GRADO EN INGENIERÍA TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

DESARROLLO DE UNA APP DE RECONOCIMIENTO FACIAL CON TÉCNICAS DE INTELIGENCIA ARTIFICIAL

Autor: Alberto Menéndez Ruiz de Azúa

Director: Álvaro Jesús López López

Madrid

Junio de 2019

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. ALBERTO MENÉNDEZ RUIZ DE AZUA

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: "DESARROLLO DE UNA APP DE RECONOCIMIENTO FACIAL CON TÉCNICAS DE INTELIGENCIA ARTIFICIAL" que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

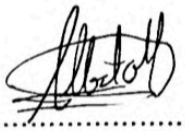
6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a ...12.... de ....Junio..... de ..2019

ACEPTA



Fdo.....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título "Desarrollo de una App de reconocimiento facial con técnicas de inteligencia artificial" en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2018/2019 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Alberto Menéndez Ruiz de Azúa Fecha: 12/06/2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Álvaro Jesús López López Fecha: 12/06/19

GRADO EN INGENIERÍA TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

DESARROLLO DE UNA APP DE RECONOCIMIENTO FACIAL CON TÉCNICAS DE INTELIGENCIA ARTIFICIAL

Autor: Alberto Menéndez Ruiz de Azúa

Director: Álvaro Jesús López López

Madrid

Junio de 2019

UNIVERSIDAD PONTIFICIA COMILLAS
Escuela Técnica Superior de Ingeniería (ICAI)
Grado en Ingeniería en Tecnologías Industriales

DESARROLLO DE UNA APP DE RECONICIMIENTO FACIAL CON TÉCNICAS DE INTELIGENCIA ARTIFICIAL

Autor: **Menéndez Ruiz de Azúa, Alberto.**

Director: López López, Álvaro Jesús

Entidad colaboradora: Cátedra de Industria Conectada del Instituto de Investigación Tecnológica.

RESUMEN DEL PROYECTO

Introducción

Este proyecto surge como reacción y solución a las exigencias de la cuarta revolución industrial, la cual trae consigo una demanda de una constante mejora y modernización por parte de las empresas. El objetivo de estos procesos de innovación es el de la optimización de procesos y servicios por parte de cualquier empresa y el de no quedarse atrás con respecto a los competidores. La digitalización e implantación de proyectos de Industria 4.0 para ofrecer mejores y más rápidos servicios a los clientes es algo que toda empresa industrial que quiera estar a la cabeza de su sector ha de hacer. No adoptar este tipo de iniciativas da la oportunidad a la competencia de hacerlo y ser ellos quien se posicionen a la cabeza del sector.

El proyecto de reconocimiento facial planteado en este documento busca poder aligerar los procesos que requieran la identificación de un individuo, como pueden ser el acceso del personal a las instalaciones de una empresa industrial o la comprobación de identidad de clientes a la hora de pagar en el sector retail. La implantación de este tipo de tecnología supondría una reducción drástica de los tiempos de espera que actualmente existen en este tipo de situaciones. La aplicación de la técnica propuesta en este documento en los casos mencionados supondría la sustitución de las acreditaciones o tarjetas, empleadas actualmente en estos procesos, por la propia cara del usuario. Esto supone una reducción del tiempo que se pierde buscando la tarjeta o acreditación a la hora de usarla y soluciona los posibles casos de pérdida u olvido de la documentación por parte del usuario.

Inicialmente, esta iniciativa surgió por una propuesta de una empresa del sector retail de elaborar un sistema de reconocimiento facial orientado a facilitar el proceso de identificación de sus clientes Premium una vez que fueran a pagar la compra y, de esta manera, poder ofrecerles de manera automática las promociones que mejor se ajusten a sus gustos en base a compras previas. Las demás aplicaciones del proyecto, mencionadas anteriormente, fueron apareciendo a medida que se iba desarrollando el proyecto y, puesto que el proyecto se ha desarrollado en el contexto de la Cátedra de Industria Conectada, estas aplicaciones poseen un marcado carácter industrial.

La decisión de hacer la aplicación para dispositivos IOS viene fundamentada por los grandes avances en sus librerías de detección facial y los buenos resultados al ser empleadas en ejemplos prácticos, que las colocan a la vanguardia en lo que al estado del arte se refiere. Los ejemplos de casos de uso de estas librerías en dispositivos IOS más similares a lo que se busca en este proyecto son, la aplicación de Face ID del Iphone X o algunas apps de reconocimiento facial que utilizan las librerías de Visión y CoreML de

Apple, como puede ser la Photo App de la propia empresa de Apple. Sin embargo, las técnicas más importante empleadas en el proyecto no son dependientes del tipo de la tecnología IOS y son extrapolables a otros dispositivos electrónicos.

Metodología

Para el proyecto, tras un estudio exhaustivo del estado del arte, se decidió emplear una técnica holística de inteligencia artificial conocida como Redes Neuronales Siamesas, basándose en el caso concreto de uso de esta técnica por parte de Google llamado FaceNet. La técnica de FaceNet ha sido usada en proyectos similares al empleado para esta solución, como puede ser el de OpenFace [1], pero nunca en dispositivos IOS en la manera en q**¡Error! No se encuentra el origen de la referencia.**. FaceNet es un método holístico con enfoque de inteligencia artificial y las razones que llevaron a la elección de la técnica de FaceNet sobre las otras usadas para el mismo propósito han sido: el coste y disponibilidad de las tecnologías, la escalabilidad del proyecto, la capacidad de ser desarrollado en el plazo propuesto, la búsqueda de una solución que permitiera alcanzar el objetivo de *one-shot learning* y la flexibilidad ante futuras mejoras del proyecto; entre otros factores

La técnica de Redes Neuronales Siamesas empleada para la aplicación de reconocimiento facial se basa en el uso de una red neuronal convolucional, la cual genera un vector característico de 128 elementos cuando recibe una imagen de una cara de tamaño 160x160 pixeles. Este vector de 128 elementos es la manera que la red neuronal tiene de parametrizar cada cara y se podría decir que es como una forma sintetizada de la información que la red neuronal es capaz de extraer de la imagen de una cara. En la Figura 1 se puede ve un esquema de como funcionaría esta técnica

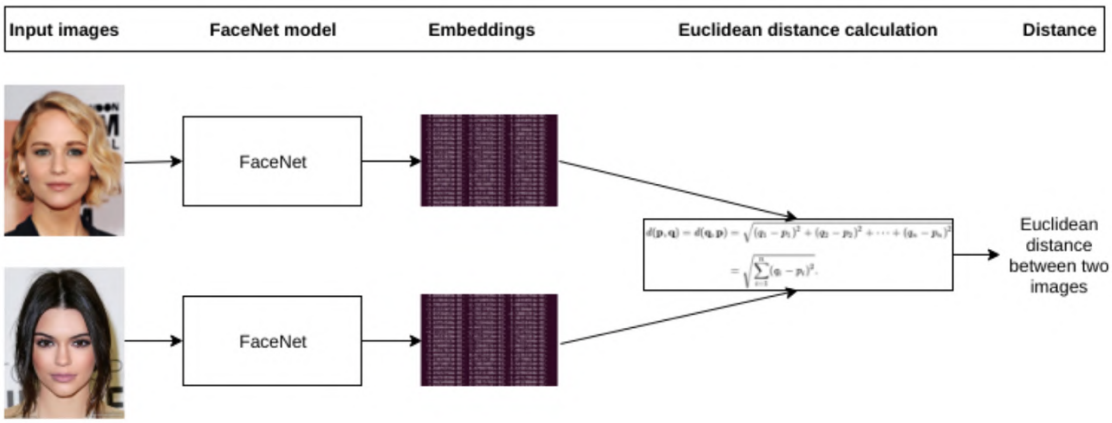


Figura 1: Ejemplo del funcionamiento de la red neuronal siamesa [2].

Esta red neuronal ha sido entrenada para minimizar la distancia en el espacio euclideo de dimensión 128 entre dos vectores de caras de una misma persona y, al mismo tiempo, para maximizar la distancia euclidea entre los vectores de cara pertenecientes a personas diferentes. La técnica de entrenamiento de la red neuronal para este tipo de aplicaciones

se llama Pérdida de Triplete, y fue presentada por primera vez por Google en el documento introductorio de FaceNet en 2015 [3].

En este proyecto se empleo un modelo pre entrenado de FaceNet, construido en Python sobre la estructura de red neuronal convolucional de Inception ResNet V1 y entrenado con la técnica de Pérdida de Triplete sobre el set de datos de MS-CELEBS 1M [4]. Este set de datos cuenta con 10 millones de imágenes correspondientes a 100 mil clases o personas diferentes. Se empleó un modelo pre entrenado por la simple razón de que el entrenamiento de la red es un proceso computacionalmente muy costoso, que requiere generalmente de meses para obtener resultados de calidad y este entrenamiento no era el objetivo principal del proyecto en si. El empleo de un modelo pre entrenado no supone una desventaja para el proyecto, si no todo lo contrario, ya que no hay necesidad en el entrenamiento de un modelo propio y usar un modelo pre entrenado permite contar con una red más fiable y que ha sido entrenada en un set de datos más rico que si tuviera que ser entrenada con unos recursos más limitados.

Para el proyecto esta red neuronal pre entrenada fue convertida a lenguaje Swift para poder emplearla en la aplicación de reconocimiento facial en los dispositivos IOS. Para la conversión de la red se empleo la librería de Coremltools de Apple, creada con el propósito de permitir la conversión de modelos de *machine learning* de Python a lenguaje Swift. Además, a la conversión del modelo se debió añadir una capa extra en Swift, llamada *Scaling Layer*, para poder convertir las capas Lambda que contiene modelo pre entrenado, las cuales no son convertibles por la librería de Coremltools.

A la hora de llevar a cabo el reconocimiento facial en la aplicación se realizan diferentes etapas y el primer paso consiste en la detección facial, es decir, determinar si hay una cara y donde está esa cara en la imagen que se está analizando. Para ello y para mejorar las prestaciones de la Red Neuronal Siamesa, se graba la realidad en tiempo real con la cámara del dispositivo y se van analizando los frames de este video en busca de caras. Para la detección facial se emplea la librería de Vision de Apple, que cuenta con las herramientas y funciones necesarias para detectar si hay caras en una imagen y en que parte de la imagen se encuentra cada una de esas caras. La herramienta de Vision emplea técnicas de *machine learning* como explicaron sus creadores en un artículo en la revista de Machine Learning Journal de Apple [5].

Una vez se ha determinado la posición de la cara en la imagen se procede a recortarla con un tamaño de 160x160 tomando como centro de la imagen el centro de la cara y dejando un pequeño margen en los bordes de la cara, ya que diversos usuarios que han trasteado con está técnica han comprobado que obtienen mejores resultados al emplear cierto margen [6]. Esta imagen recortada es la que a continuación se introduce en al modelo de la red neuronal convolucional, que había sido convertido a lenguaje Swift, para obtener el vector característico que le da este modelo a esa cara. Este vector es comparado a través de la distancia euclidea con los vectores de las caras que hay guardados en la base de datos de la aplicación y, de esta manera, se determina si esa cara que se ha detectado se corresponde con la de alguno de los usuarios que se encuentran en el sistema o por el contrario se trata de una persona desconocida.

En caso de que la cara se corresponda con la de algún usuario del sistema se coloca un cuadro rojo sobre la cara con el nombre del usuario encima y una voz dice el nombre del

usuario seguido de la palabra “Detectado”. Si, por el contrario, la cara no se corresponde con la de ningún usuario de los guardados en el sistema, se coloca el mismo recuadro rojo sobre la cara detectada, pero con la palabra “Desconocido” encima, como se puede ver en la Figura 2. En mejoras posteriores de la aplicación se incluyo la función de que junto al nombre de los usuarios detectados aparezca un valor con el porcentaje de fiabilidad que la aplicación le otorga a la decisión de que esa cara se corresponda con ese usuario.

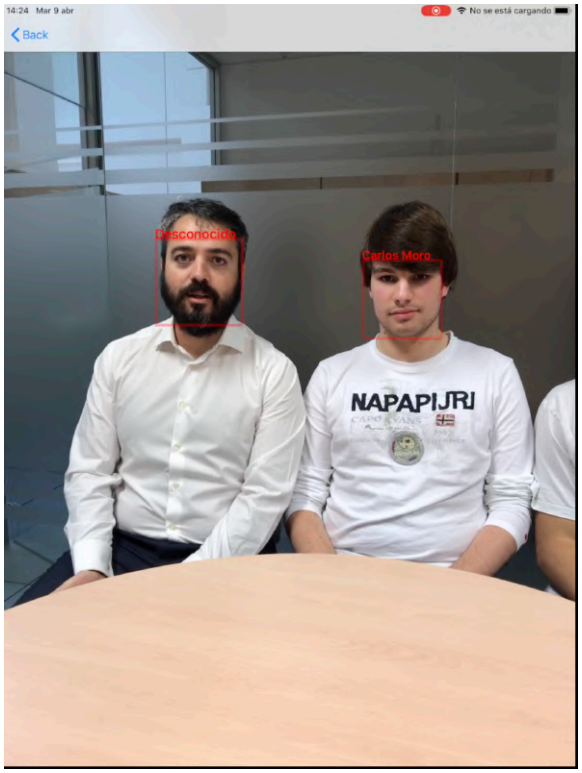


Figura 2: Pantalla de reconocimiento de la App en la que se ve a un usuario desconocido y a otro que si que se encuentra en el sistema

Resultados

El resultado final del proyecto fue una aplicación para cualquier dispositivo IOS, que ofrece la posibilidad de introducir o eliminar usuarios en una base de datos y que es capaz de reconocer a los usuarios que tenga registrados. El diseño de la estructura de la aplicación y de la apariencia de cada una de las pantallas de la aplicación es propio de este proyecto. En la Figura 3 se puede ver una muestra de las distintas pantallas que componen la aplicación y en la Figura 2 se puede ver la pantalla de reconocimiento facial, que es la que entraña mayor dificultad de la aplicación.

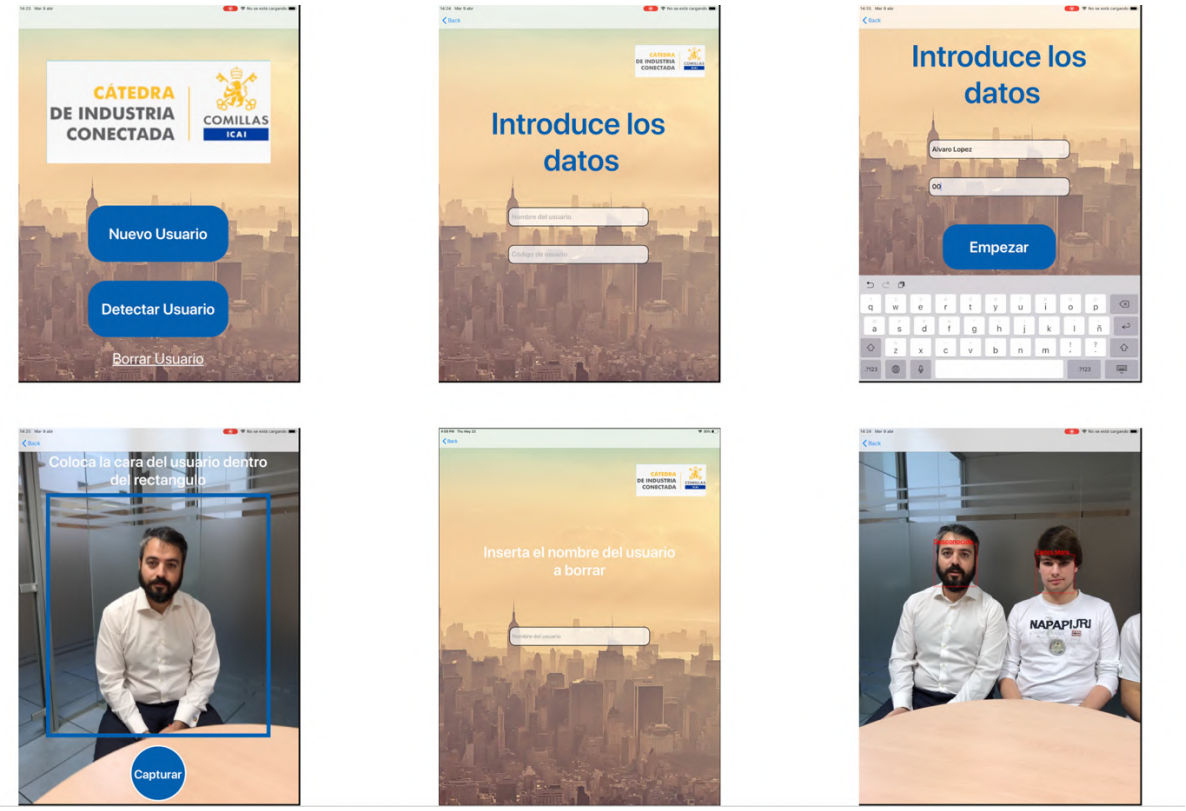


Figura 3: Muestra de lasa distintas pantallas de la App

Uno de los apartados de mayor valor de este proyecto viene del hecho de que a lo largo del desarrollo de este fueron apareciendo dificultades, que iban abriendo nuevos retos y requerían de estudios exhaustivos de lo que estaba ocurriendo. En este documento se busca plasmar tanto el proceso llevado a cabo hasta la consecución de la App de reconocimiento facial, como los resultados de los análisis de robustez llevados a cabo en cada etapa intermedia del proyecto. Estos resultados van acompañados de las conclusiones y decisiones de posibles modificaciones y mejoras que se han ido sacando a raíz de estos. Se busca, por lo tanto, no solo lograr un proyecto de calidad y robustez, sino analizar y dejar la puerta abierta a cualquier mejora que se pueda hacer en el futuro.

Es por ello, que en el apartado de resultados del proyecto y, de manera más breve, en el de conclusiones de este resumen se presentan diferentes estudios de robustez del modelo frente a diferentes cambios en las caras de entrada. Con esos análisis se busca conocer mejor la robustez de la aplicación, ver que dificultades o limitaciones presenta. Esto se hizo para tratar de encontrar soluciones que permitan dotar a la aplicación de una mayor robustez y solventar las posibles limitaciones que se detecten. En las conclusiones se exponen las mejoras que se plantean para el proyecto a raíz de los análisis de robustez elaborados, algunas de las cuales ya han sido implementadas.

Conclusiones

Tras elaborar la aplicación, esta fue sometida a diferentes análisis de robustez, que nos mostraron como la aplicación es dependiente en mayor o menos medida de cambios en la postura de la cara, el ángulo de giro de la persona respecto a la cámara, el margen de la imagen introducida en la red neuronal en torno a la cara, los cambios lineales y los cambios no lineales a los que se vea sometida la cara con respecto a la imagen con la que se hiciera el registro de esa cara. Para cada una de las limitaciones encontradas se ha propuesto una mejora y alguna de ellas ya ha sido implementada en la aplicación.

Para los cambios en el ángulo de giro se propuso lograr una mayor normalización de la cara antes de ser introducida en la red neuronal, que es una solución que ya se ha llevado a cabo en proyectos que emplean esta técnica como es el caso de OpenFace. Un ejemplo de lo que se pretende lograr con esta solución se puede ver en la Figura 4, en la cual se nos muestra el proceso que se llevaría a cabo ante la detección de una cara girada. Lo que se buscaría en esta solución sería emplear una de las herramientas de la librería de Vision, la cual nos permite determinar la posición de puntos determinados de la cara detectada. Tomando la línea que marca la mitad de la cara podemos determinar el valor del ángulo de giro de la cara con respecto al eje vertical y corregirlo antes de introducir la imagen en la red neuronal.

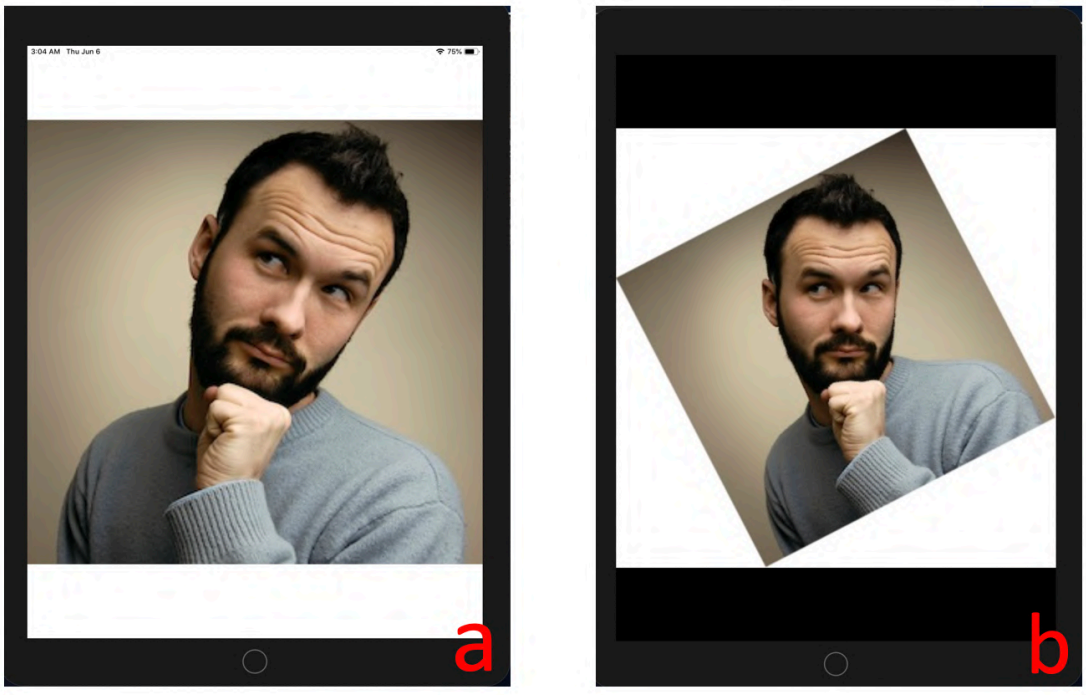


Figura 4: Ejemplo, con un pequeño código, de como funciona la mejora introducida. (a) Imagen que recibiría la App. (b) Imagen girada para poner la cara recta y que será recortada para introducir en la red neuronal.

Otra de las limitaciones que se vieron en la aplicación es la dependencia del modelo con el porcentaje de la cara que sea visible en la imagen, es decir, a los grados de giro que represente con respecto al eje vertical de la columna vertebral. Para solucionar este

problema se han estudiado distintos métodos, siendo el más prometedor el uso de técnicas de GANs (Generative Adversarial Networks), por sus siglas en inglés, o Redes Generativas Adversarias. Existen variantes de esta técnica como HoloGAN [7] o StyleGAN [8] o una técnica de Few-Shots Learning desarrollada por Samsung [9], que permiten generar un set de datos de imágenes de una persona en diferentes posiciones o estilos simplemente a partir de una sola imagen de esa persona. Algunos ejemplos de lo que son capaces estas técnicas se pueden ver en la Figura 6 y la Figura 6.

Esta técnica en combinación con un clasificador que pueda ser entrenado con el set de datos generado por la GAN, puede ser un complemento ideal para la aplicación de reconocimiento facial. De esta manera la aplicación sería capaz de reconocer a una persona en posiciones o con cambios en su aspecto con lo que no la ha visto anteriormente.

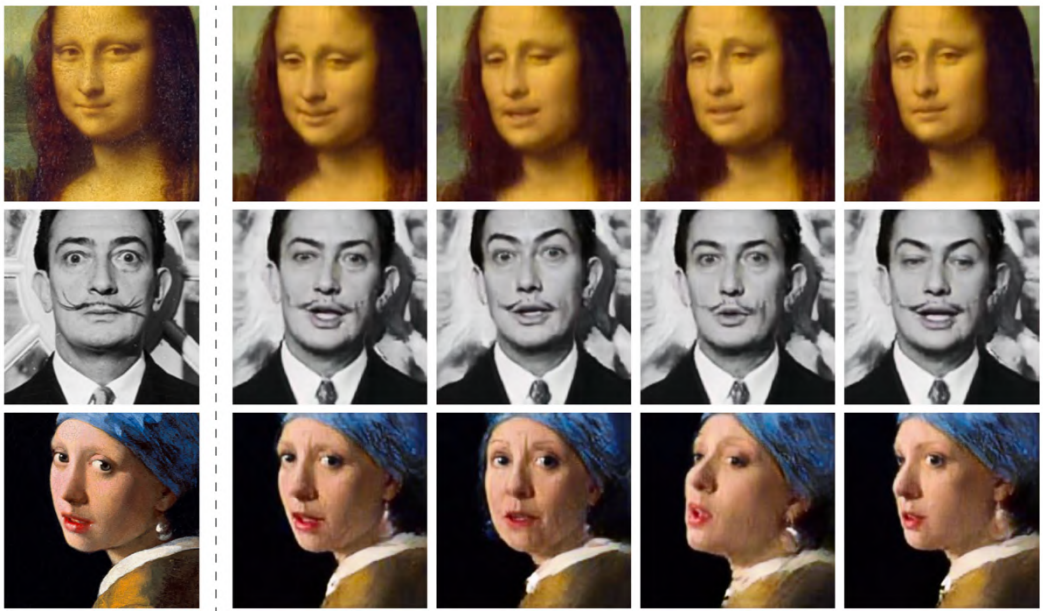


Figura 5: Ejemplo de resultados obtenidos con la técnica de Few-Shots Learning de Samsung [9].

Figura 6: Ejemplo de resultados de StyleGAN [8]

Esta última solución permite resolver la mayor parte de las limitaciones expuestas previamente, ya que tenían que ver con las dificultades encontradas a la hora de tener que enfrentar al modelo con el reto de reconocer a una persona a partir de una sola imagen inicial. Por eso el hecho de enriquecer el set de datos de esas personas empleando las GANs resulta tan efectivo para hacer frente a esta limitación.

Esta técnica de GANs puede ser usada en cualquier campo de la industria en el que por alguna razón solo se disponga de una serie reducida de datos o muestras útiles, ya que permite abstraer algunas características de estas muestras y generar datos nuevos nunca vistos antes, pero que mantengan la esencia y característica principal de los datos pertenecientes a ese set.

Referencias:

Bibliografía

[1] B. L. y. M. S. B. Amos, «OpenFace: A general-purpose face recognition,» de *IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Placid, NY, USA, Marzo 2016.

[2] D. Karunakaran, «Medium,» 27 Septiembre 2018. [En línea]. Available: https://medium.com/intro-to-artificial-intelligence/one-shot-learning-explained-using-facenet-dff5ad52bd38. [Último acceso: Noviembre 2018].

[3] F. Schroff, D. Kalenichenko y J. Philbin, «FaceNet: A Unified Embedding for Face Recognition and Clustering,» 17 Junio 2015.

[4] H. T. (nyoki-mtl), «GitHub,» Febrero 2018. [En línea]. Available: https://github.com/nyoki-mtl/keras-facenet. [Último acceso: Enero 2019].

[5] Apple´s Computer Vision Machine Learning Team, «An On-device Deep Neural Network for Face Detection,» *Machine Learning Journal,* vol. 1, nº 7, Noviembre 2017.

[6] GitHub Community, «GitHub,» Mayo 2017. [En línea]. Available: https://github.com/davidsandberg/facenet/issues/283. [Último acceso: Enero 2019].

[7] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt y Y.-L. Yang, «HoloGAN: Unsupervised learning of 3D representations from natural images,» arXiv:1904.01326v1, Abril 2019.

[8] T. Karras, S. Laine y T. Aila, «A Style-Based Generator Architecture for Generative Adversarial Networks,» arXiv:1812.04948v3, 2019.

[9] E. Zakharov, A. Shysheya, E. Burkov y V. Lempitsky, «Few-Shot Adversarial Learning of Realistic Neural Talking Head Models,» Moscow, Mayo 2019.

