Memoria de la práctica de Aprendizaje Automático 2

Simplicity is all you need

Victor Teodoro[1] & Diego Valcarce[2]

[1] Departamento de Ingeniería de Computadores

[2] Departamento de Ciencias de la Computación y Tecnologías de la Información

{[1]victor.teodoro, [2]d.valcarce}@udc.es

Diciembre de 2021

Resumen

En este trabajo presentaremos un único modelo *simple* que será capaz de predecir tanto el género como la edad de los individuos partiendo en exclusiva de una imagen facial. Para ello haremos uso de aprendizaje profundo y transferencia de aprendizaje. No construiremos un modelo arquitecturalmente complejo, sino que trataremos de sacar el máximo partido posible a modelos sencillos, teniendo muy en cuenta los sesgos presentes en los datos.

Palabras clave: Aprendizaje profundo, reconocimiento facial, transferencia de aprendizaje

1. Introducción

El problema que nos atañe consiste en el reconocimiento de género y edad en imágenes faciales, tanto atacando el problema de forma individual como conjunta. Este problema tiene casos de uso reales en nuestro día a día: por ejemplo, en el caso de que se volviese a realizar un confinamiento debido a la pandemia del COVID-19 y se volviesen a aplicar las restricciones más severas, se podría utilizar esta solución para controlar a la población que circula por la vía pública, ya que, durante parte del confinamiento, se limitaba la circulación de personas mayores a ciertas horas al día [Ministerio de Sanidad, 2020].

Otros usos podrían derivar de fines económicos y business intelligence: una pequeña empresa podría colocar varias cámaras en su local y de esta forma, mantener un registro de la población que suele acudir a su tienda y a qué horas, con fin de mejorar los ingresos teniendo en cuenta la audiencia.

En la sección 2 se abordará una explicación general de la complicación de este problema, además de realizarse un análisis exploratorio de los datos. En 3 se comentarán brevemente los métodos y materiales empleados. A continuación, en la sección 4 se comentarán los principales modelos que actualmente utiliza el sector para problemas similares. Finalmente, en 5 se desarrollarán en detalle las soluciones que proponemos, así como los resultados en 6.

2. Descripción del problema

La forma más sencilla de atacar este problema es mediante redes convolucionales (CNN, del inglés *Convolutional Neural Networks*), es por ello que construiremos redes profundas a partir de las mismas.

Además, tenemos una serie de objetivos que debemos cumplir en cuanto a lo que los errores de los modelos se refiere:

- El error del modelo que se encarga de predecir la edad debe tener un error absoluto medio -*Mean Absolute Error*- (MAE) 1 inferior a 15 años.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|x_i - \widehat{x}_i| \tag{1}$$

- En el caso del problema de la clasificación de género debe tener una precisión (*accuracy*) 2 superior al 89 %, curiosamente, debemos reseñar que el MAE y el accuracy son equivalentes si tanto x_i como $\widehat{x}_i$ son binarios.

$$\text{accuracy} = \frac{1}{N}\sum_{i=1}^{N}x_i = \widehat{x}_i \tag{2}$$

- Para el problema conjunto, debemos obtener un *Weighted MAE* (WMAE) 3 combinado de menos de 26, siendo x_i y $\widehat{x}_i$ los relativos a la edad e y_i e $\widehat{y}_i$, al género.

$$\text{WMAE} = \frac{1}{N}\sum_{i=1}^{N}\left[|x_i - \widehat{x}_i| + 100|y_i - \widehat{y}_i|\right] \tag{3}$$

2.1. Sobre los datos

Contamos con un total de tres *datasets* compuestos con imágenes faciales de celebridades:

- Un conjunto de entrenamiento, con $\sim$ 120,000 observaciones.
- Un conjunto de validación, con 200 imágenes que podemos utilizar para validar nuestros modelos, pero no para entrenarlos.
- Un conjunto de test, con 800 imágenes, conjunto del cual no tenemos etiquetas, ya que se utiliza para evaluar los modelos a través de kaggle.

Es importante partir de la premisa de que se nos informó de que el conjunto de validación se generó a partir de la misma distribución que el de test, lo cual utilizaremos para mejorar los resultados y que se estudiará más adelante 5. En la figura 1 se puede observar una muestra de seis imágenes del mismo. Nótese la presencia de imágenes tanto a color como en blanco y negro.

Figura 1: Muestra de las imágenes del *dataset*

Aunque esto pueda parecer un gran problema para la red, será el menor de ellos, el *dataset* presenta los siguientes problemas:

- Entre las imágenes encontramos cosas *extrañas*. En la figura 2 observamos un pequeño ejemplo de las imágenes que nos encontramos en el conjunto de test, lo cual limita los resultados a los que podremos aspirar.

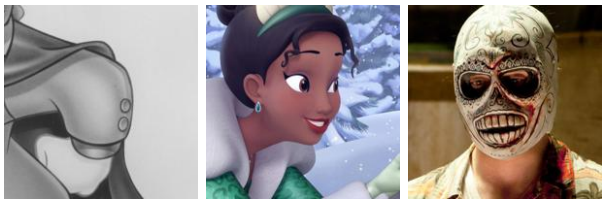


Figura 2: Muestra de imágenes *raras*

- Tenemos una cantidad ingente de imágenes mal clasificadas (véanse las imágenes de la figura 3 con las etiquetas de género y edad marcadas por el archivo de metadatos).

(a) "Hombre" (b) "Mujer"

(c) "38 años" (d) "Mujer, 25"

Figura 3: Imágenes mal clasificadas

- No sólo tenemos imágenes mal clasificadas, sino que también tenemos imágenes con ciertos problemas de resolución, figura 4, (probablemente imágenes mal capturadas o deformadas en el proceso de almacenamiento).

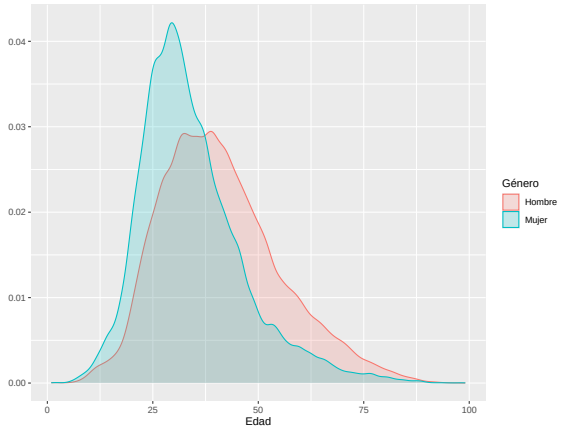
Figura 4: Fotografías mal tomadas

- En el caso del conjunto de entrenamiento, figura 5a, la distribución es campaniforme pero con asimetría positiva. Hay una ligera, pero notable diferencia entre la distribución de las edades para hombres y mujeres, habiendo mayor densidad para edades altas en el caso de los hombres.

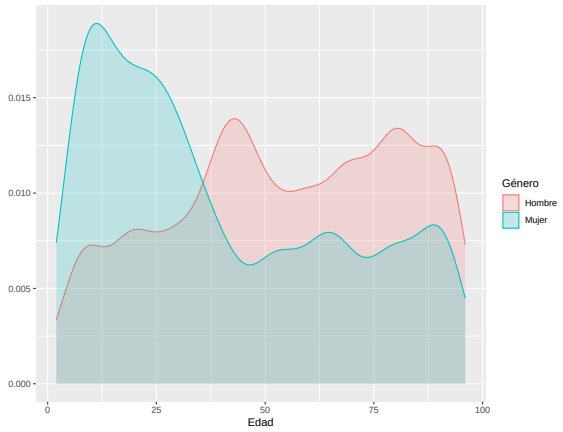
  Por otra parte, en validación, figura 5b, la diferencia de distribución en edades es mucho más pronunciada. Podemos ver que donde más densidad hay para los hombres hay menos para las mujeres y viceversa. Además, si se añaden estas densidades podemos apreciar que la distribución de la edad sin condicionar al sexo es prácticamente una uniforme, a diferencia de la distribución en entrenamiento, que se asemeja más a una normal (sin llegar a serlo).

- En el caso de los desbalanceos de clase entre los géneros, figura 6, tenemos en el conjunto de entrenamiento cerca de un 60 % de imágenes de hombres, y el restante de mujeres, prácticamente idéntico a los porcentajes de validación.

