Algoritmo de Estimación de la Edad en Imágenes y Vídeos

Image and Video Age Estimation Algorithm

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA DEL SOFTWARE
CURSO 2020–2021

Inés Prieto Centeno

Directores
Luis Javier García Villalba
Esteban Alejandro Armas Vega

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Junio de 2021

Agradecimientos

Ha sido corto pero intenso, 4 años que marcan definitivamente la vida de una persona. Aún recuerdo aquella clase de Física en 2º de Bachillerato, donde nos comentó el profesor, "a mí la Universidad se me pasó volando", y yo pensaba, ¿Cómo se le va a poder pasar volando la Universidad si son 4 años? ¡A mí se me hará eterna!, pensaba en aquellos tiempos. El primer día de pisar la facultad y de conocer a gente que aún me sigue aguantando y queriendo sabía que iba a ser diferente. Sabía que iban a ser los mejores años de mi vida, y efectivamente así ha sido, aunque con bajones y subidas hemos conseguido estar todos a flote, y digo todos porque ellos, mis amigos, son lo que realmente me llevo de la carrera. Los que me han estado apoyando siempre y han conseguido levantarme y guiarme. Gracias por todo, gracias por ser como sois.

Gracias también a mis padres Antonino e Inés por estar siempre conmigo, aconsejándome y haciéndome ver el verdadero sentido de la vida. Gracias por saber entender lo que realmente quiero en la vida, por no juzgarme y por apoyarme. Gracias por valorarme y hacerme ver lo que realmente valgo. Siempre conmigo, siempre con vosotros, os quiero.

Gracias también a mis tutores Esteban y Ana por ayudarme a realizar este trabajos, sin ellos no hubiera sido posible. Gracias por vuestra plena disponibilidad.

Gracias a la Universidad Complutense de Madrid, en especial a la Facultad de Informática y a su gente por proporcionarme los cimientos necesarios para mi carrera. Gracias también por proporcionarme el equipo necesario para poder realizar este trabajo.

Por último, dar las gracias a Paula por estar siempre a mi lado, apoyándome y dándome consejos. Gracias por darme tanta seguridad en mí misma para afrontar los problemas, por ayudarme a rematar este trabajo de la mejor forma posible, pero sobretodo muchas gracias por aguantarme a pesar de mis agobios. Muchísimas gracias por todo, kkk.

Índice General

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Lista de Acrónimos

AAM	*Active Appearance Model*
AdaGrad	*Adaptive Gradient Algorithm*
AGES	*Aging Patterns Subspace*
AI	*Artificial Intelligence*
AIX	*Advanced Interactive eXecutive*
ANN	*Artificial Neural Network*
ASM	*Active Shape Model*
BPTT	*Backpropagation Through Time*
CNN	*Convolutional Neural Networks*
CSV	*Comma-Separated Values*
DL	*Deep Learning*
DNN	*Deep Neural Network*
IBM	*International Business Machines Corporation*
ILSVRC	*ImageNet Large Scale Visual Recognition Challenge*
ITL	*Inductive Transfer Learning*

kNN	*k-Nearest Neighbors*
KRR	*Kernel Ridge Regression*
LDA	*Linear Discrimiant Analysis*
LSTM	*Long Short Term Memory*
MAE	*Mean Absolute Error*
mAP	*Mean Average Precision*
MIT	*Massachusetts Institute of Technology*
ML	*Machine Learning*
MLP	*MultiLayer Perceptron*
MSE	*Mean Squared Error*
NIN	*Network In Network*
NPR	*Non-Parameteric kernel Regression*
OD	*Object Detection*
OpenCV	*Open Source Computer Vision*
PCA	*Principal Component Analysis*
R-FCN	*Region-based Fully Convolutional Networks*
R-CNN	*Region-based Convolutional Network*
ReLU	*Rectified Linear Unit*
ResNet	*Residual neural network*
RL	*Reinforcement Learning*

