UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Exploración de topologías para el reconocimiento de género

TRABAJO FIN DE MÁSTER

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Mario Parreño Lara

Tutor: Roberto Paredes Palacios

2017-2018

Agradecimientos

Me gustaría agradecer a toda la gente que me ha ayudado y apoyado a realizar este trabajo. A Roberto Paredes y Juan Maroñas por supervisar la tesis y aportar ideas a la vez que 'no me ayudaban' para que aprendiera a indagar por mi cuenta. A Juan Sensio por su apoyo a lo largo del curso ya que sin él habría sido más duro. Por último darle las gracias a Laura por su paciencia y motivarme a seguir escribiendo.

Somos lo que hacemos repetidamente.
La excelencia, entonces, no es un acto; es un hábito.

- Aristóteles

Resumen

El campo del Aprendizaje Automático esta en constante cambio gracias a los descubrimientos de nuevos modelos y técnicas que hacen que los resultados obtenidos sean mejores. Entre el gran conjunto de algoritmos que abarca este campo, uno que está despuntando y sobre el que se están centrando la mayoría de las investigaciones son las redes neuronales. Este tipo de modelos han ganado gran popularidad en los últimos años pudiendo encontrar todo tipo de dispositivos que hacen uso de ellos para fines como el entretenimiento o la seguridad. Una de las áreas donde más destaca el uso de estos modelos es la denominada visión por computador, donde el principal recurso son las imágenes.

El principal objetivo de este trabajo consiste en el análisis del estado del arte de la tarea de reconocimiento de género a partir de imágenes faciales, donde además probaremos diferentes modelos de redes neuronales estableciendo así un marco con el que poder comparar las diferentes técnicas utilizadas en dicho estado del arte. Como último objetivo, trataremos de reducir el número de parámetros de los modelos probados con tal de poder hacer uso de estos en dispositivos con menores prestaciones, como los teléfonos móviles, tratando de mantener en la medida de lo posible una buena tasa de acierto en la clasificación del género.

Abstract

The field of Machine Learning is constantly changing thanks to the findings of new models and techniques that make the results obtained to be better. Among the great set of algorithms covered by this field, one that is emerging and on which most of the research is being focused are neural networks. This type of models have gained great popularity in recent years and today we can find all kinds of devices that make use of them for purposes such as entertainment or security. One of the areas where the use of these models stands out the most is the so-called computer vision, where the main resource are images.

The main purpose of this paper is to analyze the state of the art of the task of gender recognition from facial images, where we will also test different models of neural networks establishing a framework to compare with the different techniques used in such state of the art. As our final purpose, we will try to reduce the number of parameters of the tested models in order to use them in devices with lower facilities, such as mobile phones, trying to maintain, as much as possible, a good rate of success in the classification of the gender.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

El objetivo de este trabajo es el de presentar el estudio realizado sobre diferentes modelos de redes neuronales para la tarea de reconocimiento de género. Debemos tener en mente que todos estos experimentos han sido realizados con la idea de tratar de minimizar el número de recursos y optimizar el tiempo de inferencia de los modelos. Esto es debido a que, por una parte, es interesante el estudio del efecto de los recursos computacionales sobre las redes, a la vez que podemos vislumbrar nuestro problema del reconocimiento de género como un problema que podemos ligar al ámbito móvil donde contamos con recursos escasos.

Recientemente han aparecido un abundante número de trabajos que tratan de reducir el tamaño de las redes neuronales mientras que a la vez tratan de mantener la tasa de acierto obtenida por redes mayores, como el entrenamiento profesor-estudiante [1] o las capas lineales estructuradas [2]. Estos modelos reducidos podrían tener ventajas necesitando menos datos y siendo potencialmente más adaptables a los cambios del dominio del objetivo, debido a tener menos parámetros en nuestro modelo.

En la primera parte de este capítulo presentaremos cual es la motivación de nuestro trabajo y cómo es utilizado el reconocimiento de género en aplicaciones de la vida real. A continuación explicaremos cuales son nuestros objetivos y finalmente comentar la estructura de la memoria.

1.1 Motivación

Las imágenes faciales son probablemente la característica biométrica más utilizada por los humanos para realizar el reconocimiento de los individuos. El reconocimiento de género a partir de estas imágenes faciales es una tarea importante en el ámbito de visión por computador ya que muchas aplicaciones dependen de una correcta clasificación del género. Algunos ejemplos de estas aplicaciones son:

- **Sistemas de interacción persona-computador**: Podemos refinar estos sistemas si son capaces de identificar el género del usuario. Estos adquieren así un comportamiento más "humano", siendo capaces de responder, por ejemplo, apropiadamente según el género de la persona que interactúa con la maquina.

- **Marketing**: La publicidad dirigida a ciertos usuarios ante paneles de publicidad puede ser diferente dependiendo del género de la persona que está frente a ellos.

- **Biométricas**: En sistemas biométricos de validación en vez de dado un rostro comparar contra todas las caras de nuestra base de datos, es posible realizar una poda inicial a partir del género y así agilizar la búsqueda.

- **Sistemas de seguridad**: Es posible restringir el acceso a diferentes áreas dependiendo del género del individuo.

Aunque la importancia de asignar correctamente el género de las persona ha quedado claro, cómo realizar esta clasificación utilizando rostros puede ser discutido. En el Capítulo 3 nos centramos en estudiar diferentes técnicas que aparecen en el estado del arte para tratar de solucionar esta tarea, tomando aquellas de la literatura que utilizan las mismas bases de datos y protocolos que emplearemos nosotros en la experimentación. Dado que estas técnicas no utilizan redes neuronales del estado del arte de visión por computador, nos parece una buena idea utilizar estas redes para ver cómo funcionan ante un problema de estas características y ver si éstas consiguen funcionar mejor que las aproximaciones más complejas que requieren a veces cierto conocimiento experto.

Con tal de dar otra vuelta a la tarea y analizar otros factores relevantes para el reconocimiento de género, consideramos que el estudio del efecto de la reducción de los parámetros de las redes así como el tiempo de inferencia, la rapidez con la que somos capaces de tratar un rostro, puede ser interesante y por ello trataremos de construir redes cada vez menores para ver estudiar cómo se comportan. Cabe recordar que la puesta en marcha de este tipo de sistemas suele ir a sistemas embebidos como pueden ser dispositivos móviles donde contamos con recursos más limitados.

1.2 Objetivos

Los dos principales objetivos de nuestro trabajo son: En primer lugar la prueba de diferentes modelos de redes neuronales del estado del arte, probando diferentes técnicas para tratar de mejorar los resultados obtenidos para la tarea de clasificación de género a partir de imágenes faciales en el estado del arte; A su vez estudiaremos el efecto de la reducción de los parámetros de los diferentes modelos centrándonos para ello en el desempeño de las redes. Para ello necesitaremos:

- Estudiar el estado del arte, para así entender los diferentes planteamientos seguidos para la resolución de nuestro problema y tomar aquello que nos sea de provecho.

- Analizar cómo funciona una red neuronales y qué técnicas se emplean sobre estas para así poder comprender más fácilmente cómo se estructuran los diferentes modelos que utilizaremos.

- Por último deberemos poner en práctica lo aprendido para diseñar la experimentación de nuestro trabajo y poder posteriormente analizar los resultados obtenidos y proponer mejoras y trabajos futuros.

1.3 Estructura de la memoria

El resto del contenido del trabajo está organizado como sigue:

- **Capítulo 2**: En este capítulo tratamos de proporcionar al lector una vista más amplia, pero breve, de todos los procesos involucrados en el reconocimiento de género.
- **Capítulo 3**: Este capítulo visita algunos de las técnicas más importantes que son utilizadas con respecto a nuestra tarea.
- **Capítulo 4**: Las redes neuronales son el pilar de nuestro trabajo por lo que les dedicaremos este capítulo para verlas con detenimiento y estudiar las diferentes piezas que las componen.
- **Capítulo 5**: En este capítulos exponemos los recursos utilizados, tanto los referidos a los datos utilizados para nuestros experimentos, como aquellos recursos computacionales utilizados para realizar dichos experimentos.
- **Capítulo 6**: Aquí se presentan y explica la arquitectura de los diferentes modelos utilizados para nuestra tarea.
- **Capítulo 7**: Este capítulo reúne todos los experimentos realizados en cuanto a los diferentes modelos utilizados sobre la tarea de reconocimiento de género.
- **Capítulo 8**: En este último capítulo resumiremos todo el trabajo realizado así como los resultados obtenidos. También propondremos diferentes líneas de trabajo futuro y experimentación.

Capítulo 2

Introducción al Reconocimiento de Género

En general, cuando nos enfrentamos a un problema de reconocimiento de patrones como es el reconocimiento de género, este es dividido en una serie de pasos [4]: la detección del objeto, preprocesado, extracción de características y clasificación. En el siguiente capítulo vamos a proporcionar un breve vistazo a como estos pasos son realizados en cuanto a lo que se refiere a nuestra tarea del reconocimiento de género a través de imágenes faciales.

2.1 Detección facial

En la fase de detección del objeto a clasificar, en nuestro caso los rostros de las personas que aparecen en una imagen dada, la región del rostro debe ser detectada para posteriormente cortar la imagen con tal de capturar únicamente la cara. Para llevar a cabo esta tarea no se tiene ningún tipo de información a priori como puede ser el número de caras en la imagen, el tamaño, color, etc., sino que para cada una de estas hay que determinar su localización y su escala.

Podemos considerar la detección facial como un problema de clasificación donde contamos con dos clases: caras y no caras. Por tanto los algoritmos que tratan de resolver este problema necesitan muestras tanto de caras como de no caras. Existen muchos métodos para el problema de la detección facial [5] [6], pero por su simplicidad y su amplia utilización nos centraremos en el algoritmo de Viola-Jones [7].

P. Viola y M. Jones introdujeron la utilización de características de Haar, filtros rectangulares muy sencillos, para realizar la tarea de detección de forma rápida. En la Fig. 2.1 podemos ver un ejemplo de estos rectángulos. La suma de los píxeles que caen sobre la parte blanca de los rectángulos es sumada y posteriormente restada a la suma de los píxeles de los rectángulos negros, obteniendo un valor para esa característica de Haar. En la práctica se utiliza la imagen integral para calcular la suma de los rectángulos. Finalmente, para hallar los clasificadores a utilizar, el sistema es entrenado con AdaBoost ponderando los clasificadores en relación al error que estos obtienen. Para la clasificación se realiza un pase en cascada por las características de Haar obtenidas descartando las imágenes de no

caras.

Figura 2.1: Las dos mejores características seleccionadas por AdaBoost. En la fila inferior vemos como se superponen sobre un imagen típica de entrenamiento. Imagen extraída de [7].

Suele ser usual en la fase de detección el testeo de diferentes escalas y orientaciones. Por último, los algoritmos procesan la imagen utilizando una ventana de un tamaño prefijado y por lo tanto una misma cara varias puede ser identificada varias veces, para solucionar esto se realiza una fusión y eliminación de las hipótesis basándose en reglas, como por ejemplo si existen muchas hipótesis con similar centro entonces estas se fusionan. En la Fig. 2.2 se muestra un ejemplo de las operaciones descritas.

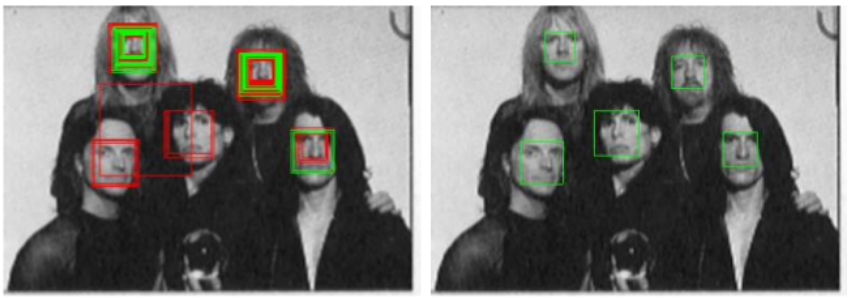

Figura 2.2: Ejemplo proceso detección facial y fusión de hipótesis.

2.2 Preproceso

Tras la obtención de los rostros de la imagen, a estos se les suele aplicar algún tipo de preproceso. Se asume que el detector facial provee una precisa normalización en cuanto a la translación, rotación y escala de la cara, de forma que el principal problema que encontramos es la iluminación de la imagen. Aunque a priori puede parecernos que no podemos extraer mucha información mediante el procesado de la iluminación en las imágenes, la verdad es que puede hacer que los resultados de nuestro clasificador mejoren notablemente. Villegas y Paredes [8] realizan una comparación de diferentes técnicas aplicadas a la identificación de individuos donde muestran una gran mejora enfrente a los resultados obtenidos sin normalizar las imágenes.

En el siguiente trabajo únicamente vamos a realizar la normalización de los rangos de valores cómo se explica en la Subsección 4.3.2. Para más información acerca de la normalización de luz, recomendamos la lectura del artículo previamente citado.

2.3 Extracción de características

Otra elemento de gran importancia en el proceso de reconocimiento de género es la extracción de aquellas características que nos permitan discriminar entre aquello que queremos clasificar. En referencia al problema que tratamos de extracción de características faciales, podemos dividir los métodos para la clasificación de género a partir del rostro en aquellos basados en:

- **Geometría**: Se fundamenta en el uso de la medición de puntos faciales. La relación geométrica entre diferentes puntos faciales pueden proveer características discriminatorias pero otra información importante puede ser desechada. Otro problema que presenta esta técnica es que el proceso de extracción de puntos faciales debe ser preciso para proporcionar buenos resultados.
- **Aspecto**: Es posible realizar diferentes operaciones o transformaciones sobre los píxeles de la imagen para generar características. Esta técnica puede ser aplicada a nivel global de la imagen o localmente, donde la imagen suele ser dividida en diferentes regiones tratando de separar elementos como los ojos, boca o nariz.

Tradicionalmente los métodos basados en el aspecto o apariencia han sido los que mejores resultados han proporcionado, debido quizás al conocimiento experto de gente que es capaz de extraer información más robusta que el uso de los píxeles simplemente. En contraste, los modelos discriminativos de aprendizaje profundo tratan de "descubrir" automáticamente estas representaciones de las imágenes para extraer las características a la vez que se ocupan de realizar el proceso de clasificación.

2.4 Clasificación

En último lugar, un clasificador es entrenado y validado con una base de datos dada. El reconocimiento de género es un problema donde los individuos deben ser clasificados como hombres o mujeres, por lo que se emplea un clasificador binario. Algunos de los clasificadores que han sido y siguen siendo, sin tener en cuenta los modelos de redes neuronales, ampliamente utilizados para la tarea de reconocimiento de genero son: Máquinas de Vectores Soporte, Adaboost, clasificadores Bayesianos, el vecino más cercano y redes neuronales.

Capítulo 3

Estado del Arte

Existe un gran número de aproximaciones para la creación de sistemas capaces de reconocer el género [3], donde podemos clasificar dichas aproximaciones en dos tipos: unos primeros sistemas que utilizan características "artesanales" obtenidas con un método de extracción de características en concreto para cada caso, y otros métodos que utilizan redes neuronales para la extracción de características y la predicción del género.

Mientras que en el siguiente trabajo vamos a utilizar las redes neuronales para nuestra tarea, en el siguiente capítulo veremos algunas de las aproximaciones que encontramos en el estado del arte para el reconocimiento de género que no utilizan estos tipos de modelos para extraer características. Con esto adquiriremos una visión más amplia de las diferentes técnicas utilizadas y que podrían utilizarse conjuntamente con las redes neuronales sin problemas.

3.1 Local Deep Neural Networks

La primera idea que vamos a estudiar es una técnica presentada en [9]. Aquí se propone la utilización de ventanas con información de características locales, denominados como *patches*, para la clasificación de los rostros dados. Se trata de una idea introducida previamente [10] pero que es particularizada para el problema de reconocimiento de género.

En primer lugar se define un marco formal para la clasificación de las imágenes en base a las características locales extraídas. Siendo c la etiqueta de clase de una imagen dada $\mathbf{x}$, de dicha imagen son extraídas F características locales asumiendo que estas tienen información incompleta pero relevante para conocer la verdadera clase de $\mathbf{x}$. De acuerdo con esta idea y tras una serie de asunciones de las que se hacen servir los autores, estos llegan a que podemos calcular la probabilidad de que una imagen $\mathbf{x}$ pertenezca a una clase c como:

$$p(c|\mathbf{x}) := \sum_{i=1}^{F} \alpha_i p(c|\mathbf{x}^{[i]}) = \sum_{i=1}^{F} \alpha_i p_c^{[i]} \tag{3.1}$$

Donde $p_c^{[i]}$ es la probabilidad de que la característica $\mathbf{x}^{[i]}$ prediga la clase global c asociada a la imagen $\mathbf{x}$. Se propondrán dos modelos para la clasificación final de una entrada $\mathbf{x}$:

- **Suma posteriors**: Se confía en que α_i sea capaz de dar poder discriminatorio a las características locales para darles mayor o menor influencia en la decisión de la clase final. Para ello se utiliza la regla de decisión de Bayes eligiendo la clase a partir de la máxima suma ponderada de las posteriors locales:

$$\mathbf{x} \rightarrow c(\mathbf{x}) = \arg\max_c p(c|\mathbf{x}) = \arg\max_c \sum_{i=1}^{F} \alpha_i p_c^{[i]} \tag{3.2}$$

- **Votación**: Para este esquema, cada característica local elige una clase de acuerdo a su máxima posterior local. Tras esto la clase más votada por todas las características locales es la seleccionada como decisión final:

$$\mathbf{x} \rightarrow c(\mathbf{x}) = \sum_{i=1}^{F} \delta(c, \arg\max_c p_c^{[i]}) \tag{3.3}$$

  De esta forma no tenemos en cuenta α_i ya que solo nos interesa el voto como 0 o 1 de la característica local como nos indica la delta de Kronecker, donde $\delta(c_i, c) = 1$ si $c_i = c$ o 0 en otro caso.

