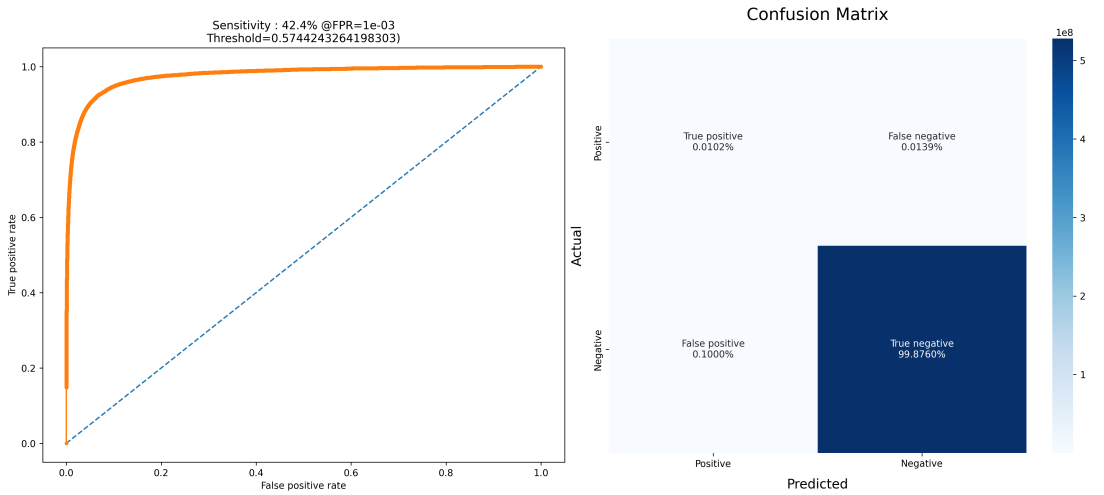
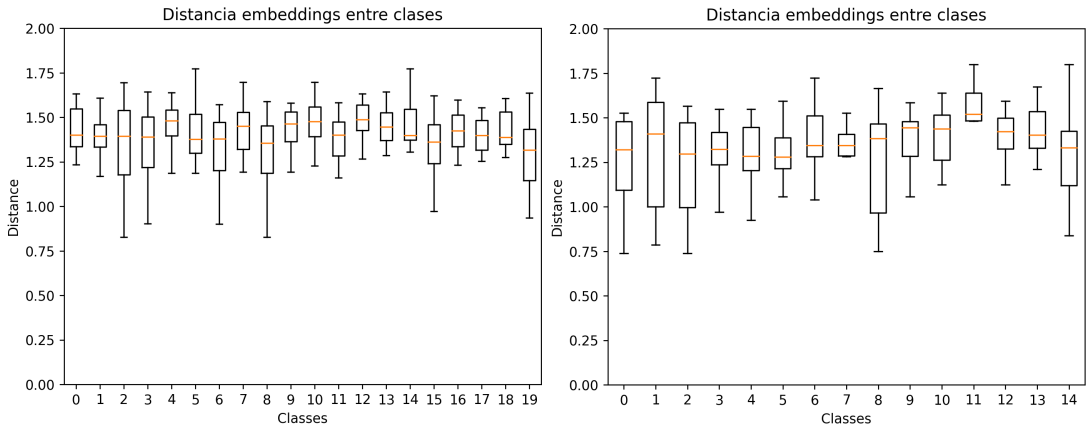


Figura 5.12 ROC y matriz de confusión Xception para CelebA. Elaboración propia.



(a) LFW. **(b)** Base propia.

Figura 5.13 Distancia entre clases de Xception. Elaboración propia.

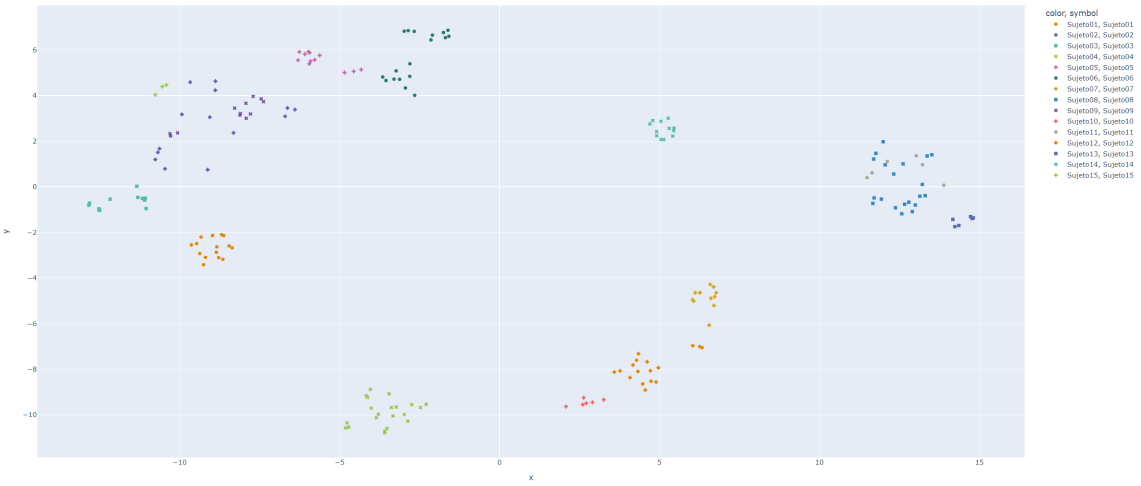


Figura 5.14 TSNE Xception a base de datos propia. Elaboración propia.

En la Figura 5.15 se vuelven a exponer ejemplos de las comparaciones en distancia utilizando Xception con la base de datos reducida de LFW.

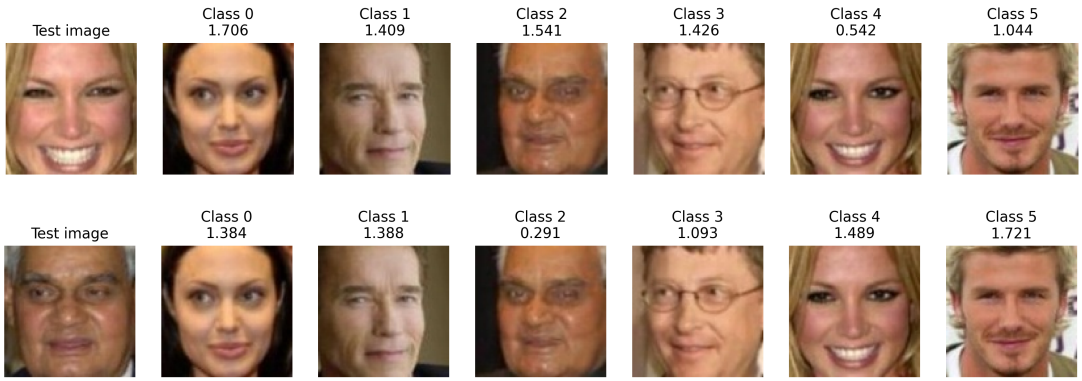


Figura 5.15 Ejemplos de distancia en LFW por Xception. Elaboración propia.

5.1.3 Inception-ResNet V1

La Inception-ResNet V1 (23.5 millones de parámetros) es una estructura comúnmente utilizada en el estado del arte del reconocimiento facial. Es por esta razón que ha sido la segunda escogida para ser entrenada. Su

entreno adopta las mismas características que el de ResNet-50. En esta ocasión, el aprendizaje fue en 50 *epochs* durando cerca de 17 horas. La función de pérdidas se representa en la Figura 5.16, la cual refleja que la red está sufriendo de *overfitting*, ya que las pérdidas en el entreno son claramente inferiores a las de validación.