RMSProp	*Root Mean Square Propagation*
RNN	*Recurrent Neural Networks*
RoI	*Region of Interest*
RPN	*Region Proposal Networks*
RSS	*Residual Sum of Squares*
SGD	*Stochastic Gradient Descent*
SL	*Supervised Learning*
SMC	*Secure Multi-party Computation*
SOM	*Self Organizing Map*
SPP	*Spatial Pyramid Pooling*
SSD	*Single Shot Detection*
SVM	*Support Vector Machine*
SVR	*Support Vector Regression*
TL	*Transfer Learning*
TTL	*Transductive Transfer Learning*
UL	*Unsupervised Learning*
UTL	*Unsupervised Transfer Learning*
WAS	*Weighted Appearance Specific*
WPS	*Weighted Person Specific*
YOLO	*You Only Look Once*

Abstract

Age detection techniques can be decisive for people who are not identified and age needs to be known for various reasons, such as crimes, sexual abuse, etc., since human intervention in these cases is a very difficult task. slow and slow down the whole process. This is why perfecting current age estimation techniques can be an improvement for forensic and security fields. This work can be a progress in this area, since many crimes are committed at night and processing the images with filters before introducing them into a detection model could be the result. For this reason, this work shows the objectives obtained from applying preprocessing to the images before introducing them into a detection model. After an arduous investigation in which there have been complicated moments, such as the search for a good team to be able to carry out the execution of the model and the processing of all the images of the dataset, to handle TensorFlow in a fluid way, etc., it is possible to solve by means of the creating an instance in Google Cloud. Once this has been achieved, a study is carried out on improving the quality of the images related to the datasets mentioned in this work. With this analysis, it is intended to address the problems that can cause the poor quality of the images, as well as their poor treatment when introducing them into a model. For the development of this study, different filters have been investigated for the improvement of images, executing and seeing the performance of each one and saving the best results, for later selection. Also, other forms of face detection have been investigated, such as profile detection. After gathering all the preprocessing with better results and introducing it into a model to see the improvements compared to the untreated images, the model trained with those images previously preprocessed was the winner, obtaining an MAE of 0.24 %.

Keywords: Datasets, Detection, Google Cloud, Mean Absolute Error, Tensorflow

Resumen

Las técnicas para la detección de la edad pueden ser determinantes para personas que no se encuentren identificadas y se necesite saber la edad por diversos motivos, como delitos, abusos sexuales, etc, ya que la intervención humana en estos casos se trata de una tarea muy lenta y ralentizarían todo el proceso. Es por esto que perfeccionar las actuales técnicas de estimación de la edad, puede constituir una mejora para los ámbitos forenses y de seguridad. Este trabajo puede resultar un progreso en este ámbito, ya que muchos delitos son cometidos con nocturnidad y procesar las imágenes con filtros antes de introducirlas en un modelo de detección, podrían ser resultantes. Por ello, en este trabajo se muestran los objetivos obtenidos de aplicar preprocesamiento a las imágenes antes de introducirlas en un modelo de detección. Después de una ardua investigación en la que ha habido momentos complicados, como la búsqueda de un buen equipo para poder realizar la ejecución del modelo y el procesamiento de todas las imágenes del dataset, manejar de manera fluida TensorFlow, etc, se consigue solucionar mediante la creación de una instancia en Google Cloud. Una vez conseguido ésto, se procede a la realización de un estudio sobre la mejora de la calidad de las imágenes relacionadas con los datasets mencionados en este trabajo. Con este análisis, se pretende abordar los problemas que pueden ocasionar la mala calidad de las imágenes, así como, su mal tratamiento a la hora de introducirlas en un modelo. Para el desarrollo de este estudio se ha investigado sobre distintos filtros para la mejora de imágenes, ejecutándose y viendo el desempeño de cada uno y guardando los mejores resultados, para su posterior elección. También, se han investigado otras formas de detección del rostro, como la detección de perfil. Tras reunir todo el preprocesamiento con mejores resultados e introducirlo en un modelo para ver las mejoras respecto a las imágenes sin tratar, sale como vencedor el modelo entrenado con aquellas imágenes preprocesadas anteriormente, obteniéndose una mejora en la detección de la edad.