DEVELOPMENT OF A FACE RECOGNITION APP WITH ARTIFICIAL INTELLIGENCE TECHNIQUES

Author: **Menéndez Ruiz de Azúa, Alberto.**

Director: López López, Álvaro Jesús

Cooperative entity: Connected Industry Chair of the Technological Research Institute

PROJECT SUMMARY

Introduction

This project arises as a reaction and solution to the demands of the fourth industrial revolution, which brings with it a demand for constant improvement and modernization on the part of companies. The objective of this type of innovation initiatives is the optimization of processes and services by any company and not to lag behind the competitors. The digitization and implementation of Industry 4.0 projects to offer better and faster services to customers is something that every industrial company that wants to be at the forefront of its sector must do. Not adopting this type of initiative gives the opportunity to the competition to do so and be them who position themselves at the head of the sector.

The facial recognition project proposed in this document seeks to lighten the processes that require the identification of an individual, such as the access of personnel to the facilities of an industrial company or the verification of identity of customers when paying in the retail sector. The implementation of this type of technology would mean a drastic reduction in the waiting times that currently exist in this type of situation. The application of the technique proposed in this document in the aforementioned cases would mean replacing the accreditations or cards currently used in these processes with the user's own face. This means a reduction in the time wasted looking for the card or accreditation when using it and solves the possible cases of loss or forgetfulness of the documentation by the user.

Initially, this initiative arose from a proposal by a company in the retail sector to develop a facial recognition system aimed at facilitating the process of identifying their Premium customers once they were to pay for the purchase and, thus, automatically offer them the promotions that best suit their tastes based on previous purchases. The other applications of the project, previously mentioned, were appearing as the project was being developed and, since the project has been developed in the context of the Connected Industry Chair, these applications have a marked industrial character.

The decision to make the application for IOS devices is based on the great advances in their facial detection libraries and the good results to be used in practical examples, which place them at the forefront in terms of the state of the art. The examples of cases of use of these libraries in IOS devices more similar to what is looked for in this project are, the application of Face ID of the Iphone X or some apps of facial recognition that use the libraries of Vision and CoreML of Apple, as it can be the Photo App of the own Apple company. However, the most important techniques used in the project are not dependent on the type of IOS technology and are extrapolable to other electronic devices.

Methodology

For the project, after an exhaustive study of the state of the art, it was decided to use a holistic technique of artificial intelligence known as Siamese Neural Networks, based on the specific case of use of this technique by Google called FaceNet. The FaceNet technique has been used in projects similar to the one used for this solution, such as OpenFace [1], but never in IOS devices in the way it is approached in this project.

Within the state of the art of facial recognition there are different techniques that were studied when choosing the technique of Siamese neural networks, such as holistic methods, methods using infrared images or 3D images. A scheme of these methods can be seen in Figure 1. FaceNet is a holistic method with an artificial intelligence approach and the reasons that led to the choice of the FaceNet technique over the others used for the same purpose have been: the cost and availability of technologies, the scalability of the project, the ability to be developed in the proposed timeframe, the search for a solution that would achieve the goal of one-shot learning and flexibility for future improvements of the project, among other factors.

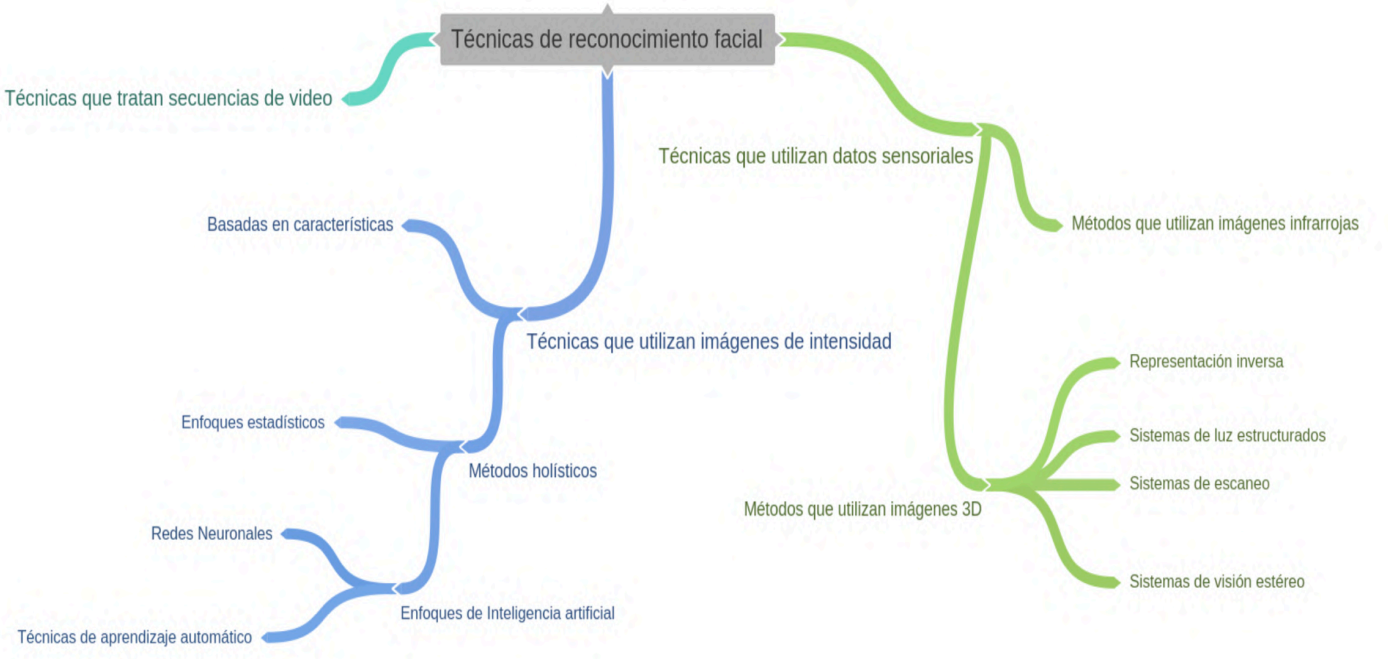


Figure 1: Scheme of techniques used for facial recognition

The Siamese Neural Network technique used for facial recognition is based on the use of a convolutional neural network, which generates a characteristic vector of 128 elements when it receives an image of a face size 160x160 pixels. This vector of 128 elements is the way that the neural network has to parameterize each face and it could be said that it

is like a synthesized form of the information that the neural network is able to extract from the image of a face. An example of how this technique works can be seen in Figure 2.

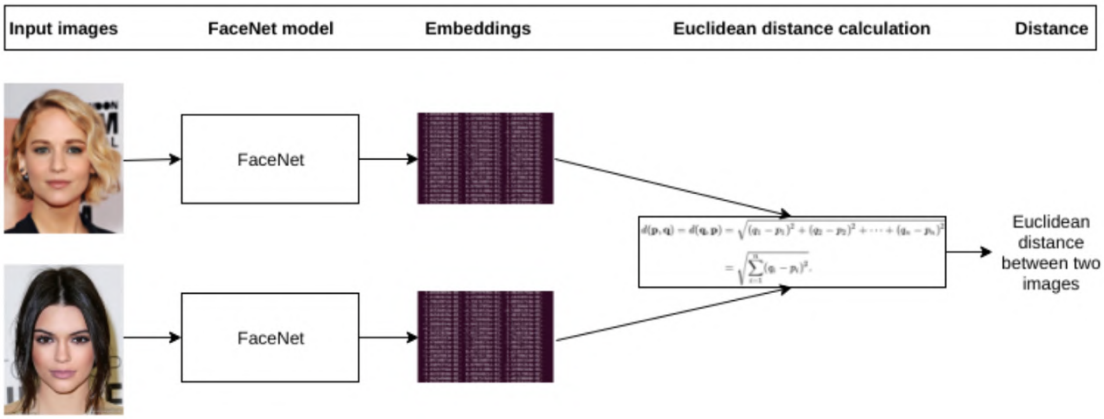


Figure 2: Example of how the Siamese Neural Network technique works [2].

This neural network has been trained to minimize the distance in the euclidean space of dimension 128 between two vectors of faces of the same person and, at the same time, to maximize the euclidean distance between the vectors of faces belonging to different people. The neural network training technique for this type of application is called Triplet Loss and was first presented by Google in the introductory FaceNet document in 2015 [3].

This project used a pretrained FaceNet model, built in Python on the convolutional neural network structure of Inception ResNet V1 and trained with the Triplet Loss technique on the MS-CELEBS 1M dataset [4]. This dataset has 10 million images corresponding to 100 thousand different classes or celebrities. A pre-trained model was used for the simple reason that network training is computationally a very expensive process, which generally requires months to obtain quality results and this training was not the main objective of the project itself. The use of a pre-trained model is not a disadvantage for the project, if not quite the opposite, since there is no need to train an own model and using a pre-trained model allows to count on a more reliable network which has been trained in a richer data set than if it had to be trained with more limited resources.

For the project this pre-trained neural network was converted to Swift language to be used in the application of facial recognition in IOS devices. For the conversion of the network, Apple's Coremltools library was used, created with the purpose of allowing the conversion of machine learning models from Python to Swift language. In addition, to the conversion of the model it was necessary to add an extra layer in Swift, called Scaling Layer, to be able to convert the Lambda layers that contained the pretrained model, which are not convertible by the Coremltools library.

At the time of carrying out the facial recognition in the application, different stages are made, and the first step consists in the facial detection, which means, to determine if there is a face and where that face is in the image that is being analyzed. To do this and to

improve the performance of the Siamese Neural Network, the reality is live recorded with the device's camera and the frames of this video are analyzed in search of faces. For face detection, Apple's Vision library is used, which has the necessary tools and functions to detect if there are faces in an image and in which part of the image is each of those faces. The Vision tool employs machine learning techniques as its creators explained in an article in Apple's Machine Learning Journal [5].

Once the position of the face in the image has been determined, the image is trimmed with a size of 160x160 taking as the center of the image the center of the face and leaving a small margin at the edges of the face, as several users who have messed with this technique have found that they get better results by using a certain margin [6]. This trimmed image is then introduced into the convolutional neural network model, which had been converted to Swift language, to obtain the characteristic vector that gives this model to that face. This vector is compared through the euclidean distance with the vectors of the faces that are stored in the database of the application and, in this way, it is determined if that face that has been detected corresponds with that of any of the users that are in the system or on the contrary it is an unknown person.

If the face corresponds to that of a user of the system, a red box is placed on the face with the user's name on top and a voice says the user's name followed by the word "Detected". If, on the other hand, the face does not correspond to that of any of the users saved in the system, the same red box is placed over the detected face, but with the word "Unknown" (“Desconocido” in Spanish) on top, as can be seen in Figure 3 . Later improvements of the application included the function that next to the name of the detected users appears a value with the percentage of reliability that the application gives to the decision that that face corresponds to that user.

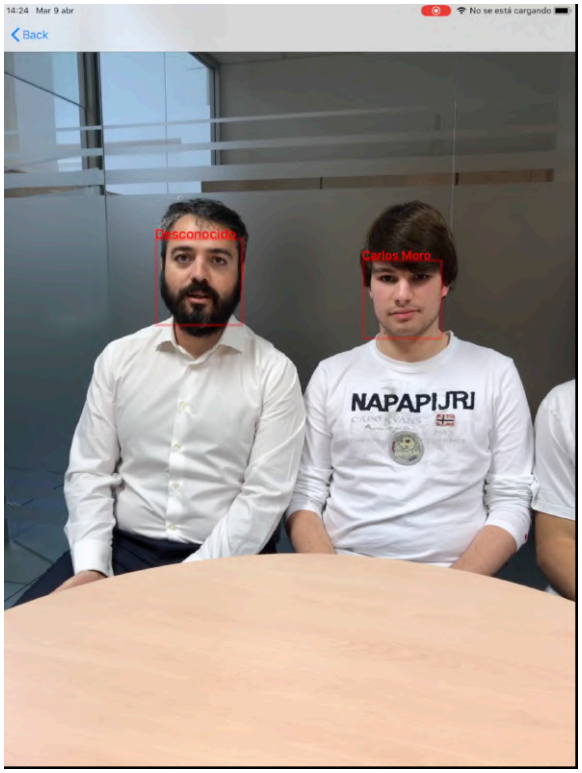


Figure 3: Recognition screen of the App in which you see an unknown user and another that is in the system

Results

The final result of the project was an application for any IOS device, which offers the possibility to enter or delete users in a database and which is able to recognize the users it has registered. The design of the structure of the application and the appearance of each of the screens of the application is specific to this project. In Figure 4 you can see a sample of the different screens that make up the application and in Figure 3 you can see the facial recognition screen, which is the one that entails the greatest difficulty of the application.

One of the most valuable sections of this project comes from the fact that throughout the development of this project difficulties were appearing, which were opening up new challenges and required exhaustive studies of what was happening. This document seeks to capture both the process carried out until the achievement of the App facial recognition, as the results of the robustness analysis carried out at each intermediate stage of the project. These results are accompanied by the conclusions and decisions of possible modifications and improvements that have been drawn from them. Therefore, the aim is not only to achieve a quality and robust project, but also to analyze and leave the door open to any improvement that may be made in the future.

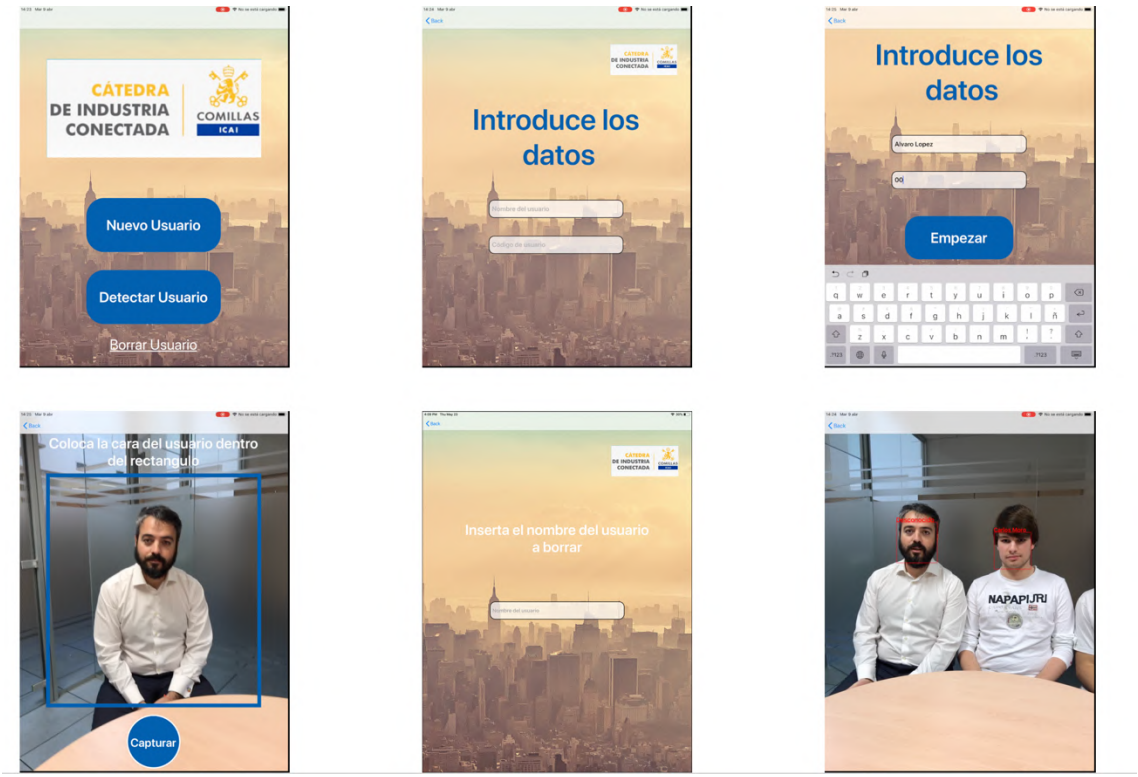


Figure 4: Sample of different screens of the App

It is for this reason that in the results section of the project and, more briefly, in the conclusions section of this summary, different studies are presented on the robustness of the model in the face of different changes in the input faces. The aim of these analyses is to find out more about the robustness of the application, to see what difficulties or limitations it presents. This was done to try to find solutions that allow the application to provide a greater robustness and solve the possible limitations that are detected. The conclusions set out the improvements proposed for the project as a result of the robustness analyses carried out, some of which have already been implemented.

Conclusions

After developing the application, this was subjected to different robustness analysis, which showed us how the application is dependent to a greater or lesser extent on changes in the posture of the face, the angle of rotation of the person with respect to the camera, the margin of the image introduced into the neural network around the face, linear changes and non-linear changes to which the face is subjected with respect to the image with which the registration of that face was made. An improvement has been proposed for each of the limitations found and some of them have already been implemented in the application.

For the changes in the rotation angle it was proposed to achieve a greater normalization of the face before being introduced into the neural network, which is a solution that has already been carried out in projects that use this technique as is the case of OpenFace. An

example of what this solution is intended to achieve can be seen in Figure 5, which shows the process that would be carried out to detect a rotated face. What we would look for in this solution would be to use one of the tools of the Vision library, which allows us to determine the position of certain points of the detected face. Taking the line that marks the middle of the face we can determine the value of the angle of rotation of the face with respect to the vertical axis and correct it before introducing the image in the neural network.

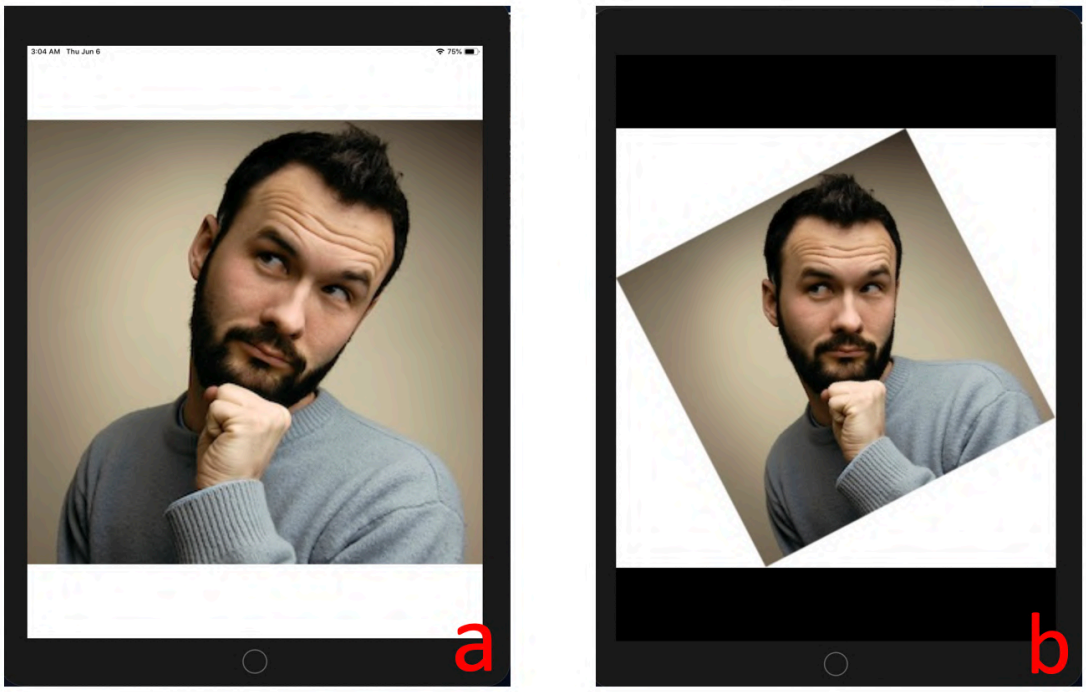


Figure 5: Example, with a small code, of how the improvement introduced works. (a) Image that the App would receive. (b) Image rotated to put the face straight and it will be cropped to enter the neural network.

Another limitation seen in the application is the dependence of the model on the percentage of the face that is visible in the image, that is, the degrees of rotation it represents with respect to the vertical axis of the spine. In order to solve this problem, different methods have been studied, the most promising being the use of GAN (Generative Adversarial Networks) techniques. There are variants of this technique such as HoloGAN [7], StyleGAN [8] or the Few-Shots Adversarial Learning Technique developed by Samsung [9] , which allow to generate a data set of images of a person in different positions or styles simply from a single image of that person. Some examples of what these techniques are capable of can be seen in Figure 6 and Figure 7.

This technique in combination with a classifier that can be trained with the data set generated by GAN, can be an ideal complement to the application of facial recognition. In this way the application would be able to recognize a person in positions or with changes in their appearance with what has not seen before.

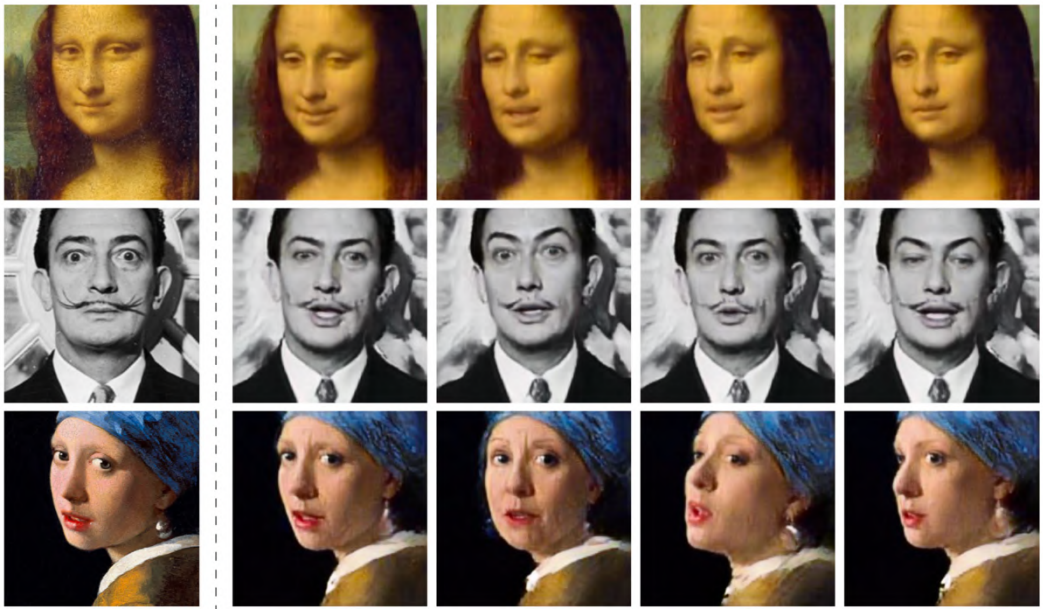


Figure 6: Example of results obtained with the Few Shots Adversarial Technique [9].

Figure 7: Example of StyleGAN results [8].

This last solution allows resolving most of the limitations previously exposed, since they had to do with the difficulties encountered when facing the model with the challenge of recognizing a person from a single initial image. That is why enriching the set of data of these people using GANs is so effective to face this limitation.

This technique of GANs can be used in any field of the industry in which for some reason only a reduced series of data or useful samples are available, since it allows to abstract some characteristics of these samples and to generate new data never seen before, but that maintain the essence and main characteristic of the data pertaining to that set.

Bibliography:

Bibliography

[1] B. L. y. M. S. B. Amos, «OpenFace: A general-purpose face recognition,» de *IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Placid, NY, USA, Marzo 2016.

[2] D. Karunakaran, «Medium,» 27 Septiembre 2018. [En línea]. Available: https://medium.com/intro-to-artificial-intelligence/one-shot-learning-explained-using-facenet-dff5ad52bd38. [Último acceso: Noviembre 2018].

[3] F. Schroff, D. Kalenichenko y J. Philbin, «FaceNet: A Unified Embedding for Face Recognition and Clustering,» 17 Junio 2015.

[4] H. T. (nyoki-mtl), «GitHub,» Febrero 2018. [En línea]. Available: https://github.com/nyoki-mtl/keras-facenet. [Último acceso: Enero 2019].

[5] Apple´s Computer Vision Machine Learning Team, «An On-device Deep Neural Network for Face Detection,» *Machine Learning Journal,* vol. 1, nº 7, Noviembre 2017.

[6] GitHub Community, «GitHub,» Mayo 2017. [En línea]. Available: https://github.com/davidsandberg/facenet/issues/283. [Último acceso: Enero 2019].

[7] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt y Y.-L. Yang, «HoloGAN: Unsupervised learning of 3D representations from natural images,» arXiv:1904.01326v1, Abril 2019.

[8] T. Karras, S. Laine y T. Aila, «A Style-Based Generator Architecture for Generative Adversarial Networks,» arXiv:1812.04948v3, 2019.

[9] E. Zakharov, A. Shysheya, E. Burkov y V. Lempitsky, «Few-Shot Adversarial Learning of Realistic Neural Talking Head Models,» Moscow, Mayo 2019.

Índice

Capítulo 1: Introducción y planteamiento del proyecto

1.1. Introducción

Este proyecto surge del interés de las empresas por dar un mejor servicio a sus clientes y optimizar su desempeño de cara a los mismos. En concreto el proyecto se empezó desarrollando de cara al sector retail y con una aplicación en la que se buscaba que los clientes, especialmente los pertenecientes al programa de fidelización, puedan ser reconocidos a través de dispositivos de reconocimiento visual (cualquier dispositivo que tenga una cámara), que informen de su presencia en un establecimiento, llegando a permitirles obtener ofertas personalizadas o incluso pagar sin necesidad de usar ningún tipo de identificación más allá de su propia cara. Posteriormente, y como se comenta en puntos posteriores de este documento, fueron surgiendo aplicaciones más industriales, como el control del acceso de los empleados a la fábrica de una empresa o para fichar.

El proyecto llega en un momento en el que las tecnologías digitales están en auge y es el grado de desarrollo de estas tecnologías el que marca la diferencia entre unas empresas y otras. Ofrecer una mejor atención al cliente y un servicio más completo es claramente una de las prioridades más fuertes de cualquier empresa, para lo cual resulta imprescindible implementar proyectos de digitalización y aumentar el grado de innovación de la empresa en este ámbito. Esto se evidencia en el informe “Painting the digital future of retail and consumer goods companies” [10] presentado por Accenture y en el cual se advierte de la necesidad de digitalizarse a las empresas del sector retail. El informe se basa en diferentes estudios llevados a cabo por esta compañía sobre los beneficios que este tipo de proyectos ya están teniendo y que tendrán en empresas que los están implementando.

Vivimos en una sociedad en la que los individuos esperan que todo sea inmediato y no suponga para ellos una innecesaria pérdida de tiempo, y los consumidores del sector retail no son una excepción, demandando cada día una mayor rapidez y efectividad de los servicios que se les ofrecen. En el contexto de la venta de bienes de cara al publico, en el cual se desarrolla este proyecto, el hecho de pagar es de los que más tiempo requiere dentro del proceso de una compra rutinaria, ya que requiere en muchas ocasiones de la búsqueda de la tarjeta o el medio de pago, perdiéndose bastante tiempo en el proceso. Ya existen métodos de pago que agilizan este proceso como Apple y Samsung pay o el chip “contactless”, incorporado en la mayor parte de las tarjetas de crédito, pero aun así estos métodos requieren llevar y buscar el móvil o la propia tarjeta en sí.

Otra aplicación muy demanda por las empresas hoy en día es la recolección de información sobre las preferencias y gustos de sus clientes. Por ello métodos que permitan obtener este tipo de información aportan un gran valor añadido a las empresas. Los datos y la información proporcionan una ventaja decisiva con respecto a los competidores y las empresas saben que si quieren ser competitivas no deben descuidar este aspecto. Estas afirmaciones vienen fundamentadas por el estudio un realizado por Strategy& la consultora estratégica de PwC (Price WaterHouse Cooper) [11], que analiza la situación de la digitalización dentro de España y en relación con el resto del mundo basándose en entrevistas a 1155 directivos de empresas de 26 países distintos. [12]

1.2. Estado del arte

La decisión de hacer la aplicación en dispositivos IOS viene fundamentada por los grandes avances en sus librerías de detección facial y los buenos resultados al ser empleadas en ejemplos prácticos, que las colocan a la vanguardia en lo que al estado del arte se refiere. Los ejemplos de casos de uso de estas librerías en dispositivos IOS más similares a lo que se busca en este proyecto son, la aplicación de Face ID del Iphone X o algunas apps de reconocimiento facial que utilizan las librerías de Visión y CoreML de Apple, como puede ser la Photo App de la propia empresa de Apple.

