Universidad Zaragoza
1542

Trabajo Fin de Máster

Estudio de técnicas de aprendizaje automático basado en redes neuronales para reconocimiento biométrico de personas.

Study of machine learning techniques based on neural networks for biometric recognition of persons.

Autor

Victoria Mingote Bueno

Director

Antonio Miguel Artiaga

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2016

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. VICTORIA MINGOTE BUENO,

con nº de DNI 73005791-L en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Máster, (Título del Trabajo) Estudio de técnicas de aprendizaje automático basado en redes neuronales para reconocimiento biométrico de personas.,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 26 de Agosto de 2016

Victoria

Fdo: VICTORIA MINGOTE BUENO

Estudio de técnicas de aprendizaje automático basado en redes neuronales para reconocimiento biométrico de personas: Resumen

En este trabajo se plantea la elaboración de un sistema de reconocimiento biométrico de personas basado en redes neuronales profundas que utiliza como característica biométrica una imagen digital del rostro humano con la que se pueda realizar la tarea de identificación facial de dicha persona.

El problema del reconocimiento facial se puede dividir en cuatro fases principales, la de detección del rostro dentro de las imágenes, el preprocesado de dichas imágenes, la extracción de la información más relevante de cada rostro y el reconocimiento de la identidad haciendo uso de dicha información relevante. Estas etapas se implementan a lo largo de este trabajo para poder crear un sistema completo de reconocimiento facial.

El sistema de reconocimiento facial creado durante este trabajo permite experimentar de manera cómoda, debido a la modularidad que este sistema presenta, con diversas arquitecturas para los procesos de extracción y reconocimiento, lo que sirve para comprobar las prestaciones del sistema obtenidas con cada arquitectura. Así como poder observar la influencia en los resultados de cambiar las bases de datos utilizadas para el entrenamiento de la etapa de extracción de características.

Study of machine learning techniques base don neural networks for biometric recognition of persons: Abstract

This work describes the development of a biometric recognition system of persons based on deep neural networks that uses as biometric feature a digital image of human face with which can perform the task of facial identification of such person.

The problem of face recognition can be divided into four main phases, the detection of the face within the images, the pre-processing of the images, the extraction the most relevant information of each face and the identity recognition using such relevant information. These stages are implemented along this work to create a complete facial recognition system.

The facial recognition system created during this work is highly modular what facilitates the experimention phase, which include the test of several architectures for feature extraction and recognition, and also test different databases for training and testing.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Motivación del proyecto

La biometría[1] es la ciencia y la tecnología dedicada a medir y analizar datos biológicos. En el ámbito de la tecnología de la información, la biometría hace referencia a las tecnologías que miden y analizan las características del cuerpo humano. Este tipo de tecnologías se usan en diversos sistemas con el objetivo principal de identificar y permitir reconocer a una persona en una aplicación determinada, ya sea para acceso a recursos, control de asistencia, etc.

Dentro de la biometría se distinguen dos grupos de registros biométricos[2] en función de las características en las que se centran como se puede ver en la figura 1.1, los basados en características fisiológicas como el ADN, las huellas dactilares, la retina, el iris de los ojos, los patrones faciales y las medidas de las manos; y los basados en características del comportamiento como la voz, la firma, la forma de caminar y la dinámica del teclado; ambos grupos de características son intransferibles de las personas por lo cual se pueden utilizar para realizar la autenticación de identidades. La autenticación mediante verificación biométrica está convirtiéndose en algo cada vez más habitual en los sistemas de seguridad, tanto privados como públicos, debido a que permite llevar a cabo un

[1] url: http://searchdatacenter.techtarget.com/es/definicion/Biometria

[2] url:http://www.biometricos.cl/equipos_biometria/que_es_la_biometria_por_huella_digital.php

control y restringir el acceso a determinados recursos, proporcionando seguridad y confiabilidad a los sistemas que cuentan con este tipo de tecnología. También se puede utilizar la biometría para la personalización de los sistemas que se usan para aplicaciones de domótica, lo cual permite mejorar la experiencia y las prestaciones de dichas aplicaciones.

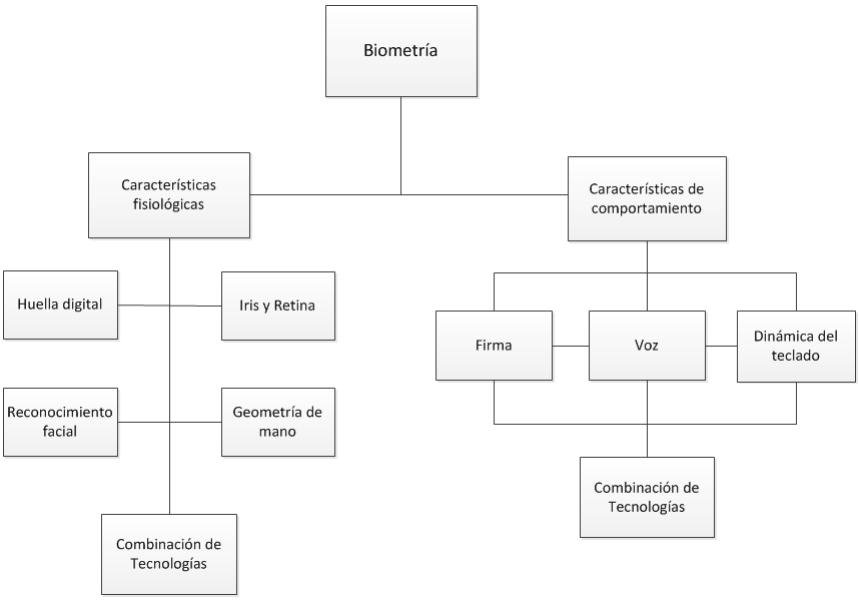


Figura 1.1: Clasificación de los grupos de registros biomédicos.

En los últimos años, dentro de estos grupos de registros biomédicos, el reconocimiento facial[3] se ha convertido en un área de investigación activa que engloba distintas disciplinas, como el procesado de imágenes, reconocimiento de patrones y redes neuronales. También se podría considerar en el campo del reconocimiento de objetos, donde se puede ver la cara como un objeto tridimensional que está sujeto a variaciones de iluminación, posición, etc., y se debe identificar basándose en su proyección en 2D.

El objetivo principal de un sistema de reconocimiento facial es, dada una imagen de la cara de una persona desconocida, o imagen de test, hallar otra imagen de la misma persona en un conjunto de imágenes conocidas, o imágenes de entrenamiento. Este tipo de sistema puede operar en dos modos:

[3]url: http://es.m.wikipedia.org/wiki/Sistema_de_reconocimiento_facial

• **Verificación o autenticación facial:** se encarga de comparar una imagen de la cara cuya identidad se encuentra verificada con otra imagen de la cara de la que queremos conocer la identidad. El sistema se encargará de confirmar o rechazar la identidad de la cara, lo cuál se puede ver como un problema de decisión.

• **Identificación o reconocimiento facial:** se encarga de comparar una imagen de una cara desconocida con todas las imágenes de caras conocidas que se encuentran en la base de datos para determinar su identidad, en este caso se puede ver como un problema de clasificación.

Los sistemas de reconocimiento automático de caras humanas ofrecen una serie de ventajas frente a otros sistemas, como los basados en ADN o huellas dactilares. Entre sus principales ventajas se encuentra la rapidez que pueden llegar a alcanzar, así como la posibilidad de realizarse el proceso sin la colaboración del sujeto e incluso a una distancia relativamente alta.

1.2 Objetivos y enfoque

El objetivo general del presente trabajo es el estudio, realización y evaluación de un sistema de biometría facial basado en imágenes. Para llevarlo a cabo, se han utilizado técnicas de aprendizaje automático de datos basadas en redes neuronales profundas [4] con el objetivo de la identificación y el reconocimiento de personas. La cara es un rasgo muy identificativo que no requiere ni de un despliegue tecnológico complejo ni de una alta participación del individuo para ser adquirida su imagen, lo que la convierte en un rasgo perfecto para realizar un sistema biométrico efectivo y poco intrusivo.

Como objetivos específicos se busca comparar diversas arquitecturas de verificación facial creadas con redes neuronales aplicadas sobre bases de datos públicas, analizando sus principales características y tasas de reconocimiento frente a varios factores. También se realizará una optimización de los parámetros de la

red para conseguir la mejor configuración posible del sistema.

Además de los objetivos mencionados anteriormente, se planteó el aprendizaje en el manejo de tarjetas gráficas (GPUs) con capacidad de paralelizar cálculos, de manera que se conseguirá aumentar la eficiencia computacional para poder desarrollar los objetivos que se habían propuesto.

1.3 Metodología y plan de trabajo

Para el correcto desarrollo y consecución de los objetivos marcados en el presente Trabajo Fin de Máster, se ha seguido un plan de trabajo organizado de la siguiente forma:

- **Estudio del estado del arte.** Como punto de partida del trabajo se empezó con una etapa de formación en la que se obtuvieron los conocimientos necesarios para el desarrollo del mismo. En concreto, se estudio el estado del arte en redes neuronales artificiales y en sistemas de reconocimiento facial. Además, como en el trabajo se desarrolla un sistema completo, en la etapa de formación también se llevó a cabo un estudio profundo sobre el lenguaje Python[4] y la librería Keras[5] utilizados en el mismo.

- **Preparación de los datos y desarrollo del sistema de reconocimiento.** La etapa intermedia del trabajo se centró en el procesado de los datos de las respectivas bases de datos utilizadas para darles un formato adecuado a lo requerido por la librería Keras y la implementación del sistema de reconocimiento facial haciendo uso de redes neuronales.

[4]url: http://www.python.org/downloads
[5]url: http://www.github.com/fchollet/keras

- **Ajustes de parámetros del sistema.** Con el objetivo de mejorar los resultados obtenidos con los parámetros iniciales del sistema se realizaron una serie de experimentos haciendo uso de tarjetas aceleradoras gráficas para así poder mejorar los resultados obtenidos en la capacidad de reconocimiento.

- **Documentación y escritura de la memoria.** En la etapa final del trabajo, se procedió a documentar todo el trabajo realizado y se distribuyó en forma de memoria para que se reflejará de manera entendible la tarea realizada durante todo el proceso de desarrollo del presente trabajo.

1.4 Organización de la memoria

La presente memoria está estructurada en 5 capítulos:

- **Capítulo 1: Introducción.** En este capítulo se exponen los motivos para la realización del proyecto así como los objetivos perseguidos para la consecución del mismo. Para lograr satisfacer las motivaciones y objetivos, se expone la metodología y el plan de trabajo que se va a seguir.

- **Capítulo 2: Revisión bibliográfica.** El segundo capítulo se centra en abordar el estado del arte actual de las redes neuronales como método de aprendizaje automático. Así como también se desarrollará la situación en este momento de los sistemas de reconocimiento facial, mostrando las diversas técnicas que se utilizan hoy en día.

- **Capítulo 3: Sistemas implementados.** En este capítulo se exponen los sistemas que se han implementado durante la duración del trabajo. En cada sistema se describen las etapas de las que consta para su funcionamiento y las diferentes estructuras que se han aplicado para su creación.

- **Capítulo 4: Experimentación y Resultados.** En el cuarto capítulo se presentan los resultados obtenidos de evaluar los sistemas implementados en el capítulo anterior con diversas bases de datos públicas de imágenes faciales.

- **Capítulo 5: Conclusiones y Líneas futuras.** El último capítulo incluye las conclusiones globales del trabajo realizado y se proponen líneas futuras de investigación que hagan posible la mejora del trabajo aquí expuesto.

Para completar la memoria, se añaden una serie de anexos con información adicional como el código desarrollado para la realización del presente trabajo y una tabla comparativa de los tiempos de computación obtenidos con los diversos dispositivos con los que se contó para la realización de este trabajo.

Capítulo 2

Revisión bibliográfica

En este capítulo se realiza un estudio de los fundamentos y los tipos de redes neuronales artificiales existentes, así como también se presentarán los antecedentes y el estado del arte sobre los sistemas de reconocimiento facial basados en diversas técnicas, aunque se explicarán principalmente los basados en el uso de redes neuronales.

2.1 Redes neuronales artificiales

2.1.1 Definición de red neuronal artificial

Las redes neuronales artificiales [5] o sistemas conexionistas son sistemas de procesamiento de la información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas, como se puede observar en la figura 2.1 existen ciertas analogías entre las redes neuronales biológicas y las redes neuronales artificiales, entre las que se encuentran las siguientes [1]:

- Las entradas X_i representan las señales que vienen de otras neuronas y que son capturadas por las dendritas.
- Los pesos W_i son la intensidad de la sinapsis que conecta dos neuronas.
- θ hace referencia a la función umbral que la neurona tiene que sobrepasar para activarse.

- Cada señal de entrada pasa a través de una ganancia o peso.
- Los pesos pueden ser positivos (excitatorios) o negativos (inhibitorios).
- El nodo sumatorio acumula todas las señales de entradas multiplicadas por los pesos y las transfiere a la salida a través de una función umbral.

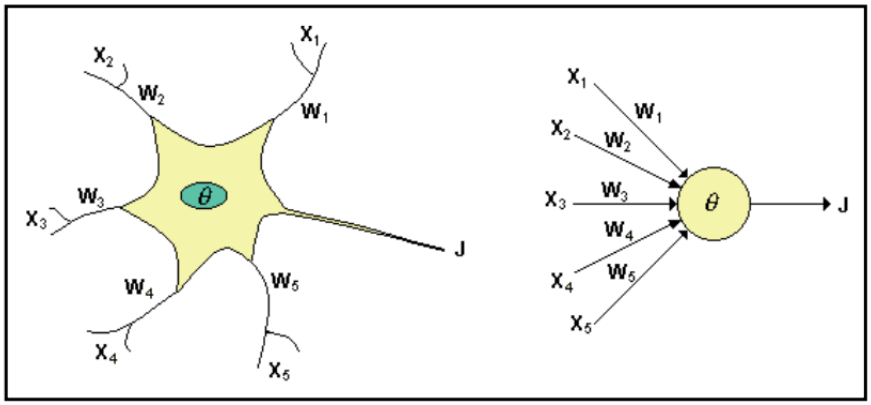


Figura 2.1: Comparativa de una neurona biológica con una neurona artificial [1].

2.1.2 Arquitectura

La estructura de una red neuronal artificial se constituye por un conjunto de elementos simples de procesamiento llamados nodos o neuronas conectadas entre sí por conexiones que tienen un valor numérico modificable llamado peso, como se puede observar en la figura 2.2 los elementos principales que se pueden distinguir en cualquier tipo de red son los siguientes:

- **Neuronas de entrada (capa de entrada)**: reciben directamente la información proveniente de las fuentes externas a la red.
- **Neuronas de salida (capa de salida)**: transfieren la información de la red hacia el exterior una vez que ha finalizado el tratamiento de la información.
- **Neuronas intermedias (capas ocultas)**: reciben estímulos y emiten salidas dentro del sistema, es decir, no tienen ningún contacto con el exterior; son las encargadas de llevar a cabo el procesamiento de la información.

- **Conjunto de conexiones o pesos sinápticos entre las neuronas**: cada conexión se define por un peso w_{ij}, el cual indica el efecto de la señal de la neurona i en la neurona j.