Para solventar el problema relativo a la figura 4, hemos optado por hacer un zoom a cada una de las imágenes, ya que como se puede observar en las figuras 1, 2 y 3, todas las caras están centradas en el centro de la imagen, por lo que no estamos perdiendo información



(a) Densidades en train



(b) Densidades en validación

Figura 5: Distribuciones de las edades en train y validación

en el camino y, empíricamente, nos ha reportado mejores resultados.

3. Métodos y materiales

Para el entrenamiento de los modelos propuestos utilizamos el framework de Tensorflow [Abadi et al., 2015] con Keras [Chollet et al., 2015] en Python [Van Rossum and Drake Jr, 1995]. Los experimentos se realizaron en 3 plataformas distintas: Google Colaboratory[1], notebooks de Kaggle[2] y en local en una máquina con un Intel i7 de 6 núcleos a 2,6 GHz y una GPU AMD Radeon Pro 5300M utilizando la librería de Apple[3].

Para las figuras hemos utilizado la librería `ggplot2`[Wickham et al., 2019] desde el lenguaje de programación R [R Core Team, 2020], para mayor belleza estética.

[1] `https://research.google.com/colaboratory/`
[2] `https://www.kaggle.com/code`
[3] `https://developer.apple.com/metal/tensorflow-plugin/`

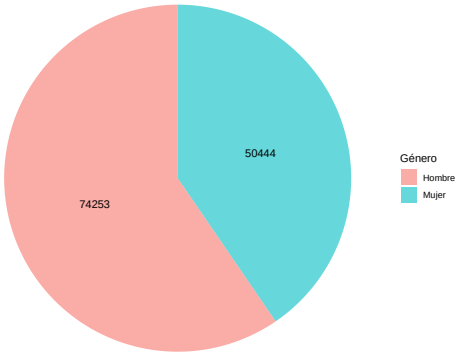


Figura 6: Gráfico de sectores del género

4. Estado del arte

En la literatura, la forma de afrontar este problema siempre termina cayendo en el uso de redes profundas con capas convolucionales. [Levi and Hassner, 2015] proponen un modelo sencillo compuesto por redes convolucionales y varias capas conectadas hacia delante y son capaces de conseguir un 86.8 % de precisión en el género. En [Dehghan et al., 2017] hacen una comparativa con varios modelos de la industria, entre ellos Microsoft, que declara un MAE de 7.62 para la predicción de la edad y un 90.86 % en la clasificación de género.

En el caso de [Duan et al., 2018], proponen un modelo híbrido entre una red convolucional y una red de aprendizaje extremo, reportando resultados de un MAE de 3.44 para el problema de la edad y una precisión del 88 % para el género.

5. Nuestra aproximación

Primeramente, probamos a utilizar algunos modelos de la literatura para entrenarlos con estos datos y, posteriormente, comparar nuestros propios modelos con los modelos consolidados de la industria.

Entre los modelos probados se encuentran arquitecturas e ideas sacadas de modelos como:

- Autoencoders para proyectar las entradas sobre un espacio latente de menor dimensión y entrenar sobre este espacio [Chen et al., 2017].
- Modelos pensados para ser más ligeros y aptos para realizar inferencia en dispositivos móviles como: GhostNets [Han et al., 2020], MobileNets [Howard et al., 2017, Sandler et al., 2018, Howard et al., 2019].
- Transformers para imágenes o Vision Transformers (ViT) [Dosovitskiy et al., 2020].

También probamos a entrenar sobre las imágenes transformadas, pasándolas a blanco y negro, ecualizando su histograma y transformándolas mediante una transformada de Wavelets de Haar

[Huang and Aviyente, 2008, Fujieda et al., 2018], ya que intuitivamente podría parecer que utilizando dicha transformada, recalcaríamos información frecuencial interesante, como las arrugas de la piel, pero sin buenos resultados.

Todo esto concluyó en que estábamos construyendo modelos de alta complejidad (gran número de capas ocultas y gran cantidad de parámetros); en esta situación, nos encontramos con que no obtuvimos ningún modelo con resultados aceptables: los modelos realizaban mucho sobreentrenamiento, ya que se obtenían errores muy pequeños en entrenamiento, pero errores mayores en validación y al subir resultados a Kaggle, obteníamos resultados con altas variaciones frente a los de validación.

Es por esto que optamos por afrontar el problema desde otra perspectiva: comenzamos con el modelo más sencillo que pudimos (una capa densa de una neurona) y fuimos añadiendo capas y parámetros de forma incremental hasta que dejamos de ver mejoras significativas (a través de la información que nos proporcionan los intervalos de predicción de *Boostrap* 6.1, tratando de conservar las menores diferencias posibles entre validación y entrenamiento); una vez encontrado este punto de inflexión, tratamos de optimizar cada uno de los parámetros involucrados en las redes. Utilizando redes pequeñas:

- Evitamos el problema del sobreentrenamiento. Sin embargo, en vez de dejar la corrección del problema en manos de métodos de regularización como en [Hinton et al., 2012b, Lee and Lee, 2020], intentaremos *evitar* el problema utilizando los mínimos parámetros posibles.

  Esto en un problema de optimización convexa, lo que se traduce en que intentaremos mantener la dimensión de Vapnik-Chervonenkis [Vapnik and Chervonenkis, 2015] más pequeña posible, para así mantener la cota del error de generalización más pequeña posible y evitar el sobreentrenamiento. Como se comenta en [Goodfellow et al., 2016], en un problema de optimización no convexa no está demostrado cómo calcular esta cota o la dimensión de Vapnik-Cherbonenkis. Sin embargo, asumimos que el comportamiento será similar y serán preferibles modelos de pocos parámetros frente a muchos por el principio de parsimonia.

  Además, en trabajos realizados anteriormente [Teodoro and Valcarce, 2021], pudimos comprobar empíricamente que soluciones más sencillas abaratan costes, tiempos y en muchas ocasiones, mejoran resultados. Saber explotar el problema, es mejor que dejar que una red aprenda por si misma.

- Reducimos considerablemente los tiempos de ejecución: los modelos propuestos tienen menos de 200000 parámetros, mientras que los que podemos encontrar en la literatura presentan siempre cifras muy superiores a los 100 millones (138 en el caso de VGG[Simonyan and Zisserman, 2014]). Es por esto también, que estos modelos son más éticos al consumir menos recursos hardware y como consecuencia, reducir la contaminación que su entrenamiento produce[Strubell et al., 2019].

  Con esto, también reducimos los tiempos de inferencia, que en este caso no son muy significativos al tener un conjunto de test de 800 imágenes, pero en caso de desplegar el modelo de forma industrial, sería una diferencia cuanto menos, importante.

- Proporcionamos una estimación más precisa de los parámetros de la red, ya que estos tendrán más grados de libertad con los que entrenarse.

Como conclusión, utilizamos una red pequeña para cada problema, compartiendo la parte convolucional, pero con distintas capas totalmente conectadas de salida.

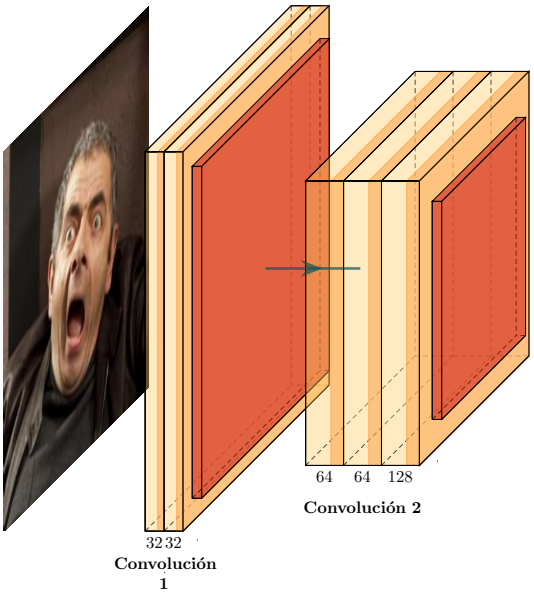


Figura 7: La arquitectura convolucional de nuestras redes

La arquitectura base de la que parten nuestras redes se puede ver en la tabla 1 y la figura 7. Esta arquitectura se compone de dos bloques formados por dos y tres convoluciones, con una capa de MaxPooling cada uno. Todas las convoluciones utilizan kernel de tamaño 3×3 y función de activación ReLU, la primera de cada bloque con strides 2 y entre los bloques hay una capa de MaxPooling2D (representadas en color naranja oscuro). Asimismo, entre cada una de las capas se realiza BatchNormalization. La entrada a la red tiene además dos capas destinadas a realizar aumentación de datos: volteo horizontal aleatorio y una traslación también aleatoria, para concluir finalmente a realizar un reescalado (1/255) para mantener las entradas a la red en el intervalo $[0, 1]$.