Definido el marco para la clasificación y dado que el problema se va a partir en la clasificación de los diferentes *patches*, se justifica la utilización de Redes Neuronales Profundas (más conocidas del inglés *Deep Neural Networks* o por su abreviatura DNN) como una medida mucho más eficiente y escalable que el método clásico de búsqueda de vecinos, ya que a diferencia de este último, no será necesario almacenar todas las características locales extraídas del entrenamiento para clasificar un nuevo objeto si no que únicamente necesitaremos una pasada por los pesos de la red para poder clasificar.

Otra parte vital para el sistema es la extracción y selección de las características locales. Inspirados en [11] lo que se hace es, dada una imagen se enfatizan los bordes y traslaciones mediante un filtro de Sobel. Después de esto se aplica un filtro de paso bajo y se binariza la imagen utilizando cierto umbral. Finalmente se extraen los *patches* centrados en cada píxel activo en la máscara binaria, eliminando aquellos que caen fuera de la imagen. Cada *patch* es normalizado para tener media 0 y varianza 1. El proceso expuesto en el *paper* original puede verse en la Fig. 3.1 .

En la parte de resultados y experimentación se proponen varias pruebas como la búsqueda del tamaño de los *patches* o el número de capas ocultas de la Red Neuronal, pero las conclusiones más destacables son que el esquema de suma de posteriors frente al esquema de votación funciona un poco mejor, siendo una mejora insignificante. También se añade que la inclusión de la información de cuáles son las coordenadas de extracción del *patch* no mejoran los resultados y lo

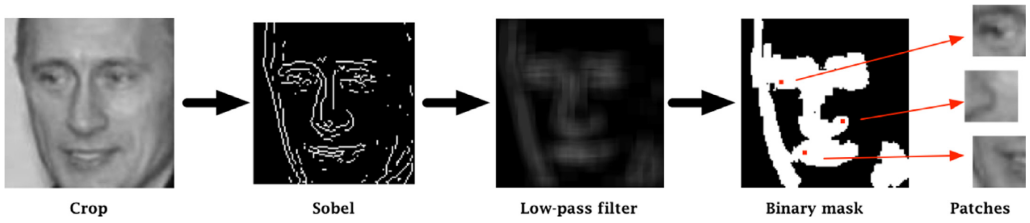

Figura 3.1: Representación gráfica de la extracción de patches en [9].

más notable es que las diferencias de resultados entre los modelos con una capa oculta contra los de dos o más capas son bastante peores achacado al bajo poder representativo del caso de uso de una única capa.

3.2 Local Phase Quantization y Imágenes Monogénicas

Local Phase Quantization (LPQ) [12] es un método de extracción de características que es utilizado para la tarea de reconocimiento de género [14]. Se basa en la intensidad y en el componente *monogénico* [13] de las imágenes para representar las características de estas. En primer lugar, tres componentes *monogénicos* son generados aplicando un filtro *monogénico* en la imagen de entrada que descompone su intensidad en la amplitud $\mathcal{A}$, orientación θ, fase ϕ, paso de banda h, h_x y h_y. Un ejemplo de la aplicación de este filtro lo tenemos en la Fig. 3.2.

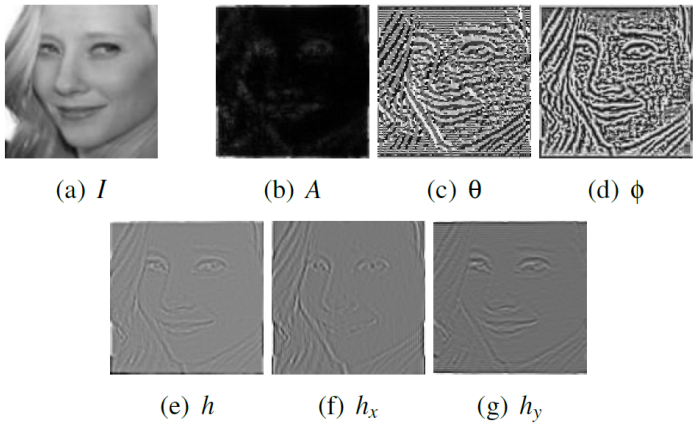

Figura 3.2: Una imagen de entrada (a) y sus imágenes *monogénicas*: amplitud (b), orientación (c), fase (d), y pasos bajos (e-g). Imagen extraída de [14].

A continuación cuatro operadores LPQ, que se basan en la utilización de la transformada rápida de Fourier, son aplicados sobre la intensidad de la imagen de entrada y sobre las tres imágenes de paso bajo h, h_x y h_y, dando como resultado cuatro nuevas imágenes. Tras esto, estas imágenes fruto de aplicar los operadores LPQ son divididas en subregiones rectangulares de forma que estas no se superpongan y para cada subregión se calcula su histograma correspondiente. A continuación los histogramas de una misma imagen son concatenados para posteriormente combinarlos todos y obtener la representación final de la imagen para así entrenar un clasificador, como para el caso de estudio [13] una máquina

de vectores soporte. La Fig. 3.3 muestra el proceso de extracción de características completo.

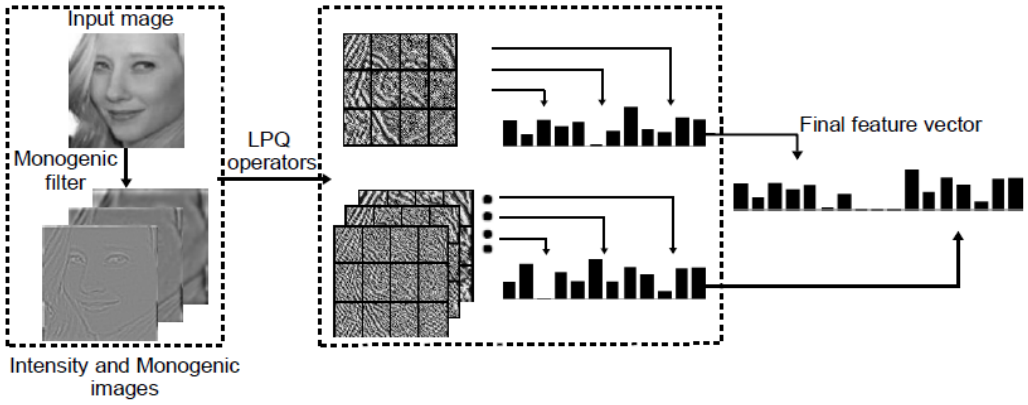

Figura 3.3: Proceso de extracción de características a partir de imágenes *monogenicas* y características LPQ. Imagen extraída de [14].

3.3 Descriptors and regions of interest fusion

En la mayoría de las técnicas que se han analizado, la clasificación del género es realizada a partir de imágenes que incluyen únicamente detalles faciales y no son incluidos otros aspectos anatómicos. Con esto en cuenta se plantea [15] la utilización de imágenes que tomen más contexto de los rostros, como los hombros, el pelo o parte del pecho, con la idea de que esto además aumentará el poder de generalización de los modelos creados.

A partir de las diferentes imágenes, tomando más o menos contexto, se propone el uso de métodos de representación de descriptores locales a nivel de píxel y la realización de pruebas donde se entrena con una base de datos y se testea con otra, para poder ver el poder de generalización que comentamos. Los métodos de representación son los siguientes:

- ***Histograms of oriented gradients* (HOG)**: La idea que hay detrás de esta técnica [16] es que la apariencia y forma local del objeto dentro de una imagen puede ser descrita por la distribución de los gradientes de la intensidad. Para esto, en la práctica la imagen es dividida en celdas donde para cada una calculamos sus correspondientes histogramas para formar el descriptor. Además, para reducir la influencia de la iluminación, cada histograma es normalizado usando su vecindario, llamado bloque, donde se utiliza el valor calculado para normalizar todas las celdas dentro de ese mismo bloque.

- ***Local binary patterns* (LBP)**: [17] Para cada píxel se compara con sus 8 vecinos donde, si el píxel central es mayor al vecino escribiremos un 0, en caso contrario un 1. Con esto obtendremos al final un número binario de 8 dígitos que usualmente es convertido a decimal. Con todos estos números formamos un histograma con la frecuencia de cada número obtenido para opcionalmente normalizar dicho histograma y finalmente obtener un vector

de características. El problema que conlleva este método para el reconocimiento de género es que implica la pérdida de información espacial, por lo que alternativamente, como se propone en [18], se divide la imagen en pequeñas regiones y aplicamos el operador LBP por cada una de ellas, para finalmente concatenar los histogramas.

- ***Neighbour Intersection - Local binary patterns*** **(NI-LBP)**: Se trata de una variante [19] del método LBP descrito donde para generar el número binario, en vez de fijarnos en el píxel central, nos centramos en la media de los píxeles.
- ***Local Oriented Statistics Information Booster*** **(LOSIB)**: Potenciador de LBP, calcula información estadística local orientada en la imagen completa [21].
- ***Local salient patterns*** **(LSP)**: Se centra en localización de las mayores diferencias de los píxeles vecinos a uno dado [20].

Dadas las diferentes representaciones se utiliza una Máquina de Vectores Soporte para la clasificación y viendo que los resultados distan de los obtenidos por el estado del arte, y dada la naturaleza de los diferentes descriptores utilizados, se propone la combinación de estos para juntar la información complementaria que proporciona cada uno y tratar de mejorar los resultados. Se utilizan imágenes tanto faciales (F), que habían obtenido los mejores resultados para los descriptores simples, como imágenes con el contexto expuesto (HS), donde primero se cogen la salida de la máquina de vectores soporte para varios descriptores y se introduce como entrada a otra (ejemplo en la Fig. 3.4).

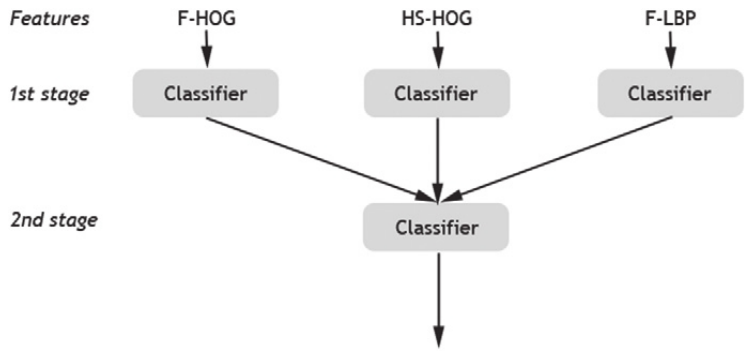

Figura 3.4: Ilustración del apilamiento de clasificadores para su combinación en [15].

Así, probando a insertar más o menos descriptores, los autores llegan a superar los resultados del estado del arte para la base de datos GROUPS.

3.4 Píxeles y Gabor jets

En la literatura, uno de los métodos más utilizados para establecer una base de resultados que superar son los píxeles. Para ello se toman todos los píxeles de la imagen y utilizando diferentes clasificadores o combinándolos se trata de discernir el sexo del rostro que aparece en dicha imagen. Para esta técnica se han dado resultados aceptables que nos sirven como referente para nuestras aproximaciones.

En este último artículo [33] que vamos a estudiar sobre el estado del arte del reconocimiento de género se realiza una comparación entre los resultados obtenidos por los píxeles, los previamente vistos LBPs y los filtros de Gabor (también conocidos del inglés como Gabor jets). El filtro de Gabor [35] es un filtro lineal que básicamente analiza donde hay contenido de frecuencia en la imagen en una dirección determinada. Su respuesta de impulso es una función sinusoidal multiplicada por una función gaussiana, donde por ejemplo, los autores han dividido la imagen con una rejilla uniforme de dimensión 10x10 y para cada reja han calculado el módulo del resultado obtenido por 40 filtros de Gabor complejos, utilizando 5 frecuencias y 8 direcciones.

Tras las pruebas en diferentes bases de datos con los métodos expuestos, se concluye que los filtros de Gabor y los LBPs proporcionan resultados similares y que estos son significativamente mejores que los obtenidos por representaciones utilizando únicamente los píxeles.

3.5 Resultados obtenidos

Explicadas brevemente las técnicas utilizadas en el estado del arte debemos presentar cuales han sido los resultados concretos, en este caso las tasas de acierto para la tarea de reconocimiento de género en ambas bases de datos, para poder comparar nuestros resultados posteriormente. En la Tabla 3.1 mostramos únicamente aquellos resultados en los que las técnicas han sido utilizadas bajo las mismas condiciones, básicamente las mismas particiones de datos y forma de validación. Esto nos proporciona un marco en el que trabajar y una base solida donde comparar de forma veraz nuestros resultados.

Referencia	**Base de datos**	**Acierto (%)**
[33]	LFW	97.23
[14]	LFW	97.00
[9]	LFW	96.25
[9]	GROUPS	91.59
[14]	GROUPS	91.58
[33]	GROUPS	86.61
[34]	GROUPS	91.59

Tabla 3.1: Tasas de acierto obtenidas por las técnicas del estado del para la tarea de reconocimiento de género. Tanto para la base de datos de LFW cómo GROUPS.

Capítulo 4

Redes Neuronales y sus Elementos

En el siguiente capítulo vamos a tratar uno de los modelos que más importancia está tomando desde hace unos años en el campo del aprendizaje automático, las redes neuronales. Estas no son algo nuevo en este campo, pero no fue hasta 2012 cuando cobraron realmente importancia dada el aumento en la tasa de acierto [22] producido por estas en la competición sobre la base de datos ImageNet [1]. Estos resultados demostraron su gran potencial y supuso un punto de inflexión desde el cual no se han parado de desarrollar teorías y procedimiento para tratar de mejorar su funcionamiento y rendimiento.

Nuestro trabajo realizado gira en torno a estas y por ello es necesario estudiar como funcionan asi como ver los elementos que las componen, tratar de mejorar su funcionamiento y finalmente cómo podemos potenciarlas a partir de distintos operadores.

4.1 Redes Neuronales

Las redes neuronales son una proyección algebraica desde una espacio vectorial de entrada a otro de salida, ambos de dimensionalidad arbitraria. Estas conforman una estructura formada por lo que se denominan neuronas, que se interconectan y mandan información entre ellas pudiendo interpretar la serie de operaciones que realizan como:

$$t = f(x);\ x \in \mathbb{R}^k,\ t \in \mathbb{R}^{k'},\ k, k' \geq 1 \tag{4.1}$$

Donde la función que modela la red es de la forma $f : \mathbb{R}^k \to \mathbb{R}^{k'}$. Esta función es interpretada en diferentes pasos donde vamos atravesando una serie de capas ocultas, véase la Fig. 4.1, que van desde la entrada hasta la salida y que están constituidas por espacios vectoriales de dimensionalidad arbitraria. Así, la función f es la concatenación de todas estas capas que forman la red.

[1] Competición conocida en el ámbito de visión por computador por su dificultada debida al gran número de clases (1000) a donde poder clasificar. `www.image-net.org/challenges/LSVRC/`

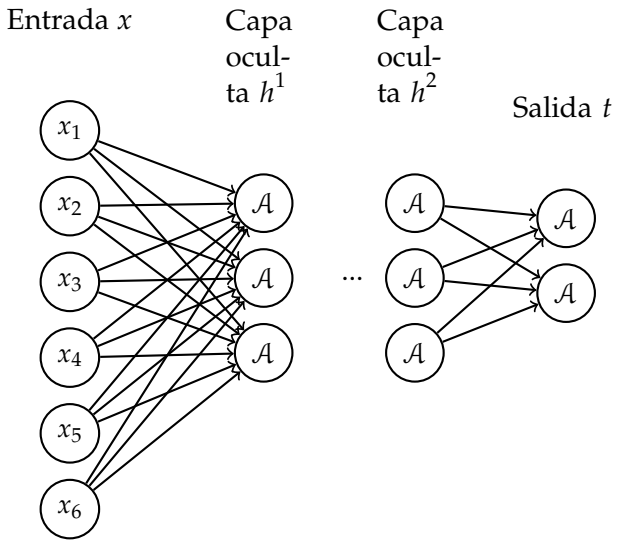

Figura 4.1: Topología de Red Neuronal. Con entrada $x \in \mathbb{R}^6$ y salida $t \in \mathbb{R}^2$ a través de al menos dos espacios vectoriales intermedios $h^1, h^2 \in \mathbb{R}^3$.

Matemáticamente, el resultado que computa cada neurona es una combinación lineal de las la salida de las neuronas previas y la aplicación de una operación no lineal a esa combinación. Con esto vemos cómo para esta topología, y las que vamos a presentar a lo largo del trabajo, no existen ciclos ya las neuronas de una capa dada no alimentan la entrada de neuronas predecesoras. La notación general para la serie de operaciones que se realizan entre dos capas sería la siguiente:

$$x^{l+1} = \mathcal{A}(W \cdot x^l + b),\ x^l \in \mathbb{R}^z,\ x^{l+1},\ b \in \mathbb{R}^y,\ W \in \mathbb{R}^{z \cdot y},\ z, y, \geq 1 \tag{4.2}$$

Donde $\mathcal{A}$ representa una operación no lineal conocida como función de activación, presentadas en el Apéndice A, W es la matriz de pesos que combina las neuronas de la capa actual con los de la capa anterior, y b el un vector llamado bias de la combinación lineal que suma a cada neurona de la capa actual. Esta matriz de pesos y bias son los parámetros que nuestro modelo debe aprender para la función f.