El iphone X realiza su reconocimiento facial usando un sistema de reconstrucción del rostro en 3D que emplea una cantidad de unos 30.000 puntos infrarrojos para hacer la reconstrucción biométrica del rostro [12]. Los principales problemas de esta tecnología son que estos rayos infrarrojos sirven para reconocer una cara hasta unas distancias determinadas y que esta tecnología no se encuentra disponible en todos los dispositivos de Apple, si no solo en los más modernos.

En cuanto a la Photo App de Apple, aunque la tecnología usada no ha sido revelada aun, sí que con la presentación de la librería de Vision y el cómo fue construida [5] podemos hacernos una idea aproximada de como funciona esta tecnología. El entorno de trabajo de Vision se nutre de una aplicación de *Machine Learning*, implementada con la librería CoreML y entrenada para detectar caras y algunas características de estas. Lo único que no parece tan obvio es el reconocimiento de las caras por parte de la aplicación, la cual, viendo tal y como funciona, utiliza un método de *clustering,* que junta todas las caras encontradas en imágenes que tienen parámetros relativamente parecidos. Este método de aprendizaje es no supervisado y requiere de una posterior realimentación por parte de un usuario que le indique quien es la persona de la foto.

Para la aplicación de este proyecto en concreto se recurrirá al empleo de la red neuronal de Google pre- entrenada, conocida como FaceNet [3], y que actualmente se considera el estado del arte en la cuestión del reconocimiento facial, por es la herramienta que mejores resultados ha obtenido en el conjunto de datos LFW (Label Faces in the Wild), con un porcentaje de acierto del 99,63%.

Esta red neuronal está construida sobre las librerías de Python de TensorFlow y Keras, que son librerías de la empresa Google, muy empleadas en el desarrollo de redes neuronales y herramientas de *machine learning* en general. FaceNet parte de una red neuronal convolucional, desarrollada sobre la estructura de capas de la red InceptionV1, desarrollada por Google y entrenada con en la base de datos de MS-Celeb-1M, formada por 10 millones de imágenes de caras correspondientes a 800 mil clases o personas diferentes. Además, el proyecto de FaceNet utiliza las técnicas de redes siamesas y de "Pérdida de triplete" mencionadas en el informe del proyecto, para entrenar la red neuronal. Esta red funciona de forma que devuelve un vector en el espacio euclideo de 128 elementos para cualquier foto de una cara que le sea introducida, de manera que ese vector es propio de esa cara. Una vez se tiene el vector de una cara, se pueden aplicar técnicas estadísticas como la distancia euclidea o PCA (Principal Component Anlysis), para comparar con futuros vectores, y ver si se corresponden con vectores de caras ya guardados anteriormente. De esta manera en la base de datos se tendrían n vectores de 128 elementos, siendo n el número de usurarios guardados en el sistema.

Existen proyectos que ya aprovechan y se basan en la tecnología de FaceNet, como un proyecto que clasifica perfiles de Tinder [7], o el famoso proyecto OpenFace [1], que permite reconocer a una persona con apenas 10 muestras de su cara. OpenFace también aporta una visión de cómo abordar nuestro problema en la captura de las caras, ya que aparte de usar Facenet, alinea las caras antes de introducirlas en la red, de manera que se compensa el posible error que se pueda dar por el hecho de que se capte una cara girada. Para nuestro proyecto nos basaremos en la estructura de trabajo del proyecto de OpenFace, ver Figura 7, por su eficacia ya demostrada en diversos conjuntos de datos y su parecido a nuestro proyecto.

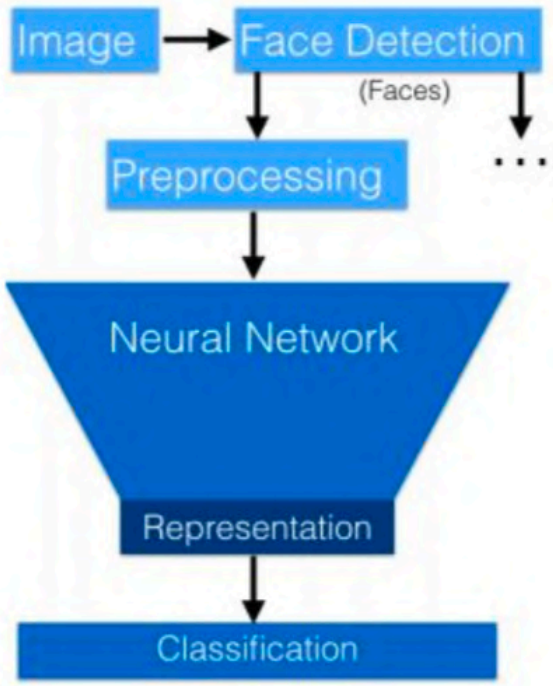


Figura 7: Estructura del proyecto OpenFace [1].

1.3. ¿Por qué redes neuronales siamesas?

Aparte del enfoque holístico de las redes neuronales siamesas, también se estudió la posibilidad de emplear otras técnicas que también han sido probadas y que han obtenido buenos resultados en cuanto a reconocimiento facial. Algunas de las técnicas descartadas se correspondían con enfoques basados en características de la cara, otras técnicas holísticas con enfoques estadísticos, técnicas de mapeado 3D mediante infrarrojos o incluso otros métodos holísticos de inteligencia artificial que no sean la de redes neuronales siamesas, como pueden ser redes neuronales convolucionales entrenadas a partir de imágenes de los usuarios.

Las razones que nos llevaron a elegir la técnica de redes neuronales siamesas frente algunas de las técnicas expuestas en el párrafo anterior fueron las siguientes:

- **El coste y la disponibilidad de las tecnologías**: si bien es cierto que algunas técnicas como el mapeado 3D de una cara, logran resultados más fiables y precisos que la técnica de redes neuronales siamesas, estas tecnologías son costosas y su complejidad no está al alcance de cualquier dispositivo. Un ejemplo de ello es el reconocimiento facial mediante infrarrojos de Apple, el cual, a pesar de ser muy fiable, solo esta disponible en los últimos dispositivos de la marca.

- **Escalabilidad del proyecto**: este factor influyo básicamente en el descarte de los métodos que se basan casi exclusivamente en el entrenamiento de una red neuronal o cualquier otra estructura de *machine learning* con datos de los distintos usuarios. En el caso de una aplicación como esta, que está orientada a ir añadiendo y quitando clases (usuarios en el caso de este proyecto), no te puedes permitir tener que entrenar la red prácticamente desde cero cada vez que quieras introducir un nuevo usuario en el sistema. Sí que es verdad que existen métodos para aligerar el proceso de entrenamiento de la red, como el uso del congelamiento de algunas capas que ya hayan aprendido a reconocer ciertos usuarios y entrenar otras para los nuevos usuarios. Sin embargo, incluso haciendo esto los tiempos de entrenamiento suelen ser grandes y limita mucho la aplicación del proyecto dentro de los objetivos que se persiguen. Cabe recalcar el hecho de que se descartan aquellas soluciones que empleen casi exclusivamente estas técnicas de *machine learning,* ya que sí que una de las ampliaciones que se plantean a este proyecto es una herramienta de inteligencia artificial situada en una etapa posterior a la de la red neuronal convolucional de la aplicación de redes siamesas y que se vaya entrenando de manera supervisada y paralela en un servidor, para, que una vez entrenada, permita hacer el proceso de reconocimiento más rápido y preciso. En el caso de la red neuronal convolucional de este proyecto, no hace falta este entrenamiento, ya que basta con emplear una red pre entrenada con un buen set de datos y no hace falta entrenarla con ningún tipo de dato de nuestros usuarios. Otro aspecto con respecto a la escalabilidad es que a la hora de añadir un usuario nuevo al sistema simplemente hay que subir online un documento actualizado que incluya los datos de este nuevo usuario y del que se pueden alimentar los demás dispositivos, que simplemente tendrían que alimentarse de ese archivo, sin tener que realizar ninguna operación compleja de entrenamiento de *machine learning*.

- **Sencillez y capacidad para ser desarrollado en el plazo propuesto**: este aspecto fue clave par decidir entre los tipos de técnicas holísticas de enfoque estadístico y de enfoque de inteligencia artificial. El aspecto clave es que los enfoques de inteligencia artificial permiten una mayor abstracción del problema biológico de tener que determinar que aspectos de una cara son los que tienen más peso a la hora de reconocerla, ya que este trabajo es llevado a cabo por la propia herramienta de inteligencia artificial, permitiendo centrarse en aspectos más ingenieriles del proyecto.

- **Búsqueda de one-shot learning**: uno de los objetivos más novedosos del proyecto es el hecho de que la técnica de redes neuronales siamesas permite, junto con algunas modificaciones, aspirar a tener un sistema que a partir de una sola imagen de una persona la pueda reconocer sin problemas las próximas veces que se enfrente al problema de su reconocimiento. Técnicas como las de

entrenamiento de herramientas de *machine learning* requieren para un funcionamiento óptimo una gran cantidad y variedad de datos, lo cual no se puede lograr en muchas de las aplicaciones a las que está destinado este proyecto, por el hecho de que básicamente se dispone de una única foto de cada usuario. En torno a esta idea también se han estudiado en este proyecto diferentes alternativas para ampliar el set de datos inicial mediante técnicas generativas de datos sintéticos, que permitan tener una mayor riqueza de datos para hacer el sistema más robusto.

- **Comodidad del usuario que va a ser reconocido**: haciendo alusión otra vez a las técnicas de mapeado 3D, estas tecnologías requieren que el usuario este detenido un rato delante del escáner o sistema de iluminación pertinente para poder realizar el modelo 3D de la cara. Sin contar además que la herramienta también requiere un tiempo para computar los datos y elaborar el modelo 3D. Como uno de los objetivos principales de la aplicación es la eliminación de tarjetas de acceso o de usuario, que puedan requerir de una perdida de tiempo en su búsqueda y uso para el usuario, este aspecto de los tiempos de espera también jugó un papel importante en el uso de la técnica de las redes neuronales siamesas.

- **Flexibilidad a futuras iniciativas de mejora del proyecto**: como se ha planteado a lo largo de puntos anteriores este método permite varias ampliaciones que le den mayor robustez y precisión al sistema de reconocimiento facial, algunas de las cuales ya se han implementado dentro del contexto de este proyecto o se analizan dentro del mismo. Estas mejoras vienen principalmente por la dificultad que entraña en si misma el hecho de lograr un sistema de reconocimiento facial partiendo de una sola imagen.

- **Privacidad:** dado que esta aplicación esta orientada a usuarios, los cuales pueden tener una sensibilidad determinada con respecto a la protección de sus datos, este es un aspecto realmente importante en este y cualquier proyecto orientado al público en general. En el caso de la técnica de redes neuronales siamesas empleada en el proyecto, la privacidad reside en que no se trabaja con imágenes de los usuarios, sino que, como se verá más adelante, solo se tiene un vector representativo de ese usuario, el cual solo podría ser interpretado con una red generativa entrenada mediante la red neuronal empleada en la aplicación.

Capítulo 2: Descripción de las tecnologías

Para el proyecto se usaron dos lenguajes de programación y varios Entornos de Desarrollo Integrados (IDEs) para programar en los dos lenguajes, técnicas de inteligencia artificial y *machine learning y* algoritmos matemáticos ligados a estas técnicas.

2.1. Lenguajes de programación

Los lenguajes de programación empleados fueron:

- **Swift**: es el lenguaje de programación de Apple para desarrollar aplicaciones para sus dispositivos. Es un lenguaje de programación multi [10]-paradigma, es decir, permite usar diferentes estilos de programación. Los paradigmas de Swift son la orientación a protocolos, orientación a objetos, programación funcional y programación imperativa. El lenguaje cuenta con varias librerías elaboradas por la propia empresa Apple, que contienen clases y funciones imprescindibles para desarrollar las aplicaciones para los dispositivos IOS.

  El entorno de desarrollo de Swift es Xcode, el cual está disponible de manera gratuita en la App Store para los usuarios de macOS. Xcode tiene algunas limitaciones, como las limitaciones que pone a la hora de publicar las aplicaciones, el hecho de tener una comunidad de desarrolladores más pequeña que la de otros lenguajes debido a su limitación a únicamente dispositivos IOS o las grandes dificultades para descargarlo en dispositivos que no sean de la familia MAC. Estas limitaciones no supusieron una imposibilidad a la hora de desarrollar el proyecto, únicamente lo hicieron más tedioso en algunos aspectos.

  Xcode tiene incorporados simuladores de los distintos dispositivos IOS en los que poder probar las aplicaciones de manera virtual. Para poder probar las aplicaciones en dispositivos físicos es necesario crear una cuenta de desarrollador y para poder usar funciones de conexión de la App a la red haya que adquirir una suscripción anual de 100$. Para este proyecto fue necesario usar un dispositivo físico y se empleó un IPad Pro de 9,7 pulgadas, ya que en los simuladores virtuales no se pueden usar las funciones de cámara de video necesarias para el funcionamiento de la App de reconocimiento facial.

  Para este proyecto se empleó la versión 4.2 de Swift en Xcode X, aunque en medio del proyecto se produjo una actualización a la versión XI, pero esto no resulto ningún problema. La información de las librerías de Apple, sus usos y contenidos de cada una se encuentran en la pagina web de Apple Developer de la compañía [13]. Para la implementación de la APP se usaron los siguientes entornos de trabajo o librerías de Apple:

  - **UIKit:** contiene las clases elaborar una interfaz de usuario gráfica, como pueden ser los botones, los cuadros para imágenes, las diferentes pantallas que contiene la aplicación o los cuadros de texto.

  - **AVFoundation:** es el framework de Apple que permite trabajar con contenido audiovisual basado en el transcurro del tiempo. En este proyecto resulta clave, ya que el reconocimiento facial se lleva a cabo en

tiempo real sobre un video grabado por la cámara del dispositivo y mostrado en la pantalla del mismo. También contiene las funciones y protocolos necesarios, para poder tomar los distintos frames del video y poder llevar a cabo la detección y reconocimiento facial sobre ellos.

- **CoreData:** este framework contiene todos los protocolos y clases necesarias parar crear y trabajar con una memoria persistente de datos en la que almacenar la información de cada usuario registrado en la App. Permite crear estructuras de datos, que en el caso de nuestra aplicación se irán guardando en el propio dispositivo. Los archivos de datos con los datos de los usuarios que tiene se guardan en formato SQL y pueden más tarde ser exportados, para incluirlos en otros dispositivos.

- **Vision:** framework que contiene las clases y funciones necesarias para poder identificar una cara en una imagen y obtener la posición de esta cara. Este framework emplea técnicas de *machine learning* para identificar las caras. Vision fue introducido en 2017, junto con un artículo por parte de sus desarrolladores que explica cómo fue creado [5]. Esta herramienta nos permite llevar a cabo el preprocesamiento de las imágenes antes de introducirlas en nuestra red neuronal.

- **CoreML:** este framework contiene todo lo necesario para emplear modelos de *maching learning* en una aplicación IOS. En nuestro proyecto se empleará para poder trabajar con los protocolos que permiten utilizar la red neuronal en la App. El framework de Vision emplea el de CoreML internamente, ya que para la detección de caras utiliza técnicas de *machine learning* y son sus propios creadores quienes han explicado en un artículo como se ha elaborado este entorno de trabajo [5].

- **Python:** el uso principal de Python en este proyecto se debe al hecho de que los principales proyectos de inteligencia artificial, así como las librerías que facilitan la implementación de estos proyectos, están desarrollados en Python. Más concretamente, la técnica de FaceNet, que se empleo en el proyecto, utiliza una estructura de red neuronal convolucional llamada InceptionV1, la cual está construida con TensorFlow, que es una librería de Google creada en Python.

  En Python también se encuentra la librería de Coremltools, la cual fue imprescindible en el desarrollo del proyecto y que está desarrollada por los ingenieros de Apple para Python. Esta librería permite convertir estructuras de *maching learning*, como una red neuronal en el caso de este proyecto, de lenguaje Python a Swift, que es el lenguaje en el que se implementó la aplicación de reconocimiento facial.

  Otros usos que se han hecho de Python en este proyecto incluyeron el análisis de datos obtenidos de la aplicación con el objetivo de mejorar su robustez o la creación de una GAN (*Generative Adversarial Network),* con el objetivo de explorar técnicas para mejorar la fiabilidad y robustez tanto de la aplicación como de la técnica de reconocimiento facial en si.

Las herramientas para trabajar con Python fueron, inicialmente, el navegador de Python Anaconda, por su facilidad para crear entornos de trabajo virtuales, y en combinación con la aplicación web de Jupyter Notebooks, que es también una herramienta muy intuitiva, visual y sencilla de usar. Posteriormente, se comenzó a usar Pycharm, que es una IDE de más nivel y potencia que los Jupyter Notebooks y que a la hora de crear inteligencias artificiales propias aporta mayor número de recursos y comodidad.

2.2. Técnicas de machine learning

La técnica y tecnología principal proyecto es la empleada en el reconocimiento facial, y es la técnica de redes neuronales siamesas de FaceNet.

FaceNet es un proyecto desarrollado por ingenieros de la empresa Google en 2015 y que consiste en el entrenamiento de una red neuronal convolucional, la cual al recibir una imagen de 160x160 pixeles (en el caso de nuestro proyecto se emplearon estas medidas, en otros ejemplos propuestos en el paper original [3] se utilizan imágenes de otros tamaños, como 220x220 o 80x80 pixeles) devuelve un vector de dimensión 128. Este vector se puede entender como la manera en la que la red resume la información de la cara que ha recibido y como codifica esta cara en el espacio euclideo de 128 elementos.

Esta red neuronal convolucional tiene la estructura de InceptionV1 desarrollada por Google, y que cuenta con la secuencia característica de una red neuronal formada por sucesivas capas de convoluciones, max pooling y normalización, que emplean la función de activación de Relu o Leaky Relu, en la Figura 8 se puede ver la estructura de esta red para una imagen de entrada de 220x220. La mayor diferencia con respecto a una red neuronal convolucional estándar es la eliminación de la última capa de softmax, empleada en las redes neuronales convolucionales por su orientación a problemas de clasificación. En este modelo la capa final de softmax es sustituida por una capa completamente conectada con 128 nodos de salida, los cuales dan lugar al vector final de salida.

El entrenamiento de esta red se realiza mediante la técnica de Pérdida de Triplete (Triplet Loss en inglés), la cual crea un error para su posterior propagación por los parámetros de la red basándose en dos criterios. En primer lugar, busca minimizar la distancia euclidea en el espacio vectorial de dimensión 128 entre dos vectores correspondientes a caras similares, mientras que, al mismo tiempo busca aumentar esa misma distancia con respecto a otro vector generado con una cara completamente distinta. De esta manera se entrena a la red no solo para que realice un proceso de *clustering* con las caras similares, sino que también está entrenada para separar lo máximo posible las caras de personas distintas. Por último, una vez empleada la Pérdida del Triplete se emplea el algoritmo de Backpropagation para actualizar todos los parámetros de la red e ir entrenándola. Un ejemplo más visual de como funciona esta técnica se puede ver en la Figura 9.

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Figura 8: Estructura de la red neuronal empleada en la técnica de FaceNet [3]

El error que se propaga por la red neuronal para su entrenamiento se saca de la Ecuación 1 y que es la base de la técnica de Pérdida de Triplete. En esta ecuación "a" es el ancla o vector normalizado que se ha obtenido de la red al introducir una imagen de la persona que se va a usar para entrenar el vector, "p" se corresponde con el vector normalizado obtenido de otra muestra de la persona de la imagen del ancla y "n" representa el vector normalizado obtenido con una muestra que se sabe que no se corresponde con la persona del ancla. "d(a, p)" y "d(a, n)" se corresponden con las distancias euclideas entre el ancla y el positivo y el ancla y el negativo respectivamente y el margen se fija y se pone para evitar que se cometa sobre aprendizaje por parte de la red y que el aprendizaje tienda a hacer que la distancia entre el ancla y el positivo se iguale a la distancia entre el ancla y el negativo.

$$L = \max\left(d(a,p) - d(a,n) + margen\,, 0\right)$$

Ecuación 1

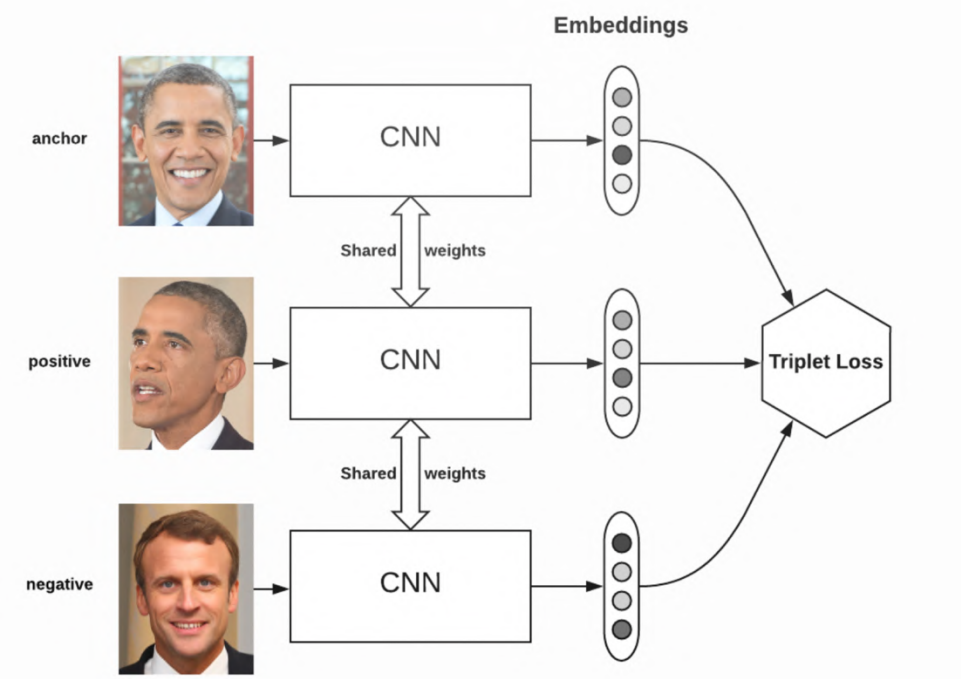


Figura 9: Ejemplo de funcionamiento de la técnica de Pérdida de Triplete

Para el proyecto descrito en este documento se empleó un modelo de FaceNet ya pre entrenado, más concretamente con el set de datos de MS-CELEB-1M, el cual contiene 10 millones de imágenes de caras de 100k celebridades diferentes. Al usar un modelo ya entrenado, lo que se hace es descargar la estructura del modelo de la red neuronal y los valores de los pesos que se han obtenido para esa estructura una vez que ya ha sido entrenada [4]. La razón principal de emplear un modelo ya entrenado es que el entrenamiento de una estructura de este estilo, con casi 10 millones de parámetros, y con un set de datos como el de MS-CELEB-1M, puede tardar en torno a las 5000 horas con el uso de equipos con gran capacidad de computación. El mero hecho de crear y entrenar una red de este estilo con garantías podría en si mismo representar un Trabajo Fin de Grado, lo cual no dejaría apenas tiempo para la consecución del objetivo de este proyecto.

.

Capítulo 3: Descripción del modelo desarrollado

3.1. Etapa inicial

Lo primero que se hizo de cara al proyecto fue formarse en las distintas herramientas que se iban a emplear en el mismo, tanto de Python como de Swift, de manera que a la hora de abordar el proyecto fuera todo más fácil.

Para coger soltura con el lenguaje Swift y las herramientas de trabajo se llevaron a cabo pequeños proyectos centrados en el uso de los Framework de Vision, CoreData y CoreML. Para practicar con el entorno de trabajo de Vision se hizo un pequeño programa capaz de recibir una imagen y colocar unos cuadros rojos sobre las caras que encuentre en esa imagen. Un ejemplo del funcionamiento de este programa se puede ve re en la Figura 10.

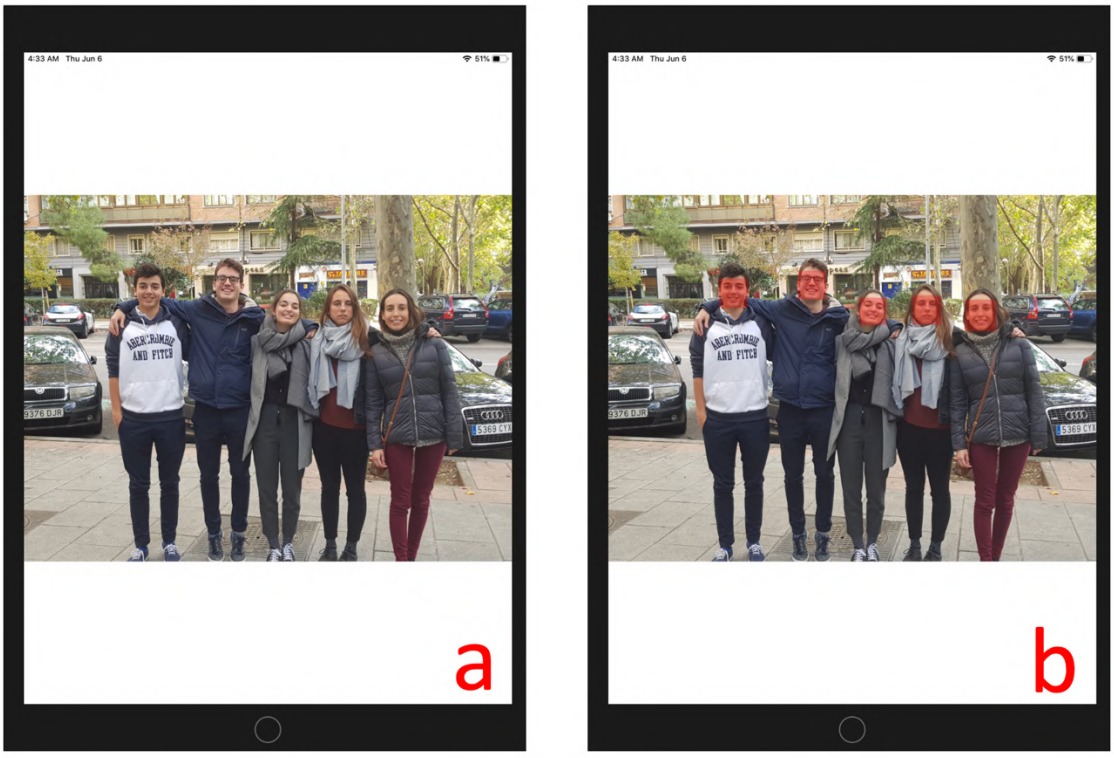


Figura 10: Ejemplo del programa de detección facial. (a) Imagen introducida al programa. (b) Imagen que devuelve el programa tras usar la librería de Vision.

Para probar la herramienta de CoreML se creo un pequeño programa capaz de detectar objetos en tiempo real usando una red neuronal convolucional clasificadora ya convertida a formato de CoreML.

3.2. Diseño de la aplicación

Una vez conocido el método a emplear (redes neuronales siamesas) se procedió al diseño de la aplicación y de las distintas pantallas que la componen. El diseño de la aplicación se hizo cuidando todo tipo de detalles, de manera que resulte fácil e intuitiva de usar de cara al usuario final.