Palabras Clave: Datasets, Detección, Google Cloud, Error Absoluto Medio, Tensorflow

Capítulo 1

Introducción

1.1. Motivación

En los últimos años se puede ver como el acceso a Internet es prácticamente total para la franja de edad más joven analizada (14 y 16 años), siendo de un 99,9 % de los hombres y un 99,6 % de las mujeres, mientras que con el aumento de edad se disminuye el acceso a este. El grupo de edad que menos uso le dan es el conjunto de edades de 65 a 74 años, con un 70,5 % de los hombres y un 68,9 % de las mujeres.

Estos datos comentados hacia los más jóvenes, conllevan a un gran consumo de contenidos inapropiados, esto sumado a la situación actual de la pandemia con el COVID-19, ha llevado a que se haya incrementado mucho más según han señalado varios estudios, entre los que se encuentra el estudio global desarrollado por la empresa de ciberseguridad Avast, donde el 90 % de los padres comentan la exhibición involuntaria al material para adultos como una de sus mayores preocupaciones acerca del uso de Internet por parte de sus hijos. Un 67 % de los niños admitieron tener malas experiencias en línea durante el confinamiento, viéndose expuestos a contenidos insultantes, groseros e inapropiados.

La exhibición tanto del material para adultos como el sexting es una de las mayores preocupaciones por los padres, a parte de que los niños también pueden poner en peligro dispositivos y domicilios ya que pueden acceder o realizar descargas accidentalmente de páginas falsas, teniendo como consecuencia la incrustación de un virus en el sistema operativo, malwares e incluso problemas con el robo de información.

Noticias muy actuales nos hacen plantearnos la importancia de la estimación de la edad en el ámbito de la Inteligencia Artificial. Instagram, una red social multiplataforma ha anunciado que está trabajando sobre varias mejoras para garantizar la protección de los usuarios más jóvenes, sobre todo haciendo hincapié en el concepto de grooming. Para esto Instagram pretende restringir los mensajes directos entre adolescentes y adultos que no se siguen, y que por lo tanto, no son de su ámbito de conocidos. Para determinar la

edad del usuario no solo se tendrá en cuenta la edad que éste indica al registrarse en la plataforma, si no que también se empleará tecnología referente al machine learning para determinar la edad del usuario, por si falsificó su edad al registrarse.

La estimación de la edad puede servir para poder sugerir diferentes tipos de productos en función de la misma. El Centro de Innovación Industrial en Inteligencia Artificial (CII.IA) quería incorporar la detección de rostros para hacer sugerencias de productos al cliente en una cadena con diferentes tipos de tiendas ya que en función de la edad hay cierta predisposición para adquirir determinados productos, en base a toda esta información, se pueden realizar promociones enfocadas a ese tipo de perfil de clientes.

Por otro lado, cabe comentar la importancia de la estimación de la edad en el ámbito forense, donde es muy importante un buen método de estimación de la edad, ya sea mediante patrones de edad, radiografías dentales, medidas antropométricas, radiografías de la región cervical, radiografías de la mano izquierda para estimar la edad cronológica en función del parámetro de la edad ósea, etc. Estas técnicas suelen ser determinantes para aquellas personas que no se encuentren identificadas y se necesite saber la edad, ya sean por diversos asuntos de delitos, abusos sexuales, etc. La intervención humana en estos casos se trata de una tarea muy lenta y por lo tanto ralentizarían todo el proceso anterior.

Por todos estos motivos se pretende abordar en este trabajo, la mejora de los métodos ya existentes de la estimación de la edad, intentando aumentar la seguridad de los más jóvenes, para intentar prevenir los problemas actuales de grooming, contenidos inapropiados hacia menores, etc.

Por último, la estimación de la edad cada vez está siendo un factor de peso para las recomendaciones de productos en las distintas empresas, dado que así se pueden sugerir productos adecuados en función de la franja de edad del usuario.

1.2. Contexto

Este Trabajo Fin de Grado ha sido realizado dentro del Grupo de Análisis, Seguridad y Sistemas (Grupo GASS, https://gass.ucm.es/, Grupo 910,623 del catálogo de grupos reconocidos por la UCM) como parte de las actividades del proyecto de investigación THEIA (Techniques for Integrity and Authentication of Multimedia Files of Mobile Devices) con referencia FEI-EU-19-04.