- **Función de activación**: calcula el estado de actividad de una neurona, convirtiendo la entrada global en un valor de activación, cuyo rango normalmente va de (0 a 1) o de (-1 a 1). Las funciones de activación de uso más común en redes neuronales son: escalón, lineal, sigmoidal, tangente hiperbólica o ReLU (*Rectified Linear Unit*) [6].

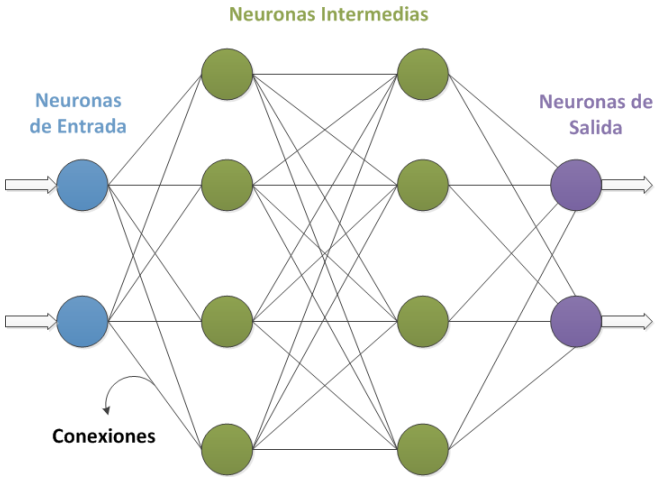


Figura 2.2: Elementos de una red neuronal artificial.

2.1.3 Características

Las características principales de las redes neuronales artificiales [7] se van a detallar a continuación:

1. *Aprendizaje adaptativo*: aprenden a realizar algunas tareas mediante un entrenamiento con ejemplos ilustrativos, sin necesidad de una elaboración de modelos a priori ni de especificar funciones de distribución de probabilidad. Las redes neuronales son sistemas dinámicos auto-adaptativos, son adaptables debido a la capacidad de autoajuste de las neuronas que componen el sistema y son dinámicos puesto que son capaces de estar constantemente cambiando para adaptarse a las nuevas condiciones.

2. *Auto-organización*: utilizan su capacidad de aprendizaje adaptativo para auto-organizar la información que reciben durante el aprendizaje. El aprendizaje es la modificación de cada neurona, mientras que la auto-organización se basa en la modificación de la red neuronal completa para conseguir un objetivo concreto. Esta auto-organización provoca la generalización, que es la facultad de las redes neuronales de responder apropiadamente cuando se les presentan datos o situaciones a la que no habían sido expuestas antes.

3. *Tolerancia a fallos*: en las redes neuronales si se produce un fallo en un número no muy grande de neuronas el comportamiento del sistema se ve influenciado pero no sufre una caída repentina como si ocurre en los sistemas de computación tradicionales, que pierden su funcionalidad cuando sufren un pequeño error de memoria. El motivo por el que las redes neuronales cuentan con esta característica es que tienen su información distribuida en las conexiones entre neuronas, existiendo cierta redundancia en este tipo de almacenamiento.

2.1.4 Tipos de redes

Dentro de las redes neuronales podemos encontrar numerosos modelos distintos a la hora de resolver un determinado problema, a continuación se describen los principales.

2.1.4.1 Perceptrón simple

El perceptron es la red neuronal más antigua. Como se puede ver en la figura 2.3, este tipo de red se basa en sumar las señales de entrada y multiplicar por los valores de pesos escogidos aleatoriamente; este valor se compara con un patrón para determinar si la neurona es activada o no. Esta red usa un algoritmo de aprendizaje supervisado, es decir, necesita conocer los valores esperados para cada entrada que se presenta a la red.

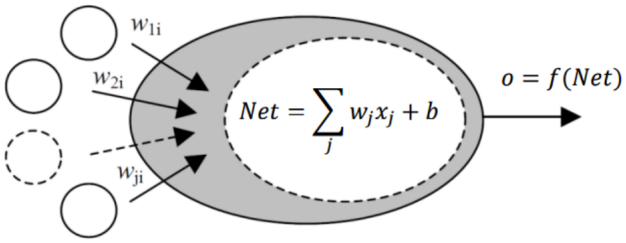


Figura 2.3: Arquitectura de red de tipo Perceptron[1].

Este tipo de red se utiliza en problemas de clasificación y se obtienen resultados perfectos si los patrones son linealmente separables, si los patrones no son linealmente separables el algoritmo del perceptrón no puede converger hacia un error nulo, por lo que los elementos que no estén claramente separados de otros no se pueden clasificar, siendo esta la principal limitación de este tipo de red.

2.1.4.2 Redes multicapa

Al igual que el perceptrón este tipo de redes son muy antiguas y a su vez han sido las más utilizadas; como se observa en la figura 2.4 el perceptrón multicapa se compone de una capa de entrada, al menos una capa oculta y una capa de salida. La principal novedad de este tipo de redes respecto al perceptrón simple es su algoritmo de aprendizaje durante su entrenamiento, el algoritmo de retropropagación (*backpropagation* o propagación hacia atrás). Dicho algoritmo es un método de aprendizaje supervisado de gradiente descendente en el que inicialmente se aplica un patrón de entrada, el cual se propaga por las distintas capas que componen la red hasta producir la salida de la misma. Esta salida se compara con la salida deseada y se calcula el error cometido por cada neurona de salida. Estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de las capas intermedias. Basándose en la influencia en el error global de cada peso, se ajustan los pesos de cada neurona.

[1]Apuntes asignatura *Aprendizaje automático en datos multimedia*

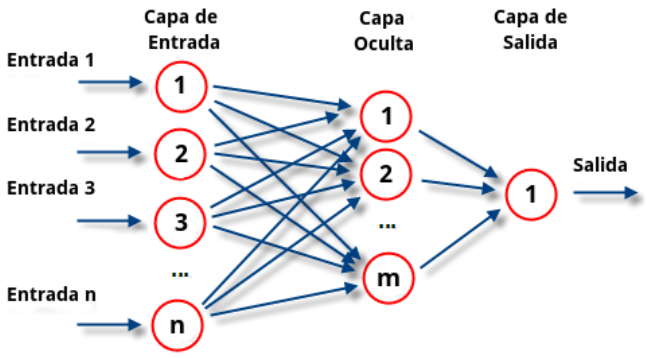


Figura 2.4: Arquitectura de red de tipo Perceptron Multicapa[2].

2.1.4.3 Redes neuronales profundas

Las redes neuronales profundas[3] [8] son redes multicapa complejas, como se puede observar en la figura 2.5, puesto que se basan en el hecho de que varias capas ocultas tienen la misma capacidad que una sola capa oculta con muchas neuronas en ella. La principal ventaja de este tipo de redes es que un mayor número de capas pueden generalizar mejor y aprender más conceptos abstractos.

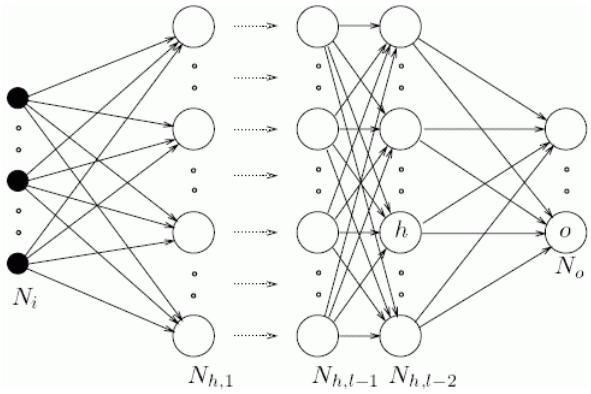


Figura 2.5: Arquitectura de red de tipo Red Densa.[4]

[2]https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa
[3]url: https://www.deeplearning4j.org/neuralnet-overview.html
[4]http://www.varpa.org/ mgpenedo/cursos/scx/Tema4/nodo4-1.html

En las redes neuronales profundas, cada capa se encarga de entrenar un conjunto distinto de características basadas en la salida de la capa previa. Cuanto más se avanza en la red, características más complejas son capaces de reconocer los nodos, ya que se agregan y recombinan características de las capas previas.

Esto es lo que se conoce como herencia de características, y es una herencia que aumenta la complejidad y la abstracción. Esto hace que las redes neuronales profundas sean capaces de manejar conjuntos de datos muy grandes y de alta dimensión con muchos millones de parámetros que pasan a través de funciones no lineales.

2.1.4.4 Redes convolucionales

Las redes convolucionales[5] son redes neuronales profundas que utilizan convoluciones en lugar de las matrices generales de multiplicación en al menos una de sus capas, dichas convoluciones son operaciones de productos y sumas entre la imagen de entrada y un filtro que genera un mapa de rasgos de carácter local, ya que la aplicación de las máscaras permite la detección de determinados efectos presentes en las imágenes, como pueden ser detección de bordes y cambios de color. Este tipo de redes pueden emplearse para clasificar imágenes (nombrar lo que ven), agruparlas por similitud (búsqueda de fotografías), y llevar a cabo el reconocimiento de objetos dentro de escenas. Pueden identificar caras, individuos, señales de tráfico y muchos otros aspectos de la información visual. En la figura 2.6 se puede ver un ejemplo de la arquitectura de una red convolucional.

[5]url:https://www.deeplearning4j.org/convolutionalnets

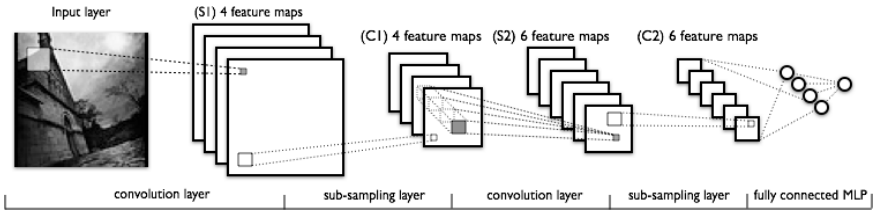


Figura 2.6: Arquitectura de red de tipo Red convolucional.[5]

La eficacia de las redes convolucionales en el reconocimiento de imágenes es una de las principales razones por las que el aprendizaje profundo está creciendo en su uso hoy en día. Se están potenciado importantes avances en visión por computación, los cuales tienen aplicaciones obvias en auto-conducción de coches, robótica, drones y tratamientos para los discapacitados visuales.

2.1.4.5 Autoencoder

Un autoencoder es una red neuronal que aprende a producir a la salida exactamente la misma información que recibe a la entrada haciendo uso de un algoritmo de aprendizaje no supervisado. Por lo cual, las capas de entrada y salida siempre deben tener el mismo número de neuronas en este tipo de redes para que sean capaces de reproducir la entrada a la red.

La clave de este tipo de red reside en la capa oculta. Dado que se exige que la salida de esta red tenga como resultado lo mismo que recibe a la entrada, y la información tiene que pasar por la capa oculta, también conocida como *bottleneck* o cuello de botella, que habitualmente tiene menos neuronas que las capas de entrada y salida, la red se verá obligada a encontrar una representación intermedia de la información en su capa oculta usando menos dimensiones. Por lo tanto, la capa oculta contendrá una versión comprimida que se puede volver a descomprimir para recuperar la versión original a la salida.

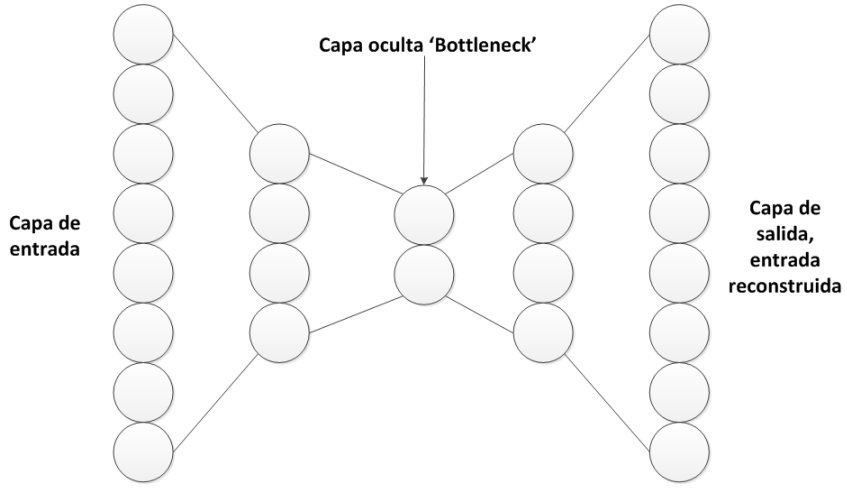


Figura 2.7: Arquitectura de red de tipo Autoencoder.

De hecho se puede dividir la red en dos como se ve en la figura 2.8, una primera red que comprendería desde la capa entrada a la capa oculta (*bottleneck*), esto sería un compresor o codificador, y la segunda red que utilizaría la capa oculta (*bottleneck*) como capa de entrada, esta parte sería un descompresor o decodificador.

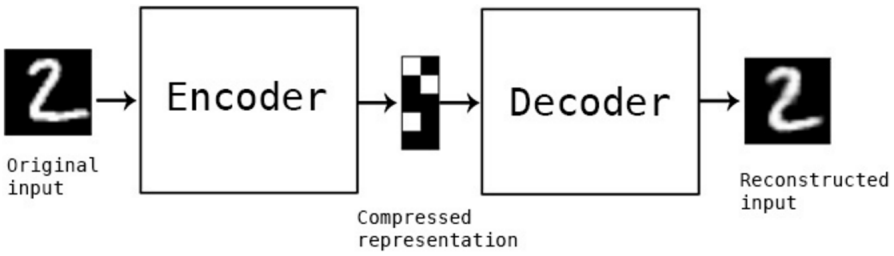


Figura 2.8: Ejemplo de autoencoder visto como codificador y decodificador con la base de datos MNIST[7].

[7]https://blog.keras.io/building-autoencoders-in-keras.html

2.2 Sistemas de reconocimiento facial

2.2.1 Reconocimiento facial

El reconocimiento facial se trata de una técnica que ha tenido un uso muy extendido durante los últimos años y que se centra en la identificación de rostros utilizando únicamente los rasgos faciales, sin utilizar otros elementos, como pudieran ser la voz, las huellas dactilares, la firma, el ADN, etc.

El rostro humano proporciona gran cantidad de información discriminativa sobre un sujeto permitiendo a la gente la capacidad de discernir e identificar a simple vista distintos sujetos. El rostro tiene un conjunto de rasgos que lo dotan de un alto poder discriminativo. Estos rasgos que componen el rostro se encuentran localizados en posiciones similares a lo largo de la población por lo que un sistema de reconocimiento facial puede aprovecharse de esta característica.

Figura 2.9: Ejemplo de los rasgos biométricos de un rostro [2].

Además de estos rasgos, se debe destacar que la forma de la cara también es una característica discriminativa, las tareas de localización o extracción de distancias se pueden beneficiar de la simetría del rostro. Sin embargo, existen ciertas características que pueden introducir mayor variabilidad en el mismo individuo como puede ser el pelo, que puede contribuir a la oclusión de los rasgos y a cambiar el aspecto de una persona, u otros elementos artificiales, que contribuyen a la pérdida de fiabilidad en el sistema, como las gorras, bufandas y gafas.