La aplicación se compone de 6 tipos de ventanas que se pueden ver en la Figura 11 , las cuales son todas de diseño propio y todas aquellas que no tienen el uso de la cámara frontal tienen un fondo común, que en la muestra que se expone en este proyecto es la de una ciudad. Detrás de cada una de las pantallas de la App hay cientos de líneas de código en Swift que permiten obtener los diseños y funcionalidades que componen el proyecto. Las 6 pantallas de las que se compone la aplicación son:

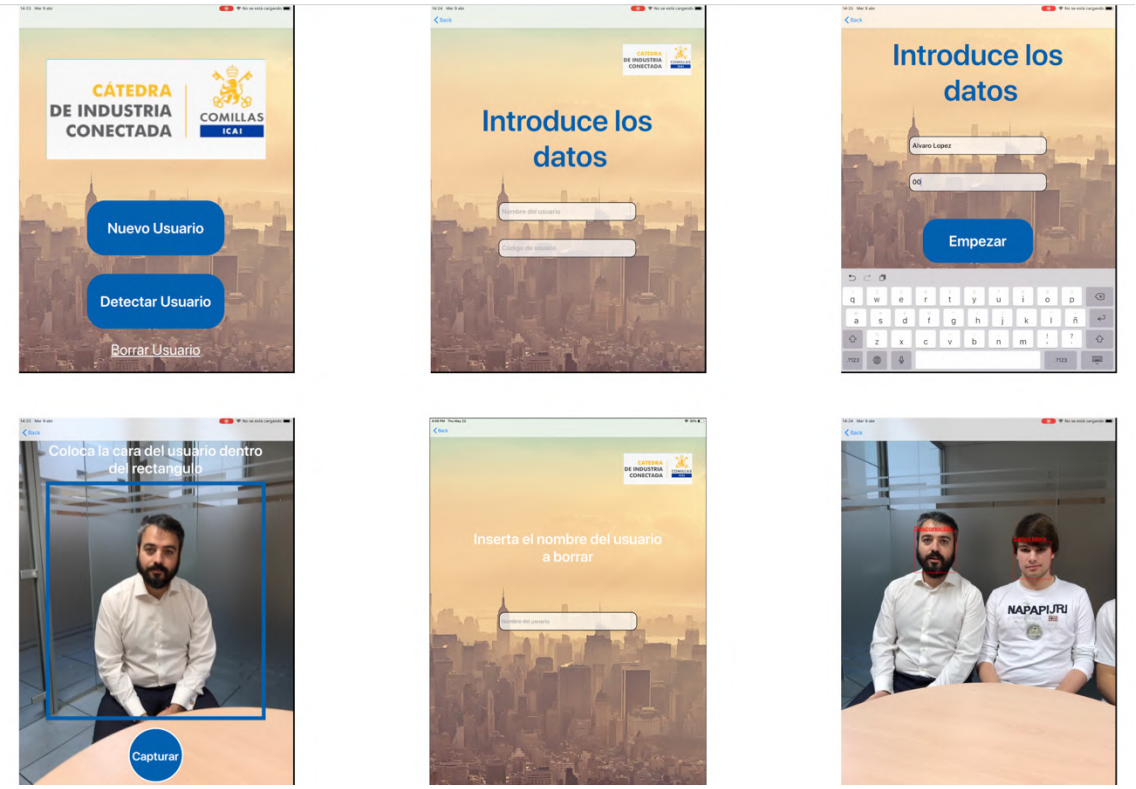


Figura 11: Muestra de las pantallas que componen la App

- **Pantalla de inicio:** es la pantalla que se muestra una vez se inicia la aplicación y consta de los siguientes elementos (ver Figura 12):

    - Logo de la entidad o empresa dueña de la aplicación: se puede poner el de cualquier empresa, para la muestra que se plantea en este documento se empleo el de la Cátedra de Industria Conectada, que es con quien se desarrollo el proyecto.

- Botón de registro de usuario: este botón tiene la función de cambiar a la pantalla de registro de un nuevo usuario en la aplicación al pulsarlo.

- Botón de detección de usuario: este botón tiene la función de cambiar a la pantalla de detección de un usuario al pulsarlo.

- Botón de borrar usuario: este botón tiene la función de cambiar a la pantalla de borrar un usuario del sistema al pulsarlo.

Figura 12: Pantalla de Inicio

- **Pantalla de registro de usuario:** esta pantalla es en la que se introducen los datos del nuevo usuario que se desea introducir en el sistema. La pantalla consta de los siguientes elementos (ver Figura 13):

  - Logo de la entidad: se muestra el mismo logo que en la pantalla de inicio solo que en la esquina superior derecha de la ventana.

  - Etiqueta de texto: contiene instrucciones para el usuario de la aplicación, en este caso pone “Introduce los datos”.

  - Cuadros de texto: permiten al usuario introducir la información que en ellos se requiere. Para un mejor resultado a nivel visual se introdujo un

formato de texto especial para las letras de dentro de estos cuadros, ya que el que viene por defecto queda muy pegado al borde del cuadro y no queda bien de cara al usuario. Hay dos cuadros de texto en la pantalla, uno para introducir el nombre de el usuario, y otro para introducir el posible número de identificación del usuario.

- El teclado: aparece cuando se pulsa sobre uno de los cuadros de texto y se ha incluido el detalle de que la pantalla se desplace hacia arriba cuando aparece el teclado, de forma que el usuario puede seguir viendo el cuadro de texto sobre el que está escribiendo.

- El botón de empezar: este botón hace aparecer la pantalla de captura de muestras de el usuario que se quiere registrar. Este botón esta configurado de manera que solo aparezca solamente una vez que haya algo escrito tanto en el cuadro de nombre de usuario como en el de código de usuario. Al pulsar este botón se transfieren los datos de los cuadros de texto a la pantalla siguiente, para guardarlos en la base de datos junto con el vector obtenido en la pantalla de la captura de muestras.

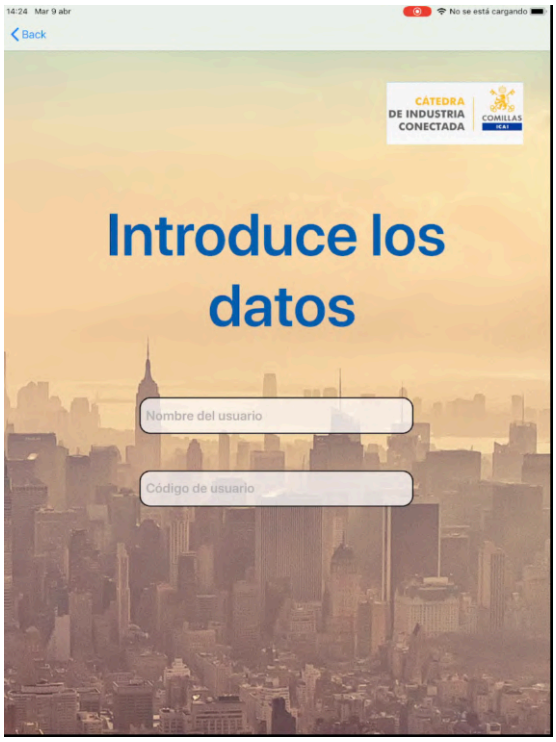


Figura 13: Pantalla de registro de usuario

- **Pantalla de captura de muestras:** en esta pantalla se emplea la cámara del dispositivo para tomar unas muestras del usuario que se quiera registrar en la aplicación. Los elementos que componen esta pantalla son (ver Figura 14):

- Sesión de video: se inicializa una vez que aparece la pantalla de capturar muestras y permite al usuario ver lo que está viendo la cámara en tiempo real.

- Etiqueta de texto: al igual que en la pantalla anterior, contiene instrucciones para el usuario de la aplicación, en este caso pone "Coloca la cara del usuario dentro del rectángulo".

- Vista de validación: consta de un rectángulo azul, dentro del cual se debe situar la cara del usuario que va a ser registrado. Esto sirve para facilitar el uso de la aplicación y hacerla más intuitiva.

- Botón de capturar: al pulsarlo se activa la función de captura de muestras, que va efectuando una detección y clusterización de la cara del usuario en cada frame del video que se vaya capturando desde que se pulsa este botón.

- Función de detección y clusterización: esta función se encarga de buscar en cada frame las caras que haya y devuelve para cada frame un vector característico de la cara del usuario a ser registrado, su funcionamiento se explicará más en detalle en la pantalla de detección y reconocimiento de usuarios. Para la aplicación propuesta en este proyecto este proceso se repetirá durante 7 veces, con un espaciado de un frame entre medias, en los cuales no se guarda el vector característico.

- Función de media de vectores: esta función se encarga de hacer la media entre todos los vectores característicos del usuario a ser registrado, sacando un vector definitivo que será el que represente al usuario y se guarde junto con sus datos. Esta función se encarga también de eliminar alguno de los vectores si este se aleja mucho de los demás en el espacio euclideo, ya que puede ser que la imagen se haya capturado mal en ese momento.

- Botón siguiente: al pulsarlo se envían todos los datos del nuevo usuario (nombre, ID y vector característico) a la pantalla de registro correcto, que se encargará de guardarlos y de normalizar el vector característico.

Figura 14: Pantalla de captura de muestras

- **Pantalla de registro correcto:** esta ventana verifica que el registro se haya llevado a cabo de manera correcta y se encarga de guardar al nuevo usuario y sus datos en la memoria persistente de la aplicación. Se compone de los siguientes elementos (ver Figura 15):

    - Imagen del usuario que ha sido registrado: esta imagen se muestra en pantalla, para que el usuario de la aplicación tenga una referencia de cual es la imagen con la que ha trabajado el modelo y de que todo ha ido bien**.**

    - Logo de tick verde: se sitúa encima de la imagen de la cara del nuevo usuario si el registro ha se ha producido correctamente.

    - Función de normalización de vectores: esta función se encarga de normalizar el vector característico de la cara del usuario que va a ser registrado antes de guardarlo en la memoria persistente de datos. Esto se hace, ya que el reconocimiento facial funciona mucho mejor cuando los vectores de las caras están normalizados, porque el propio modelo ha sido entrenado usando vectores normalizados [3].

    - Temporizador: mantiene la aplicación en la pantalla de registro correcto durante un tiempo de 3 segundos tras el correcto registro del usuario y pasado este tiempo devuelve la aplicación a la pantalla de inicio.

Figura 15: Pantalla de registro correcto

- **Pantalla de detección de usuarios:** esta pantalla es la pantalla central de lo que se buscaba para este proyecto y es la pantalla de la aplicación en la cual se produce la detección y reconocimiento facial de las personas. Esta pantalla cuenta con los siguientes elementos (ver Figura 16):

    - Sesión de video: se inicializa una vez que aparece la pantalla de capturar muestras y permite al usuario ver lo que está viendo la cámara delantera en tiempo real.

    - Función de reconocimiento facial: esta función saca el vector característico de todas las caras que haya en cada frame de la sesión de video y los compara en primer lugar con las personas del frame anterior y ,en caso de ser el vector de una persona que acaba de entrar en el video y que no ha sido reconocida anteriormente, lo compara con los vectores de los usuarios registrados en la App.

    - Función de porcentaje de fiabilidad: esta función se encarga de hallar el porcentaje de seguridad que tiene la aplicación de que la persona reconocida es quien ha dicho que era.

    - Función de colocación de cuadros rojos sobre las caras: esta función se encarga de colocar un cuadro rojo con el nombre del usuario sobre la cara

del usuario que haya sido reconocido o en caso de ser un usuario no registrado con la palabra "Desconocido" en vez del nombre. Esta función también se encarga de actualizar los cuadros de las caras de las personas que ya habían pasado por el proceso de ser identificadas, de manera que el cuadro vaya siguiendo a la cara. También se incluye la función de poder mostrar el porcentaje de parecido de esa persona con la detectada, en función de si se parece más o menos a los registros de esa persona que se tienen en la base de datos.

- Comandos por voz: esta pantalla cuenta con la función de decir por voz el nombre del usuario que ha sido reconocido, seguido de la palabra "Detectado", de manera que se informa de forma acústica cuando alguien ha sido detectado.

Figura 16: Pantalla de detección de usuarios

- **Pantalla de borrar usuario:** esta pantalla sirve para borrar los datos de la memoria persistente de datos en caso de que se desee dar de baja del sistema a un usuario. Para ello, esta pantalla cuenta con los siguientes elementos:

  - Etiqueta de texto: contiene las instrucciones que debe seguir el usuario de la App en esta pantalla. En el caso de la demo que se hizo con esta aplicación la etiqueta dice: "Inserta el nombre del usuario a borrar".

- Cuadro de texto: en el que poder introducir el nombre del usuario que se desea borrar. Al igual que en la pantalla de registro de usuario, se introdujo un formato de texto especial para las letras de dentro de estos cuadros, ya que el que viene por defecto queda muy pegado al borde del cuadro y no queda bien de cara al usuario final de la aplicación.

- Botón de borrar: una vez se pulsa este botón se elimina el usuario cuyo nombre se haya introducido de la memoria persistente de la aplicación y por tanto ya no podría ser reconocido.

Figura 17: Pantalla de borrar usuario

3.3. Conversión del modelo

La red ya pre entrenada de FaceNet con su estructura y los pesos se encontraban implementada en Python, así que fueron descargados [4], y es en esta parte donde entró en juego la librería de Coremltools. Esta librería es creación de la empresa Apple y permite convertir modelos de *machine learning* de código Python a Swift, que es lo que se necesitaba para poder implementar esta técnica de reconocimiento facial en dispositivos IOS.

El código empleado para la conversión del modelo se puede ver en la Figura 18. En este código se puede ver como se importan las herramientas de la librería de *coremltools* necesarias para convertir nuestro modelo. Otro aspecto importante es el que se ve en la primera línea, en la que se menciona las versiones de cada una de las librerías que se han empleado, ya que la conversión se complicó bastante debido a la compatibilidad de la librería de *coremlools* solo con determinadas versiones de las librerias de *Tensorflow* y *Keras*, con las cuales estaba construido el modelo de FaceNet.

Convertir_FaceNet.py

```
# usamos coremltools==0.8 keras==2.1.5 y tensorflow==1.4.0

import h5py
import coremltools
from coremltools.proto import NeuralNetwork_pb2
from coremltools.models.neural_network.quantization_utils import *
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
from keras.models import load_model
import os.path

import sys
sys.path.append('../Estructura/') # Modelo pre entrenado en https://github.com/nyoki-mtl/keras-facenet

from inception_resnet_v1 import *
model = InceptionResNetV1(weights_path='../Modelo/Pesos/facenet_keras_weights.h5')

# La funcion de conversion para las capas lambda
import coremltools
from coremltools.proto import NeuralNetwork_pb2
def convert_lambda(layer):
    if layer.function == scaling:
        params = NeuralNetwork_pb2.CustomLayerParams()

        params.className = "scaling"
        params.description = "scaling input"

        # Importante!!
        params.parameters["scale"].doubleValue = layer.arguments['scale']

        return params
    else:
        return None

# Convertimos el modelo
coreml_model = coremltools.converters.keras.convert(
    model,
    input_names="image",
    image_input_names="image",
    output_names="output",
    model_precision='float16',
    image_scale=2/255.0,
    red_bias=-1,
    green_bias=-1,
    blue_bias=-1,
    add_custom_layers=True,
    custom_conversion_functions={ "Lambda": convert_lambda })

# Guardamos el modelo convertido
coreml_model.save('FaceNetModel.mlmodel')
```

Figura 18: Código en Python para la conversión del modelo de FaceNet

Además, se tuvo que implementar un pequeño código en la aplicación de IOS, para hacer que la red neuronal funcionara como se esperaba. Este código se corresponde con la conversión de las capas Lambda de la estructura de la red construida en la librería de *keras,* ya que al ser capas que realizan funciones muy concretas y no están estandarizadas la librería de *coremltools* no cuenta con las herramientas necesarias para convertirlas. Por eso se debieron convertir de manera manual con la función convert_lambda que se ve en el código de la Figura 18 y con el código de la clase de la capa que se ve en la Figura 19.

App_TFG 〉 App_TFG 〉 scaling.swift 〉 setWeightData(_:)

```swift
//  scaling.swift
//  Pruebas_Modelo_CoreML
//
//  Created by Alberto  Menendez on 02/02/2019.
//  Copyright © 2019 Alberto  Menendez. All rights reserved.
//
import Foundation
import CoreML
import Accelerate

@objc(scaling) class scaling: NSObject, MLCustomLayer {

    let scale: Float

    required init(parameters: [String : Any]) throws {

        if let scale = parameters["scale"] as? NSNumber {
            self.scale = scale.floatValue
        } else {
            self.scale = 1.0
        }

        super.init()
    }

    func setWeightData(_ weights: [Data]) throws {

    }

    func outputShapes(forInputShapes inputShapes: [[NSNumber]]) throws -> [[NSNumber]] {

        // This layer does not modify the size of the data.
        return inputShapes
    }

    func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {

        for i in 0..<inputs.count {
            let input = inputs[i]
            let output = outputs[i]

            let count = input.count
            let inputPointer = UnsafeMutablePointer<Float>(OpaquePointer(input.dataPointer))
            let outputPointer = UnsafeMutablePointer<Float>(OpaquePointer(output.dataPointer))
            var scale = self.scale
            vDSP_vsmul(inputPointer, 1, &scale, outputPointer, 1, vDSP_Length(count))
        }
    }
}
```

Figura 19: Código de la clase de la capa Lambda del modelo de FaceNet en Swift

Para comprobar la robustez de la conversión se hicieron pruebas con 10 imágenes de personas distintas, para ver si la red neuronal sacaba los mismos valores para una misma imagen con el código en Python, que sabemos que está bien porque ha sido empleado por diversos investigadores y con la consiguiente conversión a Swift. A este respecto se obtuvieron los siguientes resultados del análisis de los 20 vectores obtenidos, 2 para cada una de las caras utilizadas. Estos resultados se pueden ver en el Apartado 4.1. de este documento, donde se recoge el análisis de los resultados de la conversión del modelo. Únicamente se muestra el análisis de la conversión satisfactoria, aunque hubo algún intento fallido de convertir el modelo que fue detectado en estos análisis posteriores.

3.4. Implantación final del modelo

Finalmente, con el modelo final convertido, se juntaron todos los códigos desarrollados para obtener la aplicación final y pulir el resultado de esta. Se obtuvo una aplicación con las pantallas mostradas en el punto 3.2. capaz de registrar y borrar usuarios y de detectar en tiempo real a una persona como se puede ver en la Figura 20. Con el resultado final se grabo un video de la aplicación como se puede ver en la Figura 20

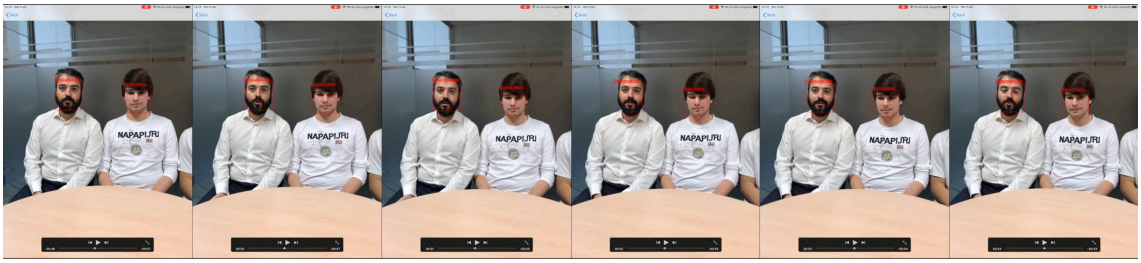


Figura 20: Extracto del video con el funcionamiento de la aplicación

La fijación de el umbral de distancia euclidea a partir del cual se considera que una persona es la que se tiene en el base de datos se fijo por tanteo y viendo los resultados de los análisis que se muestran en el Capítulo 4. Este umbral finalmente se fijó en 0.65, ya que las caras de personas diferentes se sitúan en distancias euclideas de más de 1, mientras que las de una misma persona en la posición en la que fue registrada nunca superan el 0.6.

Es a raíz de este resultado que se llevaron a cabo los diferentes estudios de robustez del modelo que se presentan en los puntos 4.3 y 4.4 de este documento, en los que se lleva a cabo un análisis de la robustez del modelo frente a cambios en la posición de la cara y en el aspecto respectivamente. El objetivo de estos análisis es detectar los puntos débiles de la aplicación y buscar métodos para poder aumentar su robustez y fiabilidad.

Uno de los apartados de valor que no se llego a aplicar es la conexión entre dispositivos para compartir la información de la base de datos entre varios dispositivos y que todos los dispositivos puedan reconocer a los mismos usuarios independientemente de en cual se hayan registrado. Esta mejora nos es complicada de aplicar, ya que Apple cuenta con una librería especifica para esta función, el entorno de trabajo de *cloudkit,* y que tiene las funciones y protocolos necesarios para poder compartir documentos de Core Data ente dispositivos Apple y que todos puedan usarlos en su aplicación. La razón por la que no se llego a implementar es porque para usar estas funciones de Apple se requiere de licencia específica de desarrollador de Apple que tiene una cuota anual y que es la misma que permite subir y monetizar las aplicaciones a Apple Store. El hecho de que no fuera a aprovecharse esta inversión al máximo hizo que no se adquiriera esta licencia por el momento, pero si en algún momento se requiere se adquirirá la licencia y se introducirán las líneas de código necesarias para lograr la conectividad entre dispositivos.

Capítulo 4: Análisis de resultados

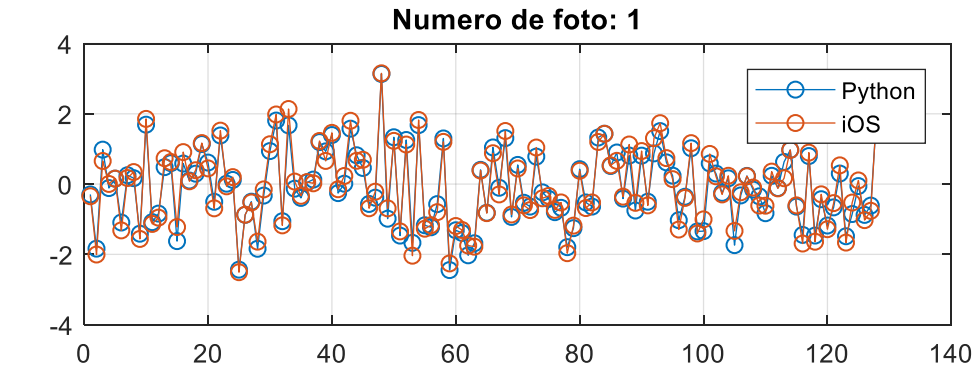
4.1. Análisis de los resultados de la conversión del modelo

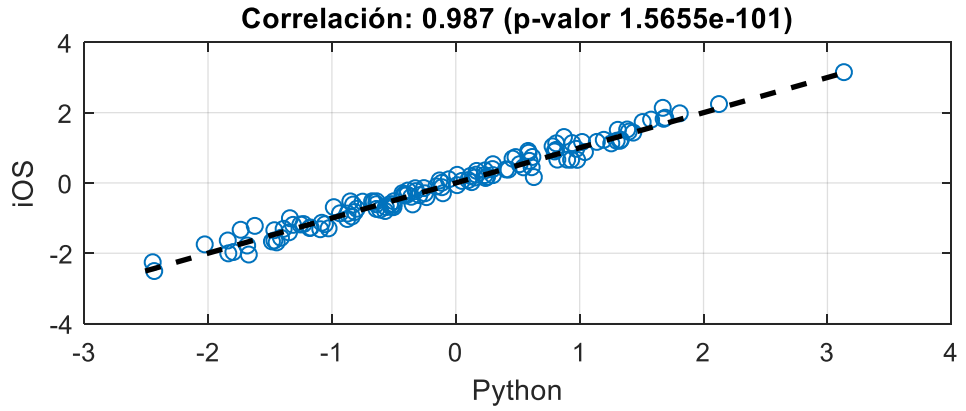
Para hacer este análisis se introdujeron 10 fotos correspondientes a 10 usuarios distintos en el modelo de la red neuronal de Python y en el modelo convertido a Swift. Una vez obtenido el vector característico que le otorgaba cada una de las redes a las imágenes se hizo un estudio de similitud entre el vector de Python y el de los dispositivos IOS, para ver si la conversión había sido hecha correctamente. En la Figura 21 se pueden ver las imágenes que se introdujeron a los modelos. Las imágenes se han numerado para poder identificarlas con las gráficas del estudio de similitud de sus vectores característicos.

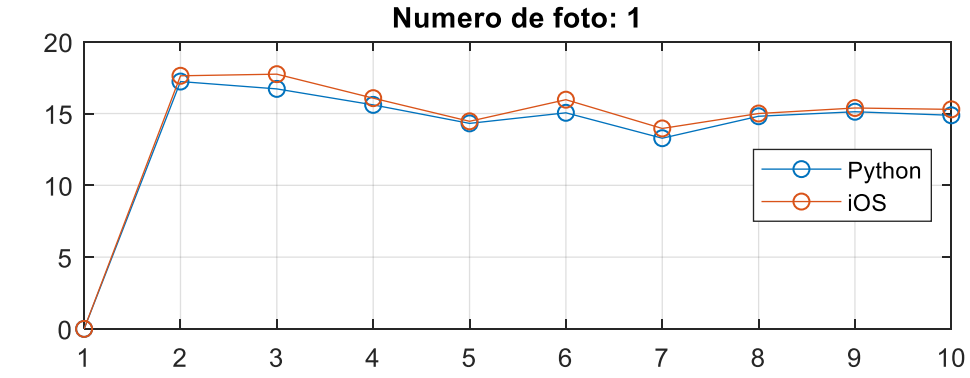
Figura 21: Imágenes que se introdujeron en los modelos para estudiar la correcta conversión del modelo de FaceNet

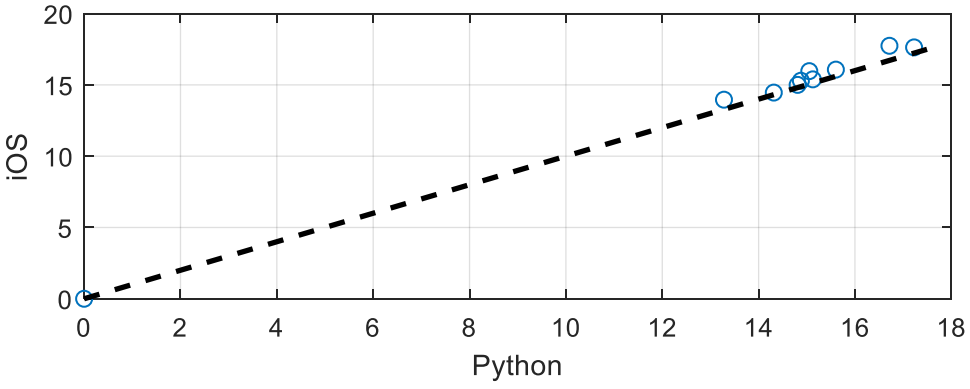
A continuación, se muestran 4 gráficas obtenidas en Matlab para cada una de las 10 muestras de la Figura 21. Las dos primeras gráficas se corresponden con el análisis de la similitud entre el vector obtenido para cada cara con el modelo en Python y el obtenido con el modelo en IOS. Cabe destacar que este análisis se hizo con los vectores obtenidos por el modelo en bruto, es decir, sin normalizar, para analizar únicamente la correcta conversión del modelo. Las dos gráficas siguientes se corresponden con el análisis de la distancia euclidea de el vector de la cara analizada con cada uno de los otros 9 vectores de las otras muestras. Lógicamente la distancia euclidea entre el vector de cara y él mismo es siempre 0.