1.3. Objeto de la investigación

Este trabajo se realizó en dos fases:

Un estudio previo de otros trabajos similares, intentando encontrar el método que más se ajuste a los objetivos de la investigación, teniendo presente las diferentes

implementaciones usadas. Una vez realizado el estudio, nos quedamos con aquellos que mejor resultado hayan dado, y se utilizarán como punto de partida para intentar realizar mejoras que produzcan un resultado con menos margen de error. Para ello incorporaremos tanto preprocesamiento de imágenes, como variaciones a los diferentes parámetros de la red neuronal utilizada en ese estudio.

1.4. Plan de Trabajo

Este trabajo se ha desarrollado a través de las actividades de la Tabla 7.1.

Tabla 1.1: Planificación del trabajo

Actividades	Fecha realizada
1. Estructura del proyecto y plazos	2ª quincena de septiembre
2. Investigación de proyectos anteriores	2ª quincena de octubre
3. Resúmenes de la investigación realizada	1ª quincena de noviembre
4. Capítulo marco conceptual	1ª quincena de diciembre
5. Capítulo estado del arte	2ª quincena de Febrero
6. Ejecución de código	2ª quincena de marzo
7. Análisis de datos. Redacción resultados y conclusiones	2ª quincena de abril
8. Revisar memoria actual	1ª quincena de mayo
9. Terminar de retocar memoria	2ª quincena de mayo

Para llevar a cabo una mejor rutina a la hora de realizar el proyecto, se hizo uso de Trello, una aplicación con la que se puede visalizar de manera sencilla y clara las tareas pendientes, en curso y realizadas. Un ejemplo de como se ha realizado la organización en Trello se puede ver en la Figura 1.1.

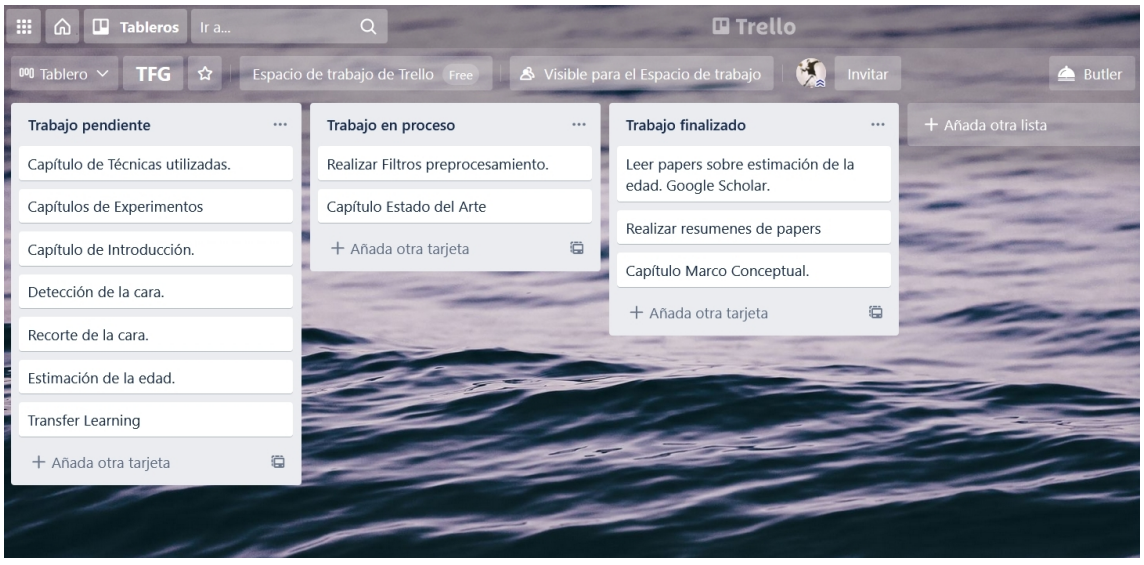


Figura 1.1: Uso de Trello en el proyecto

1.5. Estructura del proyecto

La estructura de este trabajo, excluyendo este capítulo de introducción, se basa en 5 capítulos con una estructura específica comentada a continuación.