Existen numerosas técnicas implementadas para llevar a cabo un sistema de reconocimiento facial, como se comentará en el siguiente apartado, pero independientemente de qué técnica se utilice, siempre se necesita tener dos conjuntos de datos:

- El primero se utiliza siempre para la etapa de aprendizaje o entrenamiento. Se debe tratar de que los datos que integran este conjunto sean lo más diferentes posible entre ellos, y que además, representen al problema, para poder llegar a conseguir un grado de generalización alto.
- El segundo conjunto se emplea en la etapa de reconocimiento y se conoce como conjunto de prueba o de test.

2.2.2 Etapas de un sistema de reconocimiento facial

En la figura 2.10 se muestra la secuencia de etapas principales para el reconocimiento facial, la cual habitualmente suele ser la misma en todos los sistemas:

Figura 2.10: Etapas generales de un sistema de reconocimiento facial.

- **Adquisición**: de las imágenes de entrada que se realiza mediante cualquier dispositivo capaz de tomar imágenes. Con una imagen ya se puede comenzar el proceso, el tiempo de adquisición es muy bajo y no requiere de supervisión siempre y cuando las imágenes cumplan con un mínimo de calidad.

- **Detección de caras (posición)**: se trata de una etapa muy crítica en los sistemas de reconocimiento facial puesto que el resto de etapas se verán afectadas si no se ha realizado correctamente la detección y localización. Lo primero que se debe determinar en esta etapa es si en la imagen hay caras y si hay alguna localizarla en la imagen. Esta tarea se puede volver compleja si hay factores que puedan enmascarar las características faciales como pueden ser el vello facial, maquillaje, gafas, etc. Por otro lado, hay otros dos factores que también dificultan este proceso: la iluminación que tenga la escena y la calidad de las imágenes.

  Existen diversos métodos de detección de caras pero la mayoría se basan en autocaras (*eigenfaces*) [9], redes neuronales, análisis de rasgos o análisis de bajo nivel [10], como puede ser el color de la cara.

- **Preprocesamiento**: a partir de la información obtenida en la detección se lleva a cabo la etapa de procesado en la cual se realizan una serie de transformaciones sobre la imagen para dejarla preparada para la correcta extracción de características. Dentro de esta etapa se suelen encontrar cuatro fases para normalizar y alinear la imagen:

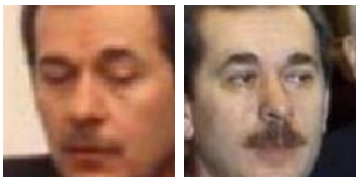
  - **Rotación.** Determinar el ángulo de giro de una cara en una imagen y compensarlo. Al tener caras sin giro, el proceso de reconocimiento dará mejores resultados.
  - **Escalado.** Acción que se lleva a cabo para conseguir que todas las imágenes tengan el mismo tamaño. Esto se convierte en necesario puesto que muchas técnicas de reconocimiento requieren que todos los datos de entrada tengan el mismo tamaño.
  - **Recorte.** Una vez que se ha aplicado la rotación y el escalado, en caso de ser necesarios, se procede al recorte de la misma para obtener solo la región de interés de las imágenes.

  - **Normalización.** Las imágenes pueden presentar variabilidad en la luminosidad y en el contraste lo que puede producir que imágenes de la misma persona sean muy diferentes por lo que hay que normalizarlas para tener un estándar en todas las imágenes que se pasen al extractor de características.

- **Extracción de características**: se emplea para obtener la información que resulta más relevante de un rostro, se elimina la información que resulte irrelevante para el reconocimiento, de esta etapa depende en gran medida el buen desempeño del sistema de reconocimiento facial. Las técnicas más clásicas son las siguientes: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) y Redes Neuronales Artificiales (RNA) o Artificial Neural Networks (ANN) [4].

- **Reconocimiento**: está última etapa se basa en alimentar al sistema con imágenes de rostros diferentes a las utilizadas durante el entrenamiento para obtener una medida como resultado con la cual se puede tomar una decisión, bajo la premisa de que en los sistemas de reconocimiento facial caras de un mismo individuo tienen características similares. Existen diversos métodos para obtener dicha medida, entre los más utilizados se pueden encontrar dos categorías, los clasificadores que requieren entrenamiento previo: K-Nearest Neighbours (KNNs) , Support Vector Machines (SVMs), Gaussian Mixture Models (GMMs) y Redes Neuronales Artificiales (RNA) [4], y las medidas de similitud o distancia que no requieren de parámetros de entrenamiento: euclídea, coseno y chi-square.

El problema que se quiere resolver con la creación de un sistema de reconocimiento facial y la dificultad de que dicho problema conlleva en algunos casos se puede observar en las figuras 2.11, ejemplo en el que se trata de imágenes de la misma persona, y 2.12, ejemplo en el que se trata de personas diferentes.



(a) Foto 1 (b) Foto 2

Figura 2.11: Ejemplo 1 de una pareja de imágenes para decidir si se trata de la misma persona o no, en este caso se trata de la misma persona.

(a) Foto 1 (b) Foto 2

Figura 2.12: Ejemplo 2 de una pareja de imágenes para decidir si se trata de la misma persona o no, en este caso se trata de distinta persona.

2.2.3 Técnicas de reconocimiento facial

A continuación, se explican algunos de los métodos utilizados en los sistemas de reconocimiento facial en los últimos años tanto para la extracción de características como para la clasificación final.

2.2.3.1 Basados en PCA

El método basado en PCA, también conocido como Eigenfaces [9], se basa en la transformación de un número de variables posiblemente correladas en un pequeño número de variables incorreladas llamadas componentes principales, es decir, se trata de un algoritmo de reducción dimensional que permite encontrar los vectores que mejor representan la distribución de un grupo de imágenes.

En el caso de las imágenes de rostros, se centra en el análisis de ciertas características faciales para reconocer a un individuo dentro de una base de datos previamente creada, al aplicarlo se reduce enormemente la cantidad de información necesaria a manipular en la fase de reconocimiento. Estas características funcionan como proyección de las imágenes en un conjunto de vectores básicos ortogonales (*eigenfaces*) de un subespacio lineal llamado Espacio Facial, un ejemplo de estos vectores de características se pueden observar en la figura 2.13. El reconocimiento del rostro se realiza proyectando una nueva imagen de un rostro en el espacio facial y comparando su posición en el espacio facial con alguna cara conocida por el sistema.

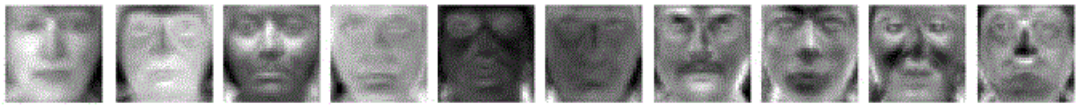


Figura 2.13: Ejemplo de vectores de características del PCA

Existen varios métodos de reconocimiento basados en la caracterización de ciertos rasgos que no cambian en una cara: la sección alta de las cavidades oculares, la zona ósea alrededor de las mejillas y los laterales de la boca. El proceso cuando se utiliza este tipo de técnicas se compone de las fases normales de una aplicación de reconocimiento facial: realizar una captura de imagen, extraer las características matemáticas de la misma y realizar una comparación con patrones de referencia.

2.2.3.2 Basados en LDA

Este método se creó como una variación del PCA, se propusó como una mejora de este, a esta técnica también se la conoce como Fisherfaces [11] y es de las más utilizadas a la hora de implementar un sistema de reconocimiento e identificación mediante el uso de los patrones faciales localizados en el rostro. Se basa en encontrar combinaciones lineales para poder reducir la dimensión del problema, de tal manera que se mantenga la habilidad de separar dos o más clases de objetos, a este tipo de métodos se les llama discriminativos.

El LDA permite utilizar la información entre miembros de la misma clase y entre clases para desarrollar un conjunto de vectores características, de los cuales se puede ver un ejemplo en la figura 2.14, donde las variaciones entre las diferentes caras se marcan más mientras que los cambios debido a la iluminación, expresión facial y orientación de la cara no. Es decir, se maximiza la variancia de las muestras entre clases, y se minimiza entre muestras de la misma clase.

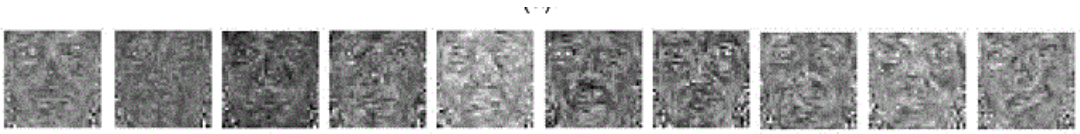


Figura 2.14: Ejemplo de vectores de características del LDA

Diversos experimentos que se han llevado acabo en los últimos años muestran que tanto la técnica del PCA como la LDA obtienen un buen rendimiento si las imágenes de prueba son similares a las imágenes de entrenamiento. Sin embargo, el método LDA obtiene mejores resultados en caso de que haya variaciones en las condiciones de iluminación y gesto que la técnica del PCA.

2.2.3.3 Basados en SVM

Las SVMs se han convertido en una herramienta de uso generalizado en el campo de reconocimiento de patrones, se utilizan como un clasificador automático que requiere de supervisión. Dado un conjunto de ejemplos de entrenamiento se puede etiquetar las clases y entrenar una SVM para crear un modelo que prediga la clase de una nueva muestra en función de una selección de ejemplos del entrenamiento llamado vectores soporte.

El objetivo de las SVMs se centra en encontrar el plano óptimo que tenga la máxima distancia con los puntos soporte. De esta forma, se separan las muestras de manera que los datos de ambas categorías queden cada uno a un lado del plano. Los coeficientes que seleccionan los vectores soporte y los parámetros del plano se

optimizan simultáneamente para conseguir la mayor distancia posible entre las dos categorías.

Los algoritmos SVM pertenecen a la familia de los clasificadores lineales, pero usando una función de kernel no lineal pueden generalizar cualquier frontera de clasificación no lineal genérica.

2.2.3.4 Basados en GMMs

Una GMM es una función de densidad de probabilidad paramétrica representada como una suma ponderada de las densidades de las componentes Gaussianas. Las GMMs comúnmente se utilizan como un modelo paramétrico de la distribución de probabilidad continua de las medidas o características en un sistema biométrico. Los parámetros de la GMM son estimados en la fase de entrenamiento usando el algoritmo iterativo EM o la estimación MAP de un modelo a priori bien entrenado.

En el caso del reconocimiento facial, las GMMs se entienden como métodos de modelado estadístico en los cuales un modelo se define como una mezcla de cierto número de funciones Gaussianas con las cuales se evalúa la proximidad a un modelo de las imágenes a clasificar.

2.2.3.5 Basados en redes neuronales artificiales

En los últimos años, los términos como *big data* (grandes cantidades de datos) y *deep learning* (aprendizaje profundo) [12] se han convertido en principales en el mundo de la visión por computador debido a la gran cantidad de imágenes que se puede encontrar en los buscadores y en las redes sociales.

Esta gran cantidad de datos y el aumento en los recursos de computación han hecho posible el uso de métodos estadísticos más potentes. Estos modelos

han mejorado drásticamente la robustez a diversas variaciones que son el núcleo de muchas aplicaciones de visión por computador. Mientras que los métodos de aprendizaje convencionales, como los presentados anteriormente, tienen una capacidad limitada para aprovechar grandes volúmenes de datos, sin embargo las redes neuronales profundas han mostrado una mejor escalabilidad de sus propiedades.

En el presente trabajo se decidió utilizar redes profundas ya que en los sistemas del estado del arte actual se ha demostrado que se consiguen resultados impresionantes, se pueden aplicar a grandes cantidades de datos de entrenamiento y se encuentran disponibles recursos de computación escalables como el uso de un gran número de CPUs y/o de GPUs.

2.2.3.6 Basados en medidas de similitud

La similitud se corresponde con una medida de cuanto se parecen o no dos vectores de características. Si la distancia entre los dos es pequeña significa que existe un alto grado de similitud; si la distancia es grande existe poca similitud entre ambos. Dentro del conjunto de medidas de similitud se encuentran las dos que aquí se presentan:

- **Distancia euclídea**: La distancia euclídea es la medida que se usa comúnmente. Dicha distancia entre dos puntos es la longitud del camino que los conecta, esta distancia viene dada por el Teorema de Pitagoras. A continuación, se puede observar la ecuación por la que se rige esta medida:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum (x_i - y_i)^2} \tag{2.1}$$

- **Similitud coseno**: La similitud coseno se trata de una métrica que encuentra el producto normalizado de dos vectores de características. Para determinar esta medida de forma eficiente se busca encontrar el coseno del ángulo entre los dos vectores. Lo que indicará que dos vectores muy parecidos tendrán una similitud coseno cercana a 1, mientras que dos vectores completamente distintos tendrán una similitud cercana a 0. Está medida se esta utilizando por su gran eficiencia al evaluar. A continuación, se puede observar la ecuación por la que se rige esta medida de similitud:

$$d(\mathbf{x}, \mathbf{y}) = sim(\mathbf{x}, \mathbf{y}) = \cos(\theta) = \frac{\mathbf{x}^T \bullet \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \tag{2.2}$$

Capítulo 3

Sistemas desarrollados

3.1 Introducción

En este capítulo, se describen con detalle los dos sistemas desarrollados e implementados a lo largo del presente trabajo, con ambos se consigue tener un sistema de reconocimiento facial completo. Para ello se analizarán cada una de las etapas principales de las que se compone cualquier sistema de reconocimiento facial y que se han implementado para la completa creación de los sistemas que se detallan a continuación.

3.2 Sistema basado en una red neuronal profunda

3.2.1 Descripción

El objetivo de este sistema se centró en encapsular todas las etapas de un sistema de reconocimiento facial desde la detección inicial hasta la decisión final. En la figura 3.1 se puede ver dicho sistema propuesto como parte principal de este trabajo, para lograr diferenciar entre diversos pares de fotografías si se trata de la misma persona o de distinta persona en ambas fotos, esto se realizó por medio de la obtención de las características particulares de cada una de las imágenes faciales

que comprenden la base de datos, tanto las que forman la parte de entrenamiento como las de la parte de evaluación, dicho proceso se llevó a cabo haciendo uso de un sistema basado en redes neuronales profundas. Una vez obtenidos los vectores de características de cada una de las imágenes se procede a la etapa de reconocimiento en la cual se utilizaron clasificadores para obtener los resultados finales. En este caso a diferencia de cuando se utilizan otros extractores de características, al tratarse de redes, se requiere para realizar el entrenamiento de la red el uso de datos diferentes a los que se pretenden clasificar posteriormente.

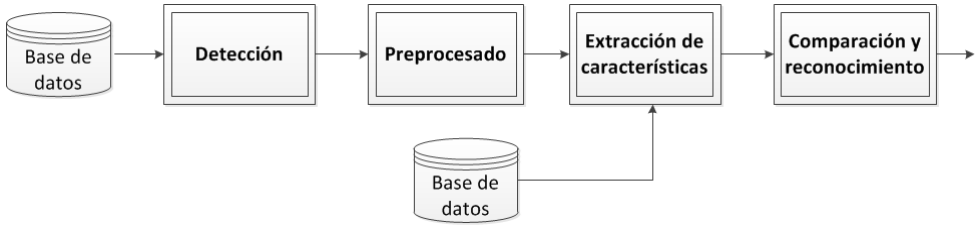


Figura 3.1: Fases del sistema basado en una red neuronal convolucional de una sola rama.