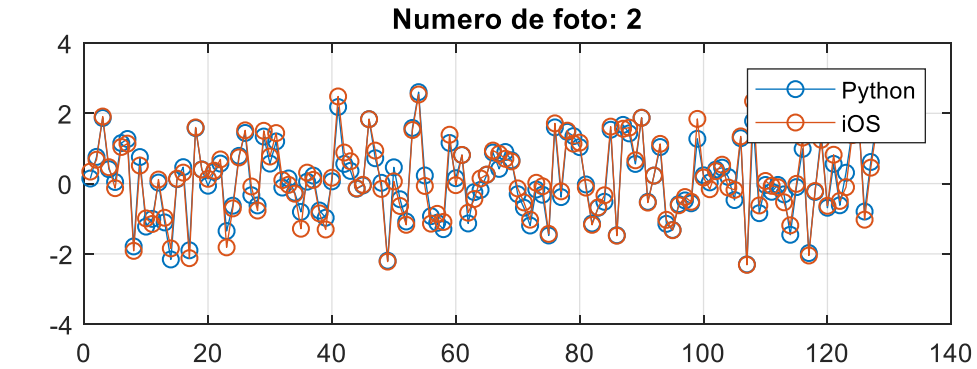
Comparación salidas modelos

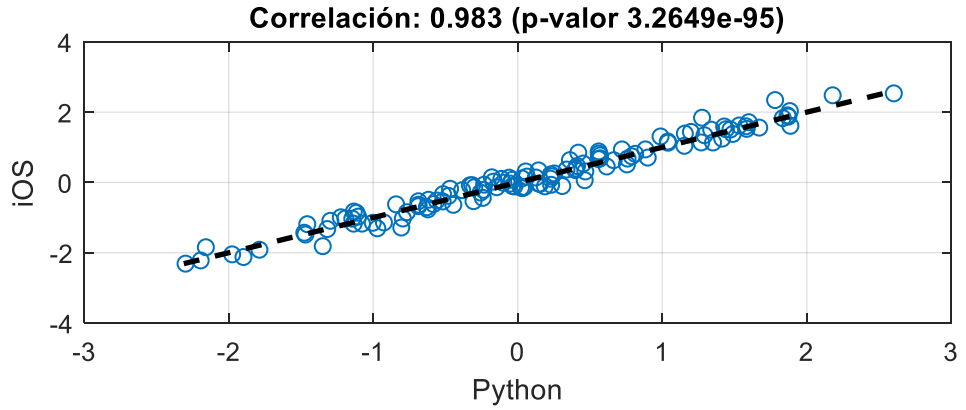


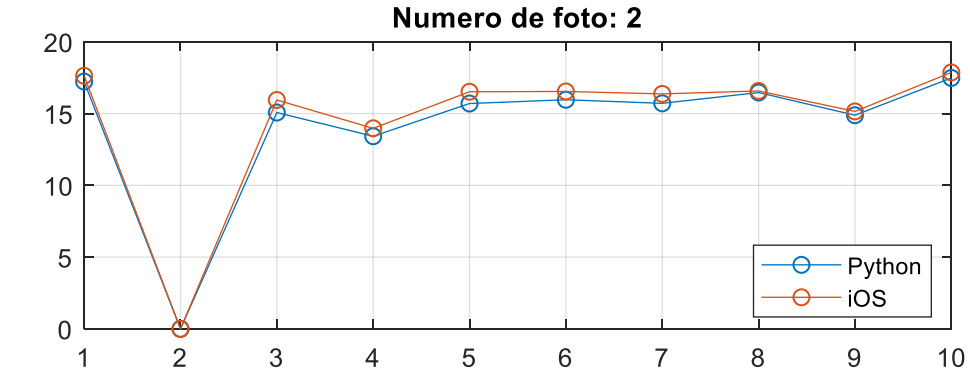


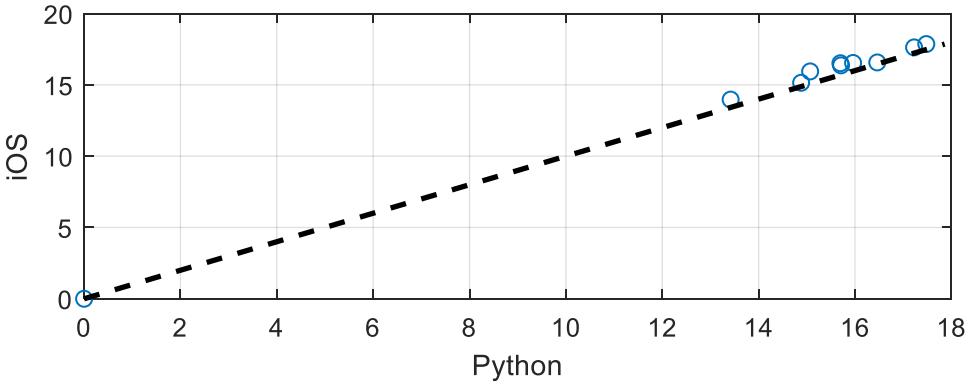


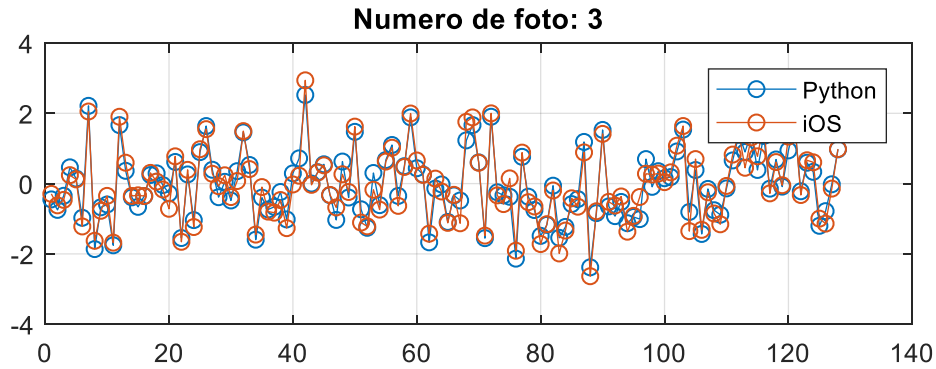


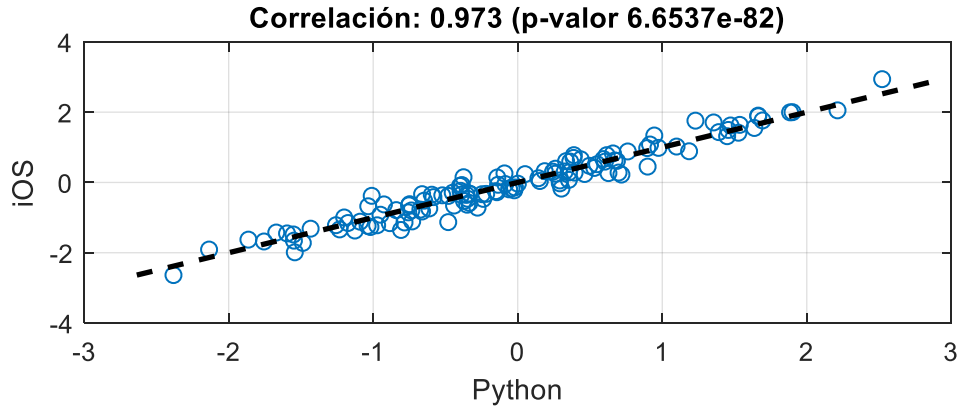

Numero de foto: 2
Python
iOS

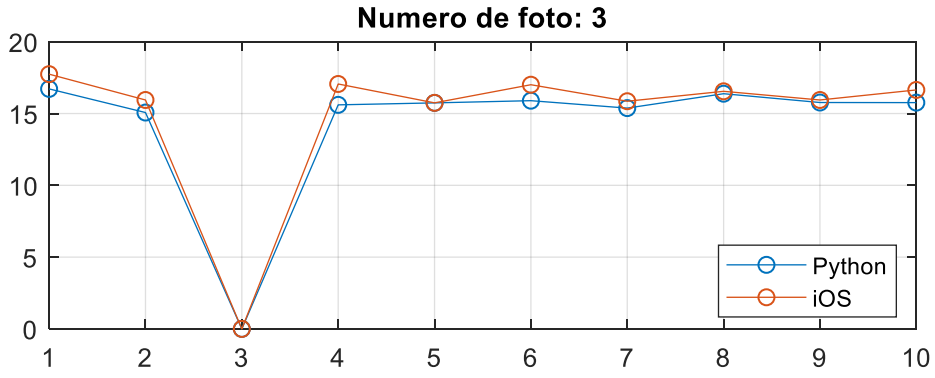

Correlación: 0.983 (p-valor 3.2649e-95)
iOS
Python

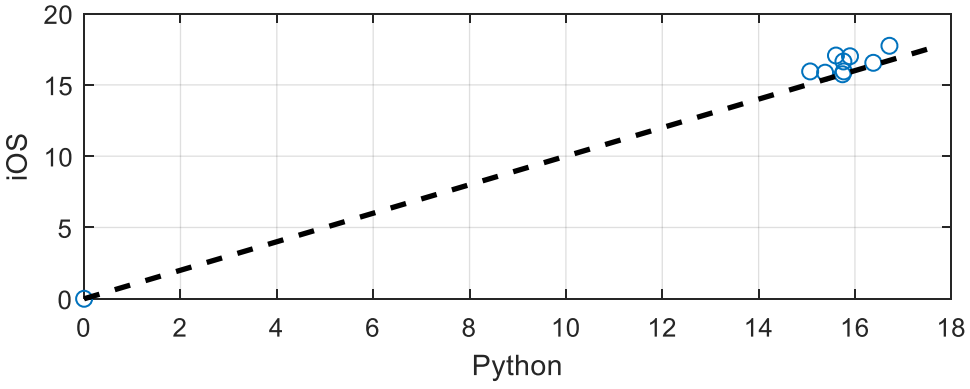

Numero de foto: 2
Python
iOS

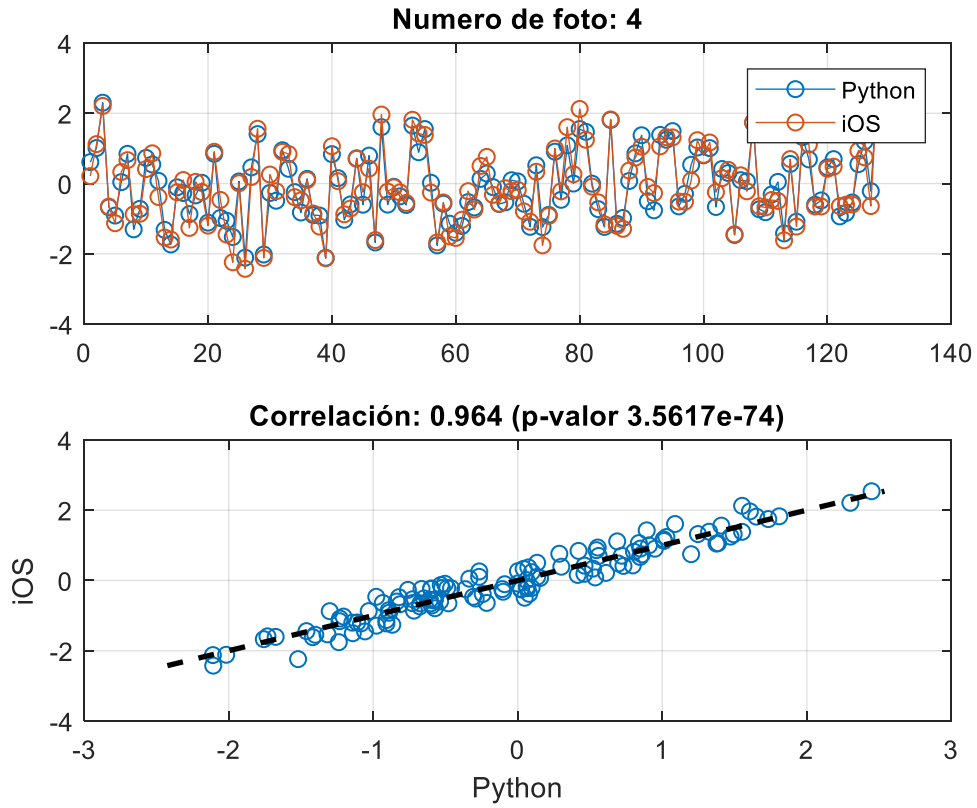

iOS
Python

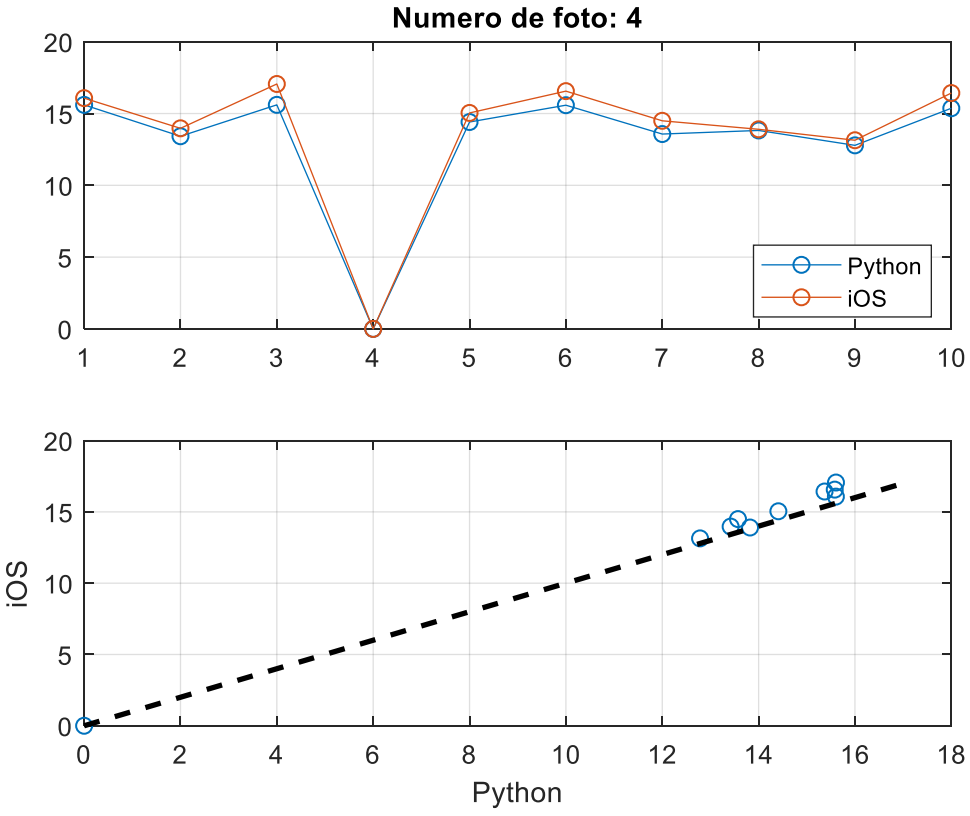

Numero de foto: 3
Python
iOS

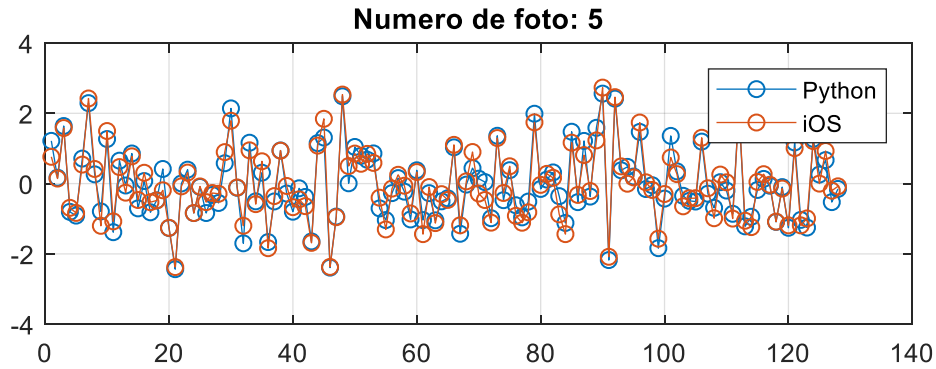

Correlación: 0.973 (p-valor 6.6537e-82)
iOS
Python

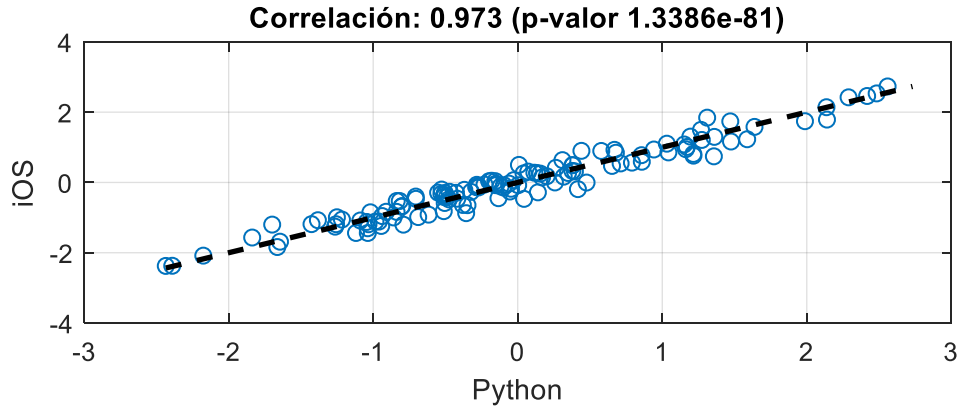

Numero de foto: 3
Python
iOS

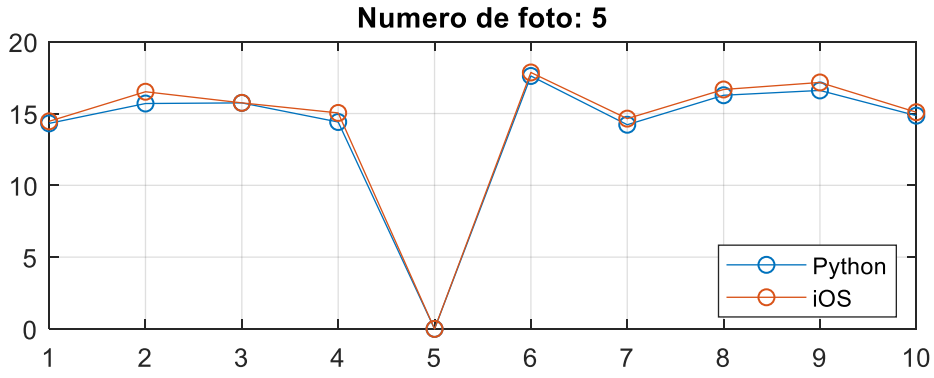

iOS
Python

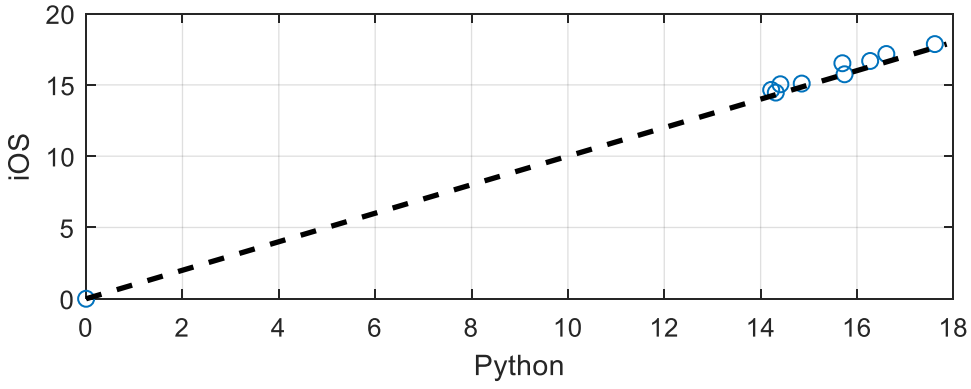

Numero de foto: 4
Python
iOS
Correlación: 0.964 (p-valor 3.5617e-74)
iOS
Python

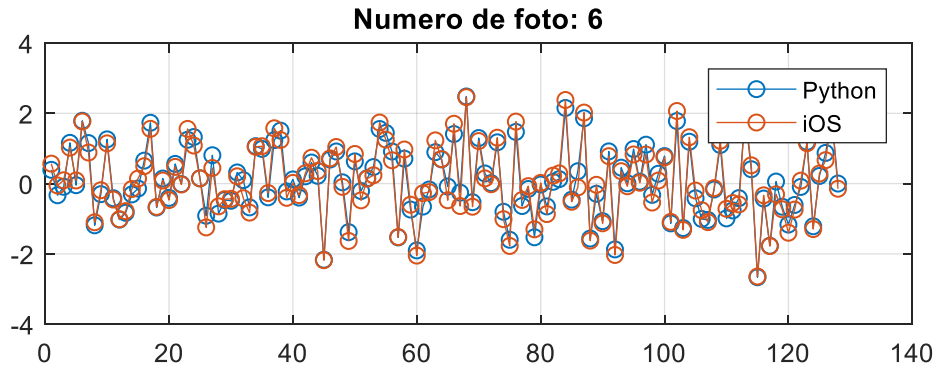

Numero de foto: 4
Python
iOS
iOS
Python

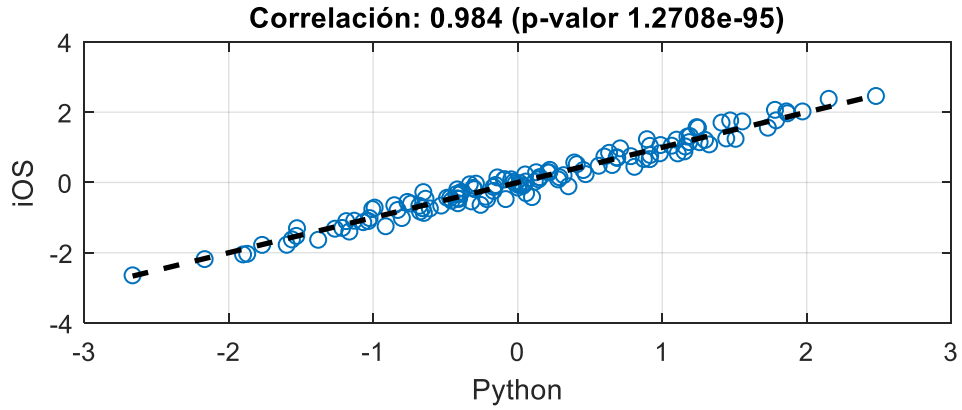

Numero de foto: 5
Python
iOS

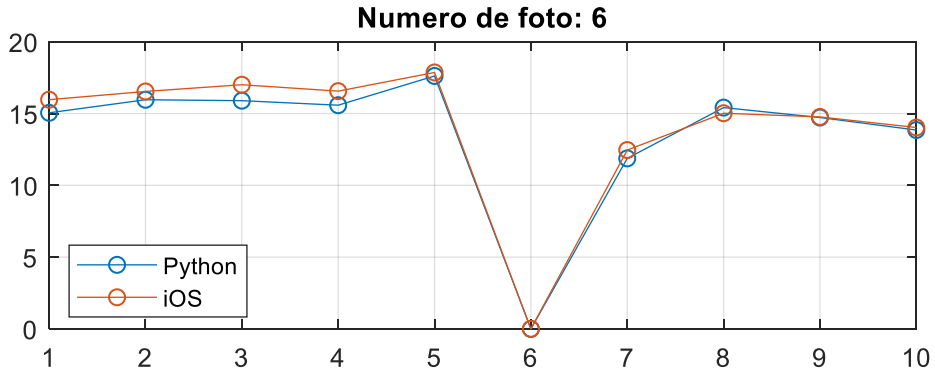

Correlación: 0.973 (p-valor 1.3386e-81)
iOS
Python

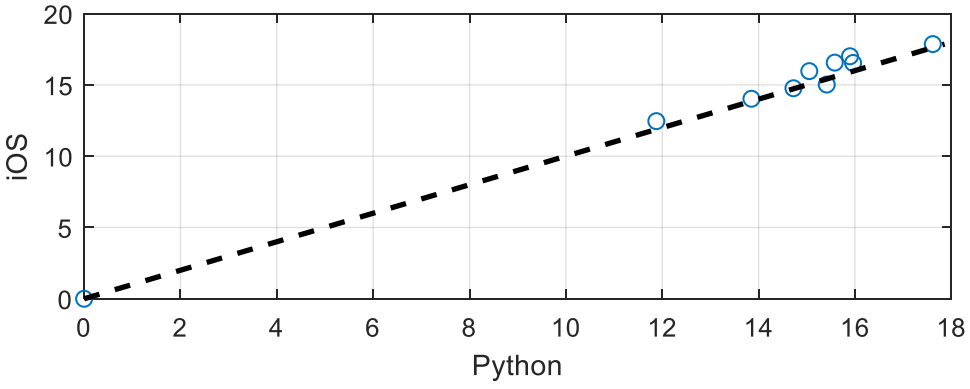

Numero de foto: 5
Python
iOS

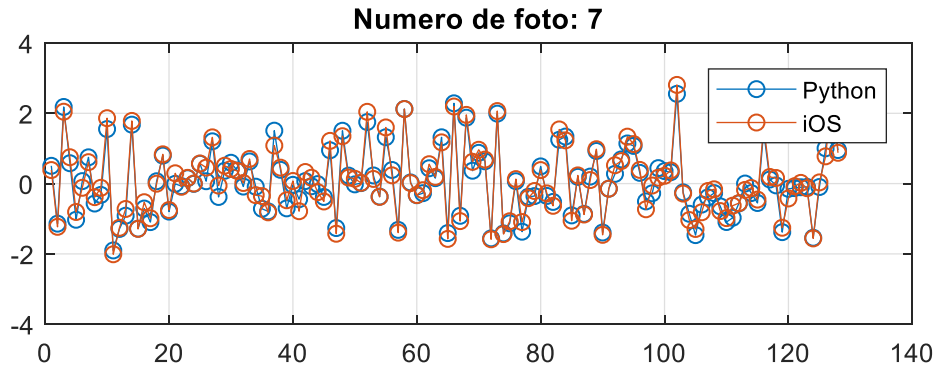

iOS
Python

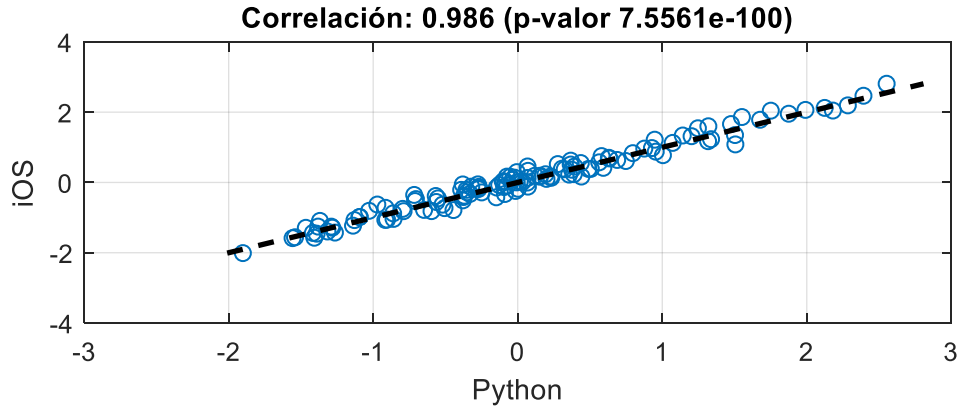

Numero de foto: 6
Python
iOS

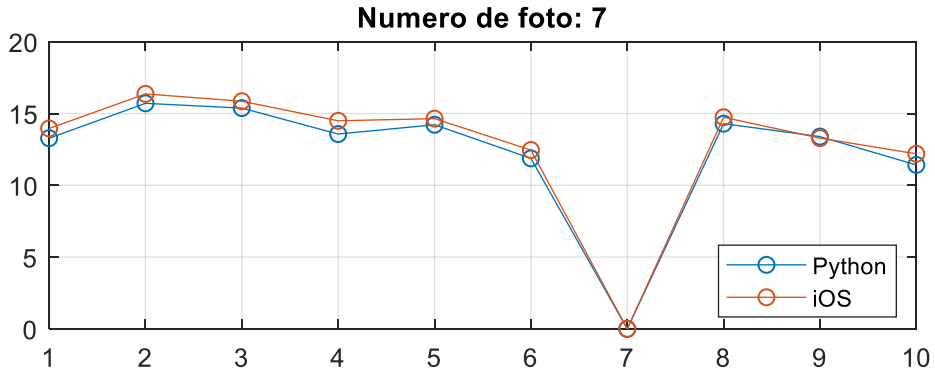

Correlación: 0.984 (p-valor 1.2708e-95)
iOS
Python

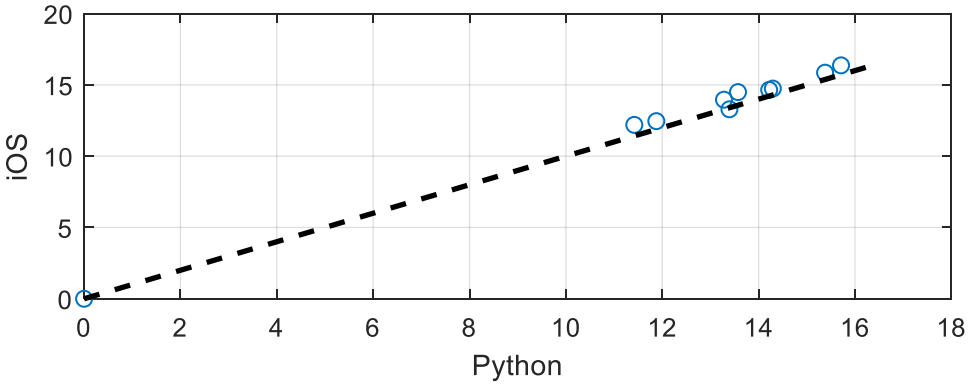

Numero de foto: 6
Python
iOS

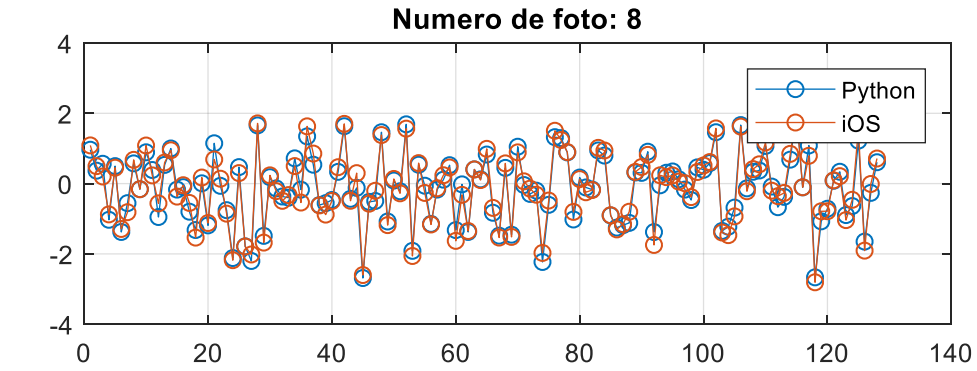

iOS
Python

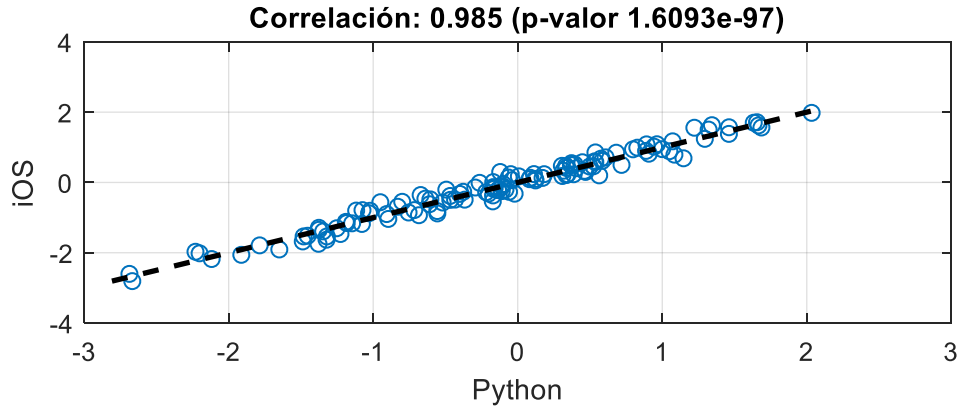

Numero de foto: 7
Python
iOS

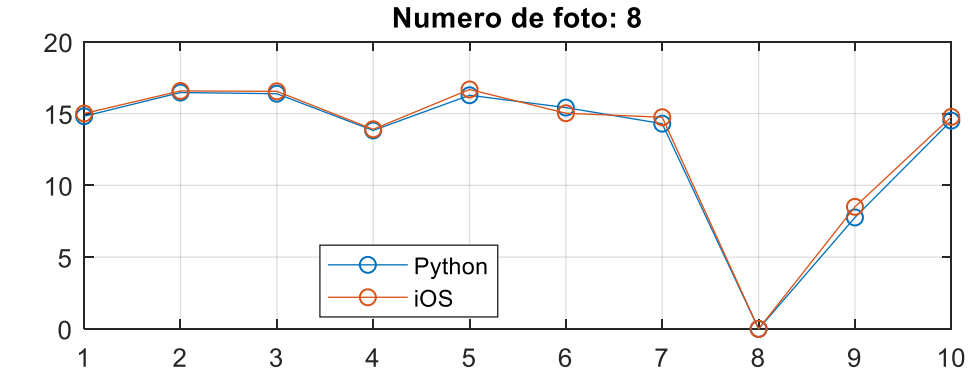

Correlación: 0.986 (p-valor 7.5561e-100)
iOS
Python

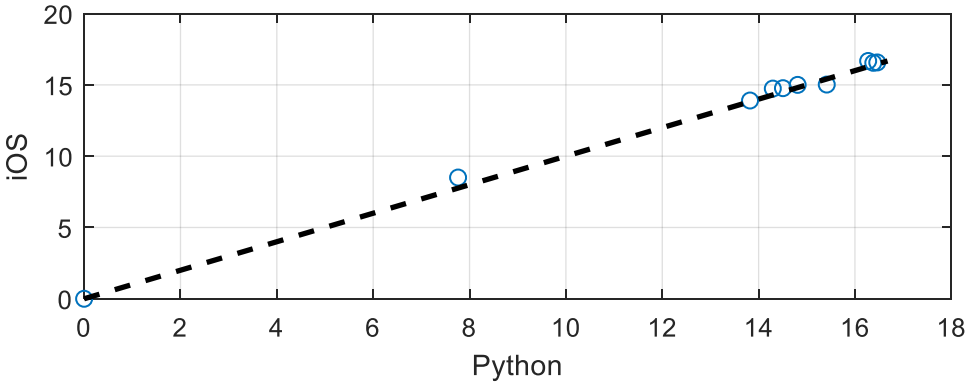

Numero de foto: 7
Python
iOS

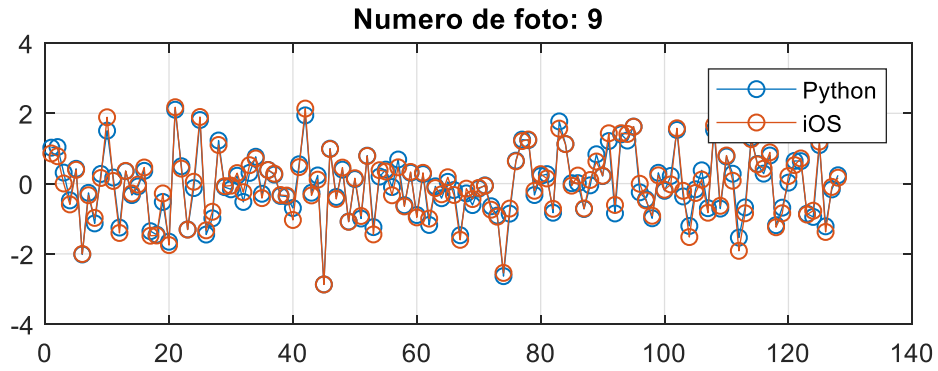

iOS
Python

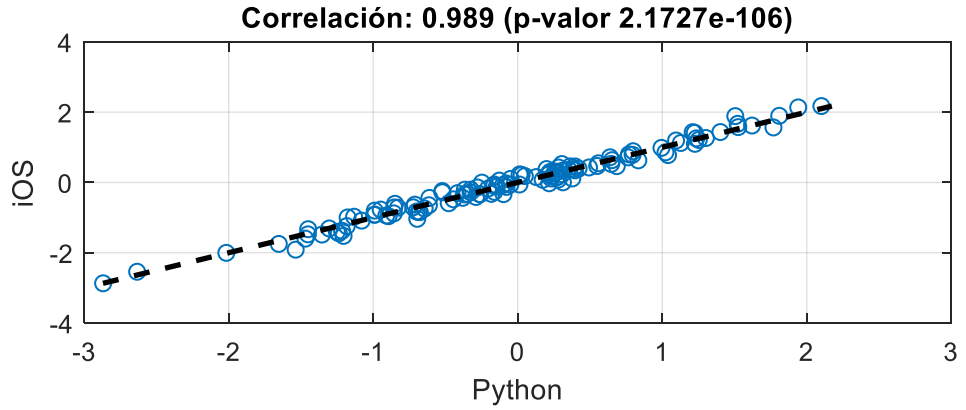

Numero de foto: 8
Python
iOS

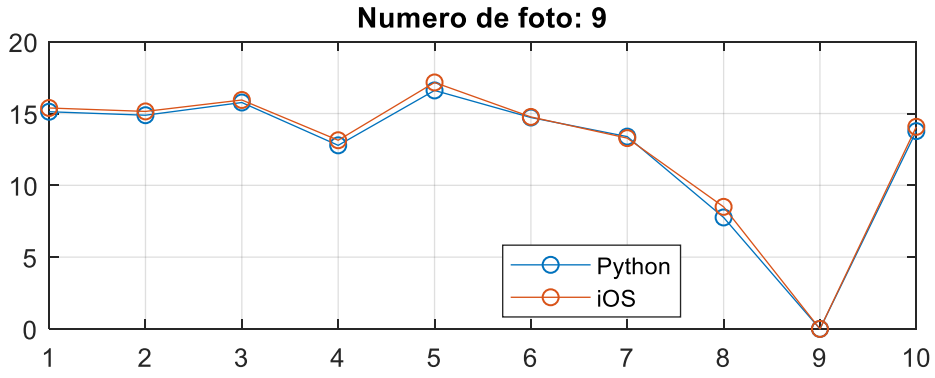

Correlación: 0.985 (p-valor 1.6093e-97)
iOS
Python

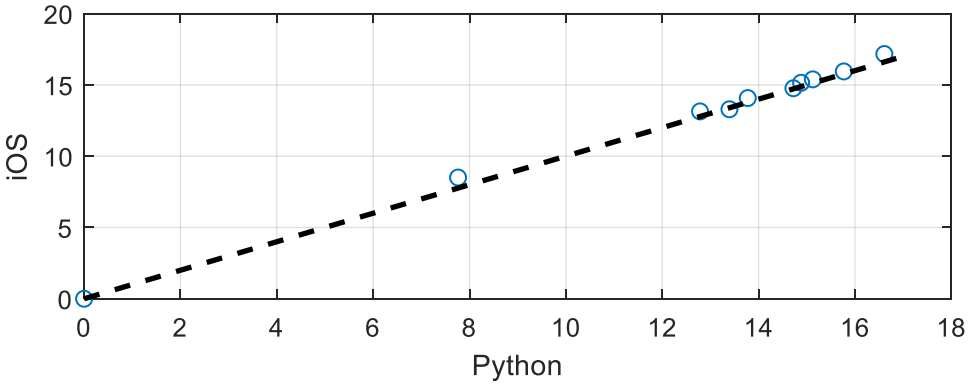

Numero de foto: 8
Python
iOS

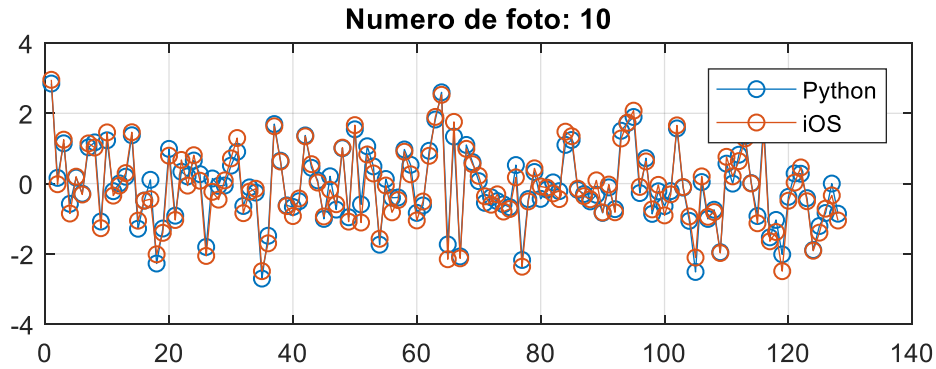

iOS
Python

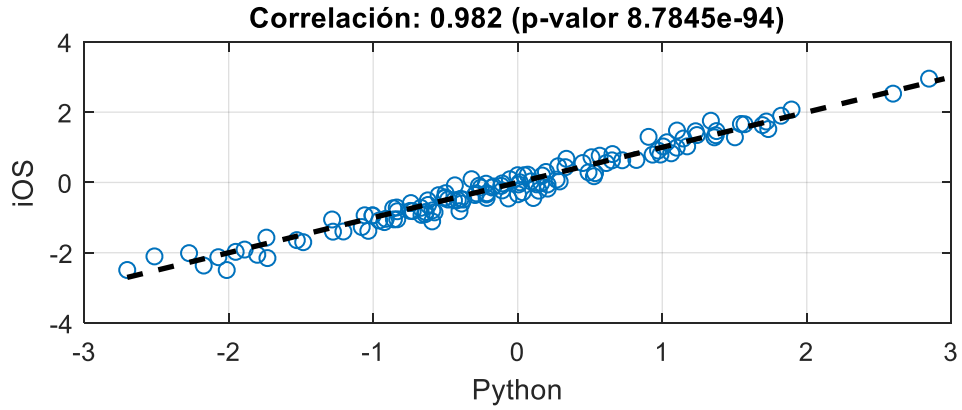

Numero de foto: 9
Python
iOS