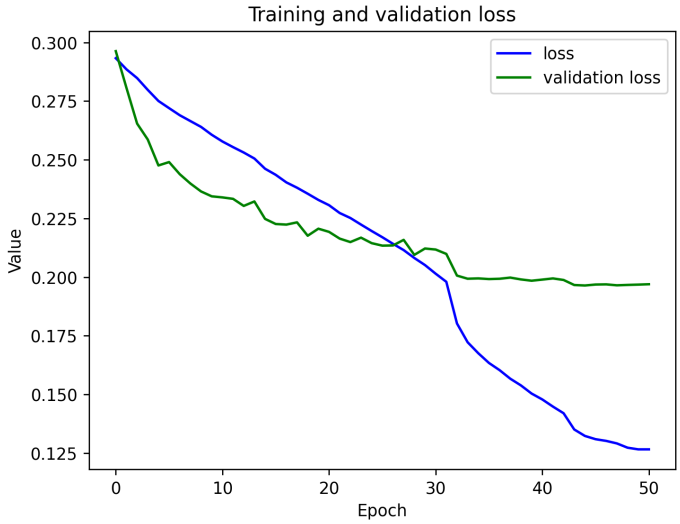


Figura 5.16 Función de pérdidas entrenamiento de Inception-ResNet V1. Elaboración propia.

Las gráficas de ROC y matriz de confusión tanto para LFW, como para la base de datos propia como para CelebA se pueden observar en las Figuras 5.17, 5.18 y 5.19 respectivamente. En comparación con la ResNet-50, se observa un considerable decremento en la sensibilidad, hecho que se puede explicar con el *overfitting* que sufre la red desde su entreno.

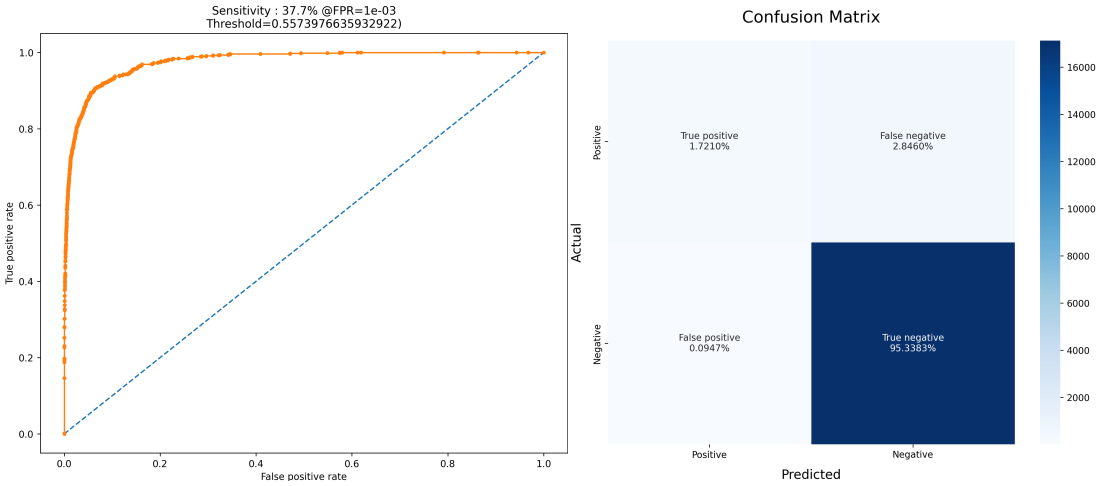


Figura 5.17 ROC y matriz de confusión Inception-ResNet V1 para LFW. Elaboración propia.

En la distancia entre clases (Figura 5.20), la media desciende también en comparación con el caso anterior, además de incluir más valores atípicos que acercan una clase a otra.

El TSNE de la Inception se visualiza en la Figura 5.21. Aún con el descenso de rendimiento, se consigue discernir entre sujetos, situando sus fotografías cercas unas de otras.

Del mismo modo que pasaba anteriormente, los famosos de LFW siguen cumpliendo el límite (Figura 5.22) que el ROC calculó.

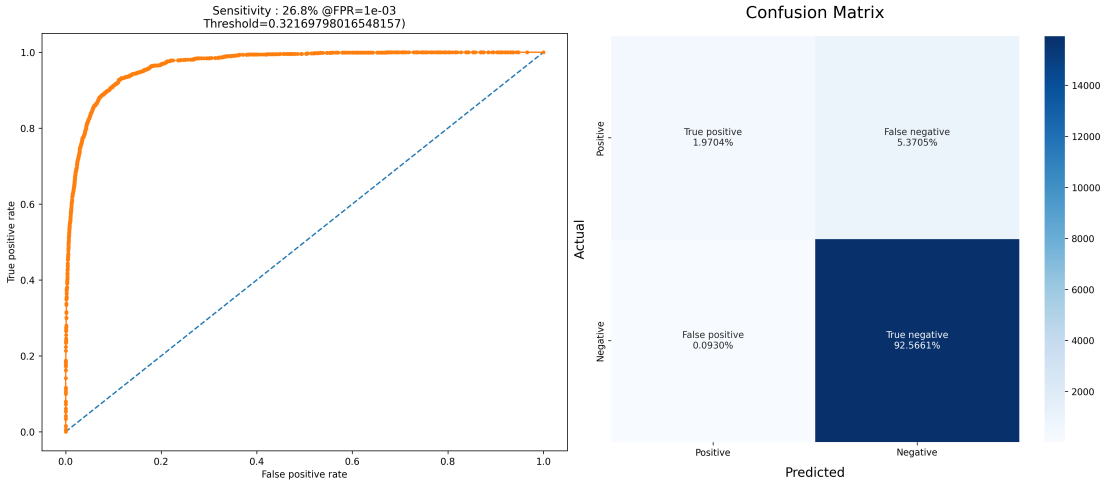


Figura 5.18 ROC y matriz de confusión Inception-ResNet V1 para base de datos propia. Elaboración propia.

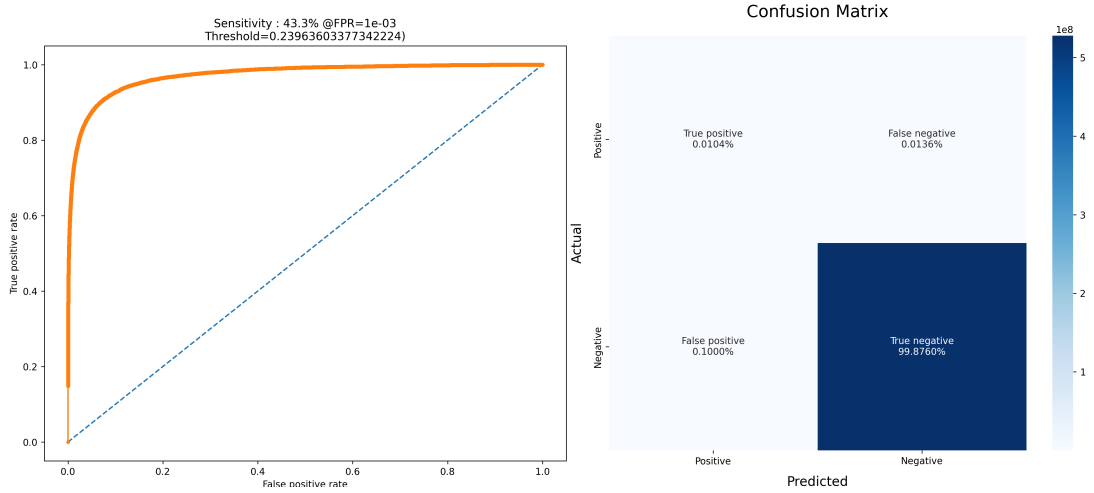
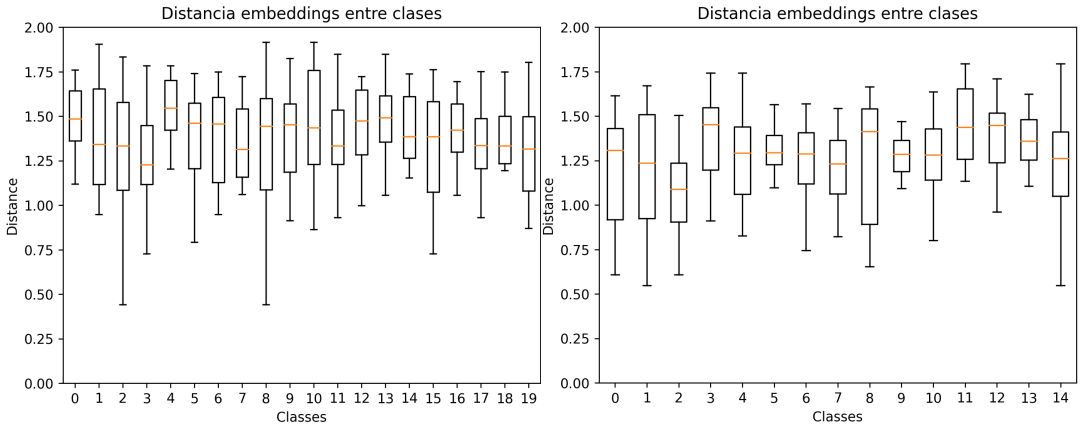