En este sistema se podrían intercambiar fácilmente diferentes técnicas en cada una de las etapas para comparar resultados, es decir, se trata de un sistema completamente modular. Esta modularidad se convirtió en un aspecto muy importante para la implementación del segundo sistema, ya que se pudo reutilizar una parte de este primer sistema para la creación del segundo como se explicará más adelante.

3.2.2 Adquisición

Para este trabajo esta primera fase del sistema de reconocimiento facial que se trataba de la adquisición de los datos, la cual, como se explicó en el estado del arte al comentar las diversas etapas de un sistema de este tipo, se podría realizar mediante la captura de las imágenes con cualquier dispositivo que fuese capaz de ello pero en este trabajo no se llevó a cabo dicha captura. En este caso se hizo uso de diversas bases de datos públicas para realizar tanto el entrenamiento como la evaluación del sistema. Dichas bases de datos son las siguientes:

- Labeled Faces in the Wild (LFW) [13]: se trata de una base de datos pública de fotografías faciales creada para estudiar el problema del reconocimiento facial sin restricciones. Esta base de datos se conforma de un conjunto de 13.323 imágenes de 5.749 personajes famosos tanto hombres como mujeres.

- CASIA-WebFace (CASIA)[1]: se trata de una base de datos pública de fotografías faciales que se compone de 493.456 imágenes faciales de 10.575 personas.

- Youtube Faces (YTF) [14]: se trata de una base de datos pública modelada de manera similar a la LFW, formada con vídeos de algunos de sus individuos, se compone por un total de 3.425 vídeos de 1.595 personas. Estos vídeos se encuentran divididos en 5.000 parejas de vídeos, las cuales se reparten en 10 conjuntos de 500 parejas. Cada uno de los vídeos se encuentra descompuesto en un número de imágenes variable en cada uno.

Las bases de datos mencionadas se componían en ambos casos de imágenes en las que los rostros de los sujetos se encontraban centrados en la imagen pero sin haberles aplicado ningún tipo de recorte para hacer uso únicamente de la zona del rostro, ni haberles aplicado ninguna transformación a escala de grises, transformaciones que posteriormente se explicará como se aplicaron en la realización de este trabajo. Un ejemplo de las imágenes que componen estas bases de datos se pueden observar en la figura 3.2. Sin embargo, a la vez que se adquirieron estas versiones de las bases de datos también se adquirió una version de las bases de datos LFW y CASIA ya recortada y en escala de grises [15] para poder realizar, como se explicará más adelante, experimentos con dichas bases de datos en las cuales se había realizado un preprocesado mucho más complejo que el que se expondrá a continuación.

[1] url: http://github.com/happynear/FaceVerification

(a) Imagen LFW (b) Imagen CASIA (c) Imagen YTF

Figura 3.2: Imágenes originales de las bases de datos.

3.2.3 Detección

La fase de detección del sistema implementado durante este trabajo se basó principalmente en la detección de la cara, en este trabajo no se tuvo en cuenta la detección de los ojos. La detección de la cara se trata de un aspecto crucial en el sistema puesto que un error al detectar la misma resultará en un error en el resto de etapas. Para esta función, se implementó la detección de la cara mediante la librería opencv[2] , que implementa el algoritmo Viola-Jones [3]. La librería incorpora un paquete para poder usarla con el lenguaje de programación Python, lenguaje de programación que se ha utilizado para realizar este trabajo, por su facilidad de implementación respecto a otras técnicas.

La detección con esta librería se realiza utilizando un clasificador en cascada basado en características Haar pre-entrenado para la cara, se trata de un efectivo método de obtención de objetos que permite detectar casi cualquier objeto presente en una imagen. Este algoritmo se eligió por su buena relación entre simplicidad, coste computacional y eficacia. En la figura 3.3 se pueden ver las características o filtros de Haar que se usan en este algoritmo.

[2]url: http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html

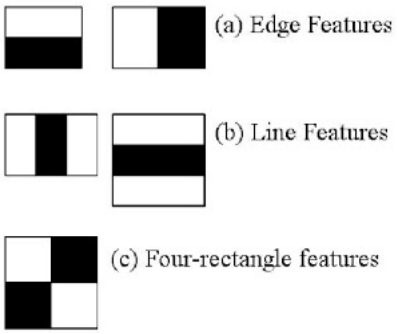


Figura 3.3: Filtros Haar utilizados en la detección con opencv [3].

3.2.4 Preprocesado

En la etapa de preprocesado se suelen llevar a cabo las operaciones de rotación, escalado y recorte de la imagen de la cara con el objetivo de tener todas las imágenes de la base de datos a utilizar en un mismo formato, en este caso la operación de rotación no se llevó a cabo puesto que se han cogido versiones de las diversas bases de datos en las que las imágenes ya estaban rotadas y centrados los rostros en la imagen, con lo cual a continuación se describirán las dos etapas de las que se compone el preprocesado que se aplicó a las imágenes de las tres bases de datos en este trabajo. En la figura 3.4 se pueden ver las etapas que componen el preprocesado que se llevó a cabo.

Figura 3.4: Fases del preprocesado del sistema.

3.2.4.1 Recorte

En la figura 3.5 se puede observar el resultado que se obtuvo de recortar una de las imágenes originales de la base de datos LFW haciendo uso de la librería opencv, este mismo proceso se aplicó a todas las imágenes de las bases de datos para facilitar el proceso posterior de extracción de características. Las imágenes de las bases de datos utilizadas se encontraban en un formato altura-anchura-canales que inicialmente era de 250x250x3, en este caso la variable canales es 3 que marca que las imágenes son RGB (Red-Green-Blue), y tras este proceso las imágenes se convirtieron al tamaño de 128x128x3. Al realizar este recorte se consiguió obtener imágenes en las que sólo estuviera la región de interés, es decir, el área de la cara.

(a) Imagen Original (b) Imagen Recortada

Figura 3.5: Ejemplo de una imagen original y su correspondiente imagen recortada con opencv de la base de datos LFW.

En este trabajo la tarea de detección y la parte del preprocesado del recorte de las imágenes se realizó haciendo uso de las CPUs de los diversos nodos que componen el cluster, lo que permitió que se realizasen de forma paralela estas tareas reduciendo el tiempo de computación.

3.2.4.2 Transformación a escala de grises

Como última etapa del preprocesado se realizó una conversión de todas las imágenes de color a escala de grises, de manera que se facilitará más el proceso de reconocimiento a partir de la extracción de características puesto que con está transformación el efecto de tener diversidad de iluminación entre imágenes no tendrá tanta influencia. Con esta transformación se pasó de tener imágenes en formato 128x128x3 a tenerlas en formato 128x128x1. El resultado de esta etapa de preprocesado se puede ver en la figura 3.6.

(a) Recortada (b) Blanco y negro

Figura 3.6: Ejemplo de una imagen recortada a color con opencv de la base de datos LFW y su correspondiente imagen transformada a escala de grises.

3.2.5 Trabajando con grandes volúmenes de datos

Respecto a las etapas tradicionales que se suelen encontrar en los sistemas de reconocimiento facial para este proyecto se tuvo que añadir esta etapa previa al proceso de extracción de características para poder manejar correctamente la gran cantidad de datos que se tiene, principalmente en el caso de la base de datos CASIA, puesto que en el caso de la base de datos LFW esta etapa no se tuvo que utilizar porque el número de imágenes que la componen se podía almacenar completamente sin problema en la memoria de las GPUs que se necesitaron utilizar para realizar de forma eficiente la extracción de características, mientras que en la base de datos CASIA la cantidad de imágenes que la componen impidió que se pudiera trabajar con todas a la vez.

Para solventar este problema lo que se realizó fue un particionamiento de la base de datos CASIA en varios ficheros de extensión .mat de tamaño considerablemente más pequeño y cuyo formato permite ser guardado y cargado con facilidad en Python.

Las etapas que se han explicado hasta ahora solo se realizaron la primera vez que se ejecutó el primer sistema y no se tuvieron que repetir en posteriores implementaciones, puesto que las imágenes procesadas se guardaron para no tener que realizar el proceso siempre que se ejecutaba alguno de los sistemas con los diversos modelos que se probaron.

3.2.6 Extracción de características

La etapa de extracción de características se encarga principalmente de quedarse con los valores que realmente dan información de cada imagen de cara al reconocimiento y desechar aquellos valores que no aportan información relevante. En este trabajo se entrenó una red con datos etiquetados y una vez entrenada se extrajeron las características de cada imagen para poder realizar la clasificación de la etiqueta de identidad, para lo cual se utilizó la salida de la penúltima capa puesto que se supuso que en ese punto del modelo ya se tiene una representación de la entrada con información suficiente para clasificar la identidad de los sujetos.

Para la realización de esta etapa se hizo uso de diversas arquitecturas basadas tanto en redes neuronales convolucionales como redes residuales [16] para comprobar las prestaciones del sistema completo en función de la arquitectura utilizada para la extracción de características. A su vez se dividió en tres partes para poder trabajar de manera más eficiente con las GPUs para esta tarea, el esquema que se siguió para esta etapa se puede ver en la figura 3.7.

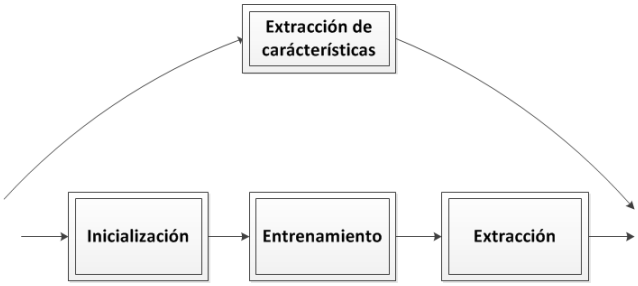


Figura 3.7: Fases de la extracción de características del sistema.

3.2.6.1 Inicialización

- **Red neuronal convolucional**. La primera de las tres etapas de las que se compone la extracción de características se trata de una inicialización de la arquitectura que se utiliza para realizar la extracción de características posteriormente, en este caso se trata de una red neuronal convolucional implementada haciendo uso de la librería de aprendizaje profundo Keras, la cual permitió la creación de las diversas arquitecturas probadas a lo largo de este trabajo de manera cómoda y sencilla.

  En el código 3.1 se puede ver el modelo creado inicialmente para realizar el proceso de extracción de características, se puede observar la simplicidad que conlleva el hecho de añadir una capa o eliminarla en la librería de Keras, así como la definición de sus diversos parámetros. Para la creación de los modelos de este trabajo se utilizaron los siguientes tipos de capas:

  - **Convolucionales** [17], [18] : este tipo de capas son el bloque principal en las redes neuronales convolucionales. Los parámetros de esta capa se componen de un conjunto de filtros que aprenden características como pueden ser: bordes, cambios de dirección, cambios de color.
  - **Pooling**: este tipo de capas se utilizan para llevar a cabo el submuestreo y tienen tres posibilidades de funcionamiento: coger el máximo (*maxpooling*), mínimo (*minpooling*) o promedio (*avgpooling*), cualquiera de ellas en una vecindad, delimitada por un núcleo o tamaño

de ventana. Este tipo de capa se utiliza porque reduce los costes computacionales de las capas siguientes y da robustez frente a una traslación o desplazamiento.

- **Densas**: se trata de capas totalmente conectadas (*FullyConnected*), es decir, las neuronas de estas capas tienen conexiones completas con todas las activaciones de las capas anteriores a ellas. Sus activaciones se pueden calcular pues con una multiplicación de la matriz seguida de un bias de desplazamiento.
- **Activación**: se tratan de capas que contienen funciones de activación, normalmente no lineales, que se aplican a la salida de una capa previa y determinan cómo y en qué casos se activará una neurona de la siguiente capa. Algunas como la ReLu hacen incluso el efecto de un umbral de activación que al ser superado se activará una neurona de la siguiente capa.
- **Dropout** [19]: este tipo de capas se utilizan para que las redes puedan generalizar mejor, la operación que realizan consiste en colocar aleatoriamente una fracción p de unidades de entrada a la capa a 0 en cada actualización durante el entrenamiento, lo que ayuda a prevenir el sobreajuste o overfitting, ya que cuando se desactivan algunas neuronas el resto tienen que aprender a realizar esa tarea.
- **Flatten**: esta capa se utiliza en Keras para poder pasar de utilizar capas convolucionales y de pooling a tener los datos en un formato que se pueda pasar a las capas densas.

También se puede ver en el código 3.1 la compilación necesaria del modelo antes de que se realice su entrenamiento, en la cual también se marca la función de optimización y de pérdidas de dicho modelo. Mientras que en la figura 3.8 que se encuentra junto al código se muestra el resultado de la creación del modelo anteriormente mencionado.

Listing 3.1: Creando el modelo inicial

```
img_rows, img_cols = 128, 128
img_channels = 1

def create_model():
 model = Sequential()
 model.add(Convolution2D(32,11,11,
     border_mode='same',dim_ordering='tf',
      input_shape=(img_rows, img_cols,
     img_channels),activation='relu'))
 model.add(MaxPooling2D(pool_size=(3,3),
     strides=(2,2),dim_ordering='tf'))
 model.add(Convolution2D(16,9,9,
     dim_ordering='tf',activation='relu'))
 model.add(Flatten())
 model.add(Dense(512,activation='relu'))
 model.add(Dropout(0.5))
 model.add(Dense(nb_classes,activation='
     softmax'))
 return model

model=create_model()
model.compile(loss='
    categorical_crossentropy', optimizer='
    sgd',metrics=["accuracy"])
```

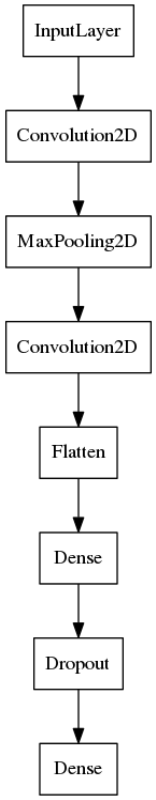


Figura 3.8: Arquitectura del modelo inicial creado.

Con el fragmento anterior se puede observar la creación de la arquitectura de la red usada en un modelo, mientras que en la figura 3.9 se puede ver como es el proceso completo de inicialización del modelo. En la cual se puede apreciar como después de crear la estructura del modelo, si ya existía el modelo previamente se pasa directamente a realizar el entrenamiento del modelo y si no existía este modelo se realiza el siguiente proceso:

- **Crear carpeta**: se crea una carpeta para ese modelo donde se guardaran posteriormente todos los ficheros intermedios que se vayan creando.
- **Guardar modelo**: se guarda el modelo en un fichero de extensión .json, el cual permite de manera sencilla almacenar la estructura del modelo creado.

– **Guardar pesos**: se guardan los pesos iniciales del modelo en un fichero de tipo .h5[3] , lo que permite posteriormente empezar directamente el entrenamiento a partir de unos pesos iniciales.