Para finalizar cabe mencionar que los parámetros de las capas ocultas así como la serie de funciones no lineales que aplicamos funcionan como extractores de características para la red, con la finalidad de que estas sean capaces de extraer los aspectos relevantes de los objetos para poder discriminarlos. Es en la última capa de nuestra red, t, donde representamos el número de clases de nuestro problema y a través de las distintas funciones, como típicamente la función softmax (véase el Apéndice A), somos capaces de tener una salida normalizada $t_i \in [0, 1]$ donde $\sum_{i=1}^{k'} t_i = 1$, que puede ser interpretado como la probabilidad posterior para cada una de las clases.

4.2 Optimización

El objetivo principal que tratamos de conseguir con las redes neuronales, al igual que con los diferentes algoritmos de aprendizaje automático, consiste en la mi-

nimización del error que comete nuestro modelo a la hora de realizar una tarea como puede ser la clasificación de imágenes en taras de visión por computador. Ahora bien, los pesos de las redes neuronales suelen ser inicializados de forma aleatoria y si infiriéramos sobre estos sin tratar de optimizarlos sería improbable que las predicciones obtenidas fueran acertadas.

En la siguiente sección trataremos algunos de los métodos más utilizados a la hora de tratar de optimizar los pesos de nuestra red, y por lo tanto su funcionamiento.

4.2.1 Descenso por gradiente estocástico y función de coste

El algoritmo más popular para la aproximación del óptimo, ya sea para hallar el máximo o el mínimo, es el llamado descenso por gradiente estocástico o SGD (por sus siglas del inglés Stochastic Gradient Descent).

Podemos definir SGD como se muestra en la Fig. 4.2. Además, en redes neuronales implementamos las derivadas en SGD mediante retropropagación del error [24].

entrada : $\mathcal{P} = \{P_1, P_2, ..., P_M\}, \mathcal{X} = \{X_1, X_2, ..., X_N\}, \alpha, C, Iteraciones$;
salida : $\mathcal{P}' = \{P'_1, P'_2, ..., P'_M\}$;
Require $\alpha \geq 0$;
Inicialización variables;
$it \leftarrow 0$;
while $it \leq Iteraciones$ **do**
  for $p \in \mathcal{P}$ **do**
    $p' = p - \alpha \cdot \frac{\partial C}{\partial p}$;
  end
  $\mathcal{P} = \mathcal{P}'$;
  it += 1;
end

Figura 4.2: Implementación estándar del algoritmo de descenso por gradiente estocástico.

Este algoritmo de retropropagación del error tiene como finalidad minimizar las funciones de coste utilizadas en las redes neuronales. Existen diferentes funciones de coste que podemos utilizar, donde principalmente elegiremos una u otra en función del tipo de problema frente al que nos encontremos. Así, fijándonos en la salida s deseada, podemos encontrarnos ante problemas de clasificación, regresión... Siendo el problema abordado en este trabajo un problema de clasificación en dos clases: hombre o mujer.

Ante un problema de clasificación en M clases (dimensionalidad de salida M), definimos esta serie de funciones sobre un conjunto de pares $\mathcal{X} = \{(X_1, S_1), (X_2, S_2), \ldots, (X_N, S_N)\}$. Donde ante nuestro problema decidimos utilizar la entropía cruzada debido a que esta función de coste se define para que la salida s de nuestra red se encuen-

tre en un rango de 0 a 1, y además que la suma de todas estas salida de como resultado 1. Se define como:

$$C = -\frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{N}\hat{S}_i(m) * log(f(X_i(m))) + (1-\hat{S}_i(m)) * log(1-f(X_i(m))) \tag{4.3}$$

4.2.2 Algoritmo Adam

A lo largo de la historia del aprendizaje profundo, muchos investigadores han propuesto algoritmos de optimización viendo que funcionaban bien para unos cuantos problemas, sin llegar a generalizar y funcionar tan bien como el algoritmo SGD. Esto causa cierto escepticismo en la comunidad de investigadores y desarrolladores acerca de estos nuevos algoritmos de optimización. Adam [25] es una de esas excepciones que se han seguido utilizando con cierta frecuencia.

Adam es definido como se presenta en la Fig. 4.3, donde como apuntan los autores, en etapas tempranas el algoritmo esta segado hacia cero y para contrarrestar dicho efecto se propone que:

$$\hat{m}_t = \frac{m_t}{-\beta_1^t} \tag{4.4}$$

$$\hat{v}_t = \frac{v_t}{-\beta_1^t} \tag{4.5}$$

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Figura 4.3: Implementación del algoritmo Adam presentada en [25].

4.2.3 Momentum

Existe una variante del descenso por gradiente que introduce lo que es denominado como *momentum* [24]. La idea básica que hay detrás es el calculo de un promedio ponderado exponencialmente de los gradientes para posteriormente utilizarlo a la hora de actualizar los pesos de la red. Con esto se trata de acelerar el proceso de optimización de los pesos y de no caer en óptimos locales.

En la Fig. 4.4 mostramos la incorporación del *momentum* en el algoritmo SGD presentado en la Fig. 4.2. Si comparamos los algoritmos vemos como estos difieren ligeramente. Al introducir *momentum* nos mantenemos avanzando más rápidamente en la dirección del gradiente que calculamos. Con esto conseguimos mayor velocidad en el descenso lo que nos previene de caer en mínimos locales y mantener la dirección principal del proceso de minimización, al tener en cuenta los pasos previos.

entrada : $\mathcal{P} = \{P_1, P_2, ..., P_M\}, \mathcal{X} = \{X_1, X_2, ..., X_N\}, \alpha, C, m, Iteraciones$;
salida : $\mathcal{P}' = \{P'_1, P'_2, ..., P'_M\}$;
Require $\alpha \geq 0$ *y* $0 \leq m \leq$;
Inicialización variables;
$it \leftarrow 0$;
$allM \leftarrow 0 \in \mathcal{R}^M$;
while $it \leq Iteraciones$ **do**
  for $p \in \mathcal{P}$ **do**
    $p' = p - \alpha \cdot \frac{\partial C}{\partial p} + m \cdot allM_p$;
    $allM_p = -\alpha \cdot \frac{\partial C}{\partial p}$;
  end
  $\mathcal{P} = \mathcal{P}'$;
  it += 1;
end

Figura 4.4: Implementación estándar del algoritmo de descenso por gradiente estocástico con *momentum*.

4.2.4 Mini-batch gradient descent

Con la implementación de los algoritmos de optimización vistos anteriormente, como por ejemplo SGD, debemos procesar toda nuestra base de datos antes de poder tomar un pequeño paso en el descenso por gradiente. El problema es que cuando contamos con una base de datos grande esto puede ralentizar el proceso de aprendizaje, y por ello se introduce la técnica del *mini-bacth gradient descent*.

Resulta que podemos agilizar estos algoritmos de aprendizaje si permitimos que se realicen modificaciones de los parámetros de nuestra red antes de tener que procesar todo el conjunto de datos. Para ello dividiremos los datos de nuestro entrenamiento en pequeños subconjuntos que utilizaremos para calcular el error del modelo y actualizar los coeficientes del mismo. Con esto, la frecuencia de actualización de los parámetros será más alta y nos permitirá una mayor convergencia y prevenir en cierta medida mínimos locales.

4.2.5 Learning rate decay

La variación del valor del factor de aprendizaje o *learning rate* introducido en los algoritmos de optimización de parámetros previos, ha demostrado en muchos casos una mejora en los resultados obtenidos. Dicha variación típicamente consiste en la reducción controlada del *learning rate* que se puede realizar de diver-

sas formas: reducir el valor cuando observamos que tras una serie de iteraciones el número de errores no varía significativamente, esperar un cierto número de iteraciones predeterminado, seguir una función decreciente... Esto nos ayuda a acelerar el proceso en una primera instancia, dando saltos por la función de coste mayores, y a converger hacia el mínimo conforme vayamos avanzando ya que el *learning rate* será menor.

4.3 Regularización

Las redes neuronales son modelos muy potentes que nos permiten aprender datos con mayor o menor facilidad. Es por esto que la iteración continua sobre los mismos datos puede llevarnos a aprenderlos y no ser capaces de generalizar bien, lo que nos llevaría a malos resultados en el test. Este problema es conocido como *overfitting* y existen una serie de técnicas que nos ayudan a evitarlo. En la siguiente sección trataremos algunas de estas técnicas más utilizadas.

4.3.1 Dropout

El dropout [23] es una potente técnica de regularización que tiene como idea subyacente la combinación de distintos modelos para nuestra tarea. Debido a que entrenar varios modelos es costoso, el dropout sugiere simular este efecto.

La idea básica de esta técnica es la inutilización de algunas de las neuronas de la red durante la fase de entrenamiento. Para ello contaremos con cierta probabilidad de poner a cero la salida de dichas neuronas. Esto se lleva a cabo esencialmente mediante una variable aleatoria $r_j^l \sim Bernoulli(p)$ controla la activación de la neurona j de la capa l, de forma que su salida s quedaría como $\hat{s}_j^l = r_j s_j^l$. Una idea más intuitiva de esto podemos verla en la Fig. 4.5.

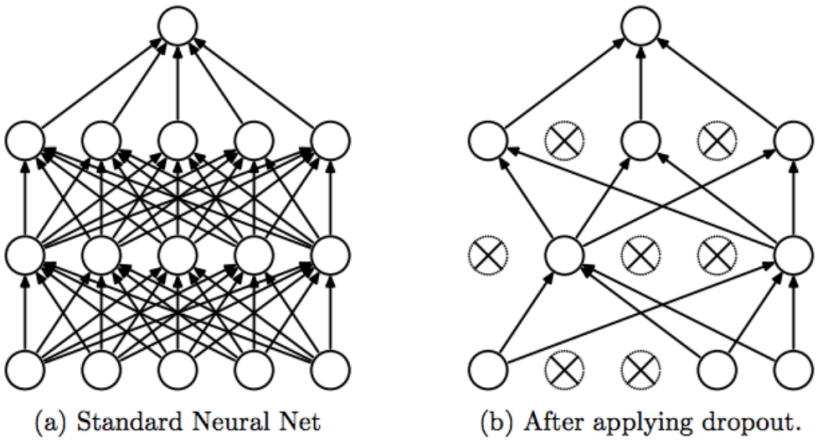

Figura 4.5: Representación técnica de regularización dropout [23].

Recordar que la cancelación de cada neurona es controlada en cada pase de datos que se realiza sobre la red, teniendo así cada vez unas neuronas u otras inutilizadas. Posteriormente en la fase de test los pesos son multiplicados por la probabi-

lidad de dropout introducida siendo la salida en tiempo de test la misma salida esperada en fase de entrenamiento.

4.3.2 Data Augmentation

La mayoría de los problemas relacionados con visión por computador podrían utilizar más datos para el entrenamiento, *Data Augmentation* es una de las técnicas más simples pero potentes para llevar esto a cabo. Este uso de más datos sobre nuestra red supone tanto un rendimiento mayor como un incremento en el poder de generalización.

La técnica consiste en aplicar diferentes transformaciones a los datos de entrada para generar más datos "nuevos" a partir de ellos que sean parecidos pero no exactamente iguales. Esto se realiza porque en la práctica contar con más datos ayuda al hacer que la red tenga en cuenta también pequeñas modificaciones de los objetos y pueda actuar mejor en escenarios diferentes, es decir, generalizar mejor. Algunas de las transformaciones típicas para tareas de visión por computador son:

- **Ruido**: Generalmente añadir ruido no correlacionado, como el caso de ruido gaussiano, es una buena forma de generalización. Este debe tener correlación con la estructura de los datos de los datos de tal manera que no debemos añadir ruido con media 0 si los datos no están centrados en 0.
- **Recortar**: Tomar partes aleatorias de la imagen con ciertas dimensiones en vez de la imágen completa.
- **Voltear**: Consiste en dar la vuelta a la imagen, tanto verticalmente como horizontalmente.
- **Desplazar**: Trasladamos la imagen mediante la aplicación de transformaciones afines. Un claro ejemplo de este método sería la rotación de las imágenes.

Si nos paramos a pensar, la aplicación de estas transformaciones tiene sentido ya que si solo contamos con rostros que siempre tienen la misma orientación, al llegar una nueva muestra con una orientación por ejemplo volteada, nuestra red podría tener diversas dificultades por no haber visto nunca una muestra así. O si el rostro está parcialmente ocluido o tiene algún tipo de zoom, etc. Un ejemplo de estas transformaciones lo podemos ver en la Fig. 4.6

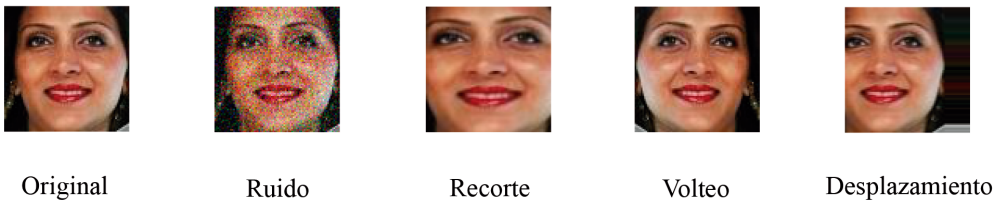

Figura 4.6: Ejemplos transformaciones *data augmentation*.

Por último, una técnica que es usual encontrar es la aplicación de algún tipo de normalización a los datos. Para nuestros experimentos probaremos la utilización de dos de ellos. El primero de ellos consistirá en hacer que nuestros datos se encuentren en le rango 0 a 1, de tal forma que dado nuestros datos $\mathcal{X} = \{X_1, X_2, ..., X_N\}$, $X_i \in \mathbb{R}^k$ tendremos:

$$X_j(i) = \frac{X_j(i) - min(\mathcal{X})}{max(\mathcal{X}) - min(\mathcal{X})} \tag{4.6}$$

Otra normalización que utilizaremos será para que el rango de nuestros datos se encuentre entre -1 y 1:

$$X_j(i) = \frac{2 \cdot X_j(i) - max(\mathcal{X}) - min(\mathcal{X})}{max(\mathcal{X}) - min(\mathcal{X})} \tag{4.7}$$

La aplicación de estas normalizaciones es debida a dos principales problemas. El primer problema hace referencia en los datos podemos tener rangos muy amplios de valores, pudiendo tener valores muy pequeños y grandes definiendo un objeto. A causa de esto la red podría verse dirigida por aquellas características de mayor valor al realizar los diferentes cálculos, cosa que para tratar de corregirse haría más lento el proceso de entrenamiento.

Por otro lado, el segundo posible problema que paliaríamos con la normalización sería la posibilidad de que nuestros datos de entrenamiento y de test provengan de distribuciones diferentes. En caso de que esto pasara, al inferir por nuestra red las características no serían extraídas de forma similar a conforme hemos entrenado.

4.4 Búsqueda de hiperparámetros

Existen infinidad de tareas diferentes con características intrínsecamente diferentes. Esto hace que tengamos que adaptar cada modelo de aprendizaje profundo a cada tarea en especial y hace tan importante la búsqueda de hiperparámetros. En redes neuronales contamos con un abundante número de parámetros que podemos modificar en cierta medida y conllevarán en la mayoría de los casos una mejora o empeoramiento de los resultados obtenidos. Un ejemplo de estos hiperparámetros podría ser el numero de capas ocultas de nuestra red, la función de activación a utilizar en cada capa o incluso el grado con el que debemos rotar las imágenes para generar muestras ligeramente diferentes.

Algunos de estos parámetros tienen un mayor o menor impacto, o incluso algunos han adquirido valores estándar por lo que rara vez se prueban nuevos valores. La forma clásica de realizar la búsqueda es mediante la definición de un conjunto de parámetros y probar las posibles combinaciones. Esto es conocido como *grid search* y su mayor inconveniente es que es una búsqueda muy lenta. Cómo alternativa encontramos que podemos realizar la búsqueda de forma aleatoria, donde está demostrado [27] que es una mejor aproximación cuando contamos con espacios de características de alta dimensionalidad. Con esta últi-

ma técnica no todos los parámetros definidos son probados ya que definimos una distribución de parámetros de cada tipo a probar y se muestrean aleatoriamente.

4.5 Rectifier Linear Unit

El desvanecimiento de los gradientes es uno de los grandes problemas que las arquitecturas de las redes neuronales más profundas sufrían. Dicho desvanecimiento era causado en parte a la derivada de algunas funciones de activación por lo que estas toman un poder importante en el desempeño de nuestro algoritmo de aprendizaje. Un claro ejemplo es la función de activación sigmoide, la cual tiene gradiente 0 en la parte de saturación de la función y poco gradiente en el resto.

La *Rectifier Linear Unit* o ReLu [26] es una función de activación que se caracteriza por valer su gradiente únicamente cero o uno, de forma que supone una mejora en el rendimiento de las arquitecturas más profundas debido a que ya no sufriremos del desvanecimiento del gradiente. Sin embargo hay que tener cuidado ya que también existe el peligro de que la activación de muchas neuronas sea cero y sean propagados junto a su derivada, llevándonos a no aprender nada. Se define como: $ReLU(x) = max(0, x)$.