Figura 5.19 ROC y matriz de confusión Inception-ResNet V1 para CelebA. Elaboración propia.



(a) LFW. **(b)** Base propia.

Figura 5.20 Distancia entre clases de Inception-ResNet V1. Elaboración propia.

Figura 5.21 TSNE Inception-ResNet V1 a base de datos propia. Elaboración propia.

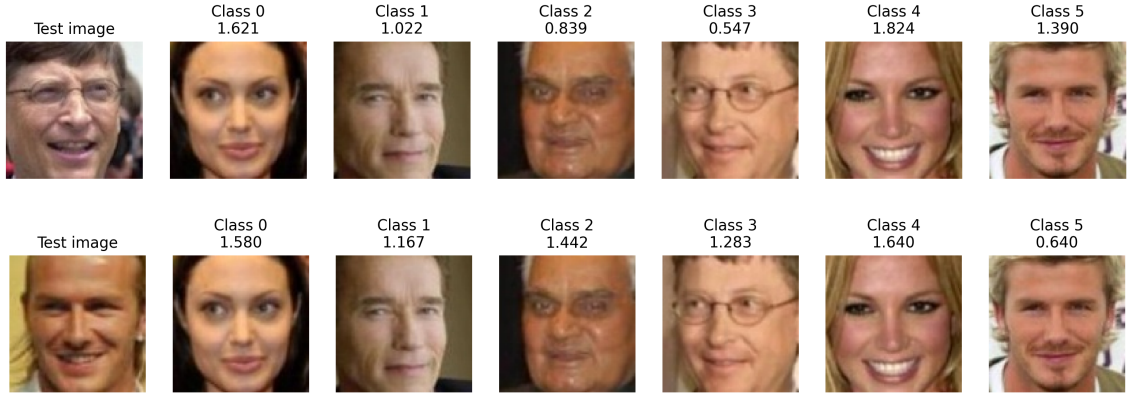


Figura 5.22 Ejemplos de distancia en LFW por Inception-ResNet V1. Elaboración propia.

5.1.4 Inception-ResNet V1 Preentrenada

Atendiendo a la misma estructura de la Inception-ResNet V1, se va a cargar unos pesos preentrenados con la base de datos MS-Celeb-1M para comparar con una red más robusta y con mayor aprendizaje.

En esta ocasión, la mejora es significativa atendiendo a la sensibilidad como se puede observar en las Figuras 5.23 y 5.24: se llega hasta casi el 90 %.

Entre clases (Figura 5.25), la distancia se concentra y se reduce los valores atípicos, dando unos mejores resultados.

La agrupación dada por TSNE en la Figura 5.26 refleja el buen discernimiento que es capaz de realizar este modelo preentrenado.

5.1.5 Inception-ResNet V1 Preentrenada y transfer learning

Una vez visto el desempeño de la Inception-ResNet V1 preentrenada, cabría preguntarse si aplicando la técnica de *transfer learning* mejoraría los resultados en relación con la base de datos propia. Este método consiste consiste en congelar la red para que no se actualicen sus pesos y sólo dejar entrenable la última capa densa para, aprovechando todo lo aprendido en todas las capas convolucionales previas, intentar que el modelo se adapte mejor a las imágenes con las que será evaluado.

La función de pérdida a lo largo del aprendizaje se visualiza en la Figura 5.27. El comportamiento era esperable al entrenarse con menos de 200 imágenes: el modelo converge a un valor de 0 en la pérdida del *training* pero en el de validación no se comporta así.

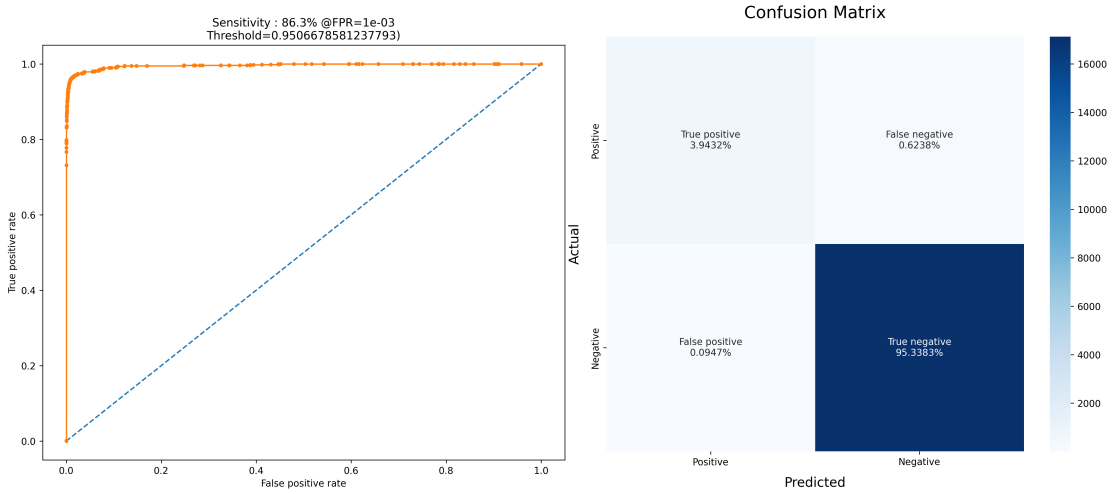


Figura 5.23 ROC y matriz de confusión Inception-ResNet V1 Preentrenada para LFW. Elaboración propia.