Este proceso se realizó así debido a que este proyecto en su conjunto se implementó haciendo uso de un cluster con recursos compartidos y un sistema de colas llamado condor[4] que prioriza los trabajos de manera interna, lo que podía llevar a que en alguna ocasión la ejecución del proceso de extracción de características se pudiese ver interrumpido en su desarrollo y el hecho de implementarlo de esta manera en tres fases diferentes permitía recuperar el proceso en el punto en el que se hubiese visto interrumpido, sin tener que empezar completamente dicho proceso.

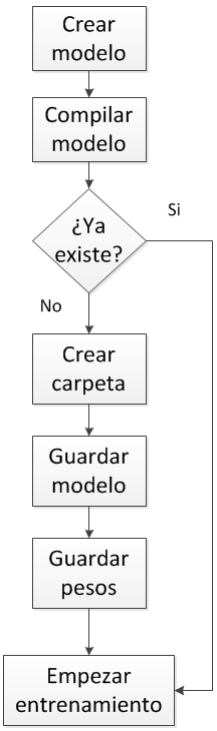
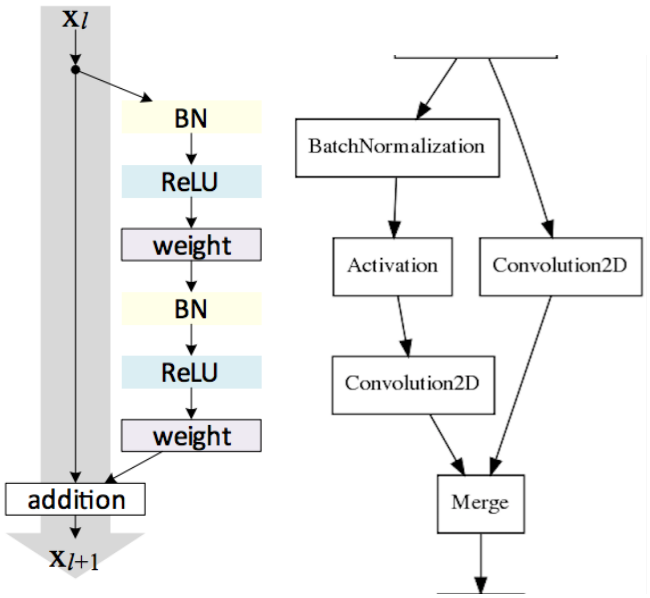


Figura 3.9: Inicialización de los modelos creados para la extracción de características.

[3]url: https://www.hdfgroup.org/HDF5/
[4]url:https://research.cs.wisc.edu/htcondor/

- **Red residual o resnet** [16]. Como una alternativa con base de partida en el modelo anterior de una red neuronal convolucional, también se implementaron diversos modelos de redes residuales con el objetivo de comprobar si mejoraba la convergencia y se mantenía la eficiencia de los modelos implementados con las redes neuronales convolucionales. En la figura 3.10 se puede ver como es la arquitectura de una red de este tipo, cuya principal diferencia con la arquitectura de red neuronal convolucional comentada anteriormente reside en que tiene un camino residual con el cual se pretendió mejorar la convergencia del sistema en el entrenamiento, ya que se evita en parte el problema de calcular los gradientes con redes profundas. Además este tipo de redes incorporá el uso de capas de normalización batch [20] que mejoran la velocidad de convergencia.



(a) Resnet artículo (b) Resnet implementada

Figura 3.10: Imagen de la resnet propuesta en el artículo de referencia y otra imagen del trozo del modelo implementado en este trabajo que se corresponde con la arquitectura resnet propuesta en el artículo.

3.2.6.2 Entrenamiento

Durante la realización de este trabajo se realizó el entrenamiento de las diversas arquitecturas implementadas con las bases de datos LFW y CASIA presentadas anteriormente siguiendo el proceso genérico que se puede ver en la figura 3.11. El protocolo en el que se utilizó la base de datos LFW para el entrenamiento, al contar con un número reducido de imágenes se cargaba completamente en memoria para cada iteración sin problema. En cambio, cuando se entrenó el modelo con la base de datos CASIA al tener una cantidad tan elevada de imágenes se dio el problema de no poder cargar todas ellas en memoria de las GPUs que se utilizaban, así que se tuvo que crear una función generadora para cargar las imágenes para el entrenamiento.

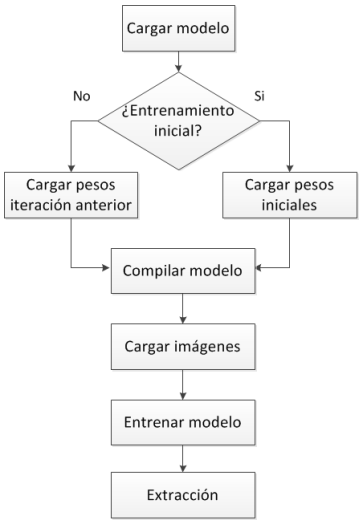


Figura 3.11: Entrenamiento genérico de los modelos creados para la extracción de características.

El proceso que se siguió al utilizar la función generadora para cargar los ficheros y entrenar el modelo se volvió más complejo que cuando no se hacía uso de dicha función como se puede ver en la figura 3.12. Los pasos que se deben llevar a cabo para realizar este proceso son los siguientes:

- **Aleatorizar orden ficheros**: se realiza el proceso de aleatorizar el orden de los ficheros que se van a cargar para conseguir mezclar lo máximo posible las imágenes de la base de datos y que no aparezcan siempre en el mismo

orden para que la red no oscile entre clasificar bien los primeros ficheros a clasificar bien los últimos.

- **Unir ficheros a cargar en bloques**: se crean los bloques en función del número de ficheros que se desea cargar en cada bloque, este parámetro se define en una variable al inicio de la función para facilitar su modificación para poder realizar diversas pruebas y así poder comprobar la importancia de la aleatorización de los ficheros en el entrenamiento.

- **Cargar ficheros bloque**: se cargan los ficheros que se corresponden a cada bloque, dichos ficheros fueron creados en la etapa previa de trabajo con grandes volúmenes de datos donde se guardó la base de datos por partes en ficheros de extensión .mat que son los que se cargan en este punto. Este paso se repite hasta que se carguen todos los ficheros correspondientes a cada bloque.

- **Añadir ficheros del bloque al entrenamiento**: se coge el conjunto de ficheros de cada bloque cargado anteriormente y se realiza el proceso de ir añadiéndolos en lotes o *batches* para el entrenamiento, dichos lotes son conjuntos de varios ejemplos para los que se calcula el gradiente.

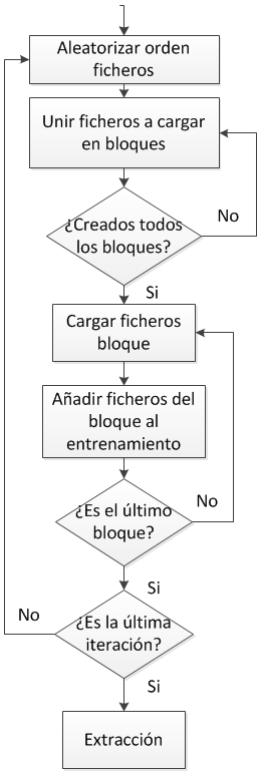


Figura 3.12: Cargando las imágenes de la base de datos CASIA para el entrenamiento.

Al igual que ocurría con el proceso de cargar las imágenes, la función de entrenamiento del modelo que se utilizó difiere en función de la base de datos empleada, cuando se realizó utilizando la base de datos LFW se usó el método *fit* para entrenarlo con un tamaño fijo de *batch* como se puede ver en el código 3.2. Mientras que cuando se utilizó la base de datos CASIA para entrenar se hizo uso del método *fit_generator*, como se puede observar en el código 3.3 que permite utilizar la función generadora creada para poder cargar todos los ficheros en formato .mat con las imágenes en cada iteración del entrenamiento.

Listing 3.2: Entrenando el modelo con la LFW

```
save_model=ModelCheckpoint("model.{epoch:02d}.h5",modelos=Lista,
    verbose=1,save_best_only=False)

model.fit(X_train, Y_train, batch_size=batch_size,nb_epoch=nb_epoch,
    verbose=2,validation_data=(X_test, Y_test), shuffle=True,
    callbacks=[save_model],epoch=epoch)
```

Listing 3.3: Entrenando el modelo con la CASIA

```
save_model=ModelCheckpoint("model.{epoch:02d}.h5",modelos=Lista,
    verbose=1,save_best_only=False)

model.fit_generator(myGenerator(), samples_per_epoch=
    samples_per_epoch, nb_epoch=nb_epoch, verbose=2, callbacks=[
    save_model], epoch=epoch)
```

En los códigos anteriores también se puede observar que se utilizó una función de guardado (*ModelCheckPoint*) para que se pudiera guardar el modelo que resultaba de cada iteración del entrenamiento, así si por cualquier problema se parase el proceso se pudiese retomar en la iteración que se encontraba haciendo uso del último modelo guardado. Esta acción se pudo realizar con relativa sencillez debido a la facilidad con la que se pueden manipular las funciones de la librería Keras.

3.2.6.3 Extracción

Como última parte del proceso de extracción de características, una vez entrenado el modelo con el número de iteraciones deseadas, se extrajo el vector de características de cada imagen del conjunto de evaluación o test. El modelo que se creó en Keras se puede ver como un sistema cerrado del que se puede obtener solo la entrada y la salida final no las salidas intermedias de cada capa, para que se pudieran obtener dichas salidas se creó una función, la cual aparece en el código 3.4, en Theano[5] , librería matemática sobre la que trabaja Keras y que está basada en Python, con la cual se pudo obtener la salida de la penúltima capa, que es la que se utilizó a lo largo de todo el trabajo como vector de características de las imágenes. Dichos vectores se guardaron cada uno en un fichero diferente para posteriormente poder realizar la comparación y reconocimiento en un proceso independiente al de extracción de características.

[5]url: http://deeplearning.net/software/theano/

Se decidió utilizar la salida de la penúltima capa como vector de características puesto que en ese punto la red ya ha procesado lo suficiente para que se pueda utilizar esa salida para comparar las imágenes que queremos determinar si son de la misma persona. A partir de esta representación se obtuvo posteriormente la métrica que permitió comprobar si el sistema se había entrenado de manera correcta y era capaz de diferenciar en una pareja de imágenes si se trata de la misma persona o no.

Listing 3.4: Función de extracción de capas intermedias

```
def get_activations(model, layer, X):
    fget_layer_output = K.function([model.layers[0].input],
                                   [model.layers[layer].output])
    layer_output = fget_layer_output([X])[0]
    return layer_output
```

3.2.7 Comparación y reconocimiento

La última etapa del sistema de reconocimiento, como se puede observar en la figura 3.13, se trató de la comparación y reconocimiento de las características extraídas de las imágenes, se pasaron las imágenes en parejas para poder concluir si se trataba de la misma persona o no, estas parejas se crearon siguiendo el protocolo propuesto en la página web de donde se descargó la base de datos LFW [6] .

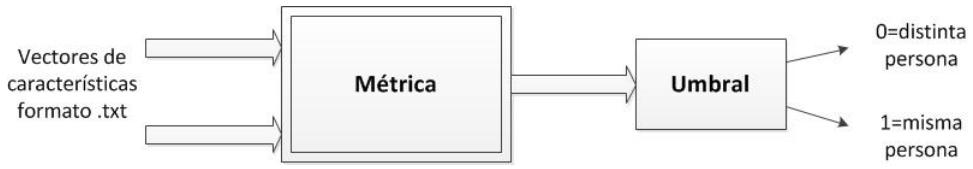


Figura 3.13: Proceso de comparación y reconocimiento implementado en este sistema.

El proceso se basó en utilizar parejas de ficheros de extensión .txt, los cuales contenían los vectores de características, para compararlos con una métrica, en

[6] url: vis-www.cs.umass.edu/lfw/

este caso se eligió la distancia coseno como métrica para comprobar la similitud entre dos imágenes. Esta métrica se escogió debido a que se encontró que se había propuesto como una alternativa efectiva y simple a la distancia Euclídea, y con la que además ya se habían conseguido grandes resultados en precision con la base de datos LFW [21]. A continuación se puede observar, como ya se comentó en el capítulo anterior, la ecuación por la que se rige esta medida de similitud:

$$d(\mathbf{x}, \mathbf{y}) = sim(\mathbf{x}, \mathbf{y}) = \cos(\theta) = \frac{\mathbf{x}^T \bullet \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \tag{3.1}$$

Una vez se tuvo calculada la métrica de similitud entre cada par de imágenes se hizo uso de un umbral (γ) con el cual se pudo obtener la predicción de la etiqueta de identidad de cada par de imágenes del conjunto total de parejas de imágenes a comparar. Con esta métrica se permite calcular el resultado de Precisión que aparecerá posteriormente en las tablas de resultados del capítulo siguiente.

$$y' = \begin{cases} 1 & \text{si} \quad d \geq \gamma \\ 0 & \text{si} \quad d < \gamma \end{cases} \tag{3.2}$$

Además, como se explicará en el siguiente capítulo, se utilizarán diversas medidas para comparar la eficiencia de los diversos protocolos probados sobre los sistemas implementados en este trabajo, estas medidas son la ya mencionada ROC, el área bajo la curva ROC (AUC) y la tasa de error igual o *equal error rate* (EER).

3.3 Sistema basado en una red neuronal convolucional siamesa

3.3.1 Descripción

Este sistema se propuso como una línea alternativa de este trabajo, más innovadora puesto que el modelo de red neuronal siamesa [17], cuya filosofía se muestra en la figura 3.14, no se encontró en ninguno de los artículos que se leyeron en la primera etapa del desarrollo de este proyecto que se hubiera utilizado un sistema basado en una red convolucional siamesa de esta forma. Sin embargo, se comprobó que en [22] se usaba una red siamesa pero sin los pesos atados, es decir, se trataba de dos redes convolucionales separadas que solo se unían en la etapa final para realizar la clasificación con una SVM, por otro lado también se pudo ver que en [23] se había usado con partes de los rostros de las imágenes, no con la imagen completa como se hizo en este trabajo.

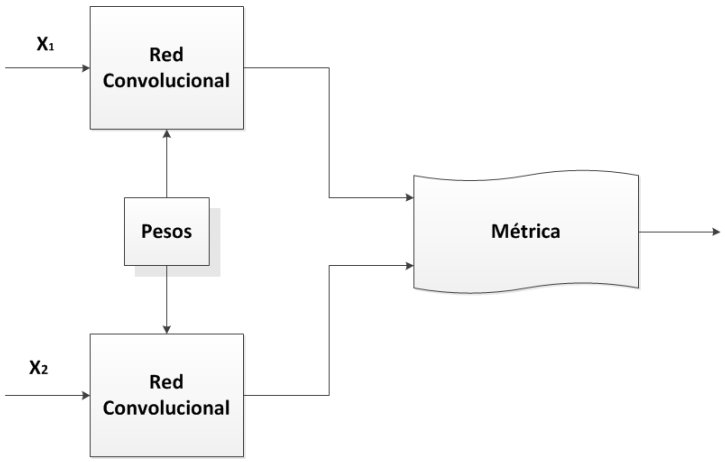


Figura 3.14: Sistema basado en una red neuronal convolucional siamesa.