Todos los modelos presentados de aquí en adelante se optimizaron utilizando el algoritmo *Root Mean Square Propagation* (RMSprop) [Hinton et al., 2012a] usando

Tipo	# filtros	Kernel	Strides
Bloque de entrada			
RandomFlip(horizontal)	–	–	–
RandomRotation(0.05, 0.05)	–	–	–
Rescaling(1 / 255)	–	–	–
Bloque I			
Convolución	32	3×3	2
BatchNormalization	–	–	–
Convolución	32	3×3	–
BatchNormalization	–	–	–
MaxPooling.	–	–	2
Bloque II			
Convolución	64	3×3	2
BatchNormalization	–	–	–
Convolución	64	3×3	–
BatchNormalization	–	–	–
Convolución	128	3×3	–
BatchNormalization	–	–	–
MaxPooling	–	–	2

Cuadro 1: Arquitectura de nuestra red base

una tasa de aprendizaje de 10^{-3}, decay rate $\rho = 0{,}9$ y sin momentum. Hicimos, entre otras, las siguientes pruebas con el algoritmo de optimización:

- Usando otras tasas de aprendizaje más altas: $2 \cdot 10^{-3}, 5 \cdot 10^{-3}, 10^{-2}, \ldots$
- Decay rates más bajos $0{,}5, \ldots, 0{,}8$ para que gradientes de iteraciones pasadas tuvieran menor peso.
- Usando un *scheduler* para bajar la tasa de aprendizaje dividiéndola entre 2 o multiplicándola por una exponencial decreciente pasado cierto número de epochs.

Pero ninguna de estas pruebas dieron mejores resultados.

5.1. Edad

Partiendo de la parte convolucional discutida en la tabla 1, se añadieron dos capas totalmente conectadas de 64 y 32 neuronas, con activación ReLU. La salida de la red la proporcionó una única neurona con activación lineal.

Inicialmente la activación que usamos para esta capa de salida era una ReLU, ya que las edades tienen que tomar valores positivos y el añadir la ReLU no solo proporcionaba una salida con más fiabilidad que utilizar una salida lineal (que podría producir edades negativas), sino que los modelos que usaban ReLU entrenaban más rápido; no obstante, en 5.1.3 se introducirá una estandarización a las edades, por lo que dejará de ser aplicable este razonamiento, de ahí que volvamos a usar la lineal.

A lo largo de esta sección se tratarán: los pesos 5.1.1 que tienen las diferentes edades a la hora de entrenar, las funciones de pérdida que mejores resultados nos han reportado 5.1.2, la estandarización de las etiquetas 5.1.3 y finalmente el proceso de entrenamiento 5.1.4.

5.1.1. Pesos

Uno de los mayores problemas que tendremos que afrontar en el entrenamiento de modelos para la edad es el de las diferencias entre la distribuciones de los conjuntos de entrenamiento y validación.

Al tener una distribución campaniforme en entrenamiento, figura 5a, un modelo entrenado sobre esos datos probablemente tendría poca capacidad predictiva sobre las colas de la distribución, esto es, para individuos de poca edad o edad muy avanzada. Además, en el conjunto de validación tenemos una distribución uniforme, figura 5b, lo cual hace que este problema se agrave, ya que contaremos con más densidad de ejemplos de estos grupos en validación que en entrenamiento, haciendo que los resultados sean inevitablemente peores.

Para intentar corregir este sesgo, lo que haremos será indicarle al modelo que preste menor atención a los ejemplos de edades muy representadas y más atención a las imágenes de edades poco representadas. Esto se puede hacer tratando la edad como un problema de clasificación como se hizo en [Dehghan et al., 2017, Liu et al., 2012]. Sin embargo, optaremos por otro método.

Lo que haremos será realizar una estimación núcleo de la densidad de las edades en el conjunto de entrenamiento, figura 8a, como las que se vieron en la figura 5a y calcular la densidad de probabilidad correspondiente a cada una de las edades presentes en el *dataset*. Posteriormente, asignaremos pesos (*sample weights*) a cada una de las imágenes según la ecuación 4 donde f representa la función de densidad estimada (resultado en la figura 8b).

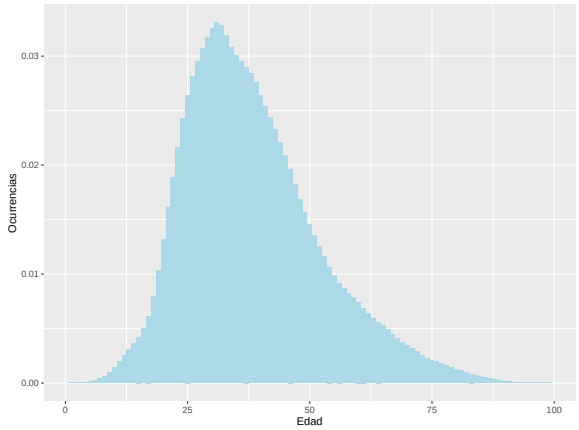
$$\text{Peso Imagen}_i = \frac{\text{máx}_x\, f(x)}{f(\text{Edad}_i)} \tag{4}$$

De esta forma haremos que las imágenes correspondientes a las edades más representadas tengan peso 1 y las correspondientes a las menos representadas tengan un peso mucho mayor.

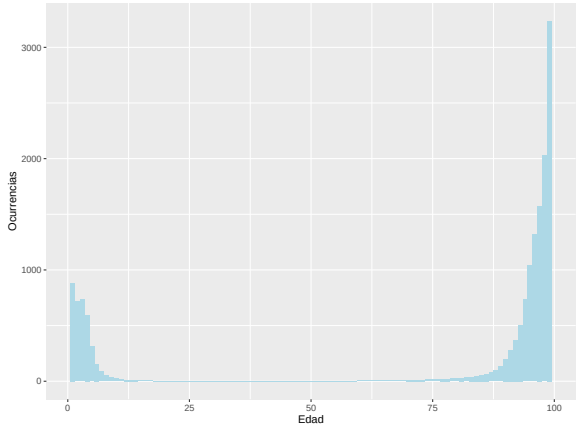
Una ventaja de este método usando una estimación de la densidad frente a contar el número de imágenes correspondientes a la edad y dividirlo entre la edad con más representación es que podemos obtener mejor información local. Al ser la estimación de tipo núcleo una convolución del número de imágenes para cada edad con una función de ventana, las estimaciones resultantes no tendrán en cuenta solo cuántas imágenes hay para una edad sino también para edades vecinas.

Esto puede ser útil para pronunciar la carencia de datos para edades menos representadas y que cuentan con menos representación local y que se tenga menos en cuenta edades muy representadas, con una vecindad con abundantes datos.

Un problema que tenemos al calcular los pesos así, será que se generarán pesos muy extremos, por lo que se



(a) Pesos obtenidos para las edades en el conjunto de entrenamiento a partir de muestras de una estimación núcleo de la densidad



(b) Inversa de la estimación núcleo

Figura 8: Estimación núcleo de la densidad de las edades

puede producir tanto el desvanecimiento del gradiente como su explosión. En nuestro dataset contamos con variables de pesos de más de 3000, como se puede ver en 8b. Utilizamos tres métodos para corregir esto:

1. Coger la raíz cuadrada de estos pesos 9. Ya que conservaría la ordenación de los pesos pero cambiaría su magnitud, haciendo que haya valores menos extremos.

2. Pasar las densidades originales al intervalo $[1, 2]$ y posteriormente aplicarles 4. De esta forma, obtendríamos pesos en el intervalo $[1, 2]$ y solucionaríamos el problema de la explosión del gradiente (imagen 10). Es más, en el caso de trabajar con la edad estandarizada esta transformación es muy positiva, ya que al trabajar con variables estandarizadas, si pensamos en el caso de una distribución normal estándar $\mathcal{N}(0, 1)$, la probabilidad de que estas variables tomen un valor en el rango $[-2, 2]$ es aproximadamente del 95 %, con lo cual estaríamos ante valores "esperables" y también evitaríamos posibles explosiones del gradiente en ese intervalo.

   Aunque lo descrito es lo correcto desde el punto de vista teórico, hemos optado por introducir a mayores una raíz cuadrada 11 para acotar ligeramente más los pesos, estando ahora en el intervalo $[1, \sqrt{2}]$, lo que se tradujo en una leve mejora.