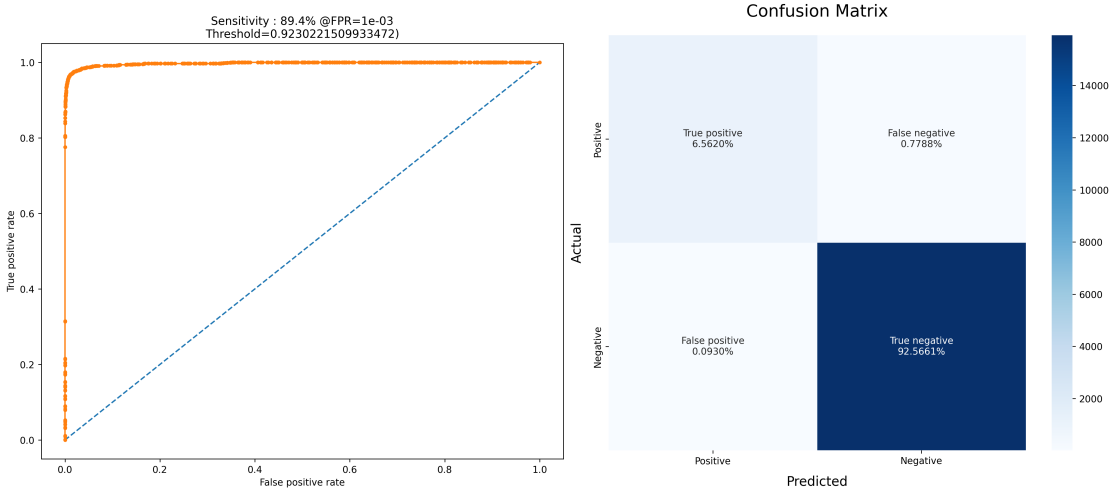
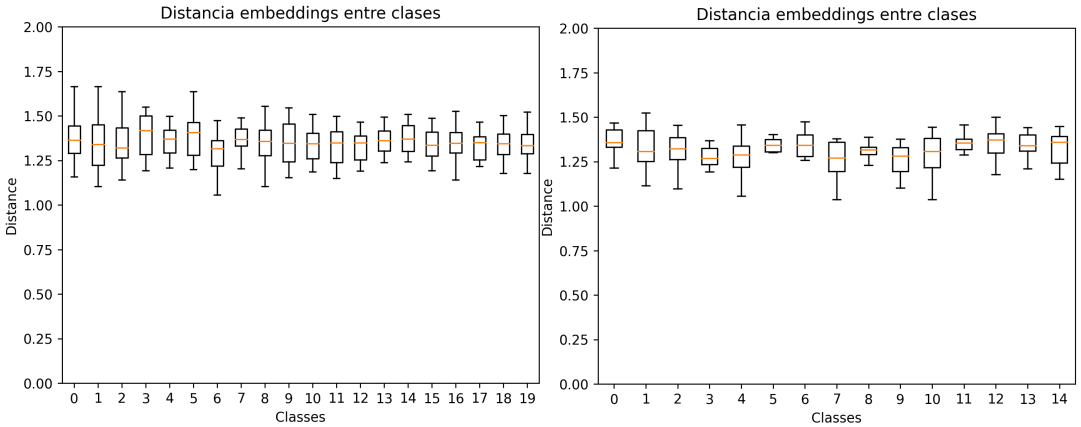


Figura 5.24 ROC y matriz de confusión Inception-ResNet V1 Preentrenada para base de datos propia. Elaboración propia.



(a) LFW. **(b)** Base propia.

Figura 5.25 Distancia entre clases de Inception-ResNet V1 Preentrenada. Elaboración propia.

Figura 5.26 TSNE Inception-ResNet V1 Preentrenada a base de datos propia. Elaboración propia.

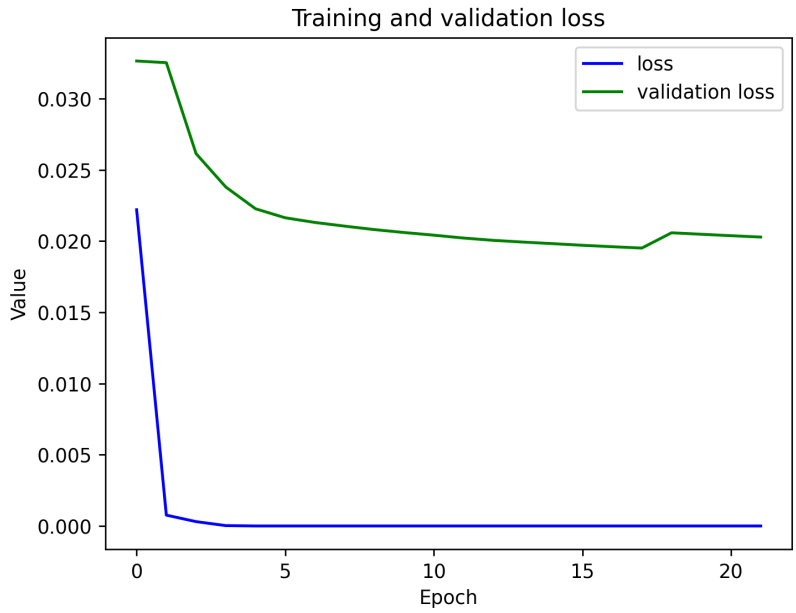


Figura 5.27 Función de pérdidas entrenamiento de Inception-ResNet V1 Preentrenada y transfer learning. Elaboración propia.

El rendimiento en la base de datos propia desciende ligeramente (Figura 5.28) aunque se sigue discriminando de manera efectiva los diferentes sujetos, hecho que se observa en la Figura 5.29. Por lo tanto, el *transfer learning* no ha resultado ser la mejor opción para el problema que nos concierne.

Para concluir, en la Tabla 5.3 se puede ver los diferentes valores de sensibilidad ofrecidos por cada una de las diferentes redes neuronales. Se refleja como la red preentrenada es la que mejor valor obtiene en este parámetro mientras que al realizar *transfer learning* este decrece. Respecto a las redes entrenadas desde cero, es claro que el mejor resultado ha sido con la ResNet-50.

5.2 Control Acceso

Una vez visto con las herramientas que contamos para poder predecir los *embeddings* de las diferentes imágenes, es el turno de implementar en un control de acceso. Esto se hará tanto en ordenador como en la

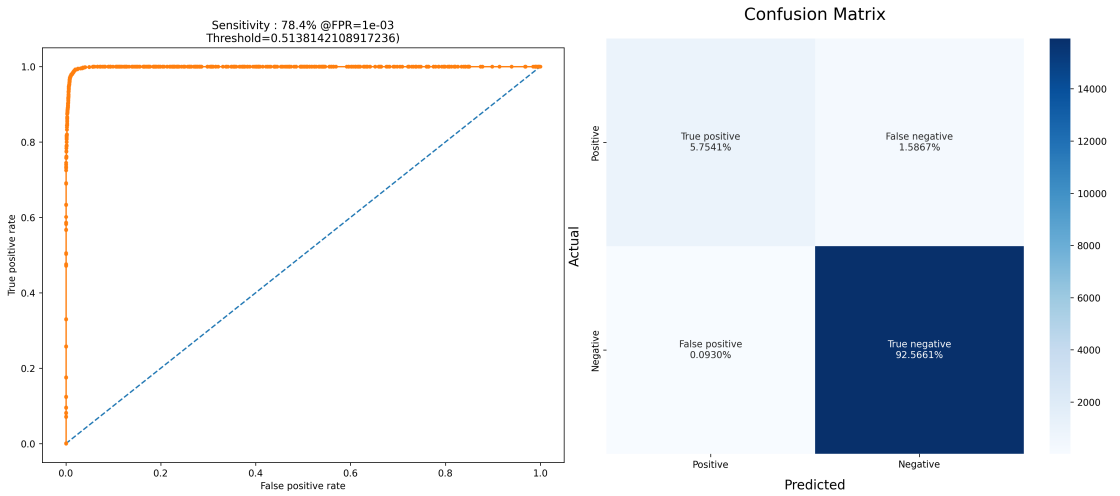


Figura 5.28 ROC y matriz de confusión Inception-ResNet V1 Preentrenada y transfer learning para base de datos propia. Elaboración propia.