Con este sistema el objetivo que se pretendía se basaba en realizar la regresión de la métrica coseno con una variable binaria (0=distintos, 1=iguales). Además, el hecho de incluir la misma métrica que se usa para decidir en el entrenamiento es una ventaja adicional, ya que los gradientes para optimizar los parámetros serán calculados para mejorar la métrica en la optimización, no haciendo uso de una clasificación intermedia como en el sistema presentado en el apartado anterior.

Este aspecto de la arquitectura también es una novedad con respecto a trabajos anteriores.

En una primera fase de implementación de este sistema se intentó el entrenamiento completo del modelo pero no se consiguió que convergiera de manera óptima, entonces se aprovechó la modularidad del sistema implementado en el apartado anterior, para utilizar el modelo ya entrenado en el proceso de extracción de características de dicho sistema como un pre-entrenamiento de la arquitectura que se creó en este segundo sistema, en la cual se utilizó los pesos finales del modelo anterior como pesos iniciales de este modelo en sus dos ramas. En este sistema las etapas que se utilizaron en el sistema anterior se tuvieron que variar ligeramente puesto que se propuso la creación de un sistema basado en una red siamesa que realizará el proceso de extracción de características y la comparación y reconocimiento todo dentro de la misma arquitectura, es decir, con este sistema se evitó tener que guardar los vectores de características en ficheros externos y luego tener que cargarlos de nuevo para realizar la comparación y el reconocimiento.

En una segunda fase de implementación, una vez comprobado el correcto funcionamiento del sistema con los pesos cargados del otro sistema y solucionados los problemas iniciales que se habían tenido, se probó la arquitectura inicial que entrenaba el modelo completo desde cero sin cargar pesos. Como se comentará en el siguiente capítulo se consiguió finalmente que convergiera y se realizaron las mismas pruebas que se habían realizado con el sistema con los pesos cargados del pre-entrenamiento para comparar los resultados obtenidos de las dos formas.

Por último, como se verá en el capítulo de resultados, con los modelos basados en redes residuales en el sistema anterior se obtuvieron buenos resultados y mejoraron los tiempos de convergencia de manera considerable, por lo cual también se implementaron los modelos basados en redes residuales en este segundo sistema para ver qué resultados se obtenían.

3.3.2 Inicialización

En este segundo sistema se procede a explicar directamente el proceso de inicialización de la red neuronal utilizada en este sistema para el posterior entrenamiento y reconocimiento puesto que las etapas de detección y preprocesado que se emplearon son las mismas que ya se explicaron en el primer sistema y las cuales solo se tuvieron que ejecutar una vez ya que las imágenes resultantes de dichas etapas se guardaron para no tener que volver a realizar este proceso.

Para que se pudiera crear el modelo de red siamesa se tuvo que utilizar la posibilidad que muestra Keras para el uso de capas con pesos compartidos, es decir, se pueden crear dos redes con las mismas capas que formen la mencionada red siamesa. En el código 3.5 se puede ver como se pueden añadir las capas compartidas a dos entradas diferentes, creando dos caminos paralelos que finalmente se pueden unir de distintas maneras, en este caso se unieron haciendo uso de la misma métrica que se comentó en el sistema anterior y la cual se mencionó al comienzo de este apartado, la distancia coseno, puesto que Keras la tenía implementada en uno de los modos de la capa de unión *merge* (*mode='cos'*). La figura 3.15 muestra el modelo de capas que se creó con el código comentado anteriormente.

Listing 3.5: Creando el modelo de la red siamesa

```
inputa=Input(shape=(img_rows,img_cols,
    img_channels,))
inputb=Input(shape=(img_rows,img_cols,
    img_channels,))
shared_conv1=Convolution2D(32,11,11,
    border_mode='same',dim_ordering='tf',
    input_shape=(img_rows, img_cols,
    img_channels),activation='relu')
shared_pool1=MaxPooling2D(pool_size=(3,3),
    strides=(2,2),dim_ordering='tf')
shared_conv2=Convolution2D(16,9,9,
    dim_ordering='tf',activation='relu')
shared_flatten=Flatten()
shared_dense1=Dense(1024,activation='relu')
shared_dense2=Dense(1024,activation='relu')
shared_dense3=Dense(512,activation='relu')

x1a=shared_conv1(inputa)
x2a=shared_pool1(x1a)
x3a=shared_conv2(x2a)
x4a=shared_flatten(x3a)
x5a=shared_dense1(x4a)
x6a=shared_dense2(x5a)
x7a=shared_dense3(x6a)
x1b=shared_conv1(inputb)
x2b=shared_pool1(x1b)
x3b=shared_conv2(x2b)
x4b=shared_flatten(x3b)
x5b=shared_dense1(x4b)
x6b=shared_dense2(x5b)
x7b=shared_dense3(x6b)

cos_distance=merge([x7a,x7b],mode='cos',
    dot_axes=1)
cos_dis=Reshape((1,))(cos_distance)
out=Dense(1,activation='sigmoid',name='out')(
    cos_dis)
model=Model(input=[inputa,inputb],output=out)
```

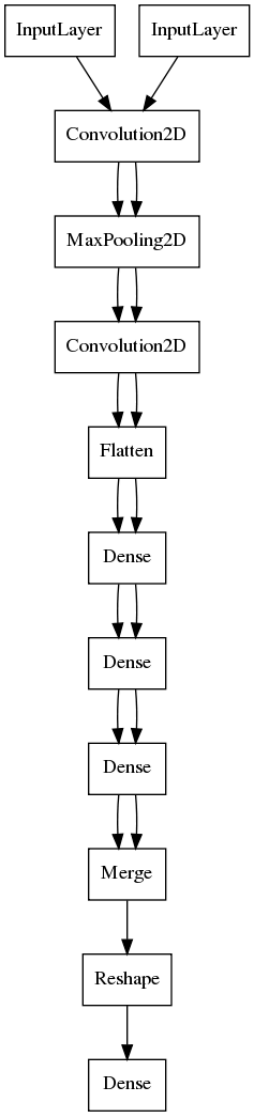


Figura 3.15: Modelo de red siamesa creado inicialmente.

Para que se pudiera llevar a cabo la primera de las implementaciones mencionadas anteriormente y que se pudieran cargar los pesos del modelo ya entrenado se tuvo que modificar una de las funciones de la librería Keras puesto que las capas finales de uno y otro modelo no coincidían y por lo tanto se debía cargar solo los pesos de las capas que sí que se correspondían entre ellas. Para llevar a cabo esto se modificó la llamada a la función de Keras para poder pasarle por parámetro en que capa debía empezar y acabar de cargar los pesos.

Para los dos primeros enfoques el modelo que se utilizó es el mismo de la figura 3.15, mientras que para el modelo creado con redes residuales la arquitectura que se implementó se puede ver en la figura 3.16.

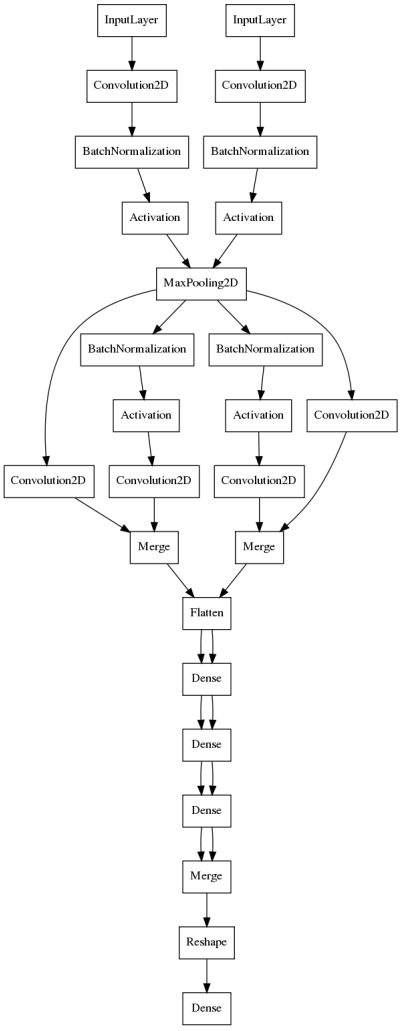


Figura 3.16: Arquitectura de red residual siamesa.

3.3.3 Entrenamiento

La fase de entrenamiento del modelo se realizó como en el sistema anterior, de manera normal con la función *fit* si se usa la base de datos LFW para el entrenamiento y con el generador en el caso de utilizar la base de datos CASIA para entrenamiento. La principal diferencia con el entrenamiento del otro sistema se encuentra en que como la arquitectura implementada en la inicialización de este sistema se basó en un modelo pre entrenado para cargar los pesos finales de ese modelo ya entrenado, en este caso el número de iteraciones que se utilizaron fueron muy pocas. Mientras que para el segundo enfoque de este sistema, entrenando el sistema sin un modelo previo como punto de partida se tuvieron que utilizar más iteraciones con lo que implica un mayor tiempo de computación para obtener unos resultados similares.

3.3.4 Comparación y reconocimiento

Por último, se realizó la etapa de comparación y reconocimiento, como en este sistema no se necesitaba guardar los ficheros con los vectores de características para luego poder calcular la métrica, sino que ya se entrena la red con el objetivo de optimizar dicha métrica, este proceso se simplificó bastante. Únicamente se utilizaron dos funciones implementadas en la librería Keras, las cuales se muestran en el código 3.6, con la función *evaluate* se puede obtener las métricas de perdidas y precision del modelo entrenado. Y con la segunda función la de *predict* se obtiene una predicción de las etiquetas en función a los datos de evaluación o test, lo que se devuelve son valores de probabilidad con los cuales luego se pueden calcular las mismas métricas que se citaron para el primer sistema: ROC, AUC y EER.

Listing 3.6: Predicción de valores en la red siamesa

```
score =model.evaluate([ X_test, X2_test], y_test, verbose=2)
print('Test score:', score[0])
print('Test accuracy:', score[1])

y_pred=model.predict([X_test,X2_test])
```

Capítulo 4

Experimentación y Resultados

En este capítulo, se explican los experimentos llevados a cabo para evaluar el rendimiento de los sistemas de reconocimiento facial que se explicaron en el Capítulo 3. Dentro de este conjunto de experimentos, se realizaron algunos en los que solo se utilizó la base de datos LFW, la cual tiene un subconjunto de imágenes de sujetos que solo se utilizaron para el entrenamiento (train) y otro subconjunto de sujetos que solo se utilizaron para la evaluación o reconocimiento (test), así como otros en los que se usó la base de datos CASIA para el entrenamiento y la base de datos LFW con el subconjunto de sujetos destinados para el reconocimiento. Por último, se realizó una prueba final en la que se entrenó con la base de datos CASIA y se utilizó la base de datos YTF para el reconocimiento.

Aparte de ver las diferencias que se consiguen con los cambios de base de datos para el entrenamiento, se exponen diversos protocolos que se probaron para ver la importancia que tiene el preprocesado previo de las imágenes que se realizó. Así como también se presenta una explicación sobre las diferencias en los tiempos de computación al usar las redes residuales (resnet) respecto a utilizar redes convolucionales.

A la finalización de cada uno de los experimentos que se exponen se incluye una tabla resumen con los resultados obtenidos y dos figuras que muestran la eficiencia

del sistema dado ese experimento concreto. Dichas figuras contienen la siguiente información:

- En la primera se representan curvas ROC, curvas en las que se presenta la sensibilidad en función de los falsos positivos para distintos puntos de trabajo, y sus respectivos valores de AUC, área que se puede interpretar como la probabilidad de que una pareja de imágenes se pueda determinar correctamente si son la misma persona o no.

- En la segunda se representan curvas de probabilidad de error de detección y el punto EER, se trata de la tasa en la cual se aceptan y rechazan los errores por igual, y en un buen sistema se debe mantener este valor tan pequeño como sea posible.

4.1 Influencia del número de imágenes e individuos en el entrenamiento del sistema

El protocolo que se utilizó para realizar este experimento se basó en comparar los resultados obtenidos de llevar a cabo el entrenamiento del modelo que se muestra en la figura 4.1 con la base de datos LFW respecto a entrenarlo con la base de datos CASIA, en ambos casos se utilizó para el reconocimiento el grupo de imágenes de evaluación de la base de datos LFW.

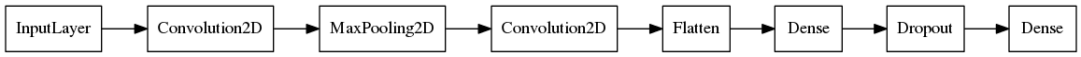
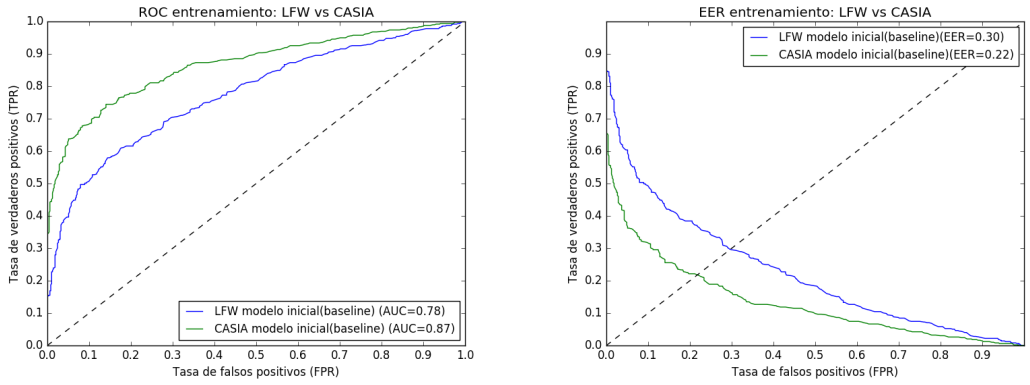


Figura 4.1: Modelo inicial (baseline).

Se planteó realizar esta comparación puesto que inicialmente se entrenó y evaluó solo con los conjuntos de imágenes de la base de datos LFW con los que se comprobó que el modelo no terminaba de aprender a reconocer correctamente entre los pares de imágenes, por lo cual se valoró la posibilidad de que se debiera al hecho de contar con un número escaso de imágenes de entrenamiento para esta tarea, ya que las redes neuronales profundas no se entrenan bien si hay pocos

ejemplos. Entonces se propuso probar a utilizar para el entrenamiento una base de datos mucho mayor en cuanto al número de imágenes y de sujetos diferentes. Para lo cual se eligió la base de datos CASIA, la cual se usó para entrenar el modelo mientras que la evaluación del modelo se realizó con el mismo conjunto de imágenes de evaluación de la base de datos LFW para poder comparar los resultados obtenidos.

En la figura 4.2 se puede ver como mejoró el rendimiento global del sistema al utilizar una base de datos más grande para entrenamiento, se obtuvo un resultado de AUC considerablemente superior al que se consiguió con el modelo inicial entrenado con la base de datos LFW. Por otro lado, también se puede ver como el EER disminuyó también en el caso de usar una base de datos más grande para el entrenamiento.