4.6 Batch Normalization

En el auge del aprendizaje profundo una de las ideas que mayor importancia ha tenido ha sido el algoritmo del *batch normalization* [28]. El entrenamiento de las redes neuronales, sobre todo de aquellas más profundas, es difícil en el sentido de que la distribución de la entrada a cada capa va cambiando a lo largo del entrenamiento en la medida de cómo los pesos de la capa anterior van evolucionando. Debido a esto el proceso de entrenamiento se hace lento ya que nos vemos obligados a utilizar un *learning rate* pequeño para tener cuidado sobre la variación de los pesos aprendidos, lo que los autores denominan como "*internal covariate shift*".

Normalizando simplemente la entrada de cada capa podríamos alterar lo que representa de cada una de estas, así los autores propusieron la utilización de dos parámetros β y γ en las capas que fuera necesario. Para asegurarse de esto las transformaciones que se utilizan en la red pueden representar la transformación identidad, donde para cada activación $x^{(k)}$ tenemos:

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)} \tag{4.8}$$

Donde estos parámetros son aprendidos junto a los pesos de la red neuronal. En la práctica, la tarea de entrenamiento es realizada *mini-batches* cómo vemos en la Subsección 4.2.4 y la tarea de aprendizaje se realiza como se describe en la Fig. 4.7.

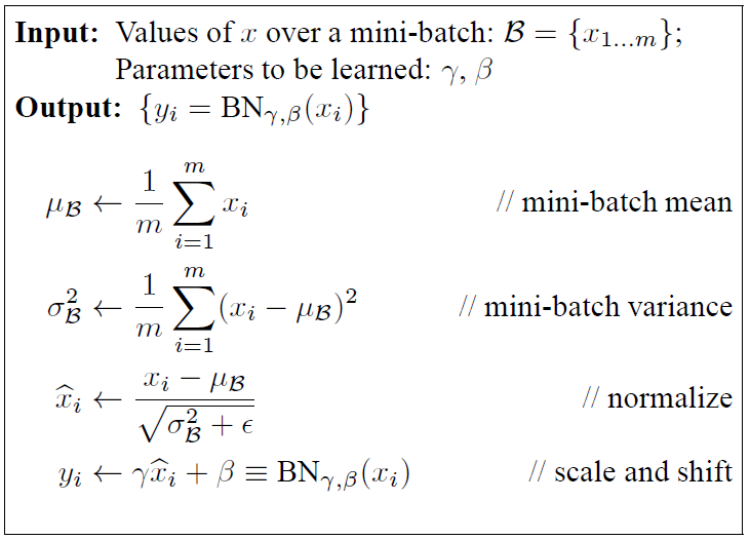

Figura 4.7: Transformación aplicada a la activación x sobre un *mini-batch* por el algoritmo de Batch Normalization. Imagen extraída de [28].

4.7 Redes Convolucionales

Las redes neuronales Convolucionales o CNNs (del inglés *Convolutional Neural Networks*) son muy similares a las redes neuronales vistas anteriormente. Están compuestas por neuronas que tratan de aprender unos pesos. Tienen una entrada y una salida con cierta función de pérdida arbitraria... La diferencia es que las cuando hablamos de CNNs se realiza la asunción de que las entradas de la red son imágenes.

Estas surgen debido a que las redes neuronales clásicas no escalan bien a este tipo de problemas. Por ejemplo, imaginemos que nos encontramos ante problema donde queremos tratar imágenes de unas dimensiones que hoy en día podríamos considerar reducidas, 300x300 píxeles a color (lo que implicaría 3 canales de color para el rojo, azul y verde), si en la primera capa oculta utilizáramos únicamente 100 neuronas, el número de parámetros que tendríamos en este primera operación (sin contar los bias) sería de 27 millones. Si deseamos utilizar más capas, el número de parámetros crecería rápidamente y un problema que podríamos encontrar sería el sobreajuste de estos parámetros o *overfitting*.

En la siguiente Sección vamos a estudiar algunas de los operadores más utilizados a la hora de utilizar redes convolucionales y cómo nos ayudan a tratar con espacios de representación tan grandes.

4.7.1 Capa Convolucional

La capa convolucional es el núcleo principal de este tipo de arquitecturas de redes neuronales. Esta toma una serie de mapas de entrada de profundidad M y mediante una serie de operaciones (definidas en 4.9) obtenemos N mapas de salida correspondientes a cada uno de los N operadores convolucionales utilizados. En la Fig. 4.9 podemos observar este proceso, mientras que en la Fig. 4.8 podemos ver en detalle cómo funciona la operación convolucional 4.9.

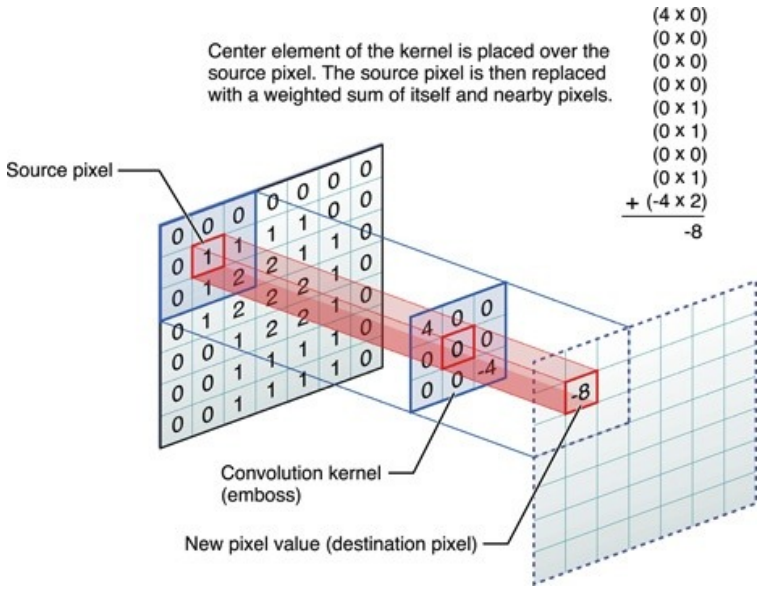

Figura 4.8: Representación gráfica de un operador convolucional en detalle.

$$O_1(r_1, c_1) = \sum_{u,v \in \Omega} I_1(u, v) \cdot K_1(u - r_1, v - c_1) + \cdots + I_M(u, v) \cdot K_M(u - r_1, v - c_1) \quad (4.9)$$

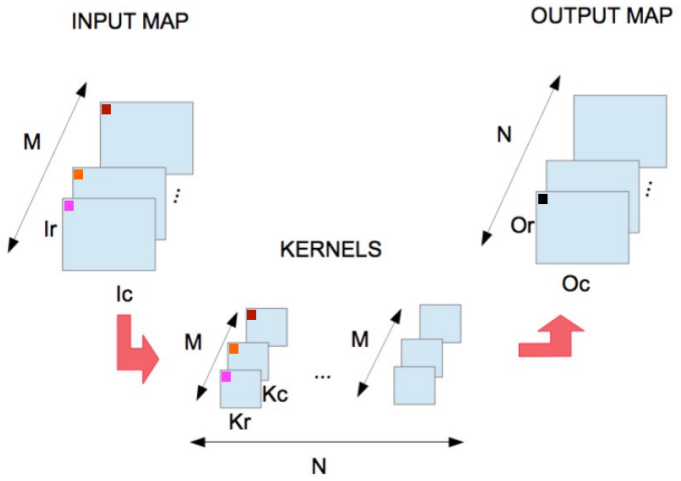

Figura 4.9: Representación gráfica de una capa convolucional.

Este operador nos ayuda a reducir en gran medida el número de parámetros que nuestro modelo debe aprender. Por ejemplo, para el caso que introducíamos con imágenes de 300x300 píxeles a color, en el caso de que utilizáramos 100 mapas de 5x5 píxeles, el número de parámetros sería de tan solo 7600.

4.7.2 Capa de Pooling

Es común insertar periódicamente capas de pooling entre sucesivas convoluciones. Su función es la reducir progresivamente el tamaño de los mapas con los que trabajamos y así reducir el número de parámetros y reducir los cálculo, y por lo tanto controlar el *overfitting*. La capa de pooling trabaja de forma independiente entre los mapas y dependiendo de la forma en la que se combinan los píxeles encontramos dos tipos de capas de pooling principalmente:

- **Average Pooling**: El valor de salida es el promedio de los valores de los píxeles que la ventana de pooling toma.
- **Max Pooling**: El valor de salida es el máximo de los valores de los píxeles que la ventana de pooling toma.

Típicamente cada píxel de salida es calculado sin sobreponer los píxeles vecinos en cada ventana. Podemos ver un ejemplo de un operador de Max Pooling de tamaño 2x2 sin sobreponer los píxeles en la Fig. 4.10. Esta serie de operaciones también nos proporcionan cierta invarianza a las transformaciones afines como las traslaciones.

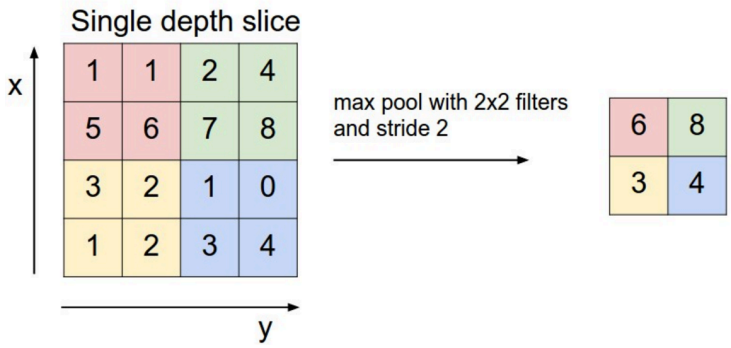

Figura 4.10: Representación gráfica de una operación de Max Pooling.

Este tipo de operador no tiene parámetros a aprender ya que únicamente calcula medias, máximos, etc., de los mapas de entrada.

4.7.3 Padding y Stride

A la hora de construir nuestras CNNs existen dos modificaciones se suele realizar con distinto fin tanto a la operación de la convolución como al pooling, la inserción de *padding* y *stride*.

Cuando aplicamos un operador convolucional de tamaño fxf a unos mapas de entrada de dimensionalidad nxn, si asumimos que el salto por el mapa del operador convolucional es de una en una posición, obtendremos como resultado mapas de menor dimensionalidad, en concreto de (n-f+1)x(n-f+1). Con esto vemos que cada vez que apliquemos nuestro operador convolucional, el tamaño de nuestros mapas se verá reducido y solo podríamos aplicarlo hasta cierto punto donde el tamaño aun fuera lo suficientemente grande. Por otra parte, si estudiamos dicho

operador convolucional, observamos como por ejemplo el píxel de la esquina superior izquierda de un mapa dado solo participará una vez en el cálculo de los mapas de salida, lo que podría conllevar una posible pérdida de información si tenemos en cuenta que un píxel central podría llegar a participar hasta en nueve ocasiones. Para solucionar estos problemas se suele añadir un relleno o *padding* p alrededor de la imagen con ceros de forma que los mapas de salida quedarían cómo (n+2p-f+1)x(n+2p-f+1), lo que nos daría la posibilidad de hacer participar más a los bordes de los mapas de entrada, y obtener mapas de salida del mismo tamaño.

Por otra parte, un parámetro configurable de estos operadores es el tamaño del salto que realizan. El tamaño que suele utilizarse es de un pixel, pero existen modelos que utilizan este salto para disminuir el tamaño de los mapas en vez de utilizar poolings. El tamaño de los mapas de salida finales, incluyendo el *padding* p y el tamaño del *stride* s sería: $\frac{n+2p-f}{s} + 1 \; x \; \frac{n+2p-f}{s} + 1$, redondeando en defecto en caso de que fuera necesario.

4.7.4 Capa Reshape

Como etapa final, las CNNs suelen incluir una parte altamente conectada o *fully connected* típica de las redes neuronales clásicas. Esta parte nos sirve para tomar las características extraídas por las convoluciones y realizar la tarea de la clasificación. Existen otro tipo de redes neuronales convolucionales en el que el objetivo no es el de clasificar, pero recordar que nosotros queremos tratar un problema de clasificación de género y por ello utilizaremos esta capa.

Esta capa toma como entrada un mapa proveniente de la etapa convolucional de la red y la transforma a un vector con la que la parte *fully connected* pueda trabajar. Así, si tenemos N mapas de dimensionalidad MxK, obtendremos un vector de dimensionalidad NxMxK.

4.7.5 Otras capas

Por último vamos a tratar otro tipo de operaciones con las que vamos a trabajar en nuestro trabajo y que son clave en algunos modelos del capítulo siguiente.

Una operación simple que aplicaremos será la de la concatenación. Está simplemente consiste en tomar diferentes mapas con las mismas dimensiones, provenientes por ejemplo de diferentes convoluciones, y los apilará. Por ejemplo, si contamos con una convolución que obtiene N mapas de KxK y otra convolución que obtiene M mapas de KxK, tras concatenarlos obtendremos N+M mapas de KxK.

Otra operación que realizaremos será la de la suma de mapas. Necesitaremos tener el mismo número de mapas en los dos componentes del operador y con la misma dimensionalidad en los mapas, de forma que tendremos de una convolución dada N mapas de KxK y de otra convolución N mapas de KxK y como

resultado obtendremos la suma de dichos mapas en N mapas de KxK.

Para finalizar nombraremos el que podríamos considerar un caso especial del operador convolucional, la convolución con kernel 1x1. Este tipo de convolución cobra sentido cuando la profundidad de los mapas de entrada es mayor a uno, ya que entonces realizará una combinación ponderada de los mapas y sus profundidades, donde la ponderación son parámetros de la red y por lo tanto aprendida.

Capítulo 5

Recursos

El siguiente capítulo está destinado a hablar acerca de los recursos que hemos utilizado para la realización de nuestro trabajo. Por una parte trataremos las bases de datos que se han utilizado para la experimentación y obtención de resultados, mientras que por otro lado describiremos cuales han sido las herramientas software y hardware para llevar a cabo dichos experimentos.

5.1 Bases de datos

Vamos a describir las bases de datos utilizadas en el trabajo así como sus protocolos de evaluación. Estas son públicas para la investigación y ampliamente utilizadas cuando tratamos el problema del reconocimiento de género. Dada la naturaleza de nuestro problema, el reconocimiento de género a través de imágenes faciales, estas bases de datos estarán compuestas únicamente por imágenes de caras.

5.1.1 Labeled Faces in the Wild

Labelled Faces in the Wild (LFW) [29] es una base de datos compuesta por imágenes de caras tomadas sin restricciones, es decir, las caras no tienen porque estar centradas en la imagen dada. Las imágenes de caras son extraídas de Internet a partir de un detector facial basado en el algoritmo de Viola-Jones [7]. La base de datos contiene 13,233 imágenes en color de un total de 5,749 celebridades, siendo 10,246 imágenes de rostros masculinos y 2,977 femeninos. Algunas imágenes de estos rostros se pueden ver en la Fig. 5.1.

Dado que las imágenes no tienen restricciones como mencionamos y con motivo de facilitar la tarea de preproceso y posterior extracción de características, hemos decidido trabajar con una versión de la base de datos que está formada por las mismas imágenes pero alineadas a una pose canónica [30]. Por último hemos utilizado el *benchmark* propuesto por *Facial Image Processing and Analysis group* (FIPA, http://fipa.cs.kit.edu), que divide los datos originales en 5 particiones [31], utilizando 4 particiones para entrenar los modelos y 1 para el test, realizando todas combinaciones posibles y promediando los resultados.

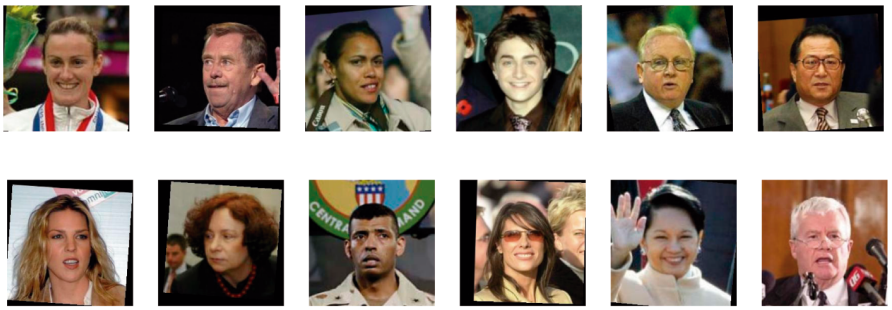

Figura 5.1: Algunos rostros originales de la base de datos LFW.

5.1.2 Groups/Gallagher

La base de datos GROUPS [32] está formada por 5,080 imágenes de grupos de personas, contando con un total de 28,231 caras anotadas con su edad y género correspondiente. Es conocida popularmente como la base de datos de Gallagher's debido a su mayor contribuyente Andrew C. Gallagher. Todas las imágenes han sido tomadas de la web Flickr, coexistiendo imágenes con todo tipo de iluminaciones, poses, expresiones, etc., haciendo que nuestra tarea sea más complicada. En la Fig. 5.2 podemos observar algunas de las imágenes de la base de datos original de *Groups*.

Figura 5.2: Algunas imágenes de la base de datos *Groups*.