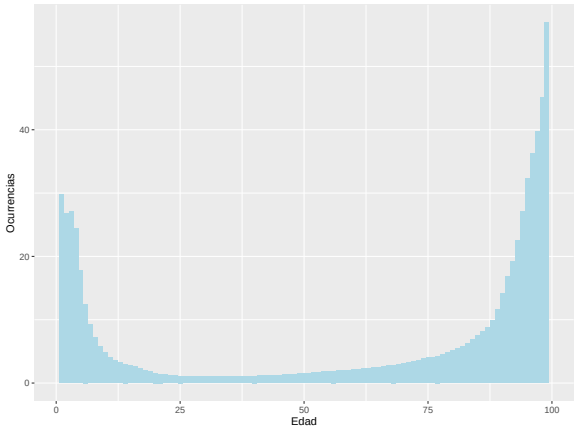


Figura 9: Estandarizar por la raíz cuadrada

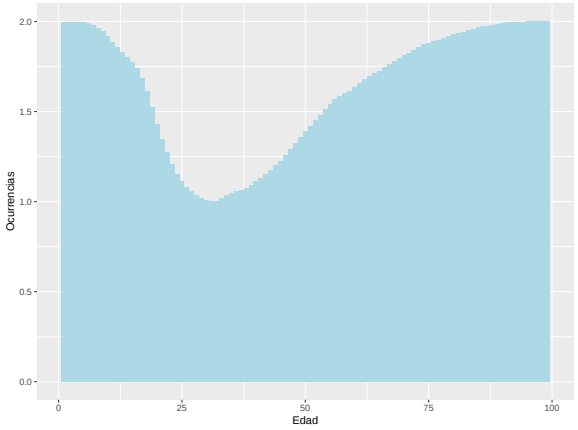


Figura 10: Estandarización en el intervalo $[1, 2]$

5.1.2. Funciones de pérdida

Inicialmente, entrenamos las redes utilizando el error cuadrático medio (MSE) como función de pérdida porque un resultado muy conocido de la estadística es que, para una variable aleatoria, se cumple que $\text{RMSE} = \sqrt{\text{MSE}} \geq \text{MAE}$. Esto es debido a la desigualdad de Cauchy-Schwarz para espacios de probabilidad [Klenke, 2013] y se puede ver, por ejemplo, en 5.

$$\begin{aligned} \text{Var}(X) &\geq 0 \\ \text{Var}(X) &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ \mathbb{E}[X^2] &\geq \mathbb{E}[X]^2 \\ \mathbb{E}[|X - \widehat{X}|^2] &\geq \mathbb{E}[|X - \widehat{X}|]^2 \\ \text{MSE} &\geq \text{MAE}^2 \end{aligned} \tag{5}$$

De esta forma, pensamos que sería preferible utilizar el MSE como función de pérdida, ya que decrementos lineales del MSE pueden provocar decrementos cuadráticos en el MAE. Así pues, podríamos conseguir

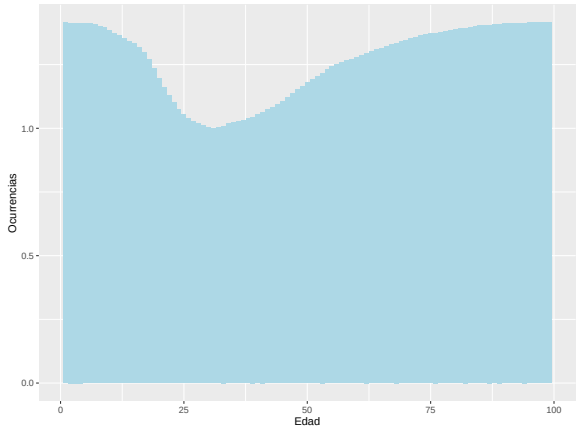


Figura 11: Estandarización en el intervalo $[1, \sqrt{2}]$

un mejor resultado para la métrica objetivo y de forma más rápida.

Sin embargo, uno de los mayores problemas a los que nos enfrentamos en este conjunto de datos es la presencia de datos mal etiquetados. Tenemos datos de personas ancianas etiquetados como infantes y viceversa 3, y esto sería muy negativo para el entrenamiento del modelo.

En base a esto, decidimos partir de otro resultado muy conocido de la estadística [Wooldridge, 2015], y es que el estimador mínimo cuadrático, es decir, el estimador que minimiza el MSE, es la media condicionada y el estimador de mínimas desviaciones absolutas, el que minimiza el MAE, es la mediana condicionada.

Debido a esto, una red entrenada con el MSE como función de coste (y que haya alcanzado el mejor MSE posible) habrá aprendido a estimar la media de la edad condicionada a los datos de entrenamiento y los parámetros de la red neuronal.

Por otra parte, una red entrenada con el MAE como función objetivo (y que haya alcanzado el MAE óptimo), habrá aprendido a estimar la mediana de la edad condicionada a los datos y los parámetros de la red neuronal.

Así pues, para este problema sería preferible el MAE frente al MSE, ya que la mediana es una medida de tendencia central robusta frente a datos atípicos, mientras que la media es muy sensible. Es decir, si una red usa MSE y le pasemos imágenes de una persona de edad avanzada mal etiquetada como muy joven, estaremos contaminando su concepto de persona de edad avanzada de forma más fuerte, ya que estaremos modificando $\mathbb{E}$[Edad|Edad aparente] en su predicción. En cambio, una red que use el MAE será más cautelosa frente estos datos atípicos, ya que estaríamos modificando su conocimiento sobre Med[Edad|Edad aparente], que es muy robusta frente a datos atípicos y, por tanto, datos mal etiquetados.

No obstante, preferimos no utilizar el MAE como función de pérdida porque esto trae consigo dos inconvenientes:

- Podría tardar más en entrenar que utilizando el MSE, por lo comentado anteriormente.
- El MAE no es una función derivable en todo $\mathbb{R}$, lo cual podría causar problemas (aunque sutiles) de estabilidad o convergencia en la red.

Lo que hicimos fue utilizar una función de pérdida que combine tanto las buenas propiedades del MSE como las del MAE: la loss de Huber [Huber, 1992], que se puede ver en 6. Cabe mencionar que esta es una de las posibles versiones de la función de Huber con corrección y no es exactamente igual que la original, pero es la que viene implementada por defecto en Keras.

$$\text{Huber}(x) = \begin{cases} \frac{1}{2}x^2 & \text{si } |x| \leq \delta \\ \frac{1}{2}\delta^2 + \delta\left(|x| - \delta\right) & \text{si } |x| > \delta \end{cases} \tag{6}$$

Esta función de pérdida depende de una variable x, que es el error de predicción, y un parámetro δ, un umbral de error que fijaremos. Lo que hará la función de Huber será utilizar el MSE o el MAE en función de si el error es mayor que δ.

Así pues, podremos aprovechar el efecto del MAE, que es robusto a atípicos, en los sitios donde más lo necesitemos (estamos cometiendo errores muy altos, puede haber posible contaminación del aprendizaje de la red por culpa de imágenes mal etiquetadas) y los efectos del MSE (es más rápido, aprende una media) donde menos nos preocupen los datos atípicos (tenemos un buen error, posiblemente no está causado por datos mal etiquetados). Además, esta función es diferenciable en todo $\mathbb{R}$ al utilizar el MSE para errores pequeños, evitando el posible problema de estabilidad que podría venir dado por el MAE.

5.1.3. Estandarización

Un preprocesado que haremos será la estandarización de las edades mediante media y varianza, ya que el darle los datos estandarizados a la red hará que sean más fáciles de predecir, porque la red no tendrá que aprender ninguno de los factores de localización y escala de la distribución de los resultados (media y varianza).

Una cosa a tener en cuenta es que ahora tendremos que cambiar el valor de δ que le pasamos a la función de Huber 6. Como δ será un error medido entre variables estandarizadas se calculará de la siguiente manera (ecuación 7).

$$\delta = \frac{X - \mu}{\sigma} - \frac{\widehat{X} - \mu}{\sigma} \tag{7}$$

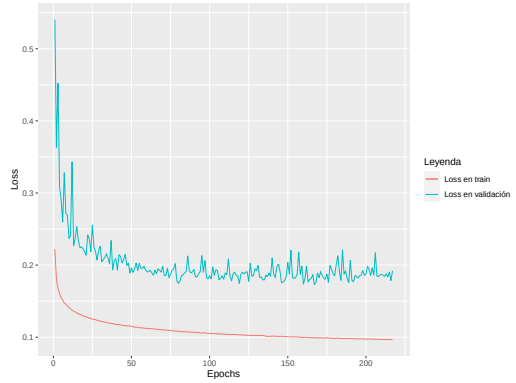
$$\delta = \frac{X - \widehat{X}}{\sigma} = \frac{\delta_0}{\sigma} \tag{8}$$

Así pues, el valor de δ que aplicaremos en la *loss de Huber* para las edades estandarizadas se calculará dividiendo la δ para las variables sin estandarizar, que llamaremos δ_0 entre la desviación típica de las edades. En nuestro caso trabajaremos con valores de $\frac{5}{\sigma}$ y $\frac{10}{\sigma}$, como habíamos comentado previamente.