(a) ROC (b) EER

Figura 4.2: Curvas ROC y EER obtenidas de realizar el entrenamiento con las bases de datos LFW y CASIA con el modelo inicial creado, donde C indica el número de capas convolucionales y D el número de capas densas.

Con este experimento se consiguió demostrar la importancia de tener una cantidad de datos suficientemente grande a la hora de entrenar un sistema de reconocimiento facial cuyo extractor de características se trata de una red neuronal profunda.

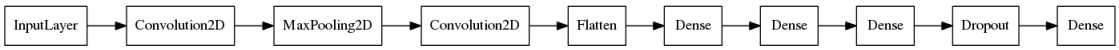
A continuación se puede ver en la tabla 4.1 de manera resumida los resultados obtenidos con este experimento. En dicha tabla se muestran recogidos los valores de AUC y EER que se pueden observar en las figuras anteriores, así como la precisión en el número de parejas acertadas en el reconocimiento con dichos modelos de la lista de parejas que se encuentran en la página web de donde se descargó la base de datos LFW.

Tabla 4.1: Tabla de resultados del experimento realizado entrenando con las bases de datos LFW y CASIA con el modelo inicial creado, donde C indica el número de capas convolucionales y D el número de capas densas.

BD entrenamiento	BD evaluación	Arquitectura	AUC	EER	Precisión
LFW	LFW	inicial(2C+2D)	0.78	0.30	708/989
CASIA	LFW	**inicial(2C+2D)**	**0.87**	**0.22**	**783/989**

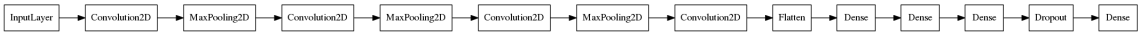
4.2 Influencia del número de capas del modelo

El protocolo que se utilizó para realizar este segundo experimento se basó en comparar los resultados obtenidos de llevar a cabo el entrenamiento de diversas arquitecturas para la creación del modelo usado en la extracción de características. Se varió el número de capas para comprobar como variaban los resultados por el hecho de contar con un modelo con mayor número de capas, se probó a añadir dos capas densas más al modelo inicial, que se componía de dos capas convolucionales más dos capas densas, en otro modelo se probó a añadir dos convolucionales más y en el último modelo se añadieron dos capas densas y dos capas convolucionales más al modelo inicial. Los modelos que se evaluaron se pueden ver en la figura 4.3, estos modelos se probaron tanto con el método de entrenar y evaluar con la base de datos LFW como con el de entrenar con la base de datos CASIA y evaluar con la base de datos LFW.



(a) Modelo con dos capas densas más

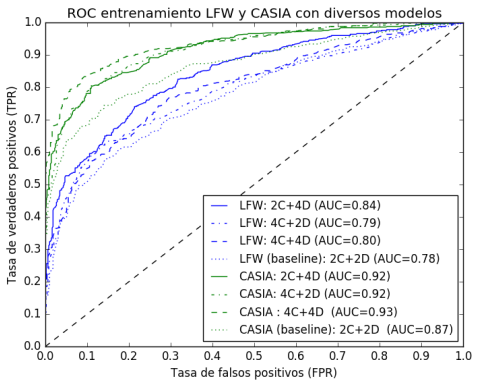
(b) Modelo con dos capas convolucionales más



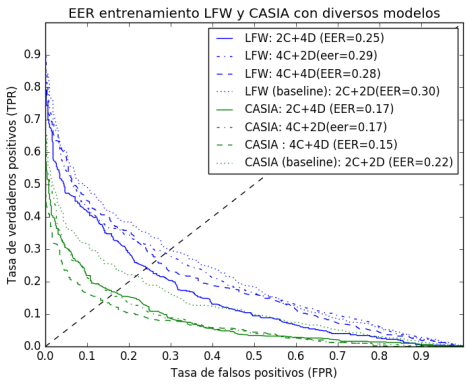
(c) Modelo con dos capas convolucionales y dos capas densas más

Figura 4.3: Modelos con mayor número de capas convolucionales y capas densas que el modelo de referencia.

En la figura 4.4 se puede ver cómo influyó en el rendimiento global del sistema el hecho de cambiar el número de capas en el modelo que se usó para la extracción de las características, obteniendo un resultado de AUC superior al del modelo inicial en el caso de utilizar cualquiera de las tres arquitecturas propuestas para este experimento y también se puede ver como el EER disminuyó también con el uso de estos otros tres modelos.



(a) ROC



(b) EER

Figura 4.4: Curvas ROC y EER obtenidas de realizar el entrenamiento con las bases de datos LFW y CASIA cambiando la estructura de la red que se utiliza, donde C indica el número de capas convolucionales y D el número de capas densas.

Con este experimento se demostró que con un modelo en el que se utilizaron un mayor número de capas densas o convolucionales que las que se tenían en el modelo inicial o baseline se consiguió mejorar la precisión del sistema implementado en este trabajo tanto en el caso de realizar el entrenamiento del modelo con la base de datos LFW (con una mejora relativa del 16,6 %) como al realizarlo con la base de datos CASIA (con una mejora relativa del 31,8 %), puesto que el añadir más capas intermedias aumenta la capacidad del modelo para aprender los patrones, aunque la mejora se puede ver que es mucho más importante en el caso de la base de datos CASIA, lo que confirmó que en redes mas profundas se requieren de muchos mas datos.

Para concluir con este experimento se recogen en la tabla 4.2 los resultados obtenidos con este experimento.

Tabla 4.2: Tabla de resultados del experimento realizado entrenando con las bases de datos LFW y CASIA cambiando la estructura de la red que se utiliza, donde C indica el número de capas convolucionales y D el número de capas densas.

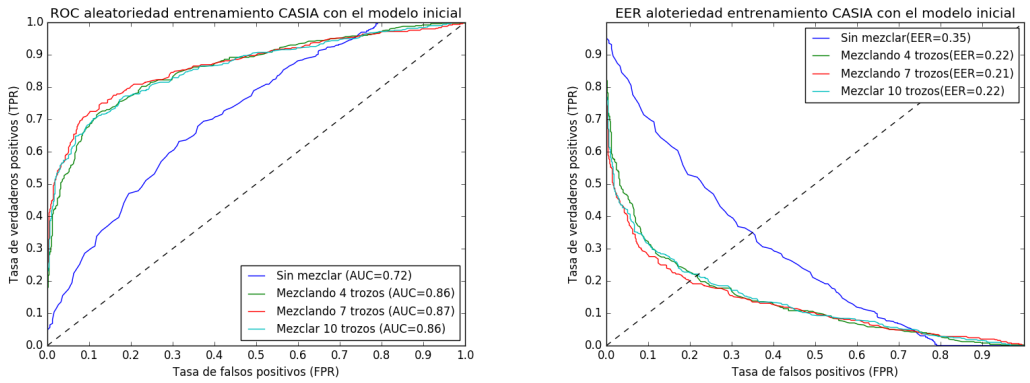
BD entrenamiento	BD evaluación	Arquitectura	AUC	EER	Precisión
LFW	LFW	2C+2D(baseline)	0.78	0.30	708/989
		4C+2D	0.79	0.29	732/989
		2C+4D	0.84	0.25	750/989
		4C+4D	0.80	0.28	723/989
CASIA	LFW	2C+2D(baseline)	0.87	0.22	783/989
		4C+2D	0.92	0.17	826/989
		2C+4D	0.92	0.17	829/989
		4C+4D	**0.93**	**0.15**	**854/989**

4.3 Influencia de la aleatoriedad de las imágenes en el entrenamiento

Para que se pudiera utilizar la base de datos CASIA en todos los experimentos que se realizaron entrenando el modelo con dicha base de datos, como ya se comentó en el capítulo anterior se tuvo que fraccionar el conjunto de imágenes que la forman puesto que todas a la vez en la memoria de las GPUs no se podían cargar. Por

lo que se crearon 200 ficheros .mat que contenían cada uno de ellos una parte de las imágenes de la base de datos CASIA, tras realizar este proceso de división se planteó a la hora de cargar dichos ficheros para el entrenamiento como se debía realizar dicho proceso y la importancia de mezclar los diferentes ficheros al cargarlos en la red para conseguir una correcta convergencia del entrenamiento del modelo.

Para comprobar cual era el mejor procedimiento a la hora de cargar los datos se realizaron cuatro pruebas haciendo uso del modelo inicial o *baseline*. Sobre este modelo se comprobó el efecto de cargar los ficheros de las imágenes sin mezclar, cargando de 4 en 4, de 7 en 7 y de 10 en 10 los ficheros de manera aleatoria.



(a) ROC (b) EER

Figura 4.5: Curvas ROC y EER obtenidas de realizar el entrenamiento con la base de datos CASIA con el modelo inicial creado, probando a no aleatorizar los ficheros al cargarlos, a cargarlos aleatoriamente de 4 en 4, de 7 en 7 y de 10 en 10.

Como se puede comprobar en la figura 4.5 cuando se realizó el experimento de entrenar el modelo con la base de datos CASIA sin mezclar los diversos ficheros creados anteriormente, es decir, se cargaron las imágenes en el orden original en todas las iteraciones, se demostró que el entrenamiento del modelo no llega a ser capaz de converger del todo. Que se diera este hecho era esperable puesto que la red se entrena aprendiendo todas las imágenes en el mismo orden en cada iteración y

no consigue aprender a generalizar para ser capaz de reconocer imágenes diferentes a las del entrenamiento en la fase posterior de reconocimiento. Mientras que en los otros tres experimentos que se realizaron se puede comprobar que no hay diferencia apreciable entre cargar los ficheros de 4 en 4, de 7 en 7 o de 10 en 10 siempre que se carguen de manera aleatoria en cada iteración.

Por último, en la tabla 4.3 se resumen los resultados obtenidos con este experimento, con los cuales se puede comprobar como aumentó la capacidad de reconocimiento del sistema por el hecho de cargar los ficheros de manera aleatoria en vez de cargarlos siempre en el mismo orden, consiguiéndose una mejora relativa del 40 %.

Tabla 4.3: Tabla de resultados del experimento realizado entrenando con la base de datos CASIA con el modelo inicial creado, probando a no aleatorizar los ficheros al cargarlos, a cargarlos aleatoriamente de 4 en 4, de 7 en 7 y de 10 en 10.

BD entrenamiento	BD evaluación	Protocolo entrenamiento	AUC	EER	Precisión
CASIA	LFW	sin mezclar	0.72	0.35	602/989
		cargar 4	0.86	0.22	783/989
		cargar 7	**0.87**	**0.21**	**795/989**
		cargar 10	0.86	0.22	785/989

4.4 Influencia del preprocesado de las imágenes

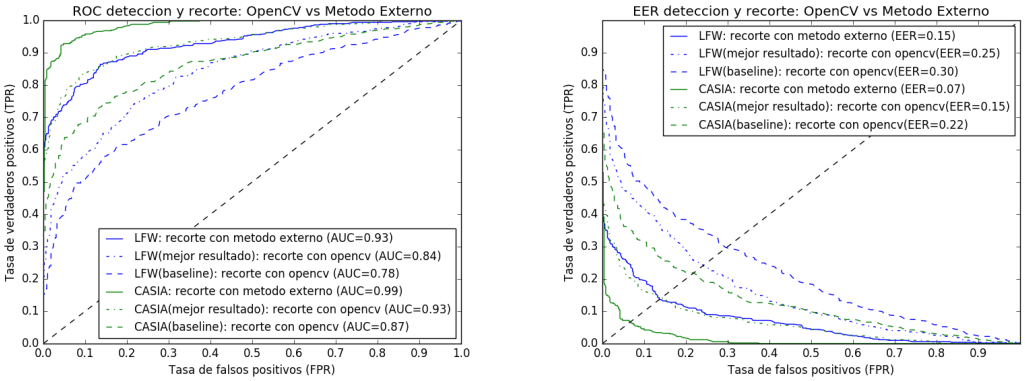
Como se mencionó en el capítulo anterior, cuando se descargaron las bases de datos que posteriormente se preprocesaron con el sistema implementado en este trabajo, también se pudo descargar a la vez una versión de las bases de datos CASIA y LFW ya normalizadas. Las imágenes de las bases de datos originales se habían normalizado [15] siguiendo el siguiente proceso:

- se construyó un modelo 3D deformable que se fue ajustando para estimar la posición de cada cara para la detección de las caras en las imágenes.
- posteriormente se utilizaron dos marcas para realizar el alineamiento de los rostros.

- por último, se aplicó un aumento de datos obteniendo la imagen en espejo de todas las imágenes de la base de datos con el objetivo de conseguir obtener representaciones más robustas a la variación de posición tras aplicar el proceso de entrenamiento basado en una red neuronal convolucional.

Por lo que se puede ver dichas bases de datos normalizadas tienen una etapa de preprocesado más compleja que la implementada en este trabajo, además de que está compuesta del doble de imágenes, así que se decidió comprobar como afecta el hecho de realizar un preprocesado más elaborado en la eficiencia del sistema implementado y contar con un número aún mayor de imágenes. Para este experimento se utilizó el siguiente protocolo que se basó en comparar los resultados obtenidos de llevar a cabo el entrenamiento del modelo que se muestra en la figura 4.3-c) con las bases de datos LFW y CASIA normalizadas con el método externo explicado anteriormente respecto a los que se obtuvieron con las imágenes normalizadas con opencv y el algoritmo de Viola-Jones [3]. Se escogió en concreto este modelo puesto que fue con el que se obtuvieron los mejores resultados en los experimentos anteriores.

En la figura 4.6 se puede ver como mejoró el rendimiento global del sistema al utilizar una base de datos en la cual se había realizado un preprocesado mucho más exhaustivo respecto a los resultados obtenidos con las imágenes preprocesadas en la fase inicial de este proyecto, tanto entrenando con la base de datos LFW (con una mejora relativa del 50 %) como con la base de datos CASIA (con una mejora relativa del 68,1 %), se obtuvo un resultado de AUC considerablemente superior al mejor resultado obtenido con las imágenes de las bases de datos detectadas y recortadas con opencv.



(a) ROC (b) EER

Figura 4.6: Curvas ROC y EER obtenidas de realizar el entrenamiento con las bases de datos LFW y CASIA preprocesada en este trabajo frente a realizar el entrenamiento con las bases de datos descargadas ya preprocesadas.

A continuación se resumen los resultados obtenidos con este experimento en la tabla 4.4 .

Tabla 4.4: Tabla de resultados del experimento realizado entrenando con las bases de datos LFW y CASIA preprocesada en este trabajo frente a realizar el entrenamiento con las bases de datos descargadas ya preprocesadas.