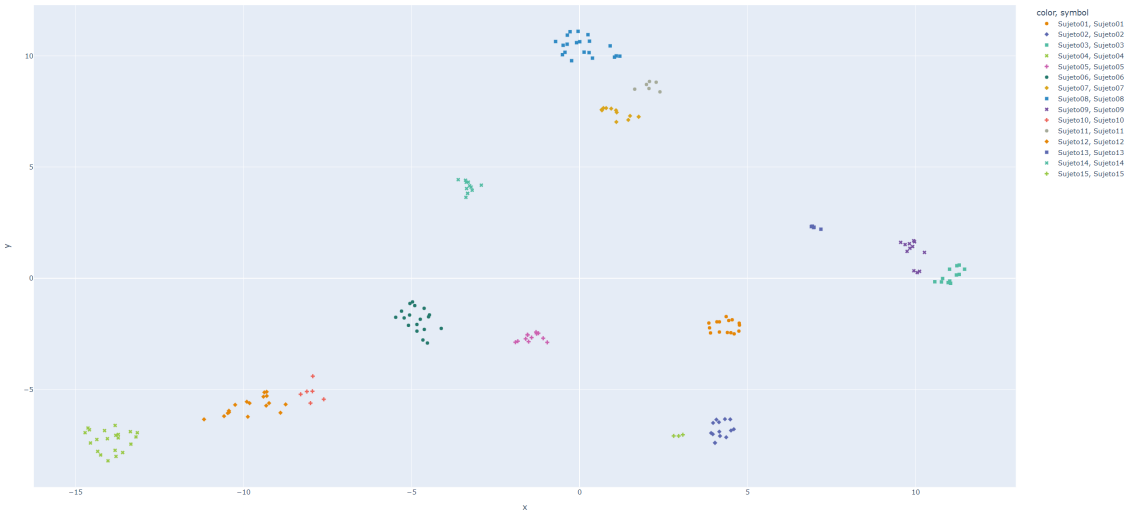


Figura 5.29 TSNE Inception-ResNet V1 Preentrenada y transfer learning a base de datos propia. Elaboración propia.

Tabla 5.3 Resumen sensibilidad redes neuronales sobre base de datos propia. Elaboración propia.

Red Neuronal	Sensibilidad
ResNet-50	55,3 %
Xception	41.6 %
Inception-ResNet V1	26,8 %
Inception-ResNet V1 Preentrenada	89,4 %
Inception-ResNet V1 Preentrenada y transfer learning	78,4 %

Raspberry para simular un sistema de bajo coste embebido. La red que se va a escoger para las pruebas es la ResNet-50 al ser el modelo entrenado que mejores resultados ha dado.

Las pautas que se seguirán son las expuestas en el capítulo anterior en la Figura 4.9. A modo de recordatorio, se tomarán imágenes de los sujetos a identificar, guardando sus prototipos en la base de datos. Esto formaría la parte *offline*, la parte en tiempo real sería detectar con una cámara al sujeto que intenta acceder, tomar una fotografía de su cara y compararla con los prototipos existentes: si es menor que un límite, se concede el acceso.

La parte de toma de imagen se hará mediante una webcam en el ordenador y mediante la Pi Camera en la

Raspberry. La interfaz es la siguiente, se ejecuta el archivo python y mediante un bucle se va tomando *frames* de la cara del sujeto. Si esta imagen, comparando la distancia, coincide con algún prototipo de la base de datos, se da el visto bueno y autoriza a la persona, recuadrando su cara y dando un mensaje de bienvenida. De forma síncrona, en la consola se muestra las coincidencias que se han encontrado junto al valor de la distancia y al tiempo que se ha tardado en hacer la inferencia.

Los resultados utilizando el ordenador personal se muestran en la Figura 5.30. Se observa como el sujeto Jaime es identificado, con su distancia asociada y un tiempo de inferencia de media de 0.17 segundos.

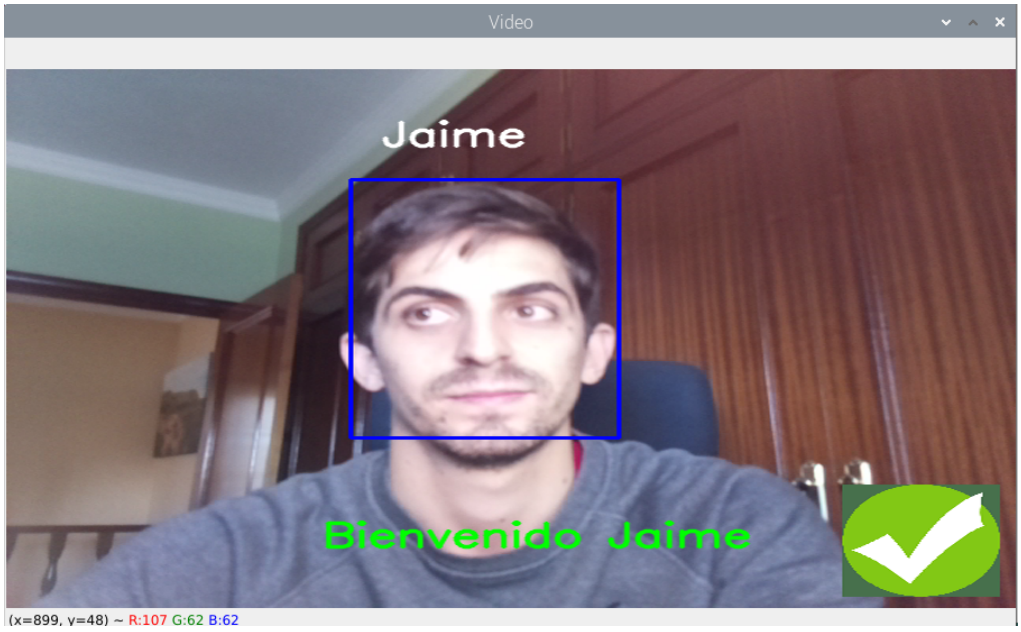
(a) a.

```
Hallado coincidencia con Jaime
Tiempo transcurrido: 0.166029 segundos.
La distancia es: 0.2763206660747528
Hallado coincidencia con Jaime
Tiempo transcurrido: 0.174038 segundos.
La distancia es: 0.2932640314102173
Hallado coincidencia con Jaime
Tiempo transcurrido: 0.185040 segundos.
La distancia es: 0.31594422459602356
Hallado coincidencia con Jaime
Tiempo transcurrido: 0.161037 segundos.
La distancia es: 0.3747425377368927
```

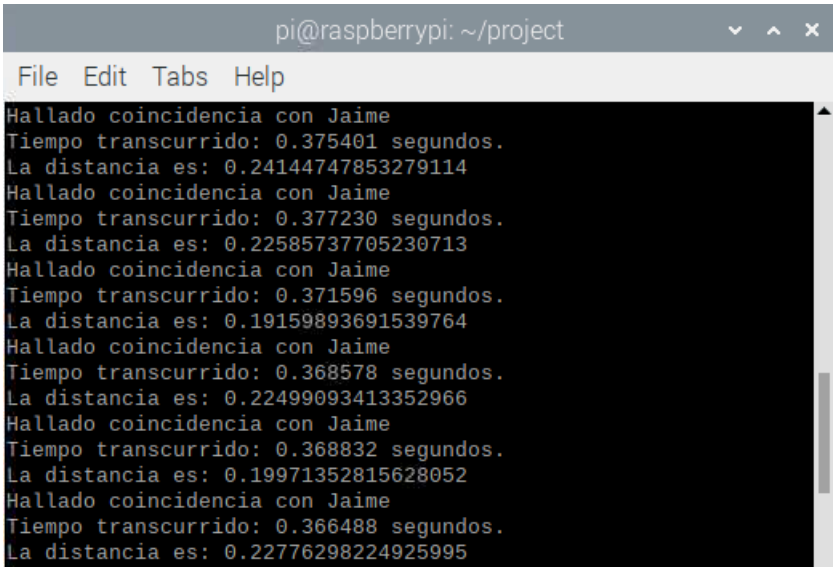
(b) b.

Figura 5.30 Caso práctico ordenador: a) Detección sujeto b) Tiempo de inferencia. Elaboración propia.

En el caso de la Raspberry, el resultado práctico se refleja en la Figura 5.31. En esta ocasión, la detección del sujeto se hace de manera satisfactoria pero el tiempo de inferencia crece al doble, debido a las limitaciones técnicas de un sistema como la Raspberry. Aun así, son valores bastante aceptables para la aplicación a tiempo real en un control de acceso.