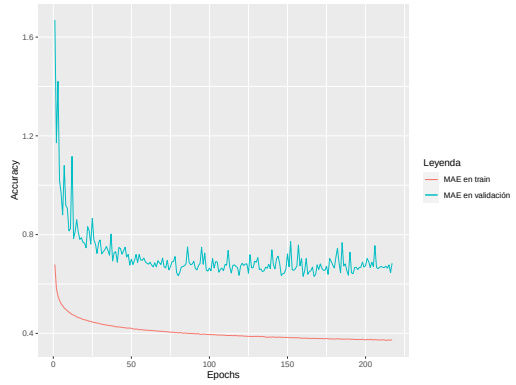
5.1.4. Entrenamiento

Para el modelo final se utilizaron los pesos estandarizados en el intervalo $[1, \sqrt{2}]$, 11 la variable edad se estandarizó, como comentamos en la sección anterior, utilizamos como función de pérdida la loss de Huber con $\delta = \frac{5}{\sigma}$ donde σ es la desviación típica de las edades. No inlcuimos *dropout* en este modelo, ya que no ayudaba a mejorar los resultados en validación, sino que empeoraba resultados, tanto en train, como en validación.

La figura 12 muestra la evolución del entrenamiento en este problema, tanto para la loss de Huber (figura 12a), como para el MAE obtenido (figura 12b). Podemos ver que se comportan de forma muy similar debido a que la mayor parte de la función de loss escogida se corresponde con el MAE, como se discutió en 5.1.2. Los resultados se discutirán en la sección 6.



(a) Evolución de la función de loss



(b) Evolución del MAE

Figura 12: Evolución del accuracy y la loss en entrenamiento y validación.

5.2. Sexo

En la figura 6 se observó la gran diferencia entre clases que tenemos en el *dataset*. Una primera solución pasaría por asignar los pesos de la clase para tener en cuenta los desbalanceos a la hora de calcular la función de loss y así, penalizar más a los errores de clasificación de mujeres.

Para ello asignamos pesos a las clases utilizando el criterio 9, donde p representa las probabilidades de cada clase, resultando en un peso de 1.5 para las mujeres y 1 para los hombres.

$$\text{Peso}(Clase_i) = \frac{\text{máx}\, p}{p(Clase_i)} \tag{9}$$

No obstante, esto no fue suficiente, ya que como se mostró en las figuras 5a y 5b, tenemos un comportamiento extraño en las densidades de las edades: tenemos muchos hombres en edades altas, al contrario que mujeres, que abarcan la parte más baja. Debido a que supusimos (y comprobamos empíricamente) que la edad es un factor clave para estimar el género, esto nos acarreerá ciertos problemas.

Para afrontar esta casuística, inicialmente se entrenó con no solo los pesos de las clases, sino que se añadieron a mayores los pesos para cada una de las edades (tal y como se discutió en la sección 5.1.1). A diferencia de la ocasión anterior, los resultados mejoraron, pero seguían siendo insuficientes, por lo que optamos por llevar esta idea un poco más al extremo: extrajimos de la red de predicción de la edad los pesos del modelo, y lo usamos como punto de partida de nuestro modelo.

Haciendo este aprendizaje transferido, estamos forzando a la red a que partiendo de la premisa de que se sabe clasificar medianamente bien la edad, aprenda a clasificar los distintos géneros; de esta forma la red tiene la información de la edad, pero de forma indirecta.

5.2.1. Funciones de pérdida

Como obtuvimos tan buenos resultados para la edad utilizando la función de pérdida de Huber, pensamos que quizás podríamos obtener los mismos buenos resultados para el problema del sexo utilizando la loss de Huber para problemas de clasificación [Yessou et al., 2020], cuya definición se puede ver en la ecuación 10.

$$\text{Huber}(x) = \begin{cases} \text{máx}(0, 1 - xy)^2 & \text{si } xy \geq -1 \\ -4xy & \text{si } xy < -1 \end{cases} \tag{10}$$

En esta función, x representa la predicción (en logits) e y la clase real (entre -1 y 1). Observemos que, para el caso de clasificación, la loss de Huber utiliza la loss de Hinge [Lin et al., 2002] donde utilizaba el MSE en regresión y una función lineal de los logits y la clase real donde utilizaba el MAE.

Esta función de coste no venía implenetada en Tensorflow o sus paquetes de herramientas, así que la implementamos de cero. Sin embargo, al hacer pruebas, no dio mejores resultados que la entropía cruzada binaria para nuestro conjunto de datos.

Posteriormente, probamos a utilizar otra función de pérdida, en este caso una muy utilizada en la literatura: la *focal loss* o entropía cruzada focal [Lin et al., 2017], cuya definición se ve en la ecuación 11. La idea de esta función de coste es que se utilizará un parámetro α para añadir peso a la clase infrarepresentada y otro parámetro γ para reducir el valor que le asigna la función de loss a los individuos bien clasificados. En el

caso de que $\gamma = 1$ y que no se incluya α en la función estaremos ante la entropía cruzada común.

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{11}$$
$$p_t = \begin{cases} p & \text{si } y_{verdadera} = 1 \\ 1-p & \text{si } y_{verdadera} = 0 \end{cases}$$

Tampoco obtuvimos resultados significativos con esta función de pérdida.

5.2.2. Aumento del conjunto de validación

Uno de los problemas que tenemos a la hora de validar nuestro modelo es que el conjunto de validación tiene solamente 200 observaciones (realmente 197 al descartar las imágenes sin etiquetar). Convendría subsanar este inconveniente para:

- Obtener una validación más precisa de nuestros modelos.
- Tener mayor poder predictivo a la hora de hacer *Bootstrap* (subsección 6.1) para predecir resultados en el conjunto de test.

Lo que haremos será aumentar el conjunto de validación; para ello quitaremos variables del conjunto de entrenamiento aleatoriamente y las introduciremos en validación. Algo con lo que tenemos que ser cautelosos al hacer esto es que, como dijimos al principio, validación comparte distribución con el conjunto de test y para poder aspirar a tener buenos intervalos de predicción usando *Bootstrap* deberemos conservar esta distribución.

En el caso de la edad esto es imposible, ya que al ser una variable continua y tener tan pocos datos el conjunto de validación no podemos estimar con suficiente precisión la función de distribución empírica como para crear una función que retire muestras de datos del conjunto de entrenamiento según la distribución original.

Sin embargo, en el caso del sexo podremos realizar un contraste de hipótesis para la diferencia de proporciones para comprobar si la proporción de mujeres en el conjunto de validación y el de entrenamiento es diferente, en el sentido de significación estadística. Lo que haremos será plantear un contraste de diferencia de proporciones asumiendo que la distribución binomial de las proporciones se puede aproximar asintóticamente por una normal y asumiendo el caso peor para el estimador de las probabilidades desconocidas para ser más conservadores [Peña, 2014]. Resultando en el siguiente estadístico de contraste 12, que con un $p-$valor de aproximadamente 0.10 nos sugiere que no hay suficiente evidencia para rechazar la hipótesis nula de igualdad de proporciones, por lo que podremos proceder a aumentar el conjunto de validación quitando el 1 % de los datos de entrenamiento.