En el Capítulo 2 se pretenden abordar las diferentes bases teóricas que se utilizarán en el resto de capítulos posteriores. Se comentarán tanto el nacimiento de la Inteligencia Artificial como su crecimiento a lo largo de los años. Posteriormente se ahondar en los diferentes métodos de detección de objetos. Por último después de realizar un paso por los avances en estos últimos años, se explicarán los principales tipos de redes neuronales, profundizando en su arquitectura y su uso. Otros temas como el Aprendizaje de Transferencia y los problemas de la Inteligencia Artificial también son comentados en este capítulo.

En el siguiente Capítulo, 3, se encuentra el estado del arte, donde se realiza una reflexión acerca de varios documentos, recopilando información relevante para su posterior estudio. De cada artículo se sacarán tanto el método utilizado como los resultados obtenidos. Posteriormente se hará una comparación de los resultados para buscar métodos eficaces para el propósito del trabajo aquí presente.

El Capítulo 4, se abordan las diferentes métricas posibles junto con las librerías utilizadas en este documento. También se comentarán las diferentes contribuciones del trabajo, explicando paso a paso las aportaciones introducidas.

Posteriormente en el Capítulo 5, se mostrarán los experimentos realizados mediante Tablas y Figuras. También se encontrarán comparaciones de los distintos modelos y del trabajo presentado con otros trabajos.

Por último, en el Capítulo 6, se expondrán las conclusiones y el trabajo futuro de este proyecto.

Capítulo 2

Marco Conceptual

Este capítulo tiene un objetivo mayoritariamente introductorio, en el cual se pretende asentar las bases teóricas que se utilizarán como hilo conductor durante todo el Trabajo de Fin de Grado. En la Sección 2.1 se describen el origen de la *Artificial Intelligence* (AI), su crecimiento teórico y práctico y la repercusión de significativos eventos que han posibilitado su crecimiento. En la Sección 2.2 se describen las técnicas de Detección de Objetos, en inglés *Object Detection* (OD), detallando las más relevantes para este trabajo. Seguidamente, se expone información relativa al Aprendizaje automático, como el aprendizaje supervisado, el no supervisado, o el aprendizaje por refuerzo en la Sección 2.3. Dentro de la Sección de 2.3 también se describen las Redes Neuronales explicando su estructura y métodos de propagación. En las Secciones 2.4 y 2.5 se exponen los aspectos teóricos más relevantes de las Redes Neuronales Convolucionales y las Redes Neuronales Recurrentes, respectivamente. La Sección 2.6 se tratan aquellos aspectos más relevantes relacionados con el aprendizaje por transferencia, así como sus subdivisiones (inductiva, el transductiva y no supervisada). Finalmente, en la Sección 2.7 se explica el aprendizaje federado.

2.1. Inteligencia Artificial

Ya en el año 1950, Adan Turing redacta un artículo en el que plantea la siguiente pregunta: *"Can machines think?"* [Tur50a]. Podría decirse que este momento dio lugar al inicio de la Inteligencia Artificial, en inglés AI, aunque varios investigadores aseguran que su verdadero inicio fue en 1956, año en el que el informático John McCarthy realizó una conferencia que estaba guiada por una idea principal: reunir a unos cuantos investigadores para "el estudio del aprendizaje o bien cualquier otra característica de la inteligencia que pueden, en principio, ser descritas con tanta precisión que puede fabricarse una máquina para simularlo".

Cabe destacar la importancia del Z3, la primera computadora programable y

automática creada en 1941 por Konrad Zuse. En ella, los cálculos era realizados mediante la aritmética en coma flotante.

El primer diseño de una Red Neuronal Artificial del inglés *Artificial Neural Network* (ANN), fue en el año 1957 por Frank Rosenblat, psicólogo estadounidense que desarrolló la primera idea de lo que conocemos como el perceptrón. Este toma varias entradas binarias y para calcular la salida binaria se introduce el término de pesos, números reales que reflejan la importancia que tiene la entrada en relación con la salida. El primer prototipo es desarrollado mediante la computadora IBM 704. Un año más tarde Frank Roseblat publica [ROS58].

El primer chatbot lo podemos situar alrededor del 1966, desarrollado por Joseph Weizenbaum en el *Massachusetts Institute of Technology* (MIT), este programa fue el primero en incorporar el procesamiento de lenguaje natural humano. El confundador de MIT, Marvin Misnky, escribió 'Perceptrones' donde mencionaba los problemas más importantes de las primeras ANN.