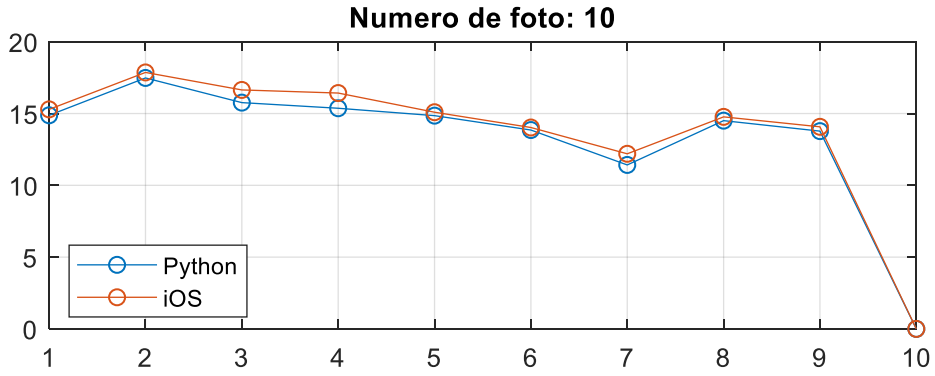

Correlación: 0.989 (p-valor 2.1727e-106)
iOS
Python

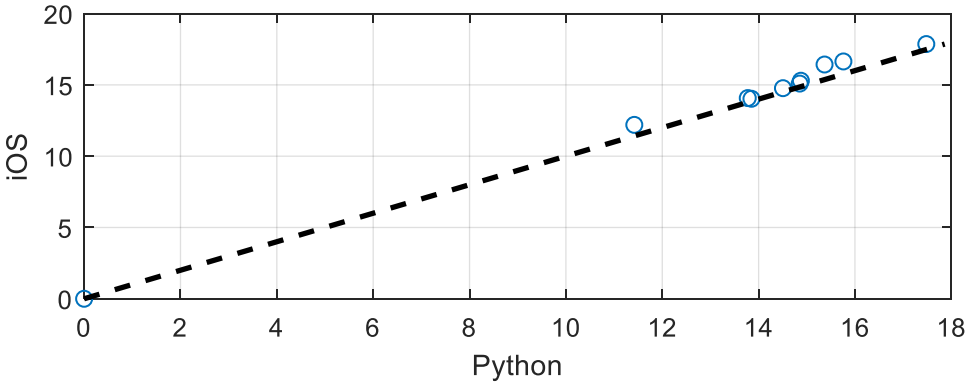

Numero de foto: 9
Python
iOS


iOS
Python


Numero de foto: 10
Python
iOS


Correlación: 0.982 (p-valor 8.7845e-94)
iOS
Python


Numero de foto: 10
Python
iOS


iOS
Python

De este estudio se pueden sacar las siguientes conclusiones:

- La conversión del modelo es correcta: ya que la correlación entre los vectores del modelo original descargado de GitHub y el modelo convertido a Swift es muy alta en todos los casos y el p-valor no permite rechazar la hipótesis de que estos vectores sean iguales.

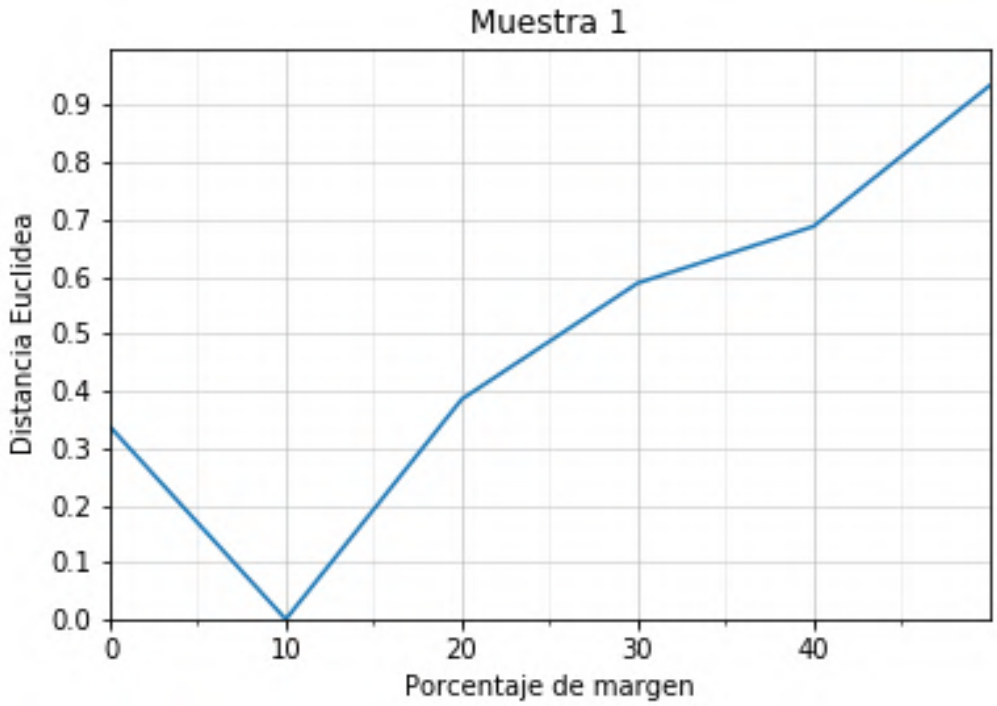
- La distancia euclidea entre todas las muestras es relativamente constante y se encuentran alejados unos de otros, excepto entre las imágenes 8 y 9 y en menor mediad entre la 7 y la 10. Esto tiene sentido, ya que las muestras de las imágenes 8 y 9 se corresponden con una imagen de la misma persona (el actor Leornado Dicaprio), solo que la imagen 8 se trata de un montaje de la imagen 9, por parte de un artista fotográfico, en la que la cara de la imagen 9 está combinada o mezclada en cierta medida con la de otra persona (el actor Sean Penn) [14]. Las personas correspondientes a las muestras 7 y 10 tienen un parentesco de madre e hija, por la tanto no es tan raro que guarden cierta similitud, aunque se observa que este parecido en sus vectores es claramente menor que el existente entre las imágenes 8 y 9.

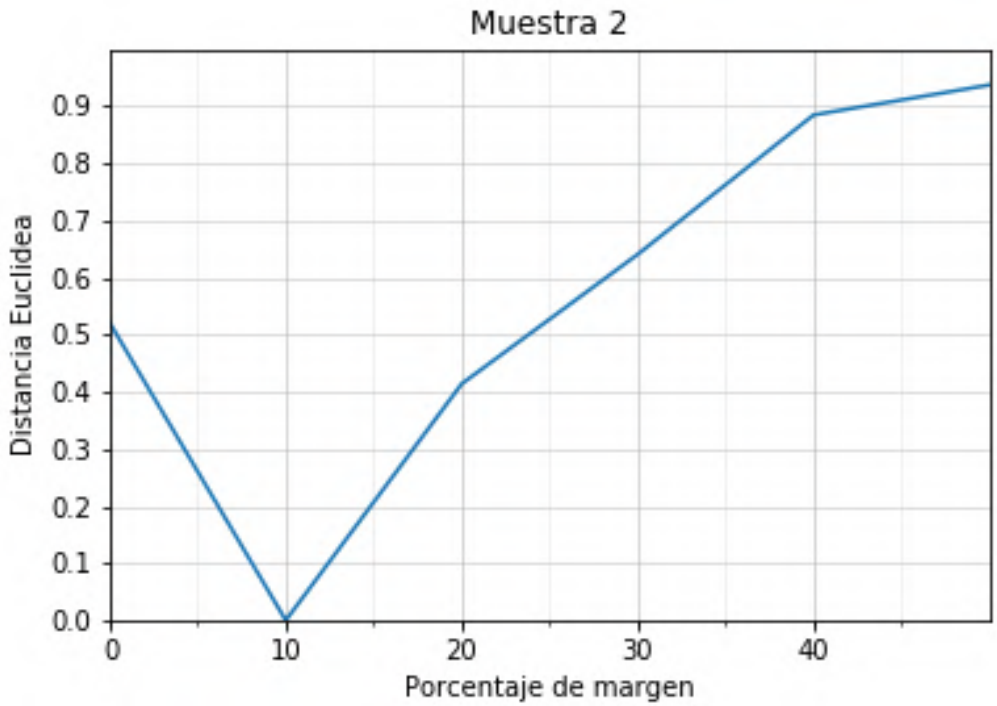
4.2. Análisis de robustez ante cambios en el tamaño del margen de la imagen de entrada

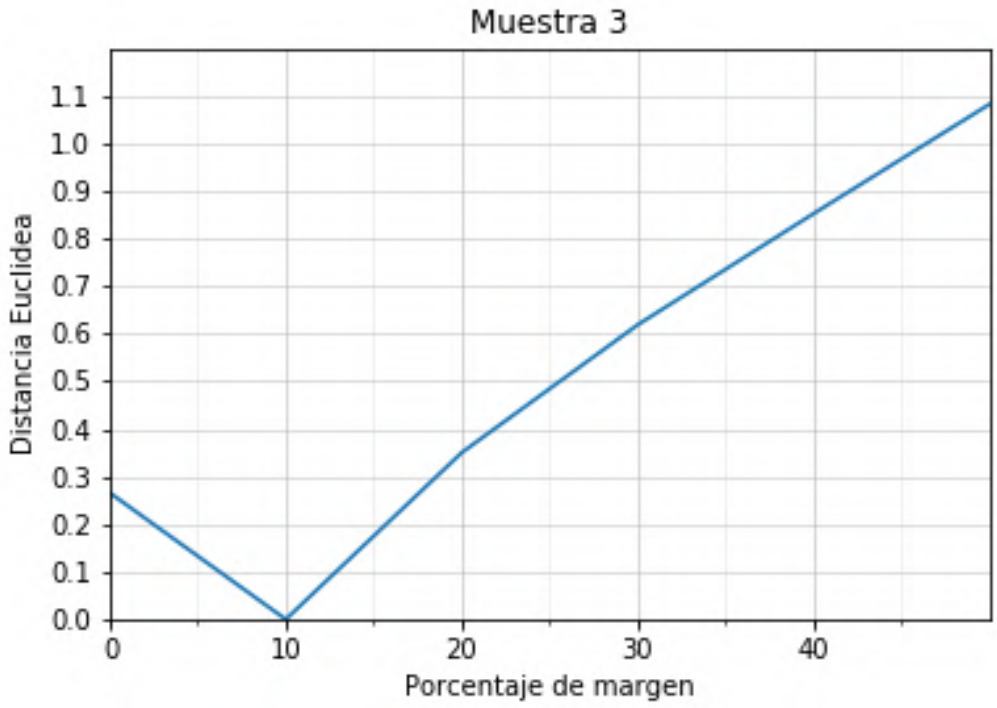
Se analizó la distancia euclidea desde cada una de las imágenes a la de margen del 10%, ya que es la que venía recomendada por distintos investigadores como el margen con el que mejores resultados de fiabilidad habían obtenido [6].

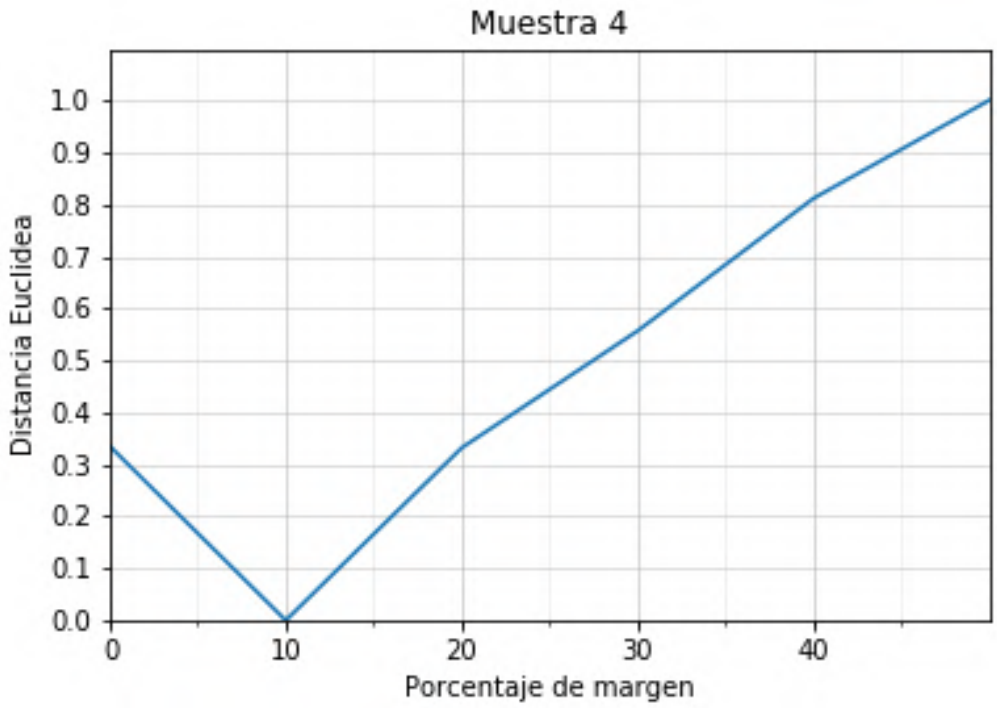
En la Figura 22 se pueden ver las muestras que se han empleado para este análisis y a continuación se muestran las gráficas del análisis para cada muestra, en las que se presenta la distancia euclidea entre las imágenes y la imagen con un 10% de margen (eje Y), frente al porcentaje de margen de cada una de las imágenes (eje X).