Una cosa que cabe comentar es que quizás podríamos estar cometiendo un error al hacer esto, ya que no hay suficiente evidencia para decir que la distribución de la variable sexo sea distinta para los conjuntos de entrenamiento y validación, pero la variable de edad claramente está distribuida de forma distinta. Podemos asumir que evitamos el problema de estimar el sexo a partir de edades diferentes ya que partiremos de un modelo entrenado con la edad para estimar el sexo mediante *transfer learning*, así que ya contaremos *a priori* con un modelo que contenga la información de la edad, pudiendo despreocuparnos de introducir edades de distribuciones distintas a la distribución de las edades de validación.

$$D = \frac{\hat{p}_{train} - \hat{p}_{val}}{\sqrt{\hat{p}(1-\hat{p})\left(\frac{1}{n}+\frac{1}{m}\right)}} \simeq N(0,1) \tag{12}$$
$$\text{Donde } \hat{p} = \frac{n\hat{p}_{train} + m\hat{p}_{val}}{n+m}$$

5.2.3. Entrenamiento

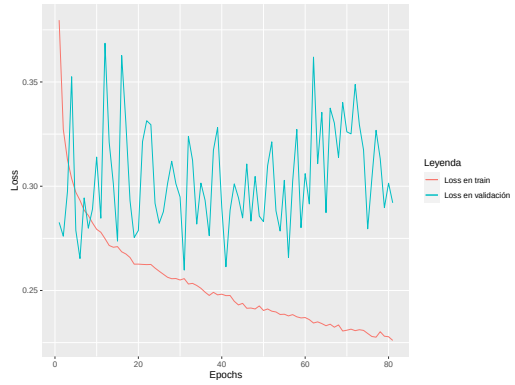
Para el modelo final del sexo, utilizamos la arquitectura propuesta en la figura 7 y tabla 1, entrenada primeramente con los datos de la edad y haciendo así *transfer learning*. A esta arquitectura base le añadimos una capa densa con función de activación ReLU y una capa de *dropout* con probabilidad del 40 %. Finalmente, ponemos una neurona de salida con función de activación sigmoidal y usamos como función de pérdida la entropía cruzada común.

En la capa de salida inicializamos los bias como $\log(p_{mujer}/p_{hombre}) = \log(2/3)$, como se comenta en [Goodfellow et al., 2016], para partir de un bias que represente los sesgos presente en la distribución de clases y que la red no tenga que aprenderlo por su cuenta.

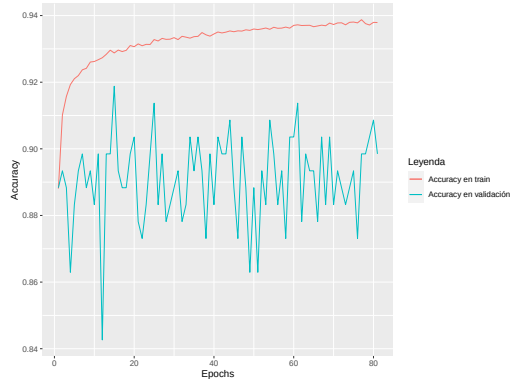
Optamos por no congelar primeramente las capas convolucionales ni por utilizar una tasa de aprendizaje baja, como sería de costumbre en *transfer learning*, porque nuestro objetivo es que la red simplemente parta con información adicional de la edad, que le puede ser útil para distinguir entre sexos para las distintas edades, pero no necesariamente que se mantenga toda esta información. En la versión final tampoco utilizamos los *sample weights* correspondientes a la edad, ya que esta información viene dada por las convoluciones transferidas, como hemos comentado.

Debido a que utilizamos una arquitectura ya entrenada para el problema de la edad, partiremos de un modelo que ya tiene convoluciones preparadas para procesar estos datos (aunque para otro problema). No obstante, este conocimiento previo que posee el modelo ya es muy bueno para predecir el género. Como podemos ver en 13 partimos de unas predicciones iniciales muy buenas, por lo que mejorarlas es difícil y hay ligeras dificultades para bajar la función de loss 13a y el accuracy 13b, pero conseguimos algunas mejoras tras unas iteraciones.

Aunque parezca contraintuitivo, mediante este método estamos obteniendo resultados muy buenos en muy pocas epochs y con apenas variaciones entre epochs: véanse resultados en 6.



(a) Evolución de la función de loss



(b) Evolución del accuracy

Figura 13: Evolución del accuracy y la loss en entrenamiento y validación.

Como este problema es de clasificación binaria, debemos ajustar un threshold a partir del cual se asignarán valores a una clase u otra. Para ello hemos optado por tres métodos distintos, que deben cumplir el objetivo de validación sobre el criterio de que las diferencias entre clases sea menor al 3 %:

- Curva precision-recall (gráfica de la izquierda en 14). A diferencia de lo habitual, donde se muestra su gráfico poniendo precision en el eje de ordenadas y *recall* en el de abcisas para que se pueda ver el área bajo la curva, optaremos por el método que se usa en [Géron, 2019] de poner probabilidades en el eje de abcisas y ambas curvas en el de ordenadas, de esta forma se verá un punto de corte entre ambas que se considera que será un buen candidato a threshold a utilizar. En este caso *recall* en naranja y *precision* en azul.
- *Receiver Operating Characteristic Curve* (curva ROC), en la gráfica de la derecha de 14. Se observa que es un clasificador muy bueno, ya que tiene un área debajo de la curva bastante grande, por lo que las probabilidades de que el clasificador asigne probabilidades más altas a las mujeres que a los hombres son elevadas (recordemos que de ser uno estaríamos ante el clasificador perfecto). Además, se ve que podemos aumentar el *recall* a valores relativamente elevados (subir la curva por el eje y) sin tener que aumentar mucho la tasa de falsos positivos (movernos en el eje x).
- Un algoritmo[4] propuesto por un usuario del foro de StackOverflow que calcula el *threshold* que optimiza el *accuracy* de nuestro modelo para un dataset con complejidad $\mathcal{O}(n)$ en vez de $\mathcal{O}(n^2)$ (asumiendo que partimos de datos ordenados). El pseudocódigo para el algoritmo se puede ver en 1.

Algorithm 1: Selección de threshold

```
Dado una lista L que contenga pares de y e ŷ
  ordenados en función de las probabilidades en
  ŷ de menor a mayor:
verdaderos_positivos = # mujeres en el dataset;
verdaderos_negativos = 0;
Inicializamos posibles_thresholds como una lista
  vacía;
for tupla en L do
    valor_real, aux_threshold = tupla;
    if valor real = 1 then
        verdaderos_positivos -= 1;
    else
        verdaderos_negativos += 1;
    end
    accuracy = (verdaderos_positivos +
      verdaderos_negativos) / # L;
    Añadirmos a posibles_thresholds la tupla
      (aux_threshold, accuracy);
end
threshold_optimo será aquel que tenga el mayor
  accuracy de posibles_thresholds;
```

Lo que hace este algoritmo 1 es partir de la premisa de que utilizamos como *threshold* para la clasificación el valor 0, es decir, clasificaremos cualquier entrada como mujer. Luego de esto, almacenará los valores de los verdaderos positivos (todos los elementos del *dataset*) y los verdaderos negativos (ninguno).

A partir de ahí, incrementaremos ese umbral de clasificación, en función de las probabilidades predichas ordenadas, teniendo en cuenta que, cada vez que se actualice el umbral, podremos estar bajando o subiendo los verdaderos positivos y negativos, respectivamente.

Esto es debido a que si $y_{verdadero} = 1$ y el nuevo *threshold* es y_{pred}, usando el criterio $y_{pred} > y_{pred}$ estaremos clasificando mal este individuo, bajando los verdaderos positivos.

Por otro lado, si $y_{verdadero} = 0$, al escoger como umbral y_{pred} estaremos clasificando ese individuo correctamente, aumentando los verdaderos negativos.

Con el algoritmo recorreremos todos los pares $y_{verdadero}$ e y_{pred}, que denotaremos por y e $\widehat{y}$, de forma ordenada en función de los valores de $\widehat{y}$. Calcularemos el *accuracy* resultante de usar $\widehat{y}$ como

[4] https://stackoverflow.com/questions/30717688/how-to-compute-optimal-threshold-for-accuracy

threshold y almacenaremos el par (*accuracy, threshold*). Finalmente, devolveremos el par que mejor *accuracy* haya generado.

En este punto, si cogiésemos como threshold 0,5, la matriz de confusión sería la que se muestra en la tabla 2 obteniendo así una tasa de positivos correctamente clasificados (TPR) del 92,8 % y una tasa de negativos correctamente clasificados (FPR) del 93,01 %. No obstante, tras realizar los ajustes pertienentes con los algoritmos mostrados con anterioridad, la matriz de confusión resultanate es la que se muestra en la tabla 3, consiguiendo así un TPR del 92,25 % FPR de 93,7 %.