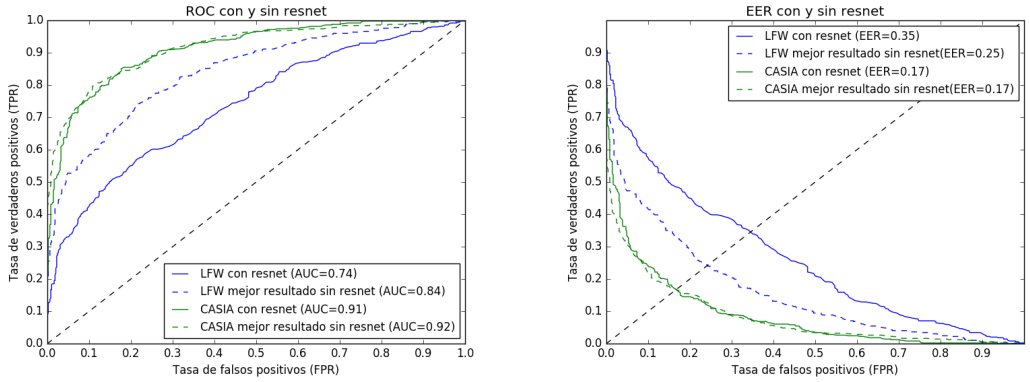
BD entrenamiento	BD evaluación	Tipo de recorte	AUC	EER	Precisión
LFW	LFW	opencv(baseline)	0.78	0.30	708/989
		opencv(mejor resultado)	0.84	0.25	750/989
		método externo	0.93	0.15	834/989
CASIA	LFW	opencv(baseline)	0.87	0.22	783/989
		opencv(mejor resultado)	0.93	0.15	854/989
		método externo	**0.99**	**0.07**	**920/1000**

4.5 Influencia en el tiempo de convergencia del uso de una red residual

En la realización del experimento se quiso comprobar la influencia del uso de redes residuales en el tiempo de convergencia del entrenamiento de los modelos, se realizaron diversos experimentos con modelos de redes convolucionales profundas, con los cuales se habían obtenido los mejores resultados en los primeros

experimentos, para implementar a partir de ellos los modelos basados en redes residuales. Con dichos modelos se pretendió comprobar si se podían conseguir resultados similares pero con un tiempo de computación bastante menor.

Como se muestra en la figura 4.7 los resultados que se consiguieron al utilizar redes residuales en el caso de entrenar con la base de datos CASIA se trataron de resultados similares pero con un número de iteraciones reducido a una cuarta parte de las que se utilizaron cuando no se usó resnet para el modelo. Mientras que en el caso de entrenar con la base de datos LFW no se consiguió alcanzar unos resultados similares puesto que este tipo de redes residuales tienen mayor utilidad en el caso de grandes redes y grandes cantidades de datos para mejorar la convergencia del modelo en el entrenamiento.



(a) ROC (b) EER

Figura 4.7: Curvas ROC y EER obtenidas de realizar el entrenamiento con las bases de datos LFW y CASIA con el modelo que mejores resultados se obtuvieron frente a realizar el entrenamiento de las bases de datos utilizando un modelo basado en una red residual (resnet).

En la tabla 4.5 se puede ver de manera resumida los resultados de este experimento y sobre todo la influencia de utilizar resnet en el número de iteraciones que se tuvieron que usar en el caso del entrenamiento con la CASIA, cuya reducción reduce el tiempo de computación a una cuarta parte respecto al modelo en el que no se utilizó resnet.

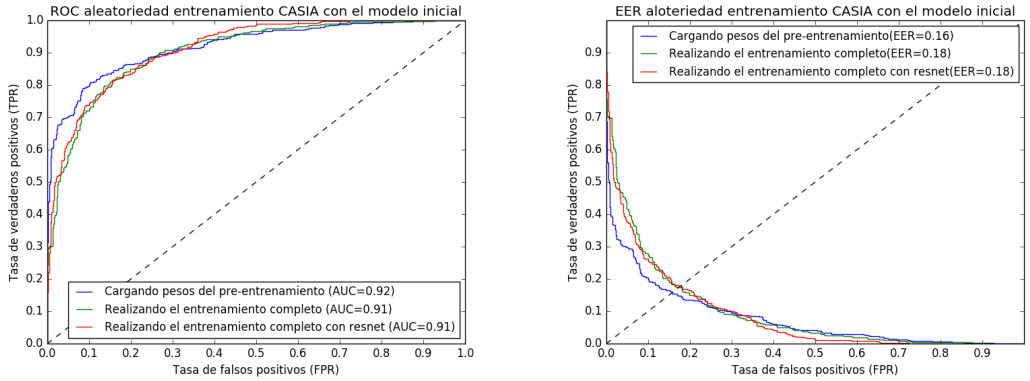
Tabla 4.5: Tabla de resultados del experimento realizado entrenando con las bases de datos LFW y CASIA con el modelo que mejores resultados se obtuvieron frente a realizar el entrenamiento de las bases de datos utilizando un modelo basado en una red residual (resnet).

BD entrenamiento	BD evaluación	Arquitectura	AUC	EER	Precisión	Nº iteraciones
LFW	LFW	sin resnet	0.84	0.25	750/989	50
		con resnet	0.74	0.35	657/989	15
CASIA	LFW	**sin resnet**	**0.92**	**0.17**	**829/989**	**100**
		con resnet	0.91	0.17	820/989	25

4.6 Influencia del protocolo de entrenamiento en el sistema basado en una red siamesa

El segundo sistema que se planteó en el Capítulo 3 se trataba de una red siamesa en la que se desarrolló el mismo sistema con las tres implementaciones distintas para la inicialización y el entrenamiento del sistema que se comentaron. Con dichos implementaciones se realizó el siguiente experimento en el que se utilizó en los tres casos la misma arquitectura para la red siamesa.

Como se puede ver en la figura 4.8 los resultados que se obtuvieron de cargar los pesos de un modelo previo ya entrenado e iterar fueron ligeramente mejores que los de entrenar la red siamesa desde cero tanto usando una red convolucional profunda como usando una resnet. Esto se debe al hecho de que conseguir que converja la red siamesa con el mismo número de iteraciones se trata de una tarea más compleja que el primer sistema del cual se cargan los pesos, puesto que conlleva el doble de tiempo computacional.



(a) ROC (b) EER

Figura 4.8: Curvas ROC y EER obtenidas de realizar el entrenamiento con la base de datos CASIA con el sistema basado en una red siamesa, probando a cargar los pesos de un modelo ya pre-entrenado, a realizar el entrenamiento completo y a realizar el entrenamiento completo con una red residual.

Por último, en la tabla 4.6 se resumen los resultados obtenidos con este experimento.

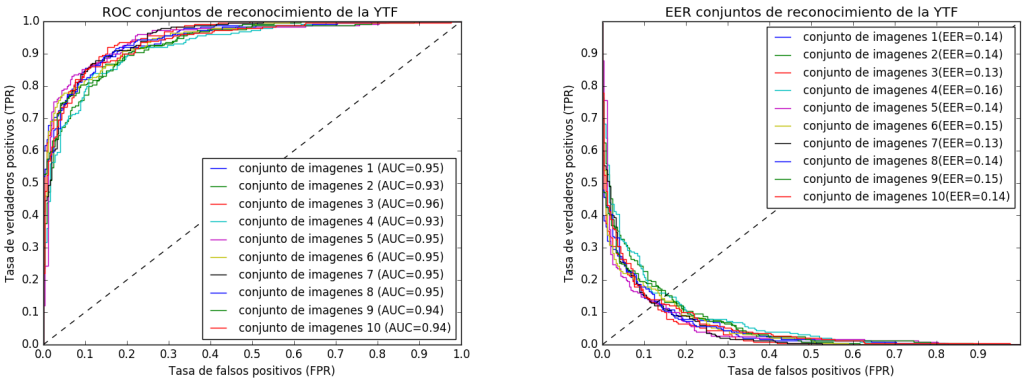
Tabla 4.6: Tabla de resultados del experimento realizado entrenando con la base de datos CASIA con el sistema basado en una red siamesa, probando a cargar los pesos de un modelo ya pre-entrenado, a realizar el entrenamiento completo y a realizar el entrenamiento completo con una red residual.

BD entrenamiento	BD evaluación	Protocolo entrenamiento	AUC	EER	Precisión
CASIA	LFW	**Con pre-entrenamiento**	**0.92**	**0.16**	**842/989**
		Sin pre-entrenamietno	0.91	0.18	820/989
		Sin pre-entrenamiento+resnet	0.91	0.18	812/989

4.7 Influencia de la calidad de las imágenes para el reconocimiento

Como último experimento que se planteó utilizar la base de datos YTF para comprobar los resultados que se obtenían al utilizar el sistema que se había entrenado con imágenes de mejor calidad y estáticas para realizar el reconocimiento con los fotogramas de los vídeos que componen la base de datos YTF y que forman por tanto una secuencia.

En la figura 4.9 se puede observar como los resultados que se consiguieron en el reconocimiento del conjunto de parejas que se proponen en la web[1] fueron muy buenos, en los 10 subconjuntos de parejas que aparecen se obtuvieron resultados de AUC por encima de los conseguidos en los experimentos que se realizaron anteriormente en los que se utilizó en todos ellos los conjuntos de parejas propuestos para la base de datos LFW.



(a) ROC (b) EER

Figura 4.9: Curvas ROC y EER obtenidas de realizar el entrenamiento con la base de datos CASIA con el modelo con el que mejores resultados se obtuvieron y realizar el reconocimiento con la base de datos YTF.

[1]url: http://www.cs.tau.ac.il/ wolf/ytfaces/

El hecho de que se obtuvieran mejores resultados al realizar el reconocimiento con la base de datos YTF se debe a la forma en la que se calculó la métrica de distancia coseno, puesto que en este caso cada pareja que se comparo se hizo por vídeo, es decir, luego dentro de cada vídeo hay varios fotogramas que se compararon uno a uno con los del otro vídeo con el que se quería ver si es la misma persona o no, obteniendo la distancia de cada uno y luego calculando la media de todas ella para obtener la distancia final con la que se realizaron los cálculos de ROC, AUC y EER.

Para finalizar se encuentra resumido en la tabla 4.7 los resultados obtenidos con este experimento.

Tabla 4.7: Tabla de resultados del experimento realizado entrenando con la base de datos CASIA con el modelo con el que mejores resultados se obtuvieron y realizar el reconocimiento con la base de datos YTF.

BD entrenamiento	BD evaluación	Conjunto de datos	AUC	EER	Precisión
CASIA	YTF	Conjunto 1	0.95	0.14	434/500
		Conjunto 2	0.93	0.14	418/500
		Conjunto 3	0.96	0.13	431/499
		Conjunto 4	0.93	0.16	422/500
		Conjunto 5	0.95	0.14	435/500
		Conjunto 6	0.95	0.15	429/500
		Conjunto 7	0.95	0.13	430/496
		Conjunto 8	0.95	0.14	432/500
		Conjunto 9	0.94	0.15	421/496
		Conjunto 10	0.94	0.14	432/500
Resultados promedio			0.945	0.142	

Capítulo 5

Conclusiones y Líneas futuras de trabajo

5.1 Conclusiones

En este trabajo se han propuesto e implementado dos sistemas principales para el reconocimiento facial, aunque también se han probado ligeras variaciones de la arquitectura de dichos sistemas. El segundo sistema propuesto basado en una red siamesa se implementó como una nueva arquitectura de red aplicada de manera novedosa al objetivo que se llevó a cabo en este trabajo y con la cual se consiguió obtener un sistema con un buen funcionamiento y buenos resultados de eficiencia en la tarea de reconocimiento facial.

Los sistemas que se crearon inicialmente para el reconocimiento facial, tras realizar algunas modificaciones a los modelos de redes iniciales que se usaban en el entrenamiento, consiguieron buenos resultados con una base de datos de entrenamiento con un número considerablemente grande de imágenes, con lo cual se pudo demostrar la importancia que tiene la cantidad de datos que se tengan en el entrenamiento de un modelo basado en una red neuronal profunda para conseguir resultados óptimos.

También a la vista de los resultados que se obtuvieron se pudo comprobar la importancia que conlleva la realización de un preprocesado de las imágenes más complejo, puesto que en el caso de utilizar una base de datos ya preprocesada se consiguieron mejorar bastante los mejores resultados que se habían conseguido con las mismas bases de datos preprocesadas en este trabajo.

Además, se probaron las redes de tipo residual que permitieron realizar un mayor número de pruebas sobre este sistema debido a la reducción en cuanto a tiempo de computación en entrenamiento.

Por último, se comprobó la robustez del sistema ante el reconocimiento imágenes de una calidad inferior y en movimiento.

5.2 Líneas futuras de trabajo

En base al presente trabajo realizado se podrían llegar a proponer las siguientes líneas futuras para continuar desarrollando este proyecto:

- Mejora de la fase de preprocesado de las imágenes de las bases de datos para intentar conseguir unos resultados similares a los obtenidos con las bases de datos que se descargaron ya normalizadas con anterioridad.

- Utilización de métodos de optimización basados en optimización bayesiana, que permitan probar otras configuraciones en las arquitecturas de los sistemas implementados, de manera que se pudiera optimizar los hiperparámetros de los sistemas creados en este trabajo.

- Implementación de otros métodos de clasificación para sustituir la métrica por distancia coseno por algún otro y comprobar si la eficiencia del reconocimiento mejora.

- Utilización de métodos usados en sistemas de identificación de locutor en audio para comprobar los resultados que se obtendrían si se utilizará la base de datos YTF en formato vídeo.

- Creación de una aplicación que utilice la cámara para adquirir la imagen de la persona que se quiere comparar con las imágenes de la base de datos, adquiridas previamente, para realizar el reconocimiento de esta persona respecto al resto de sujetos de la base de datos.

Bibliografía

[1] Eva Cristina Andrade Tepán. *Estudio de los principales tipos de redes neuronales y las herramientas para su aplicación.* Febrero de 2013.

[2] Javier Eslava Ríos. *Reconocimiento facial en tiempo real.* Julio de 2013.

[3] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* 2007.

[5] Juan José Montaño Moreno. *Redes Neuronales Artificiales aplicadas al Análisis de Datos.* 2012.

[6] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. *27th International Conference on Machine Learning*, 2010.

[7] Damián Jorge Matich. *Redes Neuronales: Conceptos Básicos y Aplicaciones.* Marzo de 2001.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[9] Matthew A. Turk and Alex P. Pentland. Face recognition using eigenfaces. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1991.

[10] Francesc J. Ferri. *Análisis comparativo de métodos basados en subespacios aplicados al reconocimiento de caras*. Septiembre de 2006.

[11] Karen Simonyan, Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Fisher vector faces in the wild. *British Machine Vision Conference*, 2013.

[12] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deep face: Closing the gap to human-level performance in face verification. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[13] Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. In *NIPS*, 2012.

[14] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[15] Dong Yi, Zhen Lei, and Stan Z. Li. Learning face representation from scratch. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[16] Kaiming He, Xiangyu Zhang, Shaoquing Ren, and Jian Sun. Identity mappings in deep residual networks. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems (NIPS)*, 2014.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning*, 2014.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32th International Conference on Machine Learning*, 2015.

[21] Hieu V. Nguyen and Li Bai. Cosine similarity metric learning for face verification. *Asian Conference on Computer Vision*, 2010.

[22] Weiwei Wu. A novel solution to test face recognition methods on the training data set. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2015.

[23] Chao Xiong, Luoqi Liu, Xiaowei Zhao, Shuicheng Yan, and Tae-Kyun Kim. Convolutional fusion network for face verification in the wild. *IEEE Transactions on Circuits and Systems for Video Technology*, 2015.