(a) a.



(b) b.

Figura 5.31 Caso práctico Raspberry: a) Detección sujeto b) Tiempo de inferencia. Elaboración propia.

6 Conclusiones

En este capítulo, se van a recapitular las conclusiones extraídas durante el presente trabajo. El objetivo ha sido presentar una aplicación que pudiera detectar sujetos de una manera resolutiva. Para esto, se ha hecho uso de la herramienta tan actual y potente como son las CNN. La idea que regía el devenir del proyecto era poder discernir entre sujetos a priori no conocidos. Es importante destacar esto ya que muchas soluciones se basan en redes que clasifican entre sujetos conocidos, pero ante una nueva persona no tienen la capacidad para distinguir quién es.

De tal forma, se ha optado por la disciplina *one-shot learning*, la cual tiene como premisa mirar dos imágenes nunca anteriormente vistas y poder discriminar si son la misma persona. En este mundo existen alternativas para tratar esta problemática, optando en este trabajo por utilizar la función del Triplet Loss.

Triplet Loss es una función con la que se ha demostrado los resultados satisfactorios que puede ofrecer, alejando los *embeddings* de sujetos diferentes y acercando los de la misma clase. El problema que trae consigo esta técnica es la carga computacional, se trabaja con redes siamesas en las que hay que calcular la función de pérdida para una entrada formada por un triplete de fotografías, ralentizando el entrenamiento.

Por esta razón, al estar limitado en recursos al trabajar con un ordenador personal, el aprendizaje ha tenido que ser efectuado con una base de datos de tamaño moderado (cerca de las 200 mil imágenes). Aún con todo esto, se han conseguido unos buenos resultados que se pueden extrapolar a dispositivos de prestaciones limitadas como es el caso de la Raspberry: con su HW y una Pi Camera el control de acceso ha sido llevado a cabo.

En definitiva, se ha demostrado el potencial que la función Triplet Loss puede aportar en el control de acceso, pudiéndose aplicar en el mundo de las empresas para facilitar la organización empresarial y, por ende, mejorar el funcionamiento de esta.

7 Futuras ampliaciones

En este último capítulo del trabajo fin de máster, se comentarán las posibles ampliaciones a realizar para expandir los límites de este.

Un primer acercamiento sería entrenar una red neuronal haciendo uso de una base de datos con un número de imágenes en el orden del millón y que presenten gran variabilidad entre ellas para que la red aprenda ante cualquier situación. Esto supondría un coste computacional tal que con un ordenador corriente sería imposible afrontar, se necesitaría de un sistema especialmente enfocado en este trabajo para procesar esta cuantiosa información.

Por otra parte, una parte crucial en todo lo que se ha trabajado es la seguridad intrínseca que el sistema puede llegar a ofrecer ante circunstancias no deseadas. Una forma de vulnerar el control de acceso podría ser utilizando imágenes 2D de fotos reales del sujeto que sí cuenta con la autenticación positiva o incluso el uso de los tan famosos *deepfakes* que están cambiando el devenir del mundo audiovisual.

Existen diversas técnicas para solventar esta problemática (Figura 7.1). En (Denning, 2001), se acuño el término de *liveness* para hacer referencia a que un sistema biométrico no tiene que ser algo secreto que no se pueda mostrar como si de una contraseña se tratase, todo lo contrario: lo que se busca es que el sistema pueda detectar vida y, por ende, identificar correctamente al sujeto por su voz, su cara o su huella, evitando así posibles falsificaciones: esto se conoce como *liveness detection*.

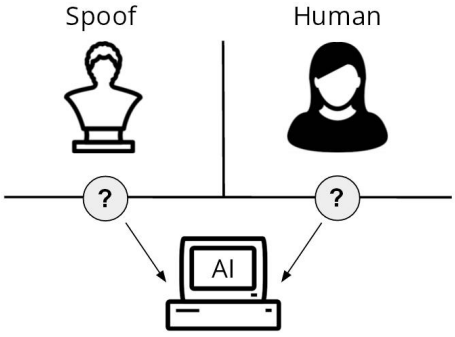


Figura 7.1 Problemática entre sujetos reales e imágenes. (Tussy y cols., 2022).

Las casuísticas de fraude son variadas, existiendo ciertos niveles de dificultad según la complejidad, empezando desde máscaras que se aplican para simular una cara real hasta simplemente inyectar en el sistema las imágenes directamente, sin pasar por el filtro que sería la cámara.

Por lo tanto, esto es algo que se vuelve realmente problemático y a las que muchas empresas han intentado dar respuesta. Una de la más brillantes y puesta en práctica en más número de usuarios es el Face ID que se comentó en el capítulo de Introducción. Según Apple, la probabilidad de que un sujeto externo acceda al teléfono es de una entre un millón (su anterior tecnología basada en huella, Touch ID, era 20 veces mayor). Consiguen detectar la autenticidad de la persona orquestando el uso de diferentes herramientas que se pueden ver en la Figura 7.2, destacando el proyector de puntos que mapea la cara en 3D y la cámara infrarroja para tomar imágenes a esta frecuencia.

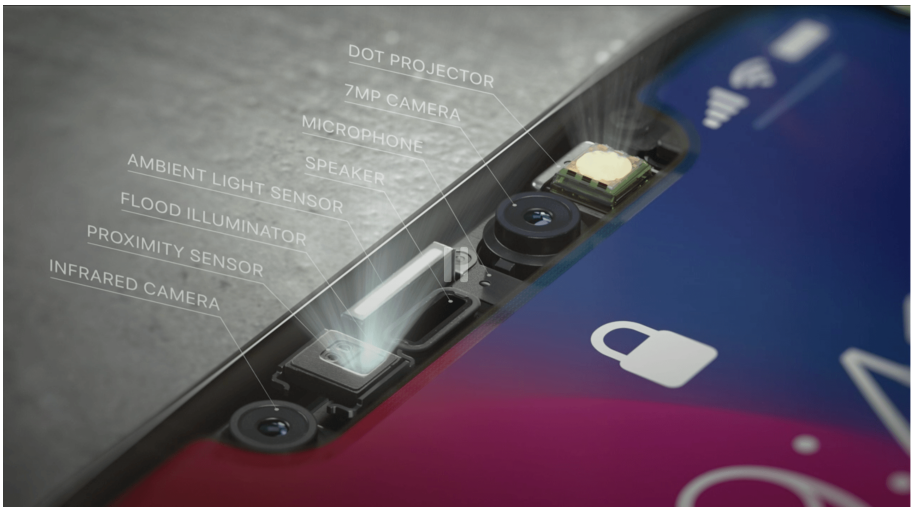


Figura 7.2 Componentes del Face ID. (Perekalin, 2017).

Utilizando el proyector y la cámara infrarroja, apoyándose con el *flood iluminator* (ilumina la cara del sujeto), se consigue una seguridad excelente. Cabe destacar que cuando hay ligeros cambios en el sujeto que hacen que la detección no sea correcta, al introducir manualmente la contraseña del teléfono, se vuelve a mapear el rostro para futuras verificaciones.

En síntesis, lo que se busca es maximizar la seguridad y la fiabilidad de un sistema biométrico para el control de acceso, y para esto podemos destacar las siguientes formas. Una de las maneras sería enfocarse en el propio algoritmo que rige la identificación, aumentando la base de datos y por tanto la complejidad de entreno, mientras que otro sería dar herramientas auxiliares al algoritmo para que, evaluando en cascada, la verificación se vaya haciendo de forma escalonada con el fin de evitar autenticaciones erróneas.

El mundo tecnológico está en un continuo cambio donde buscamos relegar tareas a las inteligencias artificiales para facilitarnos el día a día. En este camino, ciertos esfuerzos tienen que hacerse ya que hasta que una tecnología está madura muchos son los problemas que van apareciendo, sin contar que en paralelo surgen intentos de destrozar estos avances. Por tanto, el camino a seguir sería perfeccionar la técnica, pero sin olvidarse de todas las premisas que lo hacen robusto ante cualquier situación adversa que pueda ocurrir.