Debido a que no existe ningún protocolo estándar para la evaluación del problema de reconocimiento de género, hemos optado por utilizar el propuesto por [33] utilizado en la literatura y en el que nos permitirá comparar resultados para esta base de datos. Este protocolo modifica la base de datos original eliminando las

imágenes de baja resolución (aquellas con menos de 20 píxeles de distancia interocular). Además para paliar con el desbalanceo entre las clases mujer y hombre algunos rostros masculinos han sido eliminados para tener un número semejante de rostros femeninos y masculinos. Con estas modificaciones, obtenemos una base de datos final que dividida equitativamente en 5 particiones [36] con un total de 14760 imágenes. Similar a LFW, se utilizan 4 particiones para entrenamiento y 1 para test, realizando todas las combinaciones posibles y promediando el resultado.

5.1.3 Preproceso

Para la base de datos de LFW se han utilizado imágenes con las caras ya alineados por lo que este paso no ha sido necesario realizarlo. Para agilizar la computación y para eliminar posible información que no aportará mucho a la detección de género como el fondo, hemos realizado un *crop* central a las imágenes donde de unas dimensiones de 250x250 píxeles, pasábamos a 150x150 píxeles, para finalmente reescalar la cara a unas dimensiones computacionalmente más manejables de 80x80 píxeles y que aún mantuvieran la mayor parte del poder discriminativo inicial. Podemos comparar el resultado de los rostros en la versión original (Fig. 5.1) y tras el preproceso en la Fig. 5.3.

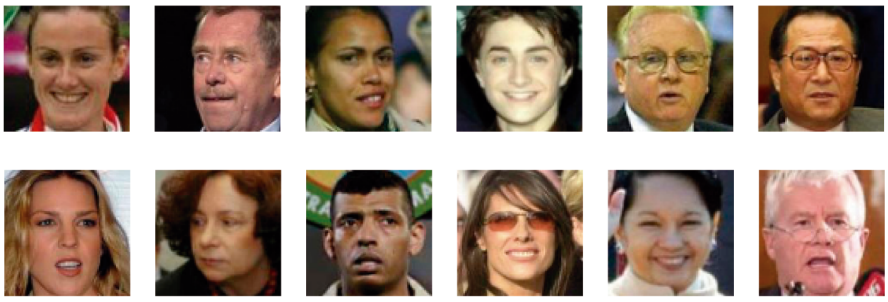

Figura 5.3: Rostros de la base de datos LFW tras el preprocesado.

Por otra parte, la base de datos GROUPS es bastante más compleja. En primer lugar se trata de una base de datos de rostros en las que se nos proporciona imágenes de grupos de personas y no la cara directamente. Para obtener los diferentes rostros se nos proporcionan las coordenadas centrales de los ojos y la separación entre estos para cada una de las personas que aparecen en la imagen. Dado esto, no existe una versión de la base de datos con las caras alineadas lo que supone trabajo adicional.

Para hacer el preproceso a esta base de datos, en primer lugar hemos recortado las caras con la información de los ojos mencionada y posteriormente hemos utilizado una herramienta basada en Dlib [1] para alinear las imágenes. Finalmente,

[1] Herramienta *Align Faces* basada en Dlib utilizada en GROUPS: `https://github.com/scotthong/dlib-align-faces`

hemos realizado el reescalado de las imágenes de forma similar a LFW a 80x80 píxeles. Todo el proceso puede observarse en la Fig. 5.4.

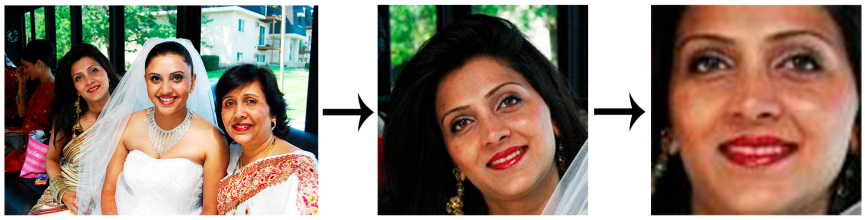

Figura 5.4: Pasos preproceso base de datos GROUPS.

5.2 Recursos computacionales

A continuación vamos a exponer los recursos computacionales, tanto *software* cómo *hardware*, que hemos utilizado para la realización de los experimentos de nuestro trabajo.

5.2.1 Recursos Software

El uso de los recursos *software* está destinado a la implementación de los diferentes modelos vistos en el Capítulo 6, basándonos en el código de Kuangliu disponible en `https://github.com/kuangliu/pytorch-cifar`. Dicho código ha sido escrito con Pytorch [44] utilizando la versión 0.3 basada en Python 3.5.

Pytorch es capaz de utilizar tanto la GPU como la CPU para realizar las diferentes operaciones internas. Para ser capaces de utilizar la GPU hemos utilizado la versión de Pytorch con CUDA 8.0 y cuDNN, donde CUDA es una API para la programación de GPUs NVIDIA y cuDNN es una librería para la aceleración de cómputo en tarea de *deep learning*. Por último, todo esto ha sido instalado sobre sistema operativo Ubuntu en su versión 16.04 LTS de 64 bits.

5.2.2 Recursos Hardware

En cuanto a los componentes *hardware*, el único que afecta directamente a nuestra tarea haciendo que podamos probar modelos con más rapidez es la GPU. Gracias al centro de investigación de la Universitat Politècnica de València *Pattern Recognition and Human Language Technology* (PRHLT), hemos dispuesto de cinco NVIDIA GeForce GTX 1080 y una NVIDIA Titan Xp para poder probar diferentes modelos simultáneamente. Mientras que el modelo GTX 1080 cuenta con 2560 núcleos CUDA y 8GB de memoria interna, la Titan Xp tiene 3840 núcleos CUDA y 12GB de memoria interna. Para más información acerca de estas GPUs véase `https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-1080`, para la

GTX 1080, y https://www.nvidia.es/titan/titan-xp/ para el modelo Titan Xp.

Por otra parte el equipo el que disponía, que utilizaba para depurar y realizar las pruebas sobre las GPUs nombradas, estaba compuesto por una GPU NVIDIA GeForce con 1280 núcleos CUDA y 6GB de memoria interna. Además, aunque no influyan inmediatamente sobre la computación, la CPU usada era una AMD Ryzen 5 1600 3.2GHZ con 6 núcleos y 2 hilos por núcleo acompañada 16GB de memoria RAM DDR4 con una frecuencia de 2400 MHz. Las especificaciones completas de la GPU GTX 1060 se pueden encontrar en https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-1060/specifications.

Capítulo 6

Modelos empleados

El objetivo del siguiente capítulo es el dar un vistazo los diferentes modelos que hemos utilizado para nuestra tarea de clasificación de género y en los que se basa nuestro trabajo. En cada sección trataremos cada uno de estos modelo así como las características principales de cada uno de ellos. Se tratan de modelos bien conocidos en el marco de las redes neuronales por sus buenos resultados.

Las configuraciones utilizadas de cada modelo son descritas en detalle en la sección 7.1.2.

6.1 Very Deep Convolutional Networks

Las redes convolucionales muy profundas o *Very Deep Convolutional Networks*, apodadas como VGG [37], nacen como una alternativa simple de arquitectura de red convolucional que pudiera dar buenos resultados y además sirviera para estudiar el efecto de la profundidad de este tipo de redes. Gracias a su simpleza y la obtención del primer y segundo puestos en las tareas de localización y clasificación respectivamente del reto de Imagenet Large-Scale VIsual Recognition Challenge[1] (ILSVRC) 2014. Utilizaremos la idea de modelo para nuestros experimentos.

La VGG está compuesta por una serie de capas convolucionales donde el tamaño de los kernel utilizados es de 3x3 píxeles. El *stride* de la convolución es prefijado a 1 píxel de forma que la resolución de lo mapas resultantes es preservada gracias al uso de *padding*. Para la reducción del tamaño de los mapas se utilizan capas de max pooling de un tamaño de 2x2 píxeles con *stride* 2, de forma que en cada pooling reducimos a la mitad la resolución de los mapas. Para finalizar, en el *paper* original se añaden tres capas *fully connected*, donde la última es la encargada de la tarea de clasificación a través de una softmax.

En la Fig. 6.1 observamos una configuración básica de modelo VGG. Las principales diferencias que observamos entre los modelos propuestos de VGG son el número de capas convolucionales entre max pooling y max pooling, siempre manteniendo la profundidad de estas capas entre dichos max pooling y manteniendo el número de cinco max pooling para llegar a la misma resolución de los

mapas a la hora de hacer el *flattening* de los mapas. La notación que se utiliza para nombrar los diferentes modelos es VGGX donde X viene dado por el número de capas con pesos que esta tiene, el número de capas convolucionales y capas de la *fully connected*. Por último destacar que el incremento de capas del modelo se hace en referencia a las capas convolucionales, donde normalmente estas se acumulan en las últimas capas.

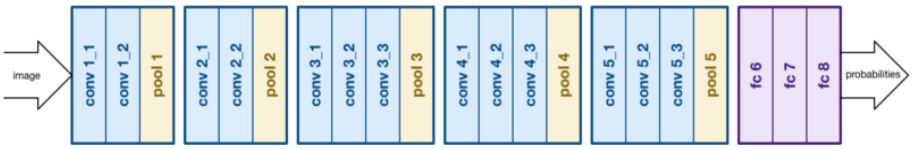

Figura 6.1: Esquema básico del modelo VGG.

6.2 Deep Residual Convolutional Networks

Se ha podido observar como la profundidad de las redes es de crucial importancia para los problemas más complejos. Por ejemplo, para el modelo VGG anterior, tan solo incrementando la profundidad de la capa éramos capaces de pasar de un error de clasificación del 29.6 % al 25.5 %. Pero, ¿se aprende con más facilidad al apilar capas simplemente? La verdad es que, como se ha visto reportado esto no es así y podemos observar un ejemplo de esta degradación en la Fig. 6.2. Con esto concluimos que el aspecto de la profundidad de las redes neuronales no es algo que pueda incrementarse indefinidamente, ya que puede presentar un obstáculo para el entrenamiento. [38]

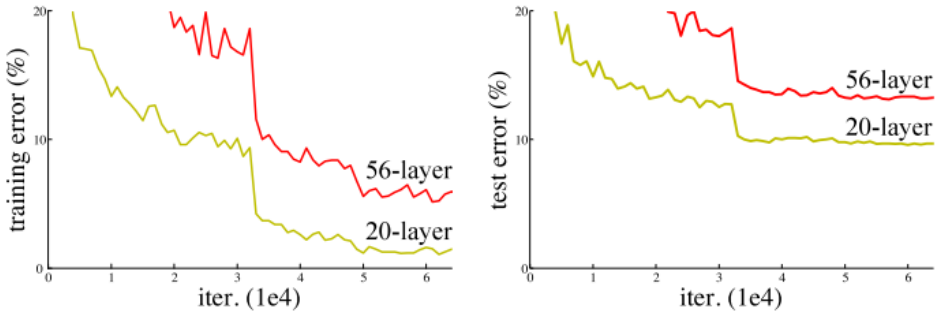

Figura 6.2: Análisis del error de entrenamiento (izquierda) y test (derecha) sobre CIFAR-10[1] con 20 capas y 56 capas en una red "plana". Imagen de [38].

La idea de las Deep Residual Convolutional Networks o ResNet es la de facilitar el entrenamiento de aquellas redes fundamentalmente profundas que se han utilizado anteriormente, como la VGG. La utilización de esta técnica acaparó muchas miradas debido a sus resultados en la competición ILSVRC de 2015 donde quedó primera en la tarea de clasificación, detección y localización.

Para solucionar el problema de degradación expuesto se introduce un marco de trabajo de aprendizaje profundo residual. Para ello, la idea principal es la de utilizar mapas residuales tras una serie de convoluciones en vez de conectar los mapas directamente. Con esto tras una serie de convoluciones que forman lo que

es llamado como bloque, se conecta la entrada x de dicho bloque con su la salida $\mathcal{F}(x)$, obtenida tras la realizar las operaciones del bloque. Así la salida del bloque quedaría como $\mathcal{F}(x) + x$. En la Fig. 6.3 vemos cómo se realizaría la conexión normal y otra variación de la misma apodada "bottleneck"por el uso de convoluciones con kernel 1x1.

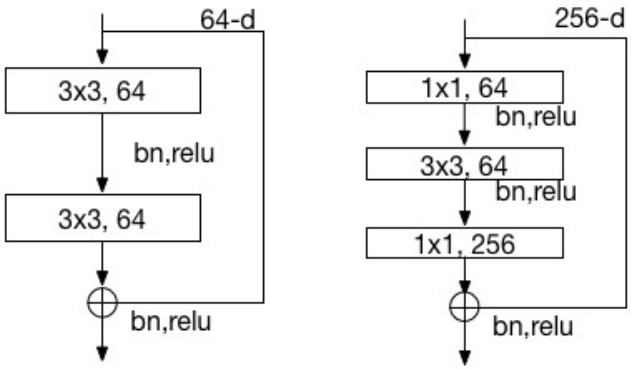

Figura 6.3: Representación de un bloque normal (izquierda) y un bloque "bottleneck". Imagen de [38].

Para poder realizar esta conexión las dimensiones de x y $\mathcal{F}(x)$ deben ser iguales para hacer la suma elemento a elemento entre estos correctamente, cuando no lo son, se opta por: (A) La conexión se realiza añadiendo ceros de relleno para incrementar las dimensiones; (B) Se proyecta el mapa a las dimensiones deseadas utilizando convoluciones con kernel de tamaño 1x1. Para ambas opciones, cuando las conexiones se realizan entre mapas de diferentes tamaños (conexiones marcadas por líneas discontinuas en la Fig. 6.4), estas son realizadas con *stride* de 2 píxeles para reducir el tamaño de los mapas.

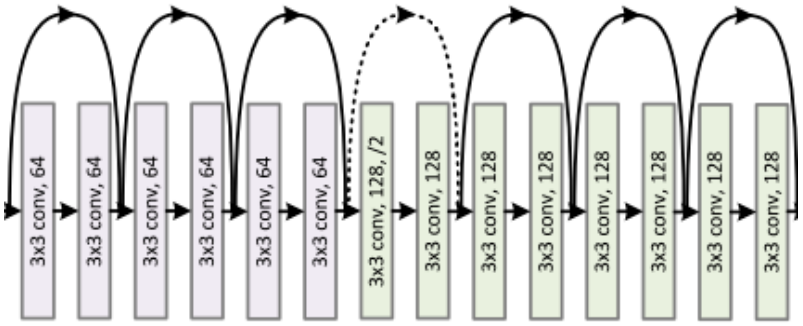

Figura 6.4: Ejemplo de capas siguiendo la idea de las conexiones introducidas en el modelo ResNet. Imagen de [38].

Aún con esta serie de conexiones la red puede seguir siendo entrenada en su totalidad con SGD y retropropagación del error. Al igual que para los otros modelos, se han implementado diferentes versiones modificando el número de convoluciones y el numero de mapas de dichas convoluciones, manteniendo en todas las pruebas el tamaño al que reducimos la imágenes (mediante las convoluciones con *stride* mencionadas) a 2x2. Finalmente hacemos un *reshape* y añadimos una capa con el número de neuronas de salida, 2 para nuestro problema.

6.3 Densely Connected Convolutional Networks

Conforme las CNNs van incrementando su profundidad, un nuevo problema surge: A medida que la información de la entrada pasa sobre las diferentes capas, esta puede irse perdiendo y puede dar lugar a distintos problemas. Las ResNet presentadas anteriormente trataban de dar solución a este problema y a continuación vamos a tratar otra metodología introducida por las Densely Connected Convolutional Networks o DenseNets [39].

Mientras que en las ResNets veíamos cómo cada cierto número de convoluciones sumamos los mapas resultantes con mapas anteriores, la esencia de las DenseNets es que estas utilizan la operación de concatenación. Para favorecer el flujo de información entre las capas se propone la conexión directa de cualquier capa con todas sus anteriores dentro de un mismo bloque denominado como *dense block*. De esta forma podemos definir la capa ℓ como

$$x_\ell = H_\ell([x_0, x_1, ...x_{\ell-1}]) \tag{6.1}$$

donde $[x_0, x_1, ...x_{\ell-1}]$ se refiere a la concatenación de los mapas producidos anteriormente dentro del mismo bloque. Además se define $H_\ell(\cdot)$ como una función compuesta por *batch normalization*, seguido de una ReLU y finalmente una convolución 3x3. La operación de la concatenación no se puede aplicar cuando el tamaño de los mapas de características son diferentes y esta es la razón por la que se encapsula en diferentes bloques esta operación (Fig. 6.5). La transición entre estos bloques consiste en realizar *batch normalization*, aplicar una convolucion 1x1 y finalmente un average pooling 2x2.

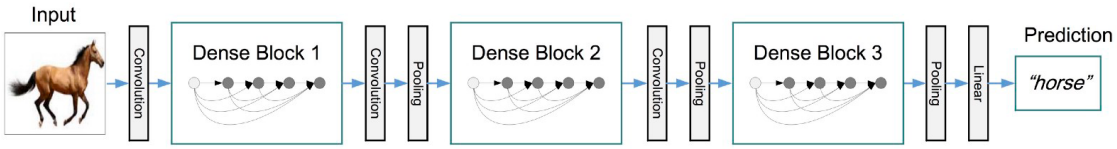

Figura 6.5: Ejemplo de DenseNet con tres bloques capas. Las capas entre bloques son referidas como capas de transición. Imagen de [39].