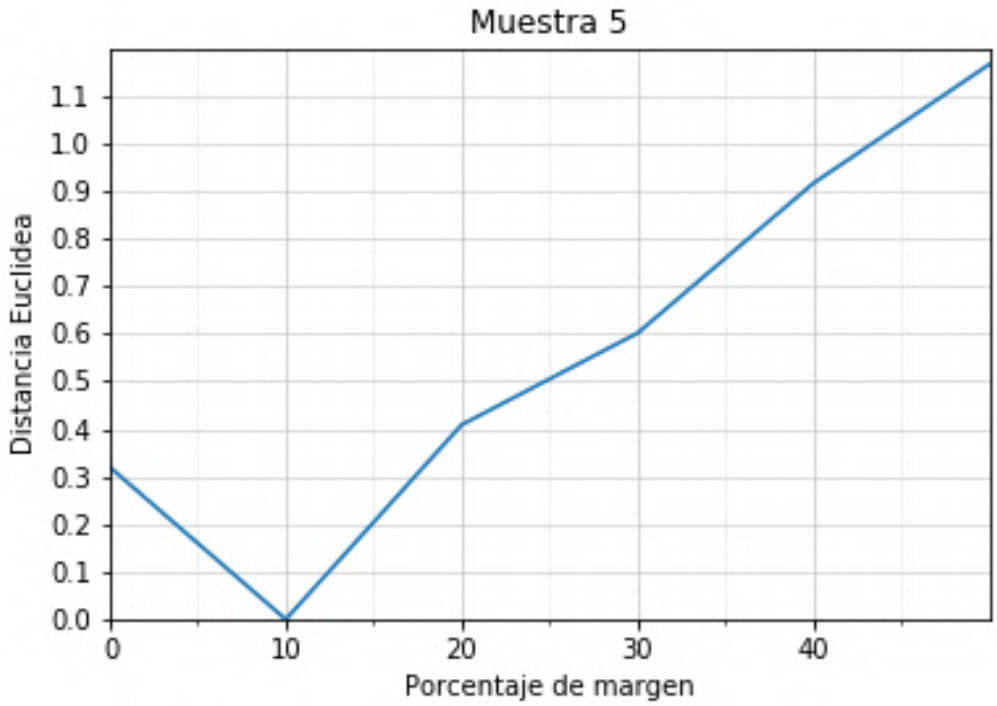
Figura 22: Muestras empleadas para el análisis de la influencia del margen

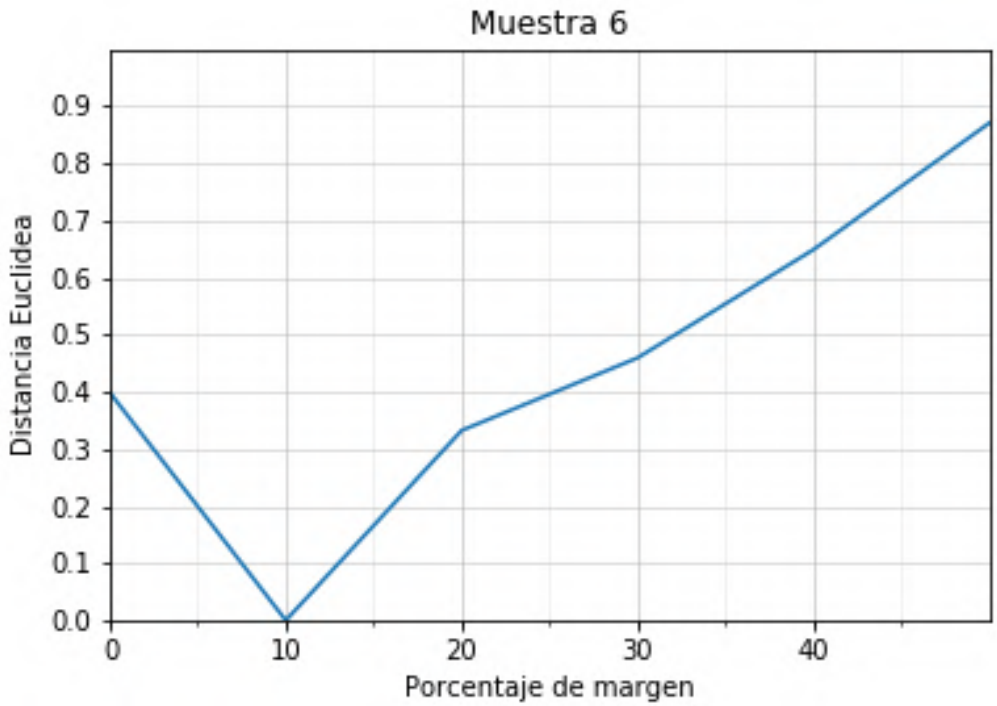

Muestra 1
Distancia Euclidea
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
Porcentaje de margen

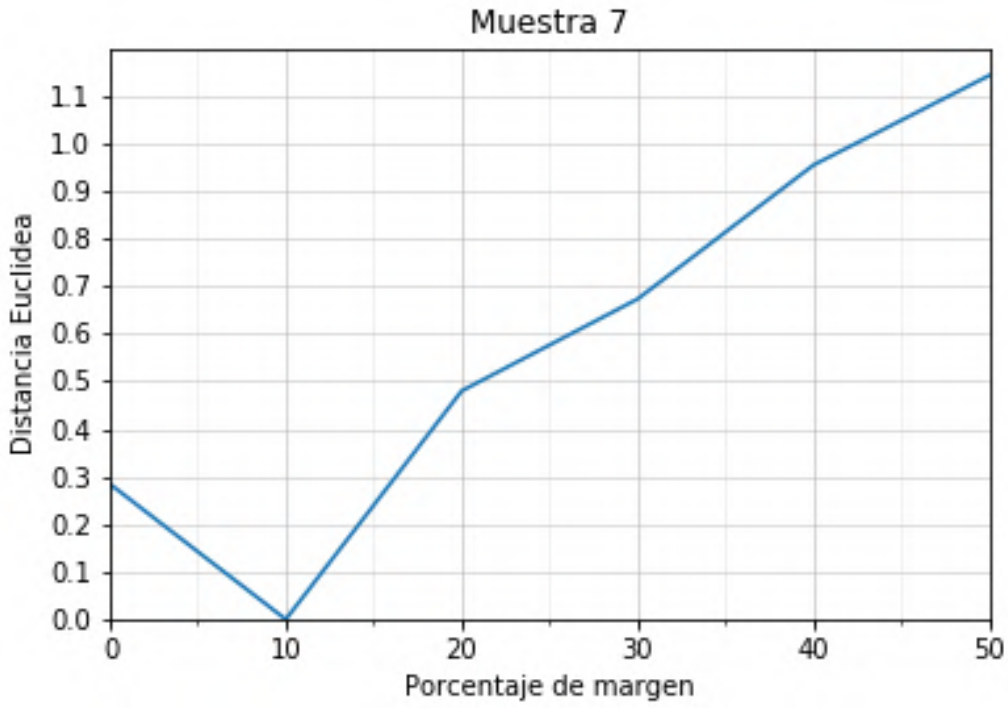

Muestra 2
Distancia Euclidea
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
Porcentaje de margen

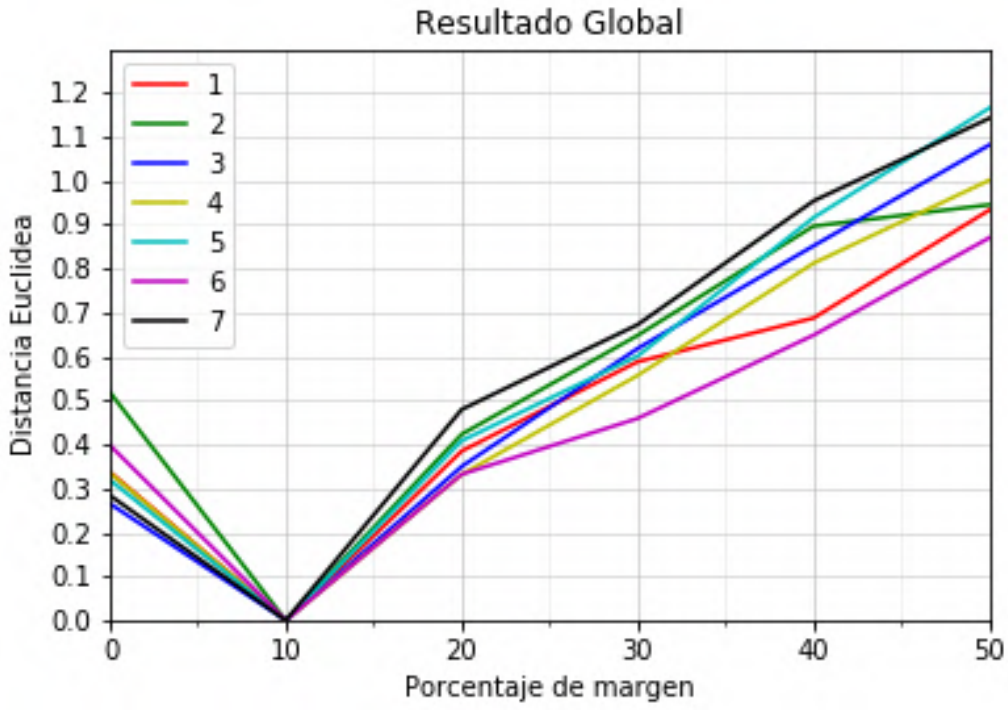

Muestra 3
Distancia Euclidea
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
Porcentaje de margen


Muestra 4
Distancia Euclidea
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
Porcentaje de margen


Muestra 5
Distancia Euclidea
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
Porcentaje de margen


Muestra 6
Distancia Euclidea
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
Porcentaje de margen


Muestra 7
Distancia Euclidea
Porcentaje de margen
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
0
10
20
30
40
50


Resultado Global
Distancia Euclidea
Porcentaje de margen
1
2
3
4
5
6
7
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
0
10
20
30
40
50

Este análisis muestra como efectivamente el modelo es dependiente del margen que se establezca entre la cara y el borde de la imagen que se introduce a la red neuronal que es siempre de 160x160 pixeles. Esto parece bastante lógico, ya que el proceso de introducir las imágenes siempre con el mismo porcentaje de margen es parte del proceso de normalizar las imágenes. Si el modelo se ha entrenado siempre con un margen determinado hace que, en el caso de introducir una imagen con un margen diferente entre la cara y el borde, esta sea detectada por el modelo como un usuario completamente distinto, ya que el tamaño de la cara y los elementos que la componen (ojos, boca, nariz…) también es un elemento diferenciador de una cara. Por eso cuanto más se aleja el margen escogido del de 10% mayor es la distancia euclidea entre las dos imágenes.

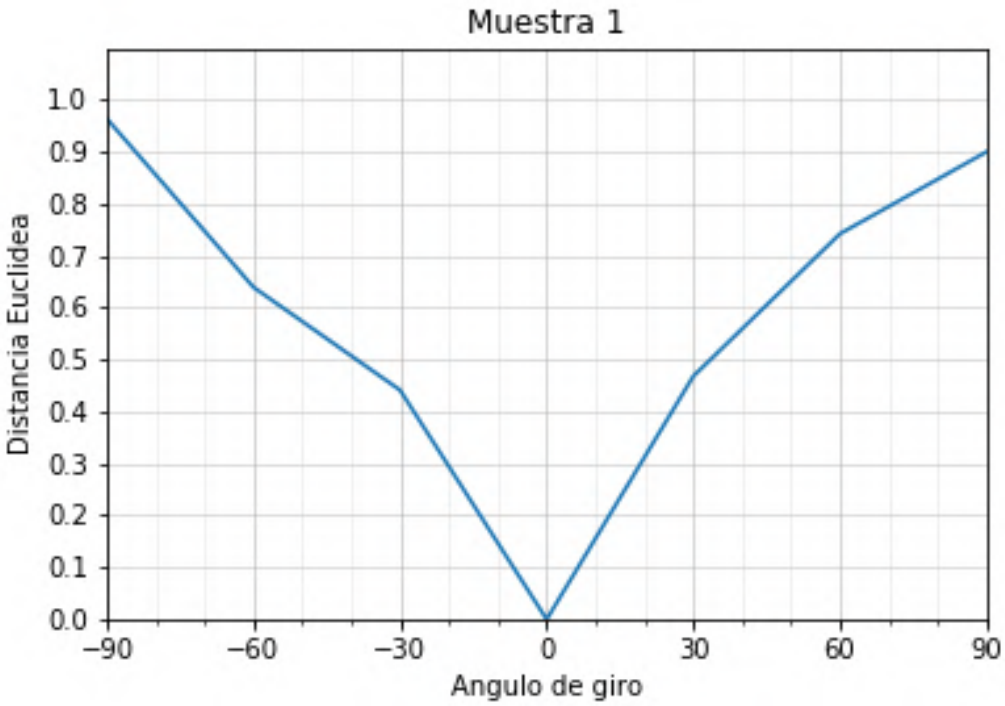
La manera de evitar la influencia de este factor en la aplicación es siendo constante en el margen que se les aplica a las imágenes entre la cara y el borde antes de introducirlas en el modelo. En el caso de esta App se ha usado un margen del 10% por, como se ha mencionado anteriormente, los estudios llevados a cabo por algunos usuarios de GitHub que ya habían empleado esta técnica [6].

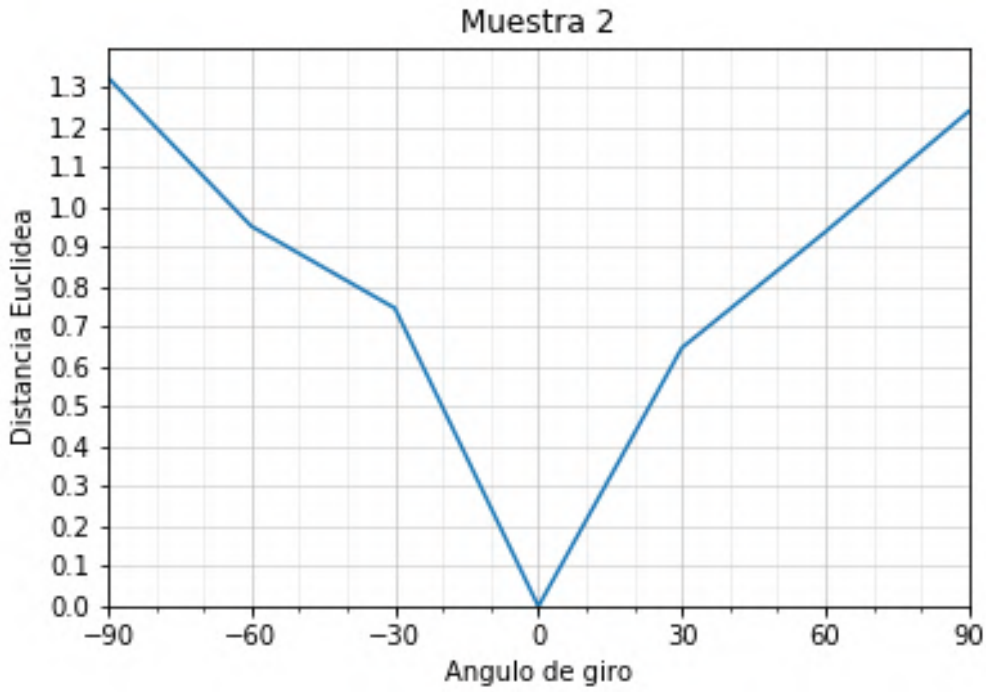
4.3. Análisis de robustez frente a giros del ángulo de perfil de la cara

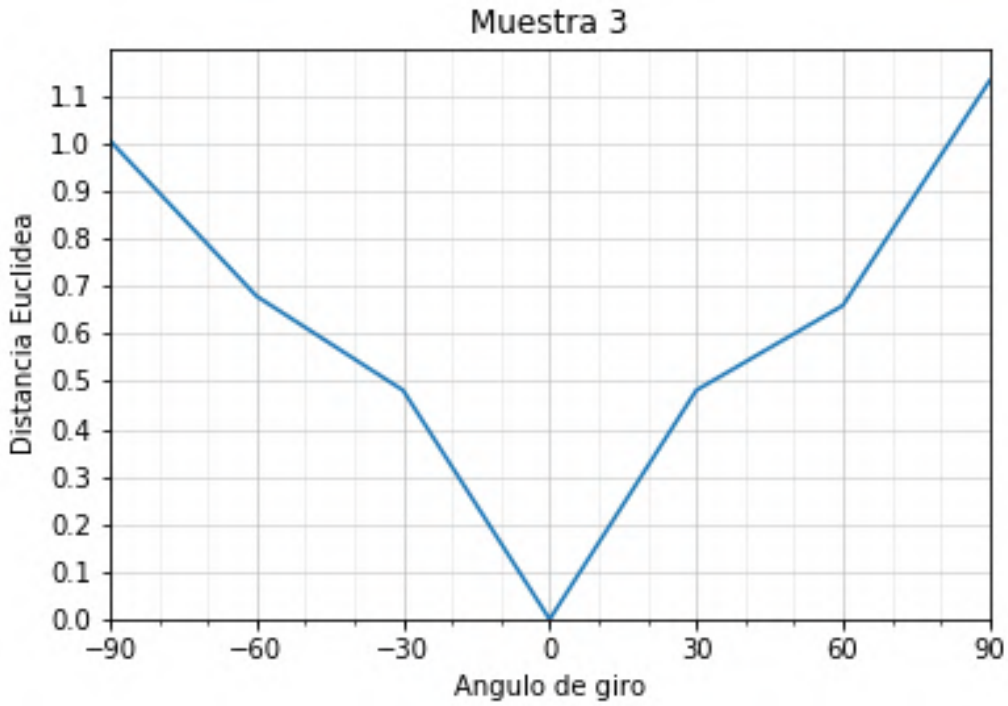
En estos ensayos se muestra como influye al modelo el hecho de que una persona sea detectada en una posición que no sea la frontal, que en teoría es en la que va a ser registrada y la que posición cuyo vector se guardaría en la base de datos. Estos análisis se hicieron cogiendo un margen del 10% respecto a las coordenadas que devuelve la librería de *Vision*.

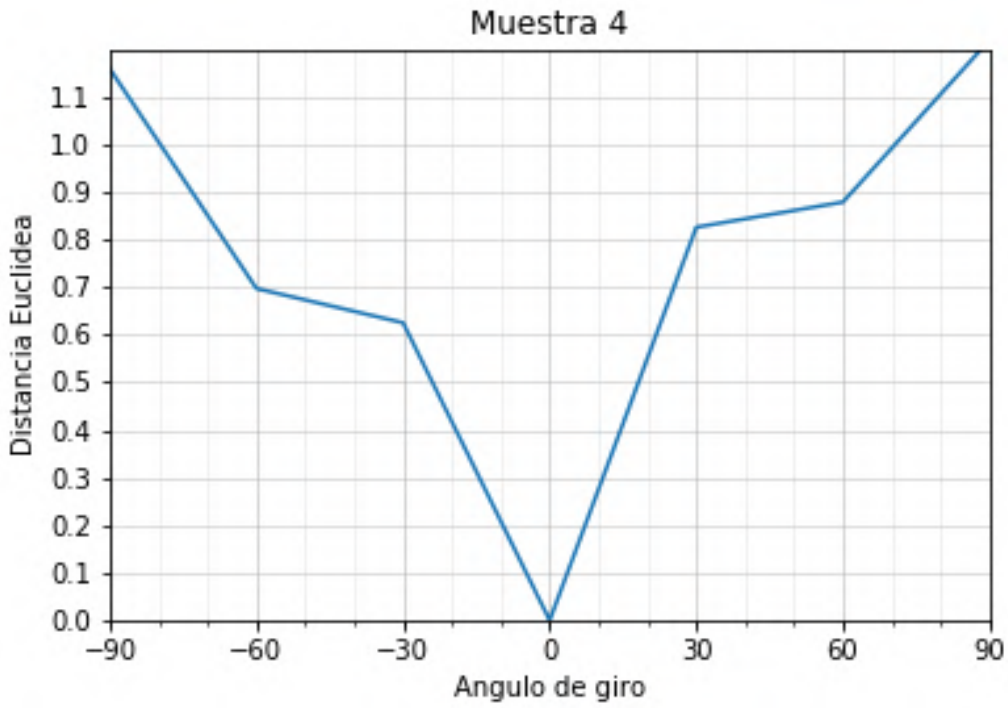
A continuación, se presentan 7 ejemplos de un estudio en el que se pretendía ver en que medida afecta la variable del ángulo de giro de la cara a la robustez del modelo. Para el estudio se tomaron 7 muestras de cada usuario en las que se observa el giro de la cara desde un perfil hasta el contrario con un intervalo de 30º de giro entre imagen e imagen. La gráfica muestra la distancia euclidea de los vectores característicos de la cara obtenidos por la aplicación para cada imagen con respecto a la imagen frontal del usuario, que es el vector que, en teoría, se almacena en la base de datos para el posterior reconocimiento del usuario. Las muestras se pueden ver en la Figura 23

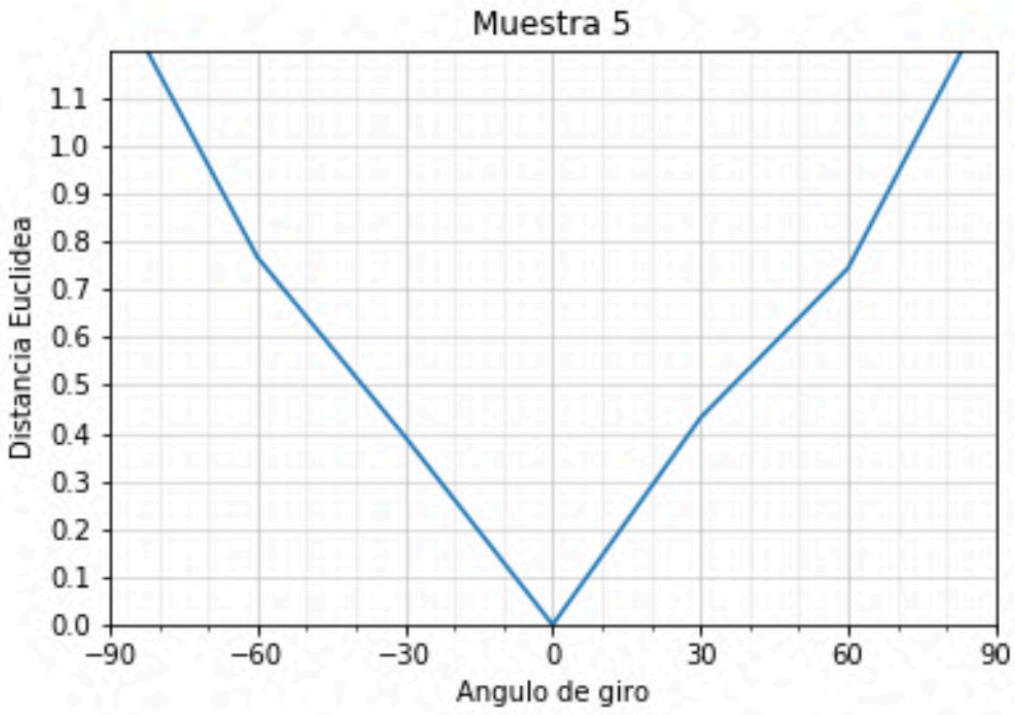
Figura 23: Muestras empleadas para el análisis de robustez frente al ángulo de giro

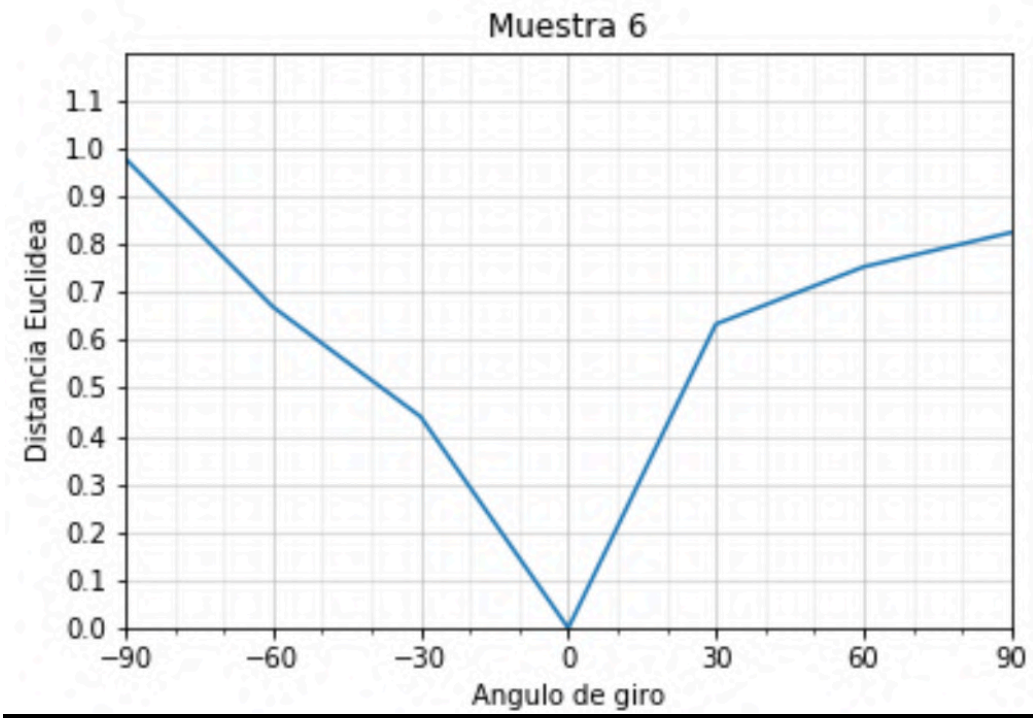

Muestra 1
Distancia Euclidea
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−90
−60
−30
0
30
60
90
Angulo de giro

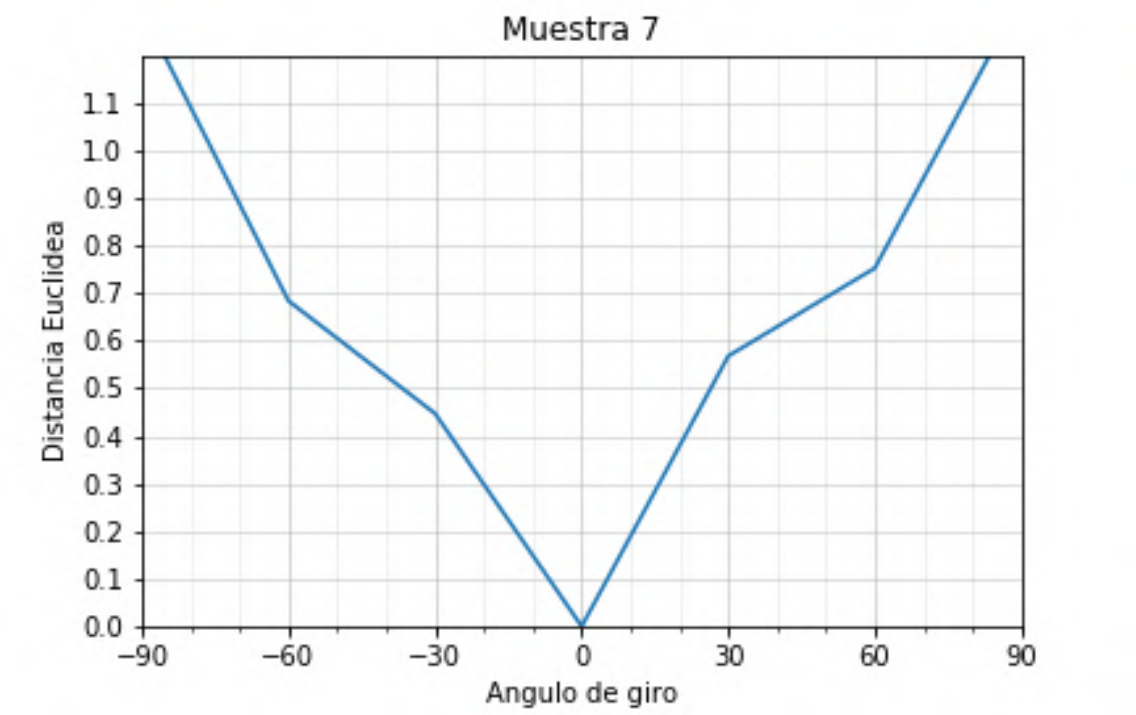

Muestra 2
Distancia Euclidea
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−90
−60
−30
0
30
60
90
Angulo de giro

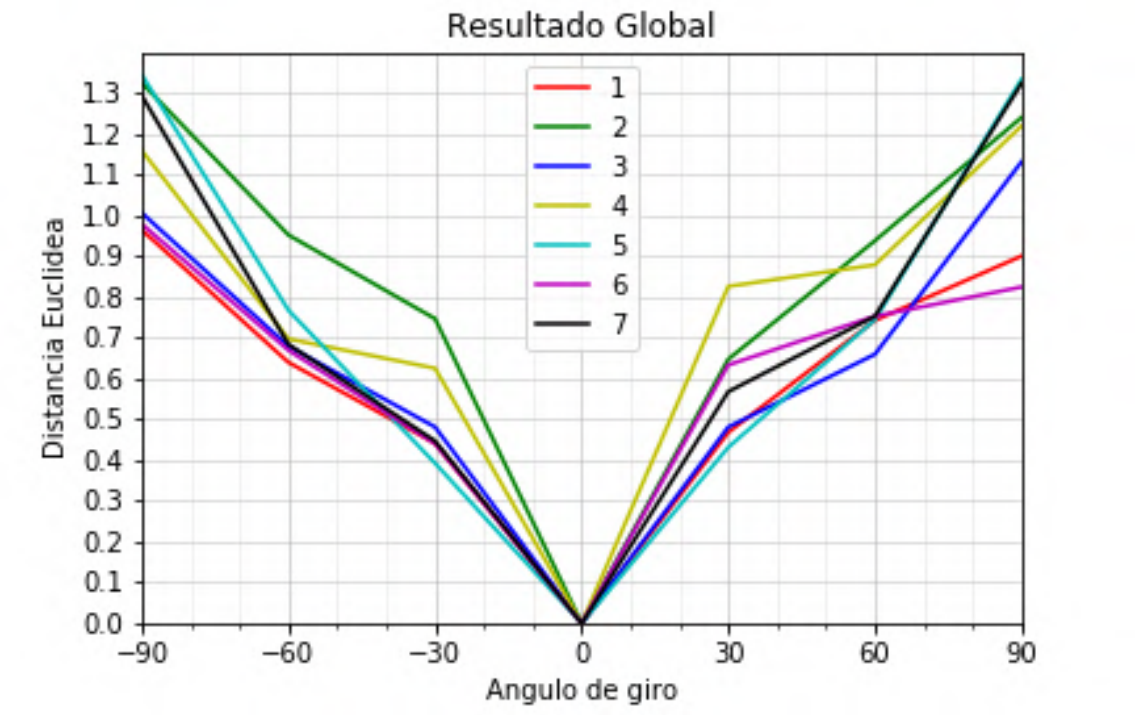

Muestra 3
Distancia Euclidea
Angulo de giro
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−90
−60
−30
0
30
60
90


Muestra 4
Distancia Euclidea
Angulo de giro
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−90
−60
−30
0
30
60
90


Muestra 5
Distancia Euclidea
Angulo de giro
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−90
−60
−30
0
30
60
90


Muestra 6
Distancia Euclidea
Angulo de giro
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−90
−60
−30
0
30
60
90


Muestra 7
Distancia Euclidea
Angulo de giro
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−90
−60
−30
0
30
60
90


Resultado Global
Distancia Euclidea
Angulo de giro
1
2
3
4
5
6
7
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−90
−60
−30
0
30
60
90

En las muestras individuales y en la de resultado global, podemos apreciar como a medida que la persona se va girando, el modelo tiene más dificultades para determinar que esa persona es la de la imagen frontal, que se supone que es la muestra que se tiene del registro. Si nos ciñéramos a el umbral de 0.65 propuesto para el reconocimiento los casos más desfavorables, que son la muestra 2 en sentido negativo de giro y la muestra 4 en sentido positivo, no nos permitirían reconocer a esos usuarios a partir de un ángulo de giro de unos 20º-25º en ambos casos, lo cual requiere de una mejora para aumentar la robustez del modelo.