		Real	
		Hombre	Mujer
Predicho	Hombre	812	41
	Mujer	61	529

Cuadro 2: Matriz de confusión en validación

		Real	
		Hombre	Mujer
Predicho	Hombre	808	45
	Mujer	54	536

Cuadro 3: Matriz de confusión en validación con el threshold ajustado

5.3. Problema conjunto

Como en el caso del problema del género, se optó por aprendizaje por transferencia, pero en este caso, llevado al extremo. Partiendo de la premisa de que nuestros dos modelos anteriores han obtenido resultados *espectaculares*, podemos crear un modelo a partir de los resultados de los mismos. Para ello se optó por la arquitectura presentada en la figura 7, y posteriormente se le añadieron dos redes distintas (para cada una de las salidas): una red de dos capas con 64 y 32 neuronas (la misma que la del modelo original de la edad 5.1) y otra con una única capa oculta de 32 neuronas, con una posterior salida con activación sigmoidal, inicialización de pesos y dropout al 40 % (como en el modelo del género 5.2).

El proceso de entrenamiento fue el siguiente:

1. Se clonaron idénticamente los pesos de la parte convolucional y la salida correspondiente del modelo de la edad.

2. Se clonaron los pesos de la salida de la red del género a las neuronas correspondientes en esta arquitectura. En cualquier otra situación, esto sería un disparate, pero al haber realizado primeramente ya aprendizaje por transferencia para este problema, es esperable que muestre un comportamiento estable ante esta concatenación.

3. Se congelan todos los pesos que se han clonado de la red de la edad (parte convolucional y salida de la edad).

4. Se entrenó con paciencia 50 este modelo, es decir, se ajustó la salida del problema del género para que se "acostumbrase" a los nuevos pesos de las convoluciones (como se mencionó, similares a los que ya había visto con anterioridad).

   En la figura 15 se observa como tiene lugar este proceso de adaptación, a través de la función de loss.

5. Se descongelan todos los pesos.

6. Se deja que la red aprenda las sutilezas que crea convenientes para ambos problemas. La figura 16 muestra la evolución de la loss para esta parte del aprendizaje, tanto para la edad 16a como para el género 16b. Como es de esperar, durante esta etapa el aprendizaje es prácticamente nulo en edad (lo hemos forzado a que así sea), mientras que en género se ve una ligera tendencia a seguir aprendiendo (esperable también, al ser el problema que no ha "heredado" los pesos del problema original).

6. Resultados

En esta sección discutiremos los resultados obtenidos a lo largo de las distintas aproximaciones. Primeramente se hará una introducción en 6.1 a nuestra técnica para decidir qué modelo era digno de realizar las predicciones en el conjunto de test; en la subsección 6.2 se discutirán los intervalos de predicción conseguidos con esta técnica, así como los resultados de Kaggle.

6.1. Bootstrap

Realizar validación cruzada para modelos de aprendizaje profundo muchas veces suele ser inviable, ya que requeriría entrenar mínimo k modelos para cada una de las k *folds* y, al ser algoritmos no deterministas, un número prefijado de ejecuciones dentro de cada *fold*. Así pues, si quisiéramos realizar validación cruzada de 10 *folds* con un número prefijado de 5 ejecuciones por *fold* tendríamos que entrenar nuestro modelo $10 \times 5 = 50$ veces, lo cual es inviable, incluso para modelos pequeños como los aquí propuestos.

Como alternativa a eso, y en aras de tener una estimación de qué tan bien generalizan nuestros modelos, propondremos un método alternativo para validar sus resultados basado en técnicas de simulación y remuestreo.

Como mencionamos previamente, se nos ha dicho que los conjuntos de validación y test comparten la misma distribución. Podemos aprovechar este hecho para obtener una estimación *a priori* del error que cometeremos en test utilizando el conjunto de validación. Lo que haremos será crear un intervalo no paramétrico para el error de predicción en validación mediante el método percentil de *Bootstrap* [Cao and Fernández-Casal, 2021].

Ya que validación y test comparten distribución y el Bootstrap nos permite aproximar la distribución de cualquier estadístico en el muestreo, aproximaremos la

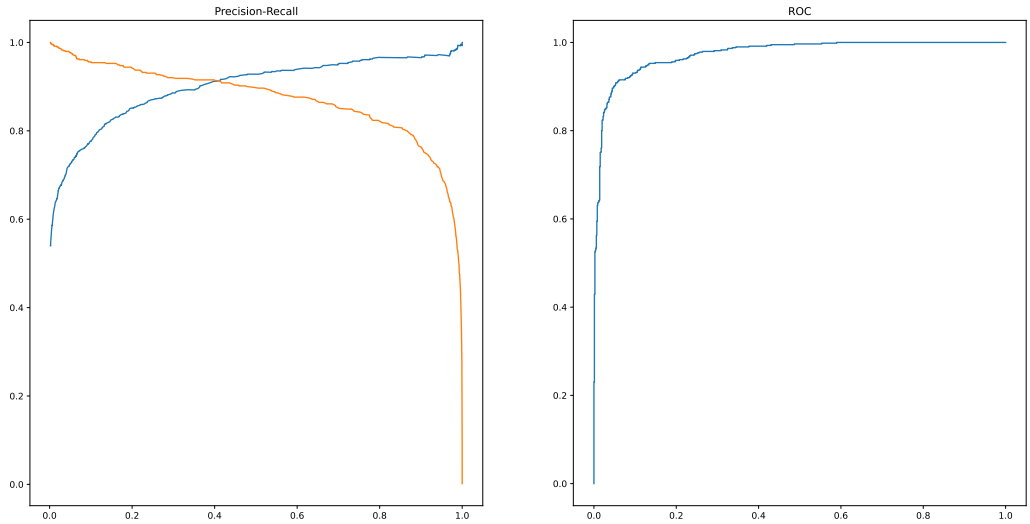


Figura 14: Curvas precision-recall y ROC para el conjunto de validación

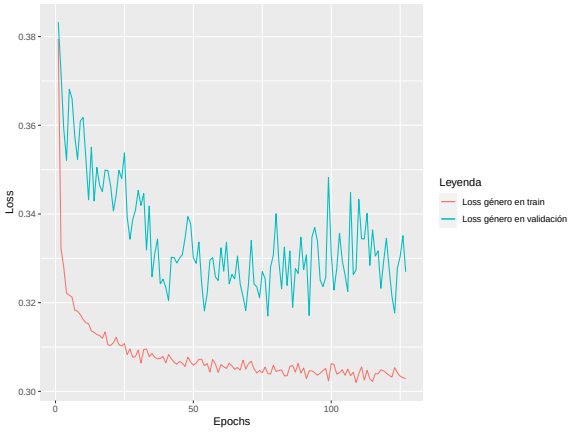


Figura 15: Evolución de la función de loss para el género.

distribución del estadístico de error condicionada a la distribución de validación y test. Esto es, calcularemos remuestras del conjunto de validación, las evaluaremos con un modelo ya entrenado y calcularemos un intervalo de confianza no paramétrico en base a estos errores.

El método *Bootstrap* nos garantizará que, a medida que el número de remuestras tienda a infinito, aproximaremos con mayor precisión la distribución real de los errores para la distribución de validación, que es la misma de test. Así pues, podremos crear un **intervalo de confianza** para el error cometido en test utilizando solamente simulaciones del conjunto de validación, lo cual es computacionalmente muchísimo menos costoso que hacer validación cruzada.

La idea es que partiremos de un conjunto de imágenes del conjunto de validación, $\{X_1, \dots X_n\}$, que tienen asociadas un conjunto de etiquetas $\{Y_1, \dots, Y_n\}$ que representarán el sexo y la edad, según el problema a tratar. Estas imágenes y etiquetas están englobadas bajo una función de distribución conjunta F y tenemos que $F = F_{\text{validación}} = F_{\text{test}}$.

Nuestro objetivo será construir un modelo de red neuronal convolucional f para estimar Y_i a partir de X_i, $\widehat{Y}_i = f(X_i)$. Una vez construido el modelo, denotaremos a los estadísticos de accuracy y MAE esperados para la distribución F (y por lo tanto, para el conjunto de test) por θ y se podrán aproximar asintóticamente por *Bootstrap* cogiendo remuestras $\{X_1^* \dots, X_n^*\}$ de F a través del conjunto de validación mediante un muestreo aleatorio simple y generando estimaciones $\widehat{\theta}^*$ de θ. De esta forma, se puede construir el intervalo de confianza 13 utilizando el MAE o accuracy obtenido en validación como $\widehat{\theta}$ y aproximando los cuantiles $x_{1-\alpha/2}$ y $x_{\alpha/2}$ mediante la distribución empírica de los distintos $\widehat{\theta}^*$ de las remuestras.

$$\mathbb{P}\left(\widehat{\theta} - \frac{x_{1-\alpha/2}}{\sqrt{n}} < \theta < \widehat{\theta} - \frac{x_{\alpha/2}}{\sqrt{n}}\right) = 1 - \alpha \qquad (13)$$

Para este trabajo, utilizaremos un α del 1 %, resultando en intervalos de 99 % de confianza.