Para la creación de esta red un parámetro a tratar es la tasa de crecimiento o *growth rate* k. Para la red cada función H_ℓ produce k mapas de características, lo que hace que al ser concatenados la capa l-ésima tenga k_0+k × (ℓ-1) mapas concatenados, donde k_0 es el número de canales en la capa de entrada. Con esto vemos que aunque una capa solo produzca k mapas, tiene normalmente muchos más de entrada. Por esta razón se introduce un cuello de botella o *bottleneck* realizando una convolución 1x1 antes de llegar a la convolución 3x3 para reducir el número de mapas de entrada. Para finalizar, se puede introducir un factor θ para reducir el número de mapas en las capas de transición de forma que si al final del bloque tenemos m mapas, generamos $[\theta m]$ mapas tal que $0 < \theta \leq 0$.

6.4 MobileNetV2: Inverted Residuals and Linear Bottlenecks

Hasta ahora hemos visto modelos del estado del arte que aunque cada vez obtienen mejores resultados a la par que son más complejos, estos suelen requerir un número mayor de recursos computacionales. El propósito de nuestro trabajo es el de estudiar diferentes modelos así como el de tratar de reducir los costes espaciales y computacionales de los mismos para ser capaces de utilizarlos en dispositivos con bajos recursos como pueden ser los móviles. Con esta idea es concebida MobileNetV2 [40], una arquitectura de red neuronal específicamente diseñada para entornos móviles y con recursos limitados.

Para llevar a cabo esto se utiliza la idea de un bloque utilizado en muchas arquitecturas de redes neuronales: *Depthwise Separable Convolutions*. Este consiste en reemplazar el operador convolucional por una versión factorizada que lo separa en varias capas. La primera capa, llamada *depthwise convolution*, realiza un filtrado aplicando un único filtro convolucional por canal de entrada, posible gracias a una convolución previa 1x1. La segunda capa es una convolución 1x1 lineal, llamada *pointwise convolution*, responsable de crear nuevas características a través de la combinación lineal de los canales de entrada. Se justifica la utilización de una convolución lineal ya que experimentalmente la no linealidades afectan a los resultados. Este tipo de convoluciones se denominan como *bottleneck* y nos referiremos al ratio entre el tamaño de la entrada a la convolución y el obtenido como *expansion ratio*.

En la Fig. 6.6 podemos observar los dos bloques utilizados en la red. El primero (a la izquierda), muestra el bloque utilizado cuando mantenemos el tamaño de entrada y salida de los mapas. Se emplean conexiones residuales con la finalidad de incrementar la habilidad del gradiente para ser propagado a través de múltiples capas. El segundo bloque (a la derecha) ilustra cómo se hace la reducción del tamaño de los mapas al aplicar un stride de 2 píxeles. En ambos bloques se utiliza una variante de la ReLU, ReLU6, debido a su robustez cuando se emplean cálculos de baja precisión [41].

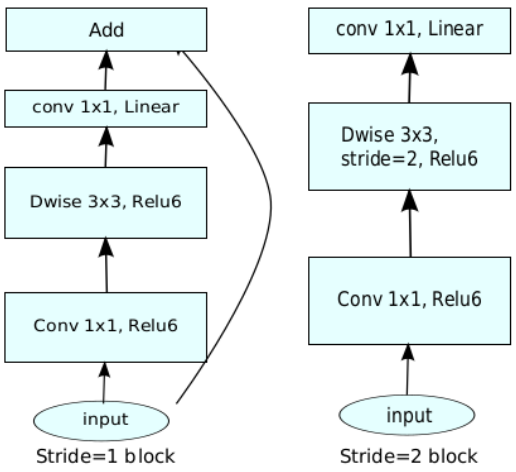

Figura 6.6: Bloques convolucionales de MobileNetV2 [40].

Por último podemos destacar que se reduce el tamaño de los mapas a 1x1 a través de un operador average pooling y posteriormente, tras un reshape, hacemos la tarea de clasificación. Las principales características que hemos estudiado en las pruebas realizadas sobre este modelo tratan el número de mapas utilizados así como el de bloques. En cambio, como aconseja el articulo, el ratio de expansión lo solemos dejar fijo a un valor de 6 ya que empíricamente da mejores resultados.

Capítulo 7

Experimentos y Resultados

En el siguiente capítulo vamos a explicar el procedimiento seguido para realizar la experimentación así como los resultados obtenidos en la misma. Dado el alto coste computacional de nuestra tarea no somos capaces de realizar una búsqueda exhaustiva de los hiperparámetros de nuestro modelo sobre el protocolo general a utilizar en las dos bases de datos. Por ejemplo, un modelo relativamente pequeño y simple de VGG tarda en ser entrenado alrededor de dos horas[1], o un modelo más costoso como una red DenseNet vemos que puede llegar a tardar hasta veinte horas[1]. Debido a esto se ha optado por la utilización de una base de datos de rostros de menor tamaño donde realizar la búsqueda de hiperparámetros de forma más rápida fijando únicamente una semilla para la inicialización de los parámetros aleatorios y tomando la misma partición de datos de entrenamiento y test.

Idealmente deberíamos haber utilizado una validación cruzada como se realiza posteriormente en el protocolo general de las bases de datos oficiales utilizadas, pero como comentábamos el coste computacional se multiplicaría por 5 y nos sería inadmisible. Así, en el capítulo comenzaremos explicando esta primera búsqueda y experimentación sobre la base de datos reducida y a continuación mostraremos los experimentos y resultados obtenidos utilizando las bases de datos y protocolos planteados en una primera instancia (Sección 5.1).

7.1 Búsqueda de hiperparámetros

En el Capítulo 4 introdujimos como funcionaban las redes neuronales y cuáles eran las técnicas más comunes utilizadas a la hora de construirlas. Estas técnicas no son las únicas utilizadas por los investigadores y desarrolladores, encontrando otras que atacan problemas muy variados: Acelerar el descenso por gradiente a través de la modificación del learning rate (como el método Adadelta [42]), utilizar funciones de activación diferentes (o simplemente variaciones como la Leaky ReLU)... Todas ellas con la finalidad de conseguir que el desempeño de las redes mejore, ya sea la velocidad con la que las redes convergen o sus resultados obtenidos. El problema es que la mayoría de estas no son capaces de generalizar a todos los problemas o los resultados obtenidos no son lo suficientemente notorios

[1] Modelo estándar entrenado utilizando la máquina con la GPU NVIDIA Titan Xp mencionada en la Subsección 5.2.2 durante 300 epochs.

para justificar un cambio en el panorama actual.

Debido a el gran número de técnicas existentes y los recursos computacionales con los que contamos, para nuestra experimentación vamos a probar un conjunto restringido de dichas técnicas, tomando aquellas más utilizadas en el estado del arte y tomando algunas de estas como estándar por su amplia utilización. Por lo tanto, definimos nuestra búsqueda sobre: La utilización del descenso por gradiente estocástico o el algoritmo Adam; El uso del Dropout (siempre aplicando un Dropout creciente hasta un valor de 0.5) o no; Y si debemos utilizar o no ruido gaussiano, probando valores de 0.1, 0.2 y 0.3 de desviación. Para las bases de datos principales (LFW y GROUPS) probaremos otras propuestas que indicaremos en la correspondiente sección. Por otra parte, se ha dejado como estándar el uso de Batch Normalization y las técnicas explicadas de Data Augmentation a excepción del añadido del ruido gaussiano mencionado.

Con esto y probando diferentes topologías de las redes propuestas, hemos realizado nuestros primeros experimentos sobre una base de datos de tamaño reducido para seleccionar aquellas con mejores resultados y utilizar estas para las bases de datos principales.

7.1.1 Base de datos reducida

Como comentábamos, la base de datos de pruebas de todas las topologías y técnicas para poder seleccionar las más convenientes es un conjunto reducido de alrededor de 10.000 imágenes, combinando imágenes de la base de datos de LFW y GROUPS. Al igual que en estas, las dimensiones que hemos tomado son de 100x100 píxeles que posteriormente son recortadas aleatoriamente durante el entrenamiento a 80x80 píxeles y de forma centrada en la fase de test.

7.1.2 Configuraciones de los modelos

Es importante presentar las diferentes configuraciones utilizadas con respecto a cada uno de los modelos presentados (Capítulos 6). En estas configuraciones plasmamos el número de filtros utilizados en las convoluciones, el tipo de pooling, número de bloques... Posteriormente mostraremos los resultados obtenidos para cada base de datos con estos modelos aplicando las configuraciones expuestas. Con la idea de estudiar el efecto de la reducción del número de parámetros sobre la tasa de acierto, realizaremos pruebas sobre configuraciones muy dispares con tal de abarcar lo máximo posible teniendo en mente nuestra capacidad computacional. También mencionar que en las siguientes tablas cuando hablamos de "*Inferencia*" hacemos referencia al tiempo que una imagen de un rostro dado tarda en atravesar la serie de operaciones de la red neuronal que se trate en cada caso, poniendo énfasis en que solo se trata de una única imagen la que es tratada. Mencionar que los tiempos de inferencia obtenidos en las siguientes tablas han sido extraídos a partir de la inferencia de 100.000 muestras una a una y finalmente han sido promediados los tiempos, todo ello utilizando la tarjeta gráfica NVIDIA GeForce GTX 1080.

En la Tabla 7.1 observamos las diferentes configuraciones para el modelo VGG, donde de izquierda a derecha vemos como más capas son añadidas, además vamos variando las profundidades. Los parámetros de cada una de estas configuraciones de VGGs se presentan en la Tabla 7.2.

Configuraciones VGGs			
VGG9	**VGG13**	**VGG14**	**VGG17**
9 capas con pesos	13 capas con pesos	14 capas con pesos	17 capas con pesos
entrada (80x80 imágenes Color/BN)			
conv3-32	conv3-16 conv3-16	conv3-32 conv3-32	conv3-32 conv3-32
maxpool			
conv3-64	conv3-16 conv3-16	conv3-64 conv3-64	conv3-64 conv3-64
maxpool			
conv3-128	conv3-16 conv3-16	conv3-128 conv3-128	conv3-128 conv3-128 conv3-128
maxpool			
conv3-256	conv3-32 conv3-32	conv3-256 conv3-256	conv3-256 conv3-256 conv3-256
maxpool			
conv3-512	conv3-32 conv3-32	conv3-128 conv3-128	conv3-512 conv3-512 conv3-512
maxpool			
FC-512			
FC-256			
soft-max			

Tabla 7.1: Configuraciones diferentes del modelo VGG probadas.

Modelo	**Parámetros**	**Inferencia GPU**	**Inferencia CPU**
VGG9	2.8M	0,750ms	7.1ms
VGG13	49K	1,258ms	3.2ms
VGG14	5.9M	1,200ms	18ms
VGG17	9M	1,572ms	24.8ms

Tabla 7.2: Número de parámetros y tiempos de inferencia de configuraciones VGGs.

Las configuraciones de las ResNets son presentadas de igual forma en la Tabla 7.3. En esta se muestran el número de bloques utilizados, formados por operadores convolucionales, y las dimensiones de los mismos. Recordar que aquí la reducción de la dimensionalidad de los mapas se hace a través de convoluciones con stride de tamaño dos al pasar entre bloques de diferentes números de mapas. Los parámetros de dichas configuraciones se presentan en la Tabla 7.4.

Con respecto a las DenseNets, debido al tiempo de cómputo que estas representan el número de configuraciones probadas sobre la base reducida es más dispar que en el resto de modelos, para así tomar aquellas primeras que obtuvieran me-

Configuraciones ResNets						
ResNet18v0	**ResNet18v1**	**ResNet18v2**	**ResNet34v0**	**ResNet34v1**	**ResNet50v0**	**ResNet101v0**
entrada (80x80 imágenes Color/BN)						
conv3-16	conv-32	conv-64	conv-64	conv-32	conv-16	conv-16
$\begin{bmatrix} 3\times3,\ 16 \\ 3\times3,\ 16 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 32 \\ 3\times3,\ 32 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 32 \\ 3\times3,\ 32 \end{bmatrix}\times3$	$\begin{bmatrix} 3\times3,\ 16 \\ 3\times3,\ 16 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1,\ 16 \\ 3\times3,\ 16 \\ 1\times1,\ 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1,\ 64 \\ 3\times3,\ 64 \\ 1\times1,\ 256 \end{bmatrix}\times3$
$\begin{bmatrix} 3\times3,\ 32 \\ 3\times3,\ 32 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times4$	$\begin{bmatrix} 3\times3,\ 32 \\ 3\times3,\ 32 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1,\ 32 \\ 3\times3,\ 32 \\ 1\times1,\ 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1,\ 128 \\ 3\times3,\ 128 \\ 1\times1,\ 512 \end{bmatrix}\times4$
$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times6$	$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1,\ 64 \\ 3\times3,\ 64 \\ 1\times1,\ 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1,\ 256 \\ 3\times3,\ 256 \\ 1\times1,\ 1024 \end{bmatrix}\times23$
$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 512 \\ 3\times3,\ 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}\times3$	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1,\ 128 \\ 3\times3,\ 128 \\ 1\times1,\ 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1,\ 512 \\ 3\times3,\ 512 \\ 1\times1,\ 2048 \end{bmatrix}\times3$
avg pool						
soft-max						

Tabla 7.3: Configuraciones diferentes del modelo ResNet probadas.

Modelo	**Parámetros**	**Inferencia GPU**	**Inferencia CPU**
ResNet18v0	701K	1,660ms	11.5ms
ResNet18v1	2.8M	1,675ms	32.8ms
ResNet18v2	11M	2,419ms	102.7ms
ResNet34v0	5.3M	3,050ms	63.3ms
ResNet34v1	1.3M	2,687ms	22.2ms
ResNet50v1	1.4M	3,876ms	31.2ms
ResNet101v0	42.5M	13,429ms	435.7ms

Tabla 7.4: Número de parámetros y tiempos de inferencia configuraciones ResNets.

jores resultados. Con tal de presentar mejor las configuraciones y sabiendo que la principal diferencia entre estas es el número de operadores convolucionales en los diferentes *Dense Blocks*, mostramos en la Tabla 7.5 un versión simplificada de las configuraciones. Entre bloque y bloque recordar que situamos una *Transition Layer* donde realizamos la reducción de dimensionalidad de los mapas a través de un *average* pooling con stride dos. Los parámetros de estas configuraciones se muestran en la Tabla 7.6.

Configuraciones DenseNets	
Modelo	**Operadores por bloque**
DenseNet41	$3 \rightarrow 6 \rightarrow 6 \rightarrow 3$
DenseNet77	$6 \rightarrow 12 \rightarrow 12 \rightarrow 6$
DenseNet121	$6 \rightarrow 12 \rightarrow 24 \rightarrow 16$
DenseNet169	$6 \rightarrow 12 \rightarrow 32 \rightarrow 32$

Tabla 7.5: Configuraciones diferentes del modelo DenseNet probadas.

Las últimas configuraciones utilizadas son para el modelo MobileNetv2 (Tabla 7.7), así como los parámetros de las diferentes variantes (Tabla 7.8). Cada línea

Modelo	Growth Rate	Parámetros	Inferencia GPU	Inferencia CPU
DenseNet41	6	44K	3,352ms	12.1ms
DenseNet41	12	170K	3,328ms	28.6ms
DenseNet77	6	122K	6,186ms	26.5ms
DenseNet77	12	472K	6,256ms	58.3ms
DenseNet121	6	258K	9,732ms	33.7ms
DenseNet121	12	1M	9,782ms	72.3ms
DenseNet121	32	7M	11,901ms	249ms
DenseNet169	6	465K	13,515ms	42ms
DenseNet169	12	1.8M	13,565ms	89.3ms
DenseNet169	32	12.5M	14,752ms	302.5ms

Tabla 7.6: Número de parámetros y tiempos de inferencia configuraciones DenseNets.

describe una secuencia de uno o más módulos repetidos n veces. Todas las capas dentro de una misma secuencia tienen el mismo número de canales de salida c, teniendo el primer canal de cada secuencia stride s y el resto 1. Todas las convoluciones espaciales usan kernels de tamaño 3x3 y se les aplica un factor de expansión t.

Configuraciones MobileNetsv2												
Operador	**MobileNetv2 36**				**MobileNetv2 45**				**MobileNetv2 54**			
	t	c	n	s	t	c	n	s	t	c	n	s
conv2d	-	32	1	1	-	32	1	1	-	32	1	1
bottleneck	1	16	1	1	1	16	1	1	1	16	1	1
bottleneck	6	24	2	2	5	24	2	2	6	24	2	1
bottleneck	6	32	2	2	6	32	2	2	6	32	3	2
bottleneck	6	64	3	1	6	64	3	1	6	64	4	2
bottleneck	6	64	2	2	6	96	3	1	6	96	3	1
bottleneck	6	128	1	2	5	128	2	2	6	160	3	2
bottleneck	-	-	-	-	5	256	1	2	6	320	1	1
conv2d 1x1	-	512	1	1	-	1024	1	1	-	1280	-	-
avgpool	5x5				5x5				10x10			
softmax												

Tabla 7.7: Configuraciones diferentes del modelo MobileNetv2 probadas.

Modelo	Parámetros	Inferencia GPU	Inferencia CPU
MobileNetv2 36	429K	2,745ms	40ms
MobileNetv2 45	1.2M	3,428ms	64.5ms
MobileNetv2 54	2.3M	4,233ms	116.3ms

Tabla 7.8: Número de parámetros y tiempos de inferencia configuraciones MobileNets.