Índice de Figuras

Índice de Tablas

Referencias

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Devin, M. (2019). *Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arxiv 2016.*

Adamczyk, J. (2021). *Robust face detection with mtcnn | towards data science.* Descargado de `https://towardsdatascience.com/robust-face-detection-with-mtcnn-400fa81adc2e`

Apple. (2017). *Face id - wikipedia.* Descargado de `https://en.wikipedia.org/wiki/Face_ID`

Bansal, A., Nanduri, A., Castillo, C. D., Ranjan, R., y Chellappa, R. (2018). Umdfaces: An annotated face dataset for training deep networks. En (Vol. 2018-January). doi: 10.1109/BTAS.2017.8272731

Barnouti, N. H., Al-dabbagh, S. S. M., y Al-bamarni, M. H. J. (2016). Real-time face detection and recognition using principal component analysis (pca)-back propagation neural network (bpnn) and radial basis function (rbf). *Journal of Theoretical and Applied Information Technology*, *15*. Descargado de `www.jatit.org`

Baron, R. J. (1981, 8). Mechanisms of human facial recognition. *International Journal of Man-Machine Studies*, *15*, 137-178. doi: 10.1016/S0020-7373(81)80001-6

Bawarith, R., Basuhail, A., Fattouh, A., y Gamalel-Din, S. (2017). E-exam cheating detection system. *IJACSA) International Journal of Advanced Computer Science and Applications*, *8*. Descargado de `www.ijacsa.thesai.org`

Belhumeur, P. N., Hespanha, J. P., y Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*. doi: 10.1109/34.598228

Bradski, G. (2000). The opencv library. *Dr. Dobb's Journal of Software Tools*.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., y Zisserman, A. (2018). Vggface2: A dataset for recognising faces across pose and age. Descargado de `http://www.robots.ox.ac.uk/`

Carlos-Roca, L. R., Torres, I. H., y Tena, C. F. (2018, 10). Facial recognition application for border control. *Proceedings of the International Joint Conference on Neural Networks*, *2018-July*. doi: 10.1109/IJCNN.2018.8489113

Chanda, S., Gv, A. C., Brun, A., Hast, A., Pal, U., y Doermann, D. (2019, 11). Face recognition - a one-shot learning perspective. *Proceedings - 15th International Conference on Signal Image Technology and Internet Based Systems, SISITS 2019*, 113-119. doi: 10.1109/SITIS.2019.00029

Chicco, D. (2021). Siamese neural networks: An overview. *Methods in Molecular Biology*, *2190*, 73-94. Descargado de `https://link.springer.com/protocol/10.1007/978-1-0716-0826-5_3` doi: 10.1007/978-1-0716-0826-5_3/FIGURES/1

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. En (Vol. 2017-January). doi: 10.1109/CVPR.2017.195

del Rio, J. S., Moctezuma, D., Conde, C., de Diego, I. M., y Cabello, E. (2016, 9). Automated border control e-gates and facial recognition systems. *Computers Security*, *62*, 49-72. doi: 10.1016/J.COSE.2016.07.001

Denning, D. E. (2001). Why i love biometrics it's "liveness," not secrecy, that counts. Descargado de `www.sensar.com`

Frossard, D. (2016). *Vgg in tensorflow · davi frossard.* Descargado de `https://www.cs.toronto.edu/~frossard/post/vgg16/`

Guo, Y., Zhang, L., Hu, Y., He, X., y Gao, J. (2016). Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. En (Vol. 9907 LNCS). doi: 10.1007/978-3-319-46487-9_6

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). *Array programming with numpy* (Vol. 585). doi: 10.1038/s41586-020-2649-2

Hatem, H., Beiji, Z., y Majeed, R. (2015, 5). A survey of feature base methods for human face detection. *International Journal of Control and Automation*, *8*, 61-78. doi: 10.14257/IJCA.2015.8.5.07

He, K., Zhang, X., Ren, S., y Sun, J. (2015, 12). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 770-778. Descargado de https://arxiv.org/abs/1512.03385v1 doi: 10.48550/arxiv.1512.03385

He, X., Yan, S., Hu, Y., Niyogi, P., y Zhang, H. J. (2005). Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*. doi: 10.1109/TPAMI.2005.55

Huang, G., Liu, Z., van der Maaten, L., y Weinberger, K. Q. (2017). *Densely connected convolutional networks.* Descargado de https://github.com/liuzhuang13/DenseNet.

Huang, G. B., Mattar, M., Berg, T., Learned-Miller, E., y Learned-Miller, E. (2008). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. Descargado de https://hal.inria.fr/inria-00321923https://hal.inria.fr/inria-00321923/document

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science and Engineering*, *9*. doi: 10.1109/MCSE.2007.55

Hurst, A. C. (2018). *Facial recognition software in clinical dysmorphology* (Vol. 30). doi: 10.1097/MOP.0000000000000677

Imran, M. A., Rahman, H., y Karmaker, D. (2015). Face recognition using eigenfaces. *International Journal of Computer Applications*, *118*, 975-8887.

Jain, V., y Learned-Miller, E. (2010). Fddb: A benchmark for face detection in unconstrained settings. Descargado de http://news.yahoo.com

Jayanthy, S., Anishkka, J. B., Deepthi, A., y Janani, E. (2019, 5). Facial recognition and verification system for accessing patient health records. *2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019*, 1266-1271. doi: 10.1109/ICCS45141.2019.9065469

Kadir, K., Kamaruddin, M. K., Nasir, H., Safie, S. I., y Bakti, Z. A. K. (2015, 1). A comparative study between lbp and haar-like features for face detection using opencv. *2014 4th International Conference on Engineering Technology and Technopreneuship, ICE2T 2014*, *2014-August*, 335-339. doi: 10.1109/ICE2T.2014.7006273

Kaur, P., Krishan, K., Sharma, S. K., y Kanchan, T. (2020, 1). Facial-recognition algorithms: A literature review. *https://doi.org/10.1177/0025802419893168*, *60*, 131-139. Descargado de https://journals.sagepub.com/doi/abs/10.1177/0025802419893168 doi: 10.1177/0025802419893168

Khosla, D. (2010). Fingerprint identification in biometric securitysystems. *Article in International Journal of Computer and Electrical Engineering*, *2*. Descargado de https://www.researchgate.net/publication/271304334 doi: 10.7763/IJCEE.2010.V2.239

Kumar, A., Kaur, A., y Kumar, M. (2019, 8). Face detection techniques: a review. *Artificial Intelligence Review*, *52*, 927-948. Descargado de https://link.springer.com/article/10.1007/s10462-018-9650-2 doi: 10.1007/S10462-018-9650-2/FIGURES/13

Li, C., Wang, R., Li, J., y Fei, L. (2020). Face detection based on yolov3. *Advances in Intelligent Systems and Computing*, *1031 AISC*, 277-284. Descargado de https://link.springer.com/chapter/10.1007/978-981-13-9406-5_34 doi: 10.1007/978-981-13-9406-5_34/TABLES/1

Li, H., Lin, Z., Shen, X., Brandt, J., y Hua, G. (2015). *A convolutional neural network cascade for face detection.*

Liu, Z., Luo, P., Wang, X., y Tang, X. (2015). *Deep learning face attributes in the wild.* Descargado de http://personal.ie.cuhk.edu.hk/

Ma, L. H., Fan, H. Y., Lu, Z. M., y Tian, D. (2020, 9). Acceleration of multi-task cascaded convolutional networks. *IET Image Processing*, *14*, 2435-2441. Descargado de https://onlinelibrary.wiley.com/doi/full/10.1049/iet-ipr.2019.0141https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-ipr.2019.0141https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-ipr.2019.0141 doi: 10.1049/IET-IPR.2019.0141

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *11*. doi: 10.1109/34.192463

Moindrot, O. (2018). *Triplet loss and online triplet mining in tensorflow | olivier moindrot blog.* Descargado de https://omoindrot.github.io/triplet-loss

Nech, A., y Kemelmacher-Shlizerman, I. (2017). Level playing field for million scale face recognition. En (Vol. 2017-January). doi: 10.1109/CVPR.2017.363

Papageorgiou, C. P., Oren, M., y Poggio, T. (1998). General framework for object detection.. doi: 10.1109/iccv.1998.710772

Parkhi, O. M., Vedaldi, A., y Zisserman, A. (2015). Deep face recognition.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Édouard Duchesnay (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*.