En 1997 se encuentra un hecho clave para el avance de la AI, *International Business Machines Corporation* (IBM) desarrolla una computadora para jugar a la ajedrez llamada Deep Blue, esta fue la primera en derrotar al campeón del mundo en ese momento de ajedrez (Gary Kasparov). Este programa de ajedrez se desarrolló en C y se ejecutó en el sistema operativo *Advanced Interactive eXecutive* (AIX). Deep Blue estaba calificado para computar 200 millones de posiciones por segundo,lo que significaba que se había conseguido mejorar la versión de 1996 yendo dos veces más rápido que la anterior.

En el campo de la robótica, se encuentra el establecimiento de las leyes de esta misma, en el año 1941, recopiladas en un cuento de ciencia ficción conocido como 'Circulo vicioso' por Isaac Asimov. En 1938 H Roselund y W Pollard fabrican el primer brazo articulado haciando un importante cambio a la industria ya que fue la primera vez que se incorporan robots a una cadena de producción humana.

El término de 'automatización' se acuyó en 1947 por DS Halder, desde ese momento se empieza a maniobrar para sustituir a las personas por robots en muchas de las tareas de proceso de frabricación de automóviles. Simplemente un año más tarde G Walter presentó un robot autónomo electrónico.

AM Turing publica [TUR50b] donde plantea realizar una prueba, test o máquina de Touring, en la cual se mida el conocimiento de la máquina intentando simular el de un humano. Si la máquina se hacia pasar por un humano, esta habría pasado el test de Touring. En 1979 se inventa el primer vehículo autónomo, Stanford Cart, creado por la Universidad de Stanford.Stanford Cart es un robot capaz de seguir una lineas en el suelo de forma autónoma.

En cuanto a los inicios del Aprendizaje Profundo, en inglés *Deep Learning* (DL), alrededor del 2006 podemos encontrar varios papers con los que se arrancó ([HOT06],

[BLPL06], [RBL07]).

Hoy en dia podemos encontrar la AI en diferentes campos como el financiero, enseñanza, medicina, neurociencia, deportes, etc. Gracias a las facilidades que ofrecese la AI muchos de estos sectores pueden optimizar diversos recursos con el fin de incrementar su efectividad y rendimiento.

2.2. Detección de objetos

En 2013, Christian Szegedy, Alexander Toshev y Dumitru Erhan redactan un artículo [STE13], donde explotan el poder que tienen las Redes Neuronales Profundas, en inglés *Deep Neural Network* (DNN), para la OD. Su estudio se basa en crear máscaras de objetos mediante regresión. La red que se trata en este artículo se compone de una DNN convolucional, la cual esta compuesta por 7 capas, las 5 primeras son capas convolucionales (3 de estas tienen un agrupamiento máximo) y las dos últimas son dos capas completamente conectadas. En la última capa en vez de usar un clasificador de softmax, utilizan una capa de regresión, como ya se había mencionado anteriormente, que genera un objeto binario:

$$DNN(x, \Theta) \in \Re^N \tag{2.1}$$

donde, Θ son los parámetros que tiene la red y N es el número total de píxeles que contiene la imágen. Esta máscara binaria representa 1 si en ese píxel se encuentra un objeto de una clase dada y 0 en caso contrario.

La red de [STE13] se entrena para minimizar el error L_2 y para predecir la máscara de verdad $m \in [0, 1]^N$ para una imagen x:

$$\min_{\Theta} \sum_{(x,m)\in D} ||(Diag(m) + \lambda I)^{1/2}(DNN(x; \Theta) - m)||_2^2 \tag{2.2}$$

Ese mismo año se encuentra la aparición de las Redes Convolucionales basada en Regiones, en inglés *Region-based Convolutional Network* (R-CNN), redes fáciles de implementar y entrenar en comparación con las CNN de ventana deslizante, a parte de que proporcionan una solución unificada para la detección y segmentación de objetos. [GDDM] la OD mediante R-CNN se base en apartados, el primero donde se generan respuestas que no tienen que ver con las categorías, estas respuestas van a definir las posibles detecciones posibles para el detector, el segundo apartado se compone de una CNN que por cada