7.1.3 Análisis de primeros resultados

Se han probado un total de 150 modelos con diferentes configuraciones, variando el uso de las técnicas anteriormente descritas. Debido al gran volumen de resultados obtenidos, vamos a ir presentando varias conclusiones de forma separada. La primera y más notoria en los resultados es que la utilización del algoritmo Adam no funciona tan bien como SGD. En la Tabla 7.9, podemos observar una muestra de los resultados de diferentes modelos, donde el algoritmo Adam suele funcionar significativamente peor.

Modelo	**Dropout**	**Ruido (std)**	**Acierto (%)**	
			SGD	**Adam**
VGG9	No	0.0	98.58	97.26
VGG9	Si	0.3	97.54	76.48
VGG13	No	0.0	97.16	95.93
VGG14	No	0.2	98.32	96.97
VGG17	Si	0.1	97.82	97.23

Tabla 7.9: Tasa de acierto obtenida por los diferentes modelos. Base de datos reducida. SGD frente Adam.

Dados estos primeros resultados y teniendo en mente que debemos ganar tiempo para la experimentación con las bases de datos principales, descartamos el uso de Adam, que aunque a veces no proporciona resultados tan dispares de SGD, la tónica indica que SGD nos proporcionará mejores resultados.

Por otra parte, la utilización de ruido gaussiano o no así como el del dropout no muestran diferencias significativas por lo que, como vemos en las siguientes tablas de esta subsección, para los modelos más costosos ResNet y DesneNet nos hemos inclinado por la no utilización de estas técnicas. Además, tomando la utilización del Batch Normalization por norma, existen trabajos [43] que llegan a la conclusión de que la utilización combinada del Dropout y Batch Normalization no funciona adecuadamente.

Para finalizar, mencionar que la selección de los modelos a utilizar con las bases de datos principales ha sido realizada de forma totalmente arbitraria con la idea de probar las configuraciones más dispares teniendo en más en cuenta los modelos que proporcionan para esta base de datos deducida mejores resultados.

Modelo	Dropout	Ruido (std)	Acierto (%)	
			SGD	Adam
VGG9	No	0.0	98.58	97.26
VGG9	Si	0.3	97.54	76.48
VGG9	Si	0.1	98.21	97.82
VGG9	No	0.1	98.11	96.97
VGG9	Si	0.0	98.01	97.63
VGG9	Si	0.2	97.82	76.48
VGG9	No	0.3	97.45	96.31
VGG9	No	0.2	97.26	96.53
VGG13	No	0.2	97.82	96.67
VGG13	No	0.0	97.16	95.93
VGG13	No	0.3	97.45	95.56
VGG13	No	0.1	96.88	96.41
VGG13	Si	0.1	96.22	96.41
VGG13	Si	0.2	96.12	76.48
VGG13	Si	0.0	95.46	76.48
VGG13	Si	0.3	93.29	76.48
VGG14	No	0.2	98.32	96.97
VGG14	No	0.0	98.32	96.22
VGG14	No	0.3	98.31	96.69
VGG14	Si	0.0	98.31	76.48
VGG14	No	0.1	98.01	97.82
VGG14	Si	0.1	98.01	76.48
VGG14	Si	0.2	97.92	76.48
VGG14	Si	0.3	97.73	76.48
VGG17	Si	0.1	97.92	76.48
VGG17	No	0.1	97.82	97.23
VGG17	Si	0.0	97.63	76.48
VGG17	No	0.3	97.63	78.84
VGG17	Si	0.3	97.45	76.48
VGG17	No	0.0	97.35	97.16
VGG17	Si	0.2	97.35	76.48
VGG17	No	0.2	97.26	96.22

Tabla 7.10: Tasa de acierto obtenida para la base de datos reducida con el modelo VGG.

Modelo	Acierto (%)
ResNet18v1	97.63
ResNet50v0	97.54
ResNet18v0	97.26
ResNet18v2	97.16
ResNet101v0	96.63
ResNet34v0	95.93
ResNet34v1	95.84

Tabla 7.11: Tasa de acierto obtenida para la base de datos reducida con el modelo ResNet. Utilizando SGD.

Modelo	Growth Rate	Acierto (%)
DenseNet41	12	98.23
DenseNet41	6	98.01
DenseNet121	32	97.92
DenseNet169	12	97.92
DenseNet121	6	97.82
DenseNet77	12	97.73
DenseNet121	12	97.63
DenseNet169	32	97.35
DenseNet77	6	97.07
DenseNet169	6	96.88

Tabla 7.12: Tasa de acierto obtenida para la base de datos reducida con el modelo DenseNet. Utilizando SGD.

Modelo	Dropout	Ruido (std)	Acierto (%)
MobileNetv2 54	No	0.0	98.11
MobileNetv2 36	No	0.0	98.01
MobileNetv2 54	No	0.2	97.82
MobileNetv2 36	No	0.1	97.73
MobileNetv2 36	No	0.2	97.54
MobileNetv2 45	No	0.1	97.45
MobileNetv2 36	No	0.3	97.16
MobileNetv2 45	No	0.3	96.97
MobileNetv2 45	No	0.2	96.88
MobileNetv2 54	No	0.1	96.78
MobileNetv2 45	No	0.0	96.22
MobileNetv2 54	No	0.3	95.84
MobileNetv2 36	Si	0.3	93.62
MobileNetv2 36	Si	0.0	93.47
MobileNetv2 45	Si	0.1	93.33
MobileNetv2 54	Si	0.2	93.32
MobileNetv2 36	Si	0.1	93.11
MobileNetv2 54	Si	0.1	92.86
MobileNetv2 36	Si	0.2	92.74
MobileNetv2 54	Si	0.0	92.71
MobileNetv2 54	Si	0.3	92.55
MobileNetv2 45	Si	0.3	92.55
MobileNetv2 45	Si	0.0	92.36
MobileNetv2 45	Si	0.2	92.21

Tabla 7.13: Tasa de acierto obtenida para la base de datos reducida con el modelo MobileNetv2. Utilizando SGD.

7.2 Análisis de resultados

Los principales experimentos llevados a cabo en este trabajo son presentados en la siguiente sección. Para estos hemos utilizado dos bases de datos de rostros conocidas, LFW y GROUPS, que son descritas previamente en las Subsecciones 5.1.1 y 5.1.2 respectivamente. El protocolo de evaluación para ambas bases de datos consiste en dividir todas las imágenes que componen dichas bases de datos en 5 particiones para así realizar un proceso de validación cruzada de forma que 4 particiones son utilizadas para el entrenamiento del modelo y la restante para el test. Así, los experimentos son realizados realizando las 5 posibles combinaciones para finalmente promediar los resultados obtenidos. Las particiones utilizadas pueden encontrarse en [31] y [36] para LFW y GROUPS respectivamente.

Por otra parte, podríamos calcular los intervalos de confianza usando una aproximación normal a partir de la expresión $\hat{p} \pm z\sqrt{\hat{p}(1-\hat{p})/n}$, donde $\hat{p}$ es la tasa de acierto promediada por las 5 combinaciones con las particiones comentadas, n es el número de muestras y z debe ser 1-96 si deseamos calcular los intervalos de confianza estándar al 95 %. Debido al gran número de muestras de entrenamiento utilizadas en los experimentos, estos intervalos podrían ser demasiado pequeños, por lo que dichos intervalos no han sido incluidos en las tablas presentadas a continuación.

Debido a que la mayoría de los experimentos realizados en los trabajos relacionados con nuestra tarea para el estado del arte utilizan las imágenes de las bases de datos en blanco y negro, se ha decidido realizar el entrenamiento y evaluación de los modelos bajo esta casuística para compararlos con el caso a color. Además, aunque en la fase anterior de búsqueda de los modelos a probar hemos aplicado únicamente la normalización de los valores en rango de 0 a 1, también proponemos el estudio del rango de -1 a 1. Para todas estas pruebas el entrenamiento ha sido similar: el entrenamiento ha sido realizado a lo largo de 300 epochs de forma que hemos ido reduciendo el learning rate manualmente desde 0.1 hasta 0.001 a lo largo de las 250 primeras epochs para en las 50 últimas ir reduciéndolo linealmente hasta 0.

7.2.1 Labeled Faces in the Wild

En las siguientes tablas mostramos los diferentes resultados obtenidos por los modelos VGG (Tabla 7.14), ResNet (Tabla 7.15), DenseNet (Tabla 7.16) y MobileNetv2 (Tabla 7.17), con cada una de las configuraciones probadas. Cómo mencionábamos al principio de la sección, la prueba del entrenamiento con las imágenes a color y en blanco y negro, así como la normalización de los valores en los rangos de 0 a 1 y de -1 a 1, son configuraciones comunes a todos los modelos. El mejor resultado de cada modelo ha sido resaltado cambiando el fondo de la celda en el que se encuentra.

Modelo	Dropout	Ruido (std)	Acierto (%) Color		Acierto (%) BN	
			$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$
VGG9	No	0.0	97.54	97.68	97.55	97.56
VGG9	Si	0.1	97.51	97.50	97.45	97.33
VGG17	Si	0.1	97.49	97.49	97.26	97.19
VGG14	Si	0.1	97.27	97.37	97.28	97.43
VGG13	Si	0.1	96.04	95.72	93.58	95.51

Tabla 7.14: Tasa de acierto obtenida para la base de datos LFW con el modelo VGG. Se prueban los rangos de normalización de 0 a 1 y de -1 a 1, así como el base de datos a color y blanco y negro (BN).

Modelo	Acierto (%) Color		Acierto (%) BN	
	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$
ResNet18v0	97.33	97.51	97.46	97.46
ResNet18v2	97.09	97.22	97.08	97.43
ResNet18v1	97.27	97.15	97.23	97.39
ResNet50v1	96.87	96.67	97.16	97.11

Tabla 7.15: Tasa de acierto obtenida para la base de datos LFW con el modelo ResNet. Se prueban los rangos de normalización de 0 a 1 y de -1 a 1, así como el base de datos a color y blanco y negro (BN).

Modelo	Growth Rate	Acierto (%) Color		Acierto (%) BN	
		$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$
DenseNet41	6	96.91	97.10	97.16	97.17
DenseNet77	12	97.28	97.40	97.53	97.71
DenseNet121	32	96.69	97.09	97.28	97.33

Tabla 7.16: Tasa de acierto obtenida para la base de datos LFW con el modelo DenseNet. Se prueban los rangos de normalización de 0 a 1 y de -1 a 1, así como el base de datos a color y blanco y negro (BN).

Modelo	Dropout	Ruido (std)	Acierto (%) Color		Acierto (%) BN	
			$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$
MobileNetv2 36	No	0.1	97.52	97.74	97.50	97.46
MobileNetv2 54	No	0.0	97.63	97.69	97.56	97.55
MobileNetv2 45	No	0.2	97.29	97.30	97.41	97.42
MobileNetv2 36	Si	0.1	95.09	95.19	94.69	95.01

Tabla 7.17: Tasa de acierto obtenida para la base de datos LFW con el modelo MobileNetv2. Se prueban los rangos de normalización de 0 a 1 y de -1 a 1, así como el base de datos a color y blanco y negro (BN) utilizando SGD.

Cómo observamos, el mejor resultado es obtenido por el modelo MobileNetv2 con una tasa de acierto del 97.74 %. Este resultado mejora los alcanzados en el estado del arte (véase la Sección 3.5) para esta base de datos y establecen una nueva meta. Además, los resultados obtenidos por el resto de modelos no distan tanto de esta marca y cómo la diferencia entre tasas de acierto no es significativa, decidimos analizar estos resultados teniendo en cuenta uno de los objetivos de nuestro trabajo, la rapidez, el tiempo de inferencia de las imágenes a través del modelo. Con los tiempos de inferencia presentados en este capítulo así como con las tasas de acierto, podemos construir la Tabla presentada en la Figura 7.1.

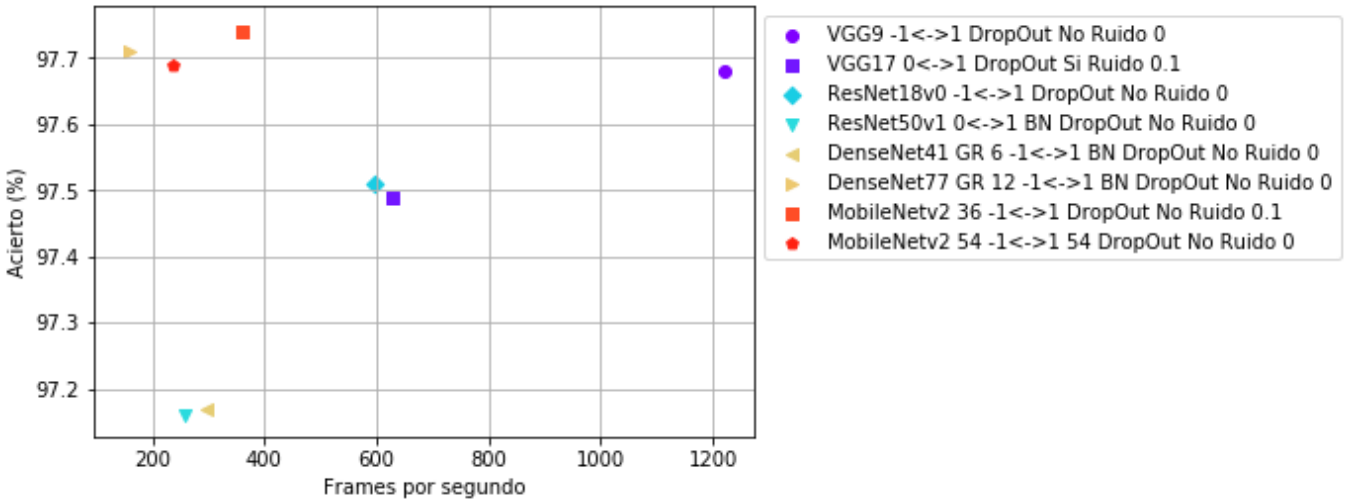

Figura 7.1: Comparación inferencia/accuracy de algunas configuraciones probadas para la base de datos de LFW.

Gracias a esta tabla podemos concluir que, aunque la tasa de acierto obtenida por el modelo MobileNetv2 es mejor, si deseáramos portar nuestro modelo a un dispositivo con recursos más reducidos, la velocidad que nos proporciona el modelo VGG, es parte por su simplicidad y menor número de operaciones, hace que nos decantamos por este. Además la principal razón por la que nos decantemos por un modelo u otro sería el tiempo de inferencia ya que las tasas de acierto son muy similares, obteniendo tasas muy similares tanto para el caso de blanco y negro como a color.

Por último recordar que la base de datos LFW es una base de datos desbalanceada donde el 77.4 % de las muestras son masculinas. Podemos analizar los errores cometidos por nuestros clasificadores y ver si el desbalanceo de muestras es notable en el funcionamiento de los mismos. Para ello probamos a clasificar las muestras de test de los diferentes folds con dos modelos diferentes: VGG y MobileNetv2 (figuras 7.2 y 7.3 respectivamente). Para el modelo VGG las muestras incorrectas dentro del conjunto masculino representa el 1.4 % de estas mientras que para las femeninas el 6.6 %, con esto vemos que el desbalanceo de las muestras es notorio levemente, si tenemos en cuenta que el 77.4 % de las muestras son masculinas, y el modelo ha sido capaz de generalizar. Para el modelo MobileNetv2 el porcentaje de mujeres es similar con el 6.1 % de las muestras erróneas y el de hombre también con el 1.6 % por lo que ambos modelos generalizan de forma similar.

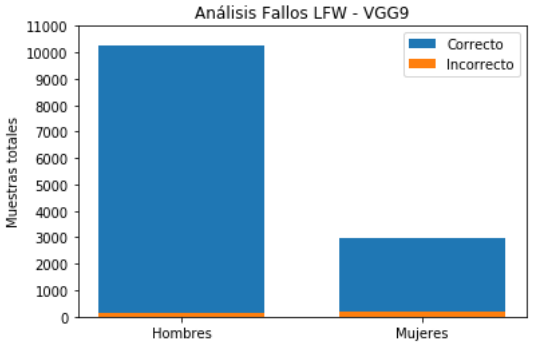

Figura 7.2: Fallos por sexo con el modelo VGG9 para la base de datos LFW.

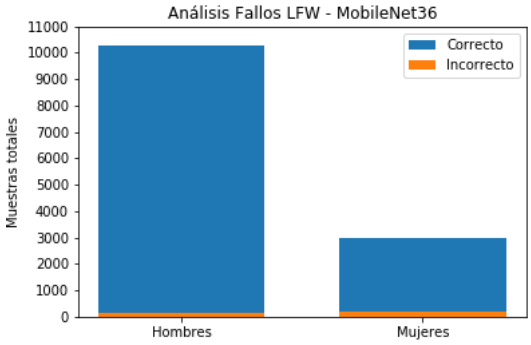

Figura 7.3: Fallos por sexo con el modelo MobileNetv2 36 para la base de datos LFW.

7.2.2 Groups/Gallagher

Al igual que para la base de datos anterior, probamos los modelos siguiendo las mismas configuraciones para la base de datos GROUPS. En las siguientes tablas mostramos los diferentes resultados obtenidos por los modelos VGG (Tabla 7.18), ResNet (Tabla 7.19), DenseNet (Tabla 7.20) y MobileNetv2 (Tabla 7.21), con cada una de las configuraciones mencionadas anteriormente. El mejor resultado de cada modelo ha sido resaltado cambiando el fondo de su celda.