En las conclusiones se plantearán las posibles mejoras que se pueden introducir en el sistema de reconocimiento facial para que esto no sea un problema a la hora de reconocer a un usuario.

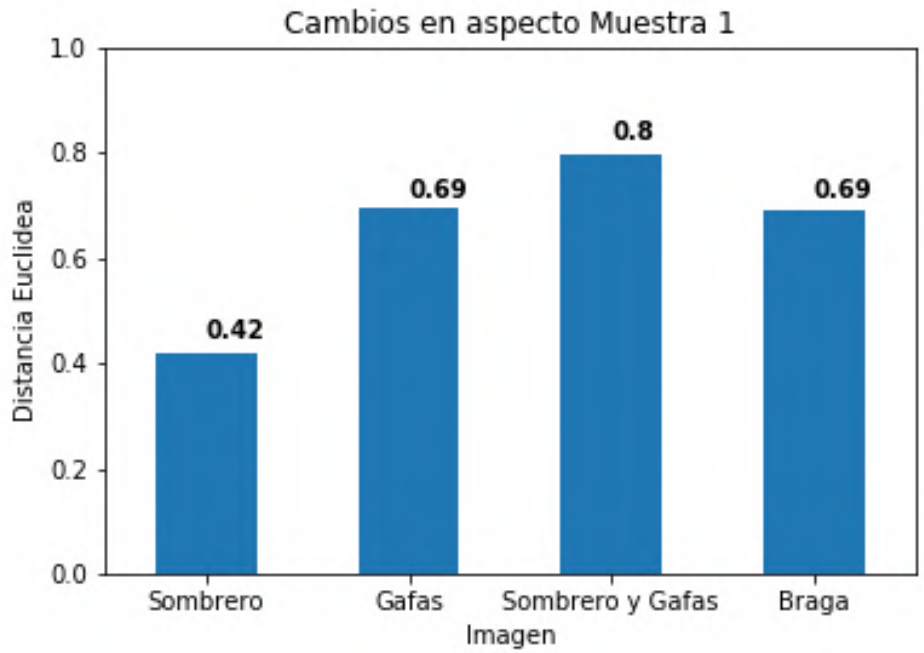
4.4. Análisis de cambios en el aspecto

En estos estudios se plantea la robustez del modelo frente a cambios del día a día, como pueden ser el hecho de llevar o no gafas, ponerse una bufanda o braga o llevar gorra o sombrero. Se presenta un análisis con la distancia euclidea de diversas imágenes de un usuario que ha cambiado su aspecto con respecto a su imagen de registro, en la cual no lleva puesto ningún accesorio.

Se emplearon muestras de dos usuarios para este análisis. Las imágenes usadas para el análisis del primer usuario junto con su gráfica de resultados se muestran en la Figura 24 y la del segundo usuario en la . Todas las imágenes se introdujeron en el modelo empleando un margen del 10% respecto a la caja de la cara devuelta por el sistema de detección facial empleado, para más información ver el apartado 4.2. de este documento.

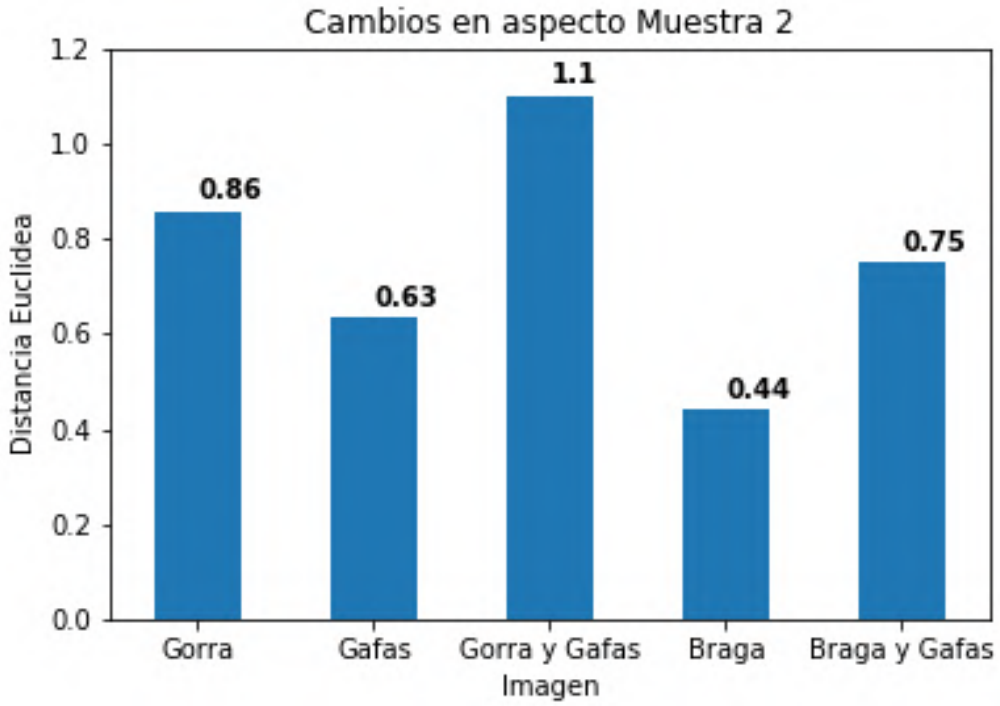
Análisis Muestra 1

Figura 24: Imágenes de la muestra 1 empleadas en el análisis de la robustez frente a cambios en el aspecto



Análisis Muestra 2

Figura 25: Imágenes de la muestra 1 empleadas en el análisis de la robustez frente a cambios en el aspecto



En los análisis de ambas muestras podemos ver como los cambios cotidianos de aspecto también afectan a la robustez del modelo. Si nos ceñimos al umbral de 0.65 que se empleó en la aplicación, vemos como más de la mitad de las muestras no serían reconocidas como esa persona, lo cual es algo que no se puede permitir y para solucionarlo se presentan algunas mejoras en el Capítulo 5. También cabe destacar que el hecho de coger un determinado margen también influye en algunas de las imágenes, ya que la imagen del sombrero de la muestra 1 presenta una distancia euclidea pequeña, ya que el margen hace que la influencia del sombrero apenas se vea.

Capítulo 5: Conclusiones y mejoras

5.1. Conclusiones generales

En las conclusiones del proyecto cabe destacar en primer lugar, que se logró el objetivo principal del proyecto, el cual consistía en la elaboración una aplicación de reconocimiento facial funcional. Se consiguió elaborar una App capaz de reconocer personas y, además, se llevaron a cabo análisis y estudios para sentar las bases de cara a futuras mejoras. Esta App fue elaborada para dispositivos IOS y probada en varios dispositivos físicos con resultados satisfactorios.

5.2 Mejores implementadas y analizadas

Se han llevado a cabo mejora para la App y también se ha dejado la puerta abierta a diferentes mejoras para el proyecto, que son las siguientes:

- **Conectividad entre dispositivos:** esto se logra consiguiendo conectar la App a icloud y permitiéndole compartir la información de su memoria persistente de datos con otros dispositivos.

  En principio esta mejora no debería ser difícil de lograr, ya que simplemente hay que comprar la licencia de desarrollador necesaria para poder usar ese tipo de tecnología y a partir de ahí hay diversos tutoriales en plataformas de internet que explican de manera sencilla como poder implantar una solución para intercambiar información entre dispositivos móviles.

- **Aumento de la robustez de la aplicación:** en respuesta a las limitaciones encontradas en la aplicación en los análisis del Capítulo 4, se han valorado las siguientes soluciones, llegando alguna de ellas a avanzarse o desarrollarse:

  - **Métodos para la normalización de las caras antes de ser introducidas en la red neuronal:** en este apartado se engloban todos aquellos métodos que tienen como objetivo mejorar la eficiencia del modelo de la red neuronal siamesas haciendo un tratamiento previo a las imágenes de las caras, de manera que todas ellas estén normalizadas en el parámetro sobre el que influye la normalización.

    En este proyecto se llevaron a cabo dos de estos métodos. El primero de ellos es el hecho de que todas las imágenes posean el mismo margen desde el cuadro de la cara que proporciona la herramienta de *visión* y el borde de la imagen que va a ser introducida en la red neuronal. Este margen se fijó en un 10%, de manera, que todas las imágenes tendrán el mismo y cualquier variación que se pudiera producir en los resultados a causa de esta variable queda totalmente eliminada. Un estudio sobre la influencia de esta variable en la salida se puede ver en el apartado 4.2 de este documento.

    El segundo método tiene que ver con la normalización con respecto a la variación que se produce en los resultados por los ladeos de cabeza de un usuario. En este caso la solución se implantó de manera sencilla

empleando la librería de *vision* Apple. Para ello, se utilizo la función de esta librería que devuelve la línea de la mitad de la cara que detecta y otra función que se creó en Swift para devolver el ángulo entre la línea de la mitad de la cara y el eje Y de la imagen. Una vez se tiene el ángulo se rota la imagen de manera que la cara queda alineada con el eje Y y es esta nueva imagen la que se recorta y se introduce a la red neuronal. En la Figura 26 se puede ver con un pequeño ejemplo como funcionaría esta técnica dentro de la App

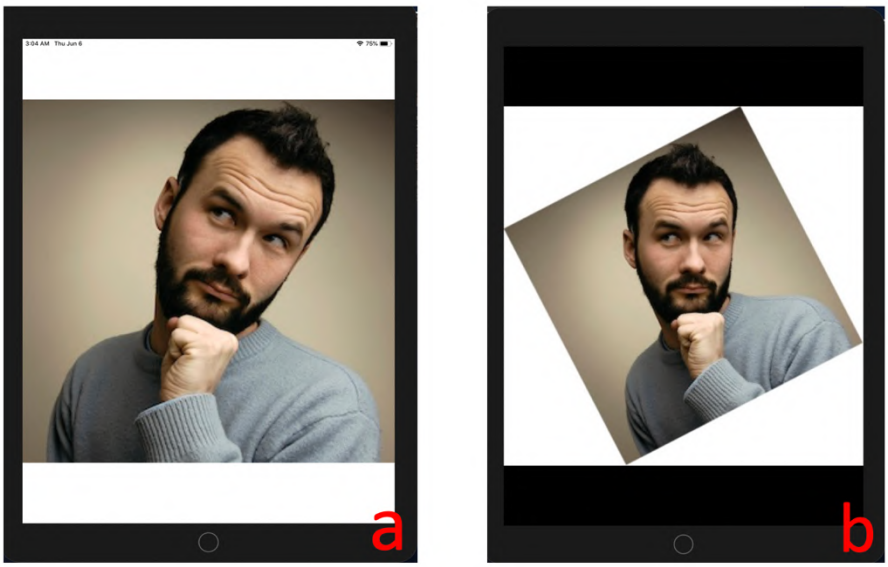


Figura 26: Ejemplo, con un pequeño código, de como funciona la mejora introducida. (a) Imagen que recibiría la App. (b) Imagen girada para poner la cara recta y que será recortada para introducir en la red neuronal.

- **Métodos para la mejora de los resultados de reconocimiento tras la red neuronal siamesa:** con estos métodos se busca mejorar los resultados a la hora de tomar una decisión sobre quién es la persona detectada. Estas soluciones irían orientadas a mejorar las debilidades mostradas por el modelo en los resultados de los apartados 4.3 y 4.4.

  La solución más prometedora de las estudiadas para solventar estas limitaciones tiene que ver con las GANs (Generative Adversarial Networks) o Redes Neuronales Generativas Adversarias. Esta técnica se orientaría a la generación de datos sintéticos, permitiendo obtener sets de datos de la cara de un usuario mas ricos tanto en variedad como en cantidad de datos. Esta técnica se complementa a la perfección con la idea de *one-shot learning* de este proyecto como se puede ver en una investigación de Samsung en la que han conseguido generar videos completos de personas a partir de una sola imagen, como por ejemplo de

la Mona Lisa [9]. En la Figura 27 se pueden ver algunos de los resultados para hacerse una idea de lo que es capaz esta técnica.

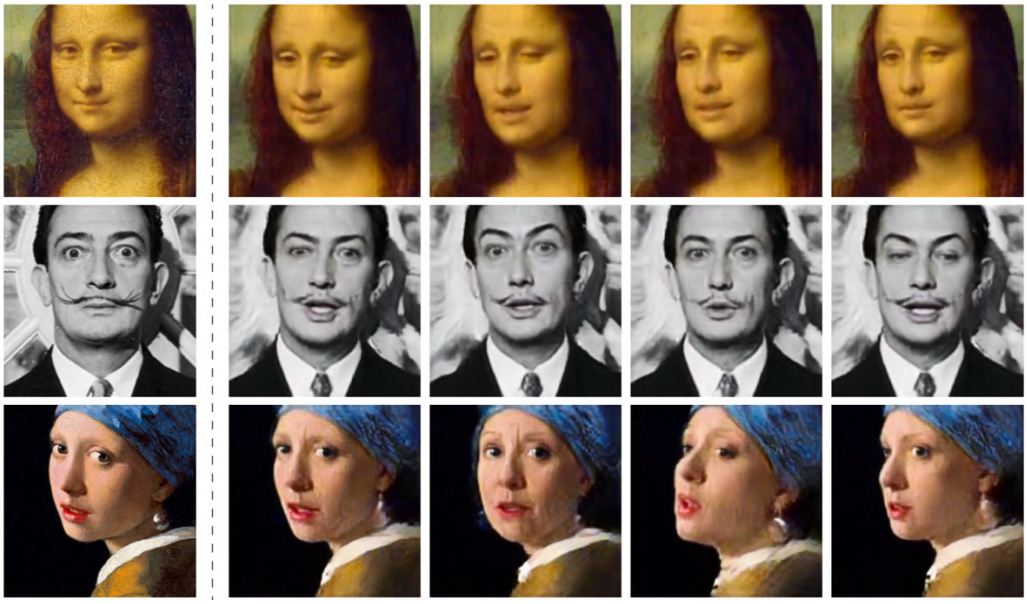


Figura 27: Ejemplo de resultados obtenidos con la técnica adversaria de Few Shots [9].

En la Figura 28 se muestra como funcionaría esta técnica a nivel de la red neuronal. Donde la red llamada como *embedder* sería FaceNet y los demás elementos se corresponderían con la estructura clásica de una GAN.

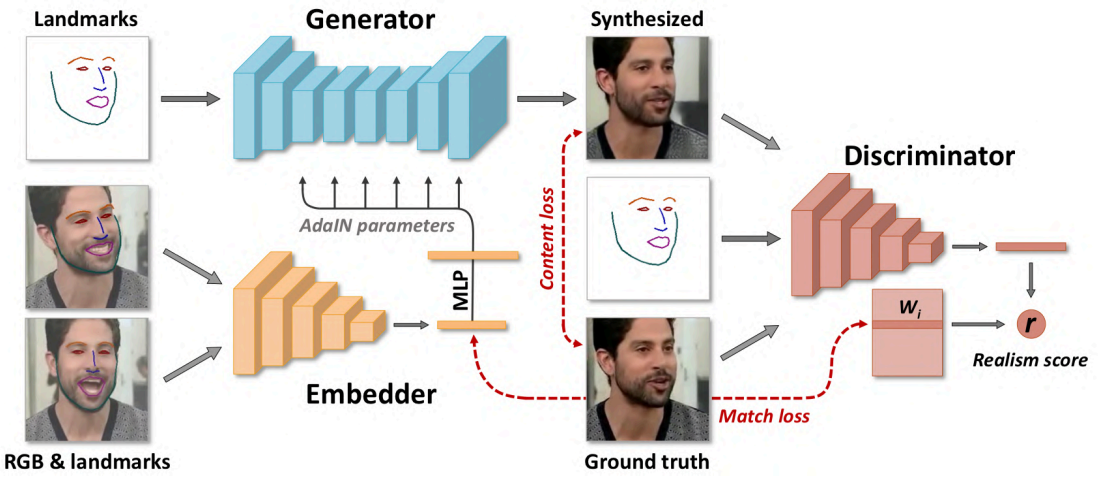


Figura 28: Esquema del funcionamiento de la técnica de aprendizaje adversario de Few-Shots [9].

Técnicas como la de Samsung u otra técnica similar llamada HoloGAN [7], cuyos resultados se muestran en la Figura 29, servirían para solucionar principalmente el problema mostrado en el apartado 4.3. Sin embargo, existe otro tipo de GAN, llamada StyleGAN, desarrollada por Nvidia y que permite aprender diferentes estilos o características, como puede ser el hecho de llevar gafas, un peinado distinto o barba, como se muestra en la Figura 30. Esta técnica resulta ideal para resolver el problema presentado en el apartado 4.4., de falta de robustez por cambios en el aspecto de un usuario.

El objetivo de estas técnicas sería el de generar una gran cantidad de imágenes de un usuario únicamente a partir de una sola imagen y de su vector característico. De esta forma crearíamos un set de datos de ese usuario en diferentes circunstancias que nos permitiría entrenar una herramienta de *machine learning* para actuar como clasificador de manera más eficiente.

Figura 29: Ejemplo de los resultados que se pueden obtener con HoloGan a partir de la imagen central [7].

Figura 30: Ejemplo de resultados obtenidos con StyleGAN [8].

De cara a poder aplicar estas técnicas en un futuro y al igual que se hizo con las herramientas del proyecto en los inicios del mismo, se comenzó la formación en estas nuevas técnicas de las GANs y se llevó a cabo un mini proyecto de una GAN que genera caras nuevas a partir dos sets de datos descargados de la plataforma Keegle. Para ello se siguió un proyecto propuesto en el portal web de Medium [15] y se creo la estructura clásica de una GAN, cuya estructura de discriminante y generador se muestran en la Figura 31 y la Figura 32 respectivamente.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	896
leaky_re_lu_1 (LeakyReLU)	(None, 32, 32, 32)	0
dropout_1 (Dropout)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496
zero_padding2d_1 (ZeroPaddin	(None, 17, 17, 64)	0
batch_normalization_1 (Batch	(None, 17, 17, 64)	256
leaky_re_lu_2 (LeakyReLU)	(None, 17, 17, 64)	0
dropout_2 (Dropout)	(None, 17, 17, 64)	0
conv2d_3 (Conv2D)	(None, 9, 9, 128)	73856
batch_normalization_2 (Batch	(None, 9, 9, 128)	512
leaky_re_lu_3 (LeakyReLU)	(None, 9, 9, 128)	0
dropout_3 (Dropout)	(None, 9, 9, 128)	0
conv2d_4 (Conv2D)	(None, 9, 9, 256)	295168
batch_normalization_3 (Batch	(None, 9, 9, 256)	1024
leaky_re_lu_4 (LeakyReLU)	(None, 9, 9, 256)	0
dropout_4 (Dropout)	(None, 9, 9, 256)	0
conv2d_5 (Conv2D)	(None, 9, 9, 512)	1180160
batch_normalization_4 (Batch	(None, 9, 9, 512)	2048
leaky_re_lu_5 (LeakyReLU)	(None, 9, 9, 512)	0
dropout_5 (Dropout)	(None, 9, 9, 512)	0
flatten_1 (Flatten)	(None, 41472)	0
dense_1 (Dense)	(None, 1)	41473

Total params: 1,613,889
Trainable params: 1,611,969
Non-trainable params: 1,920

Figura 31: Estructura de la red discriminante de la GAN

Layer (type)	Output Shape	Param #
dense_2 (Dense)	(None, 4096)	413696
reshape_1 (Reshape)	(None, 4, 4, 256)	0
up_sampling2d_1 (UpSampling2	(None, 8, 8, 256)	0
conv2d_6 (Conv2D)	(None, 8, 8, 256)	590080
batch_normalization_5 (Batch	(None, 8, 8, 256)	1024
activation_1 (Activation)	(None, 8, 8, 256)	0
up_sampling2d_2 (UpSampling2	(None, 16, 16, 256)	0
conv2d_7 (Conv2D)	(None, 16, 16, 256)	590080
batch_normalization_6 (Batch	(None, 16, 16, 256)	1024
activation_2 (Activation)	(None, 16, 16, 256)	0
up_sampling2d_3 (UpSampling2	(None, 32, 32, 256)	0
conv2d_8 (Conv2D)	(None, 32, 32, 128)	295040
batch_normalization_7 (Batch	(None, 32, 32, 128)	512
activation_3 (Activation)	(None, 32, 32, 128)	0
up_sampling2d_4 (UpSampling2	(None, 64, 64, 128)	0
conv2d_9 (Conv2D)	(None, 64, 64, 128)	147584
batch_normalization_8 (Batch	(None, 64, 64, 128)	512
activation_4 (Activation)	(None, 64, 64, 128)	0
conv2d_10 (Conv2D)	(None, 64, 64, 3)	3459
activation_5 (Activation)	(None, 64, 64, 3)	0

Total params: 2,043,011
Trainable params: 2,041,475
Non-trainable params: 1,536

Figura 32: Estructura de la red generativa de la GAN

Los resultados de diferentes épocas del proceso de entrenamiento de la red se muestran en la Figura 34 y en la Figura 33 se muestran algunos ejemplos de las imágenes que componían los sets de datos. Los resultados tampoco son realmente asombrosos, pero hay que tener en cuenta que los sets de datos empleados tampoco son muy grandes, únicamente 11682 imágenes, y, aunque se intentó usar otros más amplios, fue imposible a nivel computacional y con los recursos disponibles lograr unos resultados decentes. A pesar de todo en la se puede ve como al final la red ha aprendido como tiene que ser una cara con su forma y sus atributos más característicos.

Figura 33: Muestra del tipo de imágenes que contiene el set de datos empleado para entrenar la GAN

Figura 34: Resultados obtenidos con la GAN

Capítulo 6: Bibliografía

Bibliografía

[1] B. L. y. M. S. B. Amos, «OpenFace: A general-purpose face recognition,» de *IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Placid, NY, USA, Marzo 2016.

[2] D. Karunakaran, «Medium,» 27 Septiembre 2018. [En línea]. Available: https://medium.com/intro-to-artificial-intelligence/one-shot-learning-explained-using-facenet-dff5ad52bd38. [Último acceso: Noviembre 2018].

[3] F. Schroff, D. Kalenichenko y J. Philbin, «FaceNet: A Unified Embedding for Face Recognition and Clustering,» 17 Junio 2015.

[4] H. T. (nyoki-mtl), «GitHub,» Febrero 2018. [En línea]. Available: https://github.com/nyoki-mtl/keras-facenet. [Último acceso: Enero 2019].

[5] Apple´s Computer Vision Machine Learning Team, «An On-device Deep Neural Network for Face Detection,» *Machine Learning Journal,* vol. 1, nº 7, Noviembre 2017.

[6] GitHub Community, «GitHub,» Mayo 2017. [En línea]. Available: https://github.com/davidsandberg/facenet/issues/283. [Último acceso: Enero 2019].

[7] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt y Y.-L. Yang, «HoloGAN: Unsupervised learning of 3D representations from natural images,» arXiv:1904.01326v1, Abril 2019.

[8] T. Karras, S. Laine y T. Aila, «A Style-Based Generator Architecture for Generative Adversarial Networks,» arXiv:1812.04948v3, 2019.

[9] E. Zakharov, A. Shysheya, E. Burkov y V. Lempitsky, «Few-Shot Adversarial Learning of Realistic Neural Talking Head Models,» Moscow, Mayo 2019.

[10] C. D. y. O. Wright, «Painting the digital future of retail and consumer goods companies,» Accenture Strategy, 2017.

[11] E. L. S. S. y. S. P. R. Geissbauer, «Global Digital Operations Study,» PwC Strategy&, 2018.

[12] J.Chamary, «How Face ID Works On iPhone X,» *Forbes,* Septiembre 2017.

[13] Apple, «Apple Developer,» [En línea]. Available: https://developer.apple.com. [Último acceso: 2019].

[14] Artículo, «The Sun,» 6 Junio 2016. [En línea]. Available: https://www.thesun.co.uk/living/1237976/can-you-recognise-the-celebs-in-these-photoshopped-face-morphs/. [Último acceso: Enero 2019].

[15] AI Insider, «Medium,» 17 Febrero 2019. [En línea]. Available: https://medium.com/datadriveninvestor/generating-human-faces-with-keras-3ccd54c17f16. [Último acceso: Abril 2019].

Agradecimientos

En primer lugar, dar las gracias a Álvaro Jesús López López, director del proyecto, por saber guiarlo tan bien y poner la pausa necesaria en cada momento para complementar mi desbocado ímpetu. Gracias también por haberme dado la oportunidad de desarrollar este proyecto en un ambiente como el de la Cátedra de Industria Conectada, que me ha permitido aprender tanto de Industria 4.0, y por haber confiado mi para sacar este proyecto adelante.

Mostrar mi agradecimiento a toda mi familia y amigos que han contribuido a este proyecto prestando sus fotos para llevar a cabo los diferentes análisis de robustez. En especial a mis hermanos Pablo e Isabel y a mis padres Álvaro y Teresa que les tuve más de un día ocupados tomándose fotos.

Un agradecimiento especial a Carlos Moro, Alfredo Baldo y a Álvaro López por haber colaborado en la grabación del video sobre el funcionamiento de la App y por haber sido los principales testigos del crecimiento del proyecto.

Gracias a María Antonia García Huerres por haber prestado su Ipad para que se pudiera grabar el video del funcionamiento de la App en un dispositivo de calidad y a Carlota Carrasco que presto su MacBook para que pudiera comenzar el proyecto antes de comprarme uno.