Perekalin, A. (2017). *¿qué es face id de apple y cómo de seguro es? | blog oficial de kaspersky.* Descargado de https://www.kaspersky.es/blog/apple-face-id-security/14364/

Pi, F. R. (2020). *Raspberry pi os - wikipedia, la enciclopedia libre.* Descargado de https://es.wikipedia.org/wiki/Raspberry_Pi_OS

Rashid, R. A., Mahalin, N. H., Sarijari, M. A., y Aziz, A. A. A. (2008). Security system using biometric technology: Design and implementation of voice recognition system (vrs). *Proceedings of the International Conference on Computer and Communication Engineering 2008, ICCCE08: Global Links for Human Development*, 898-902. doi: 10.1109/ICCCE.2008.4580735

Redmon, J., Divvala, S., Girshick, R., y Farhadi, A. (2015, 6). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 779-788. Descargado de https://arxiv.org/abs/1506.02640v5 doi: 10.48550/arxiv.1506.02640

Redmon, J., y Farhadi, A. (2018, 4). Yolov3: An incremental improvement. Descargado de https://arxiv.org/abs/1804.02767v1 doi: 10.48550/arxiv.1804.02767

Rosebrock, A. (2017). *Imagenet: Vggnet, resnet, inception, and xception with keras - pyimagesearch.* Descargado de https://pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

Rossum, G. V., Drake, F. L., Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., ... Oliphant, T. E. (2009). *Python 3 reference manual* (Vol. 585).

Rowley, H. A., Baluja, S., y Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 23-38. doi: 10.1109/34.655647

Schroff, F., Kalenichenko, D., y Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. En (Vol. 07-12-June-2015). doi: 10.1109/CVPR.2015.7298682

Simonyan, K., y Zisserman, A. (2014, 9). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. Descargado de https://arxiv.org/abs/1409.1556v6 doi: 10.48550/arxiv.1409.1556

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. En (Vol. 07-12-June-2015). doi: 10.1109/CVPR.2015.7298594

Taigman, Y., Yang, M., Ranzato, M., y Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification.. doi: 10.1109/CVPR.2014.220

Taniai, H. (2018). *nyoki-mtl/keras-facenet: Facenet implementation by keras2.* Descargado de https://github.com/nyoki-mtl/keras-facenet

Turk, M., y Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, *3*. doi: 10.1162/jocn.1991.3.1.71

Tussy, K. A., Wojewidka, J., y Rose, J. (2022). *Liveness.com - biometric liveness detection explained.* Descargado de https://www.liveness.com/

Tyagi, M. (2021). *Viola jones algorithm and haar cascade classifier | by mrinal tyagi | towards data science.* Descargado de https://towardsdatascience.com/viola-jones-algorithm-and-haar-cascade-classifier-ee3bfb19f7d8

Uri, E. (2020). *Converting sandberg's facenet pre-trained model to tensorflow lite (using an "unorthodox way") | by esteban uri | medium.* Descargado de https://medium.com/@estebanuri/converting-sandbergs-facenet-pre-trained-model-to-tensorflow-lite-using-an-unorthodox-way-7ee3a6ed02a3

Valdes-Ramirez, D., Medina-Perez, M. A., Monroy, R., Loyola-Gonzalez, O., Rodriguez, J., Morales, A., y Herrera, F. (2019). *A review of fingerprint feature representations and their applications for latent fingerprint identification: Trends and evaluation* (Vol. 7). doi: 10.1109/ACCESS.2019.2909497

Venayagamoorthy, G. K., Sandrasegaran, V. K. M., Moonasar, V., Sandrasegaran, K., Sandrasegaran, K., y massey ac nz. (1998). Scholars'mine voice recognition using neural networks voice recognition using neural networks. Descargado de http://scholarsmine.mst.edu/ele_comeng_facworkhttp://dx.doi.org/10.1109/COMSIG.1998.736916 doi: 10.1109/COMSIG.1998.736916

Viola, P., y Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *1*. doi:

10.1109/CVPR.2001.990517

Wang, C.-F. (2018). *A basic introduction to separable convolutions | by chi-feng wang | towards data science.* Descargado de https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Wang, F., Chen, L., Li, C., Huang, S., Chen, Y., Qian, C., y Loy, C. C. (2018). The devil of face recognition is in the noise. En (Vol. 11213 LNCS). doi: 10.1007/978-3-030-01240-3_47

Wang, M., y Deng, W. (2021). Deep face recognition: A survey. *Neurocomputing*, *429*. doi: 10.1016/j.neucom.2020.10.081

Waskom, M. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, *6*. doi: 10.21105/joss.03021

Yang, W., Wang, S., Hu, J., Zheng, G., y Valli, C. (2019, 1). Security and accuracy of fingerprint-based biometrics: A review. *Symmetry 2019, Vol. 11, Page 141*, *11*, 141. Descargado de https://www.mdpi.com/2073-8994/11/2/141/htmhttps://www.mdpi.com/2073-8994/11/2/141 doi: 10.3390/SYM11020141

Yi, D., Lei, Z., Liao, S., y Li, S. Z. (2014, 11). Learning face representation from scratch. Descargado de https://arxiv.org/abs/1411.7923v1 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*. doi: 10.1109/LSP.2016.2603342

Zhang, X., Gonnot, T., Saniie, J., Zhang, X., Gonnot, T., y Saniie, J. (2017, 5). Real-time face detection and recognition in complex background. *Journal of Signal and Information Processing*, *8*, 99-112. Descargado de http://www.scirp.org/journal/PaperInformation.aspx?PaperID=76264http://www.scirp.org/Journal/Paperabs.aspx?paperid=76264 doi: 10.4236/JSIP.2017.82007

Zhang, Y., Deng, W., Wang, M., Hu, J., Li, X., Zhao, D., y Wen, D. (2020). Global-local gcn: Large-scale label noise cleansing for face recognition.. doi: 10.1109/CVPR42600.2020.00775

Ángel, E., y Ambrogio, J. (2020). ArtÍculos presentados a radi | tecnologÍa de la informaciÓn y comunicaciÓn.

Glosario

ABC Automated border control. 4

API Application Programming Interface. 20

CNN Convolutional neural network. 7, 17, 20, 41

DL Deep Learning. 10, 17

FN False negative. 26

FP False positive. 26

FPR False positive rate. 26

GPU Graphics processing unit. 8

HW Hardware. 19, 20, 41

LDA Linear Discriminant Analysis. 9

LFW Labeled Faces in the Wild. 22, 25, 31, 32, 45

LPP Locality Preserving Projections. 9

ML Machine learning. 5, 20

MTCNN Multitask Cascaded Convolutional Networks. 7, 8, 45, 47, IX

NN Neural network. 5, 7, 9, 12, 13, 20, IX

PCA Principal component analysis. 5, 9, IX

ROC Receiver Operating Characteristic. 26, 32

SNN Siamese Neural Network. 13, 14, 45

SW Software. 19

TN True negative. 26

TP True positive. 26

TSNE T-distributed Stochastic Neighbor Embedding. 26, 28, 32, 34

YOLO You Only Look Once. 8