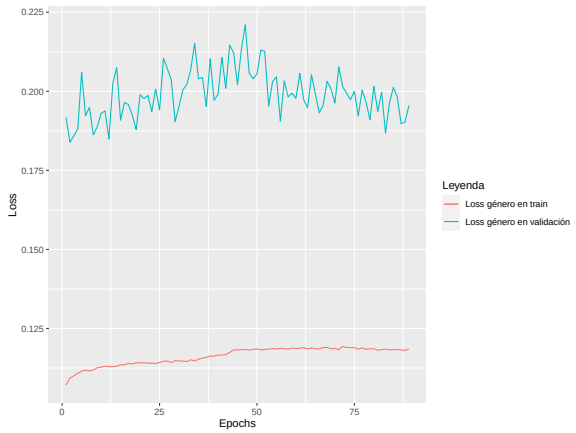
6.2. Análisis de los resultados

La tabla 4 muestra los intervalos de predicción obtenidos mediante *Boostrap* para cada uno de los problemas, así como los resultados de validación y finalmente test.

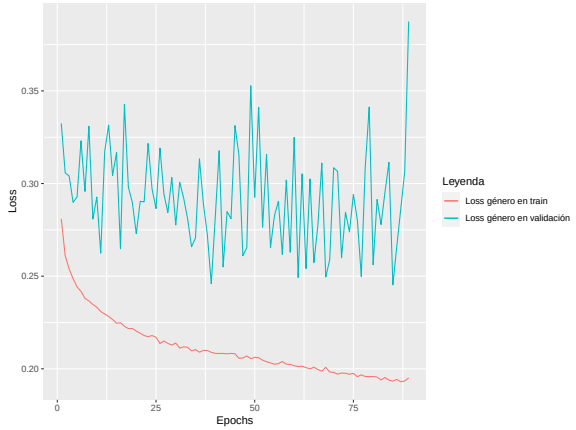
Problema	Train	Validación	Boostrap inferior	Boostrap superior	Test
Edad (MAE)	6.2	9.45	6.78	10.23	9.06
Género (Accuracy)	0.942	0.917	0.915	0.943	0.923

Cuadro 4: Resultados de los distintos modelos e intervalos *Boostrap* de predicción

Nótese que tanto los resultados de validación como los de test, para ambos problemas entran dentro de



(a) Evolución de la función de loss para la edad



(b) Evolución de la función de loss para el género

Figura 16: Evolución de la loss en la segunda parte del aprendizaje

los intervalos de predicción calculados. Para el caso del problema conjunto, como se discutió anteriormente, el comportamiento esperable es que tenga resultados similares a los obtenidos en los problemas por separado, debido a las técnicas empleadas. Es así pues, que el resultado esperable en el caso ideal en test debería haber sido 8,34 de *WMAE* 14.

$$\frac{9{,}06 + 0{,}07625 \cdot 100}{2} = 8{,}34 \tag{14}$$

No obstante, debido al overhead introducido por tener que solucionar los dos problemas a la vez, lo redujo hasta 9,12. Sabemos por las funciones de loss 16a que el error en la edad se mantuvo estable, por lo que no es descabellado suponer que en este problema mantuvimos el *MAE* en ~ 9. Por lo que para que *WMAE* obtuviese el valor que dio, supondría un error en el problema de género del mismo calibre.

7. Conclusión

A lo largo de las secciones anteriores, hemos presentado un único modelo, capaz de generalizar para solucionar todos los problemas propuestos. Este modelo presenta todos los beneficios que se podrían esperar de un buen modelo: presenta muy pocos parámetros en relación a su competencia, entrenamientos muy rápidos (número razonable de epochs, muy cortas en tiempo de ejecución) y muy buenos resultados; de esta forma corregimos de forma preventiva el sobreentrenamiento (véase 4). Estos modelos, al explotar al máximo los conocimientos aprendidos (gracias al aprendizaje por transferencia), son, como se comentó, muy éticos desde el punto de vista ecológico.

8. Trabajo futuro

Como la función de *loss* de Huber dio tan buenos resultados en la edad, siendo la predicción de la edad un resultado clave para resolver los demás problemas en nuestra aproximación, se podría probar a utilizar ciertas variaciones de esta función de *loss*. En [Barron, 2019] introducen una generalización de las funciones de pérdida robustas que extrapola el concepto de funciones como la de Huber o Cauchy a un caso general. Además, para obtener estimaciones *Bootstrap más precisas* se podría probar a utilizar el método percentil t [Cao and Fernández-Casal, 2021], que con un número mayor de remuestras propondría una ratio de convergencia más elevada que el aquí planteado.

Simplicity is all you need.

Índice

Referencias

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Barron, 2019] Barron, J. T. (2019). A general and adaptive robust loss function.

[Cao and Fernández-Casal, 2021] Cao, R. and Fernández-Casal, R. (2021). Técnicas de remuestreo. `https://rubenfcasal.github.io/book_remuestreo/`.

[Chen et al., 2017] Chen, M., Shi, X., Zhang, Y., Wu, D., and Guizani, M. (2017). Deep features learning for medical image analysis with convolutional autoencoder neural network. *IEEE Transactions on Big Data*.

[Chollet et al., 2015] Chollet, F. et al. (2015). Keras.

[Dehghan et al., 2017] Dehghan, A., Ortiz, E. G., Shu, G., and Masood, S. Z. (2017). Dager: Deep age, gender and emotion recognition using convolutional neural network. *arXiv preprint arXiv:1702.04280*.

[Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[Duan et al., 2018] Duan, M., Li, K., Yang, C., and Li, K. (2018). A hybrid deep learning cnn–elm for age and gender classification. *Neurocomputing*, 275:448–461.

[Fujieda et al., 2018] Fujieda, S., Takayama, K., and Hachisuka, T. (2018). Wavelet convolutional neural networks. *arXiv preprint arXiv:1805.08620*.

[Géron, 2019] Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Han et al., 2020] Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., and Xu, C. (2020). Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589.

[Hinton et al., 2012a] Hinton, G., Srivastava, N., and Swersky, K. (2012a). Overview of mini-batch gradient descent.

[Hinton et al., 2012b] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[Howard et al., 2019] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324.

[Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[Huang and Aviyente, 2008] Huang, K. and Aviyente, S. (2008). Wavelet feature selection for image classification. *IEEE Transactions on Image Processing*, 17(9):1709–1720.

[Huber, 1992] Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer.

[Klenke, 2013] Klenke, A. (2013). *Probability theory: a comprehensive course*. Springer Science & Business Media.

[Lee and Lee, 2020] Lee, S. and Lee, C. (2020). Revisiting spatial dropout for regularizing convolutional neural networks. *Multimedia Tools and Applications*, 79(45):34195–34207.

[Levi and Hassner, 2015] Levi, G. and Hassner, T. (2015). Age and gender classification using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 34–42.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

[Lin et al., 2002] Lin, Y., Wahba, G., Zhang, H., and Lee, Y. (2002). Statistical properties and adaptive tuning of support vector machines. *Machine Learning*, 48(1):115–136.

[Liu et al., 2012] Liu, L., Liu, J., and Cheng, J. (2012). Age-group classification of facial images. In *2012 11th International Conference on Machine Learning and Applications*, volume 1, pages 693–696. IEEE.

[Ministerio de Sanidad, 2020] Ministerio de Sanidad, G. d. E. (2020). Orden snd/380/2020, de 30 de abril.

[Peña, 2014] Peña, D. (2014). *Fundamentos de estadística*. Alianza editorial.

[R Core Team, 2020] R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[Sandler et al., 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Strubell et al., 2019] Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

[Teodoro and Valcarce, 2021] Teodoro, V. and Valcarce, D. (2021). Efectos de la reducción de la dimensionalidad en la detección de dígitos manuscritos. `https://teo-val.github.io/papers/digitos.pdf`.

[Van Rossum and Drake Jr, 1995] Van Rossum, G. and Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

[Vapnik and Chervonenkis, 2015] Vapnik, V. N. and Chervonenkis, A. Y. (2015). On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer.

[Wickham et al., 2019] Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686.

[Wooldridge, 2015] Wooldridge, J. M. (2015). *Introductory econometrics: A modern approach*. Cengage learning.

[Yessou et al., 2020] Yessou, H., Sumbul, G., and Demir, B. (2020). A comparative study of deep learning loss functions for multi-label remote sensing image classification. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 1349–1352. IEEE.