Modelo	**Dropout**	**Ruido (std)**	**Acierto (%) Color**		**Acierto (%) BN**	
			$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$
VGG9	No	0.0	91.49	91.55	91.50	91.68
VGG9	Si	0.1	90.87	90.92	91.03	91.08
VGG13	Si	0.1	74.59	82.92	76.43	70.14
VGG14	Si	0.1	91.21	91.28	91.46	91.42
VGG17	Si	0.1	91.31	91.19	85.27	87.00

Tabla 7.18: Tasa de acierto obtenida para la base de datos GROUPS con el modelo VGG. Se prueban los rangos de normalización de 0 a 1 y de -1 a 1, así como el base de datos a color y blanco y negro (BN).

Modelo	**Acierto (%) Color**		**Acierto (%) BN**	
	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$
ResNet18v0	91.47	91.32	91.63	91.18
ResNet18v1	91.42	91.43	91.17	91.26
ResNet18v2	90.06	90.05	90.90	90.58
ResNet50v1	89.41	89.33	89.63	90.50

Tabla 7.19: Tasa de acierto obtenida para la base de datos GROUPS con el modelo ResNet. Se prueban los rangos de normalización de 0 a 1 y de -1 a 1, así como el base de datos a color y blanco y negro (BN).

Modelo	Growth Rate	Acierto (%) Color		Acierto (%) BN	
		$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$
DenseNet41	6	90.48	90.10	90.17	90.45
DenseNet77	12	91.20	91.50	91.25	91.55
DenseNet121	32	90.69	91.07	91.00	91.17

Tabla 7.20: Tasa de acierto obtenida para la base de datos GROUPS con el modelo DenseNet. Se prueban los rangos de normalización de 0 a 1 y de -1 a 1, así como el base de datos a color y blanco y negro (BN).

Modelo	Dropout	Ruido (std)	Acierto (%) Color		Acierto (%) BN	
			$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$	$0 \leftrightarrow 1$	$-1 \leftrightarrow 1$
MobileNetv2 36	No	0.1	91.73	91.61	91.55	91.64
MobileNetv2 54	No	0.0	91.43	91.36	91.37	91.69
MobileNetv2 45	No	0.2	91.25	91.08	91.10	91.59
MobileNetv2 36	Si	0.1	83.15	82.37	83.42	84.01

Tabla 7.21: Tasa de acierto obtenida para la base de datos GROUPS con el modelo MobileNetv2. Se prueban los rangos de normalización de 0 a 1 y de -1 a 1, así como el base de datos a color y blanco y negro (BN) utilizando SGD.

El mejor resultado es obtenido de nuevo por el modelo MobileNetv2, con una tasa de acierto del 91.73 %, mejorando de nuevo los resultados alcanzados en el estado del arte (véase la Sección 3.5) para esta base de datos. Además, los resultados obtenidos por el resto de modelos no distan tanto de esta marca y cómo la diferencia entre tasas de acierto no es significativa, decidimos analizar estos resultados teniendo en cuenta uno de los objetivos de nuestro trabajo, la rapidez, el tiempo de inferencia de las imágenes a través del modelo. Con los tiempos de inferencia presentados en este capítulo así como con las tasas de acierto, podemos construir la Tabla presentada en la Figura 7.4. En esta volvemos a apreciar que el modelo VGG es más veloz que el MobiletNetv2, obteniendo tasas de acierto similares. Aquí sería interesante tener en cuenta que mientras el modelo VGG9 cuenta con 2.8 millones de parámetros, el modelo MobileNetv2 es más liviano con tan solo 429.000.

De igual forma que para la base de datos LFW, es interesante analizar los fallos que cometen los modelos para tener pistas sobre cómo mejorar estos. Aquí, a diferencia que en LFW, las clases están balanceadas por lo que tenemos el mismo número de muestras masculinas como femeninas. Así pues los errores también son repartidos de forma más o menos equitativa teniendo el 9.9 % de las muestras masculinas etiquetadas erróneamente y el 8.4 % para el casos de los rostros femeninos, utilizando el modelo VGG9 (Figura 7.5). Para el modelo los resultados son similares, obteniendo un diferencia menor con el 8.9 % de las muestras masculinas mal clasificadas y un 9.1 % para el caso de muestras femeninas (Figura 7.6). Con esto observamos que el desbalanceo de las muestras de LFW causa un desbalanceo en la tasa de acierto obtenida si la separamos por géneros.

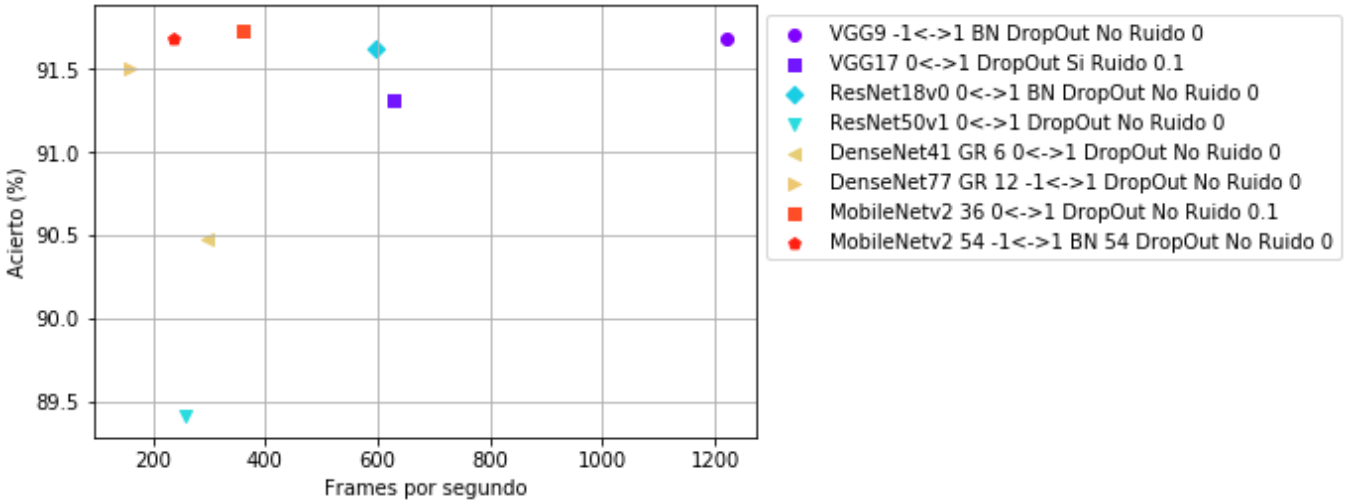

Figura 7.4: Comparación inferencia/accuracy de algunas configuraciones probadas para la base de datos de GROUPS.

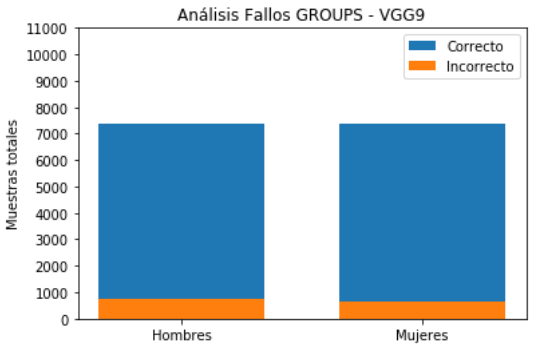

Figura 7.5: Fallos por sexo con el modelo VGG9 para GROUPS.

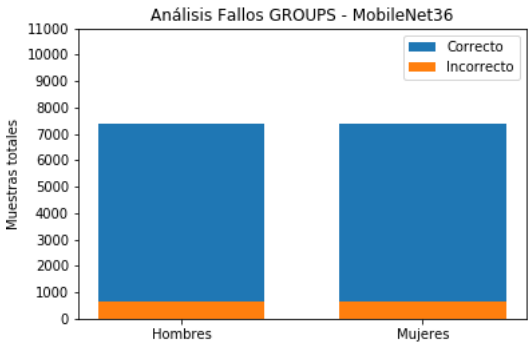

Figura 7.6: Fallos por sexo con el modelo MobileNetv2 36 para GROUPS.

Capítulo 8

Conclusiones

En este último capítulo vamos a resumir las conclusiones que extraemos del trabajo aquí presentado. Podemos dividir nuestro trabajo en dos grupos principales: Por una parte hemos estudiado el marco general de un problema de la clasificación de género, mientras que por otra parte hemos llevado de la teoría a la práctica algunos de los modelos más relevantes de redes neuronales en la actualidad. Para llevar a cabo nuestro primer objetivo hemos buscado y analizado los trabajos relacionados con el reconocimiento de género a través de las imágenes de los rostros, centrándonos así en una tarea de visión por computador, para estudiar cuál es la metodología empleada en este ámbito así como las bases de datos utilizadas. Existen numerosas bases de datos pero a la hora de elegir no todas contaban con un mismo método de entrenamiento y evaluación de los resultados y nos decantamos por dos que sí que lo hacían: la base de datos Labelled Faces in the Wild (LFW) y la base de datos Groups/Gallagher.

A partir del análisis de esta tarea, y en concreto centrándonos en las bases de datos elegidas, nos damos cuenta de que no existen trabajos que prueben el desempeño de los modelos de redes neuronales más conocidos en la actualidad por lo que nos motiva a realizar nuestro trabajo. Como resultado obtenemos que hemos sido capaces no sólo de implementar y utilizar correctamente los modelos propuestos, si no de mejorar los resultados de los compañeros que han realizado esta tarea bajo las mismas condiciones que nosotros, obteniendo una tasa de acierto del 97.74 % para la base de datos LFW y un 91.73 % para la base de datos GROUPS. También hemos visto cómo al realizar una tarea en la que no contamos con muchas clases en las que clasificar, los modelos más sencillos nos proporcionan unos mejores resultados a la vez que la carencia de operadores más complejos los hacen más veloces que el resto.

Como trabajo futuro existen varias posibilidades que podrían acarrear una mejora en los resultados obtenidos. Por una parte podemos plantearnos la utilización del esquema de ensamblaje de clasificadores, donde contamos con diferentes modelos que aportan su 'voto' a la hora de etiquetar una imagen dada, donde la clase más votada es la elegida. De igual forma podemos copiar este esquema donde en vez de contabilizar el voto como 1 o 0, sumamos las probabilidades dadas por los modelos de que pertenezca a la clase hombre o mujer. Otra tarea que hemos visto en varios trabajos y consideramos interesante consiste en probar modelos entre-

nados con una base de datos y ver cómo se comportan al presentarle rostros de otra base de datos diferente, probando así el poder de generalización del modelo entrenado. Con lo referente a nuestro objetivo de hacer este tipo de aplicaciones accesibles a dispositivos con recursos limitados, podríamos investigar y probar redes neuronales que trabajaran únicamente con aritmética entera. Por último, una forma de agilizar las redes sería la supresión de la etapa convolucional de las redes, haciendo que la inferencia se hiciera únicamente a través de una red *fully-connected*.

Bibliografía

[1] R. Adriana, B. Nicolas, K. S. Ebrahimi, C. Antoine, G. Carlo and B. Yoshua. Fitnets: Hints for thin deep nets. In Proc. ICLR, 2015.

[2] M. Moczulski, M. Denil, J. Appleyard and N. de Freitas. ACDC: A Structured Efficient Linear Layer. In Proc. ICLR, 2016.

[3] V. Santarcangelo, G. M. Farinella, and S. Battiato. Gender recognition: Methods, datasets and results. In Multimedia and Expo Workshops (ICMEW), 2015 IEEE International Conference on. IEEE, pp. 16, 2015.

[4] C.B. Ng, Y.H. Tay, B.M. Goi. Vision-based human gender recognition: A survey. In CoRR, 2012.

[5] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In IEEE Trans. Pattern Anal. Mach. Intelligence, 20(1):2338, January 1998.

[6] H. Schneiderman and T. Kanade. Object Detection Using the Statistics of Parts. In International Journal of Computer Vision 56(3), 151177, 2004.

[7] P. Viola and M. J. Jones. Robust real-time face detection. In Second International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing and Sampling, 2001.

[8] M. Villegas and R. Paredes. Comparison of Illumination Normalization Methods for Face Recognition. In Third COST 275 Workshop, 2005.

[9] J. Mansanet, A. Albiol, R. Paredes. Local deep neural networks for gender recognition. In Pattern Recogn. Lett. 70, 80–86, 2016.

[10] M. Villegas, R. Paredes, A. Juan, E. Vidal. Face verification on color images using local features. In CVPR Workshops., pp. 1–6, 2008.

[11] R. Paredes, J. C. Prez, A.Juan, E.Vidal. Local representations and a direct voting scheme for face recognition. In Workshop on Pattern Recognition in Information Systems, pp. 71–79, 2001.

[12] T. Ahonen, E. Rahtu, V. Ojansivu, and J. Heikkila. Recognition of blurred faces using local phase quantization. In ICPR, pp. 14, 2008.

[13] M. Felsberg and G. Sommer. The monogenic signal. In IEEE Trans. Signal Process., vol. 49, no. 12, pp. 31363144, Dec. 2001.

[14] Huu-Tuan Nguyen, Trinh Thi Ngoc Huong. Gender classification by LPQ features from intensity and Monogenic images. In Information and Computer Science, 2017 4th NAFOSTED Conference, 96-100, 2017.

[15] M. Castrillón-Santana, J. Lorenzo-Navarro, E. Ramón-Balsameda. Descriptors and regions of interest fusion for in- and cross-database gender classification in the wild. In Image and Vision Computing 57, 15-24, 2017.

[16] N. Dalal, B. Triggs. Histograms of Oriented Gradients for Human Detection. C. Schmid, S. Soatto, C. Tomasi (Eds.), International Conference on Computer Vision and Pattern Recognition (CVPR), 2, pp. 886–893, 2005.

[17] T. Ojala, M. Pietikäinen, T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. 24 (7), 971–987, 2002.

[18] T. Ahonen, A. Hadid, M. Pietikäinen. Face description with local binary patterns: application to face recognition. IEEE Trans. Pattern Anal. Mach. Intell. 28 (12), 2037-2204, 2006.

[19] L. Liu, P. Fieguth, L. Zhao, Y. Long, G. Kuang. Extended local binary patterns for texture classification. Image and Vision Computing 30 (2), 86-99, 2012.

[20] Z. Chai, Z. Sun, T. Tan, H. Mendez-Vazquez. Local salient patterns - A novel local descriptor for face recognition. International Conference on Biometrics (ICB), 2013.

[21] O. García-Olalla, E. Alegre, L. Fernández-Roble, V. González-Castro. Local Oriented Statistics Information Booster (LOSIB) for Texture Classification. International Conference on Pattern Recognition (ICPR), 2014.

[22] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc, 2012.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overtting. J. Mach. Learn. Res., 15(1):1929–1958. ISSN 1532–4435, 2014.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 323:533–536, doi: 10.1038/323533a0, 1986.

[25] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980, 2014.

[26] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In G. J. Gordon and D. B. Dunson, editors, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS- 11), volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.

[27] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13:281305, Feb. 2012.

[28] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[29] Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[30] G. B. Huang, M. A. Mattar, H. Lee, and E. G. Learned-Miller. Learning to align from scratch. In NIPS, pages 773–781, 2012.

[31] FIPA.Particiones para la evaluación de genero en LFW. 2011. `http://fipa.cs.kit.edu/download/LFW-gender-folds.dat`.

[32] Gallagher, A. and Chen, T. Understanding images of groups of people. In Proc. CVPR, 2009.

[33] Dago-Casas, P., González-Jiménez, D., Yu, L. L., and Alba-Castro, J. L. Single- and cross- database benchmarks for gender classification under unconstrained settings. In ICCV Workshops, pages 2152-2159. IEEE, 2011.

[34] Ehsan Fazl-Ersi, M. Esmaeel Mousa-Pasandi, Robert Laganière, Maher Awad. Age and gender recognition using informative features of various types. IEEE International Conference on Image Processing, ICIP, 2014.

[35] D. Gabor. Theory of Communication. In Journal of Institute for Electrical Engineering, 93, 1946.

[36] FIPA.Particiones para la evaluación de genero en la base de datos Gallagher's. 2011. `http://fipa.cs.kit.edu/download/Gallagher_gender_5folds.txt`.

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[38] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In arXiv preprint arXiv:1512.03385, 2015.

[39] Huang, Gao, Liu, Zhuang, and Weinberger, Kilian Q. Densely Connected Convolutional Networks. In arXiv:1608.06993, 2016.

[40] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In arXiv:1801.04381, 2018.

[41] Andrew G. Howard, M. Zhu, Bo Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In CoRR, abs/1704.04861, 2017.

[42] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. In Technical report, arXiv 1212.5701, 2012.

[43] X. Li, S. Chen, X. Hu, J. Yang. Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift. In arXiv:1801.05134v1 [cs.LG], 16, 2018.

[44] Paszke, Adam and Gross, Sam and Chintala, Soumith and Chanan, Gregory and Yang, Edward and DeVito, Zachary and Lin, Zeming and Desmaison, Alban and Antiga, Luca and Lerer, Adam. Automatic differentiation in PyTorch. In NIPS-W, 2017.

Apéndice A

Funciones de activación

En el siguiente apéndice se muestran algunas de las funciones de activaciones que encontramos más frecuentemente en la implementación de las redes neuronales.

Lineal

$$f(z_i) = z_i$$

Sigmoidea

$$f(z_i) = \frac{1}{1+e^{-z_i}}$$

Tangente Hiperbólica

$$f(z_i) = \frac{e^{z_i}-e^{-z_i}}{e^{z_i}+e^{-z_i}}$$

Softmax

$$f(z) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Rectifier Linear Unit (ReLU)

$$f(z_i) = max(0, z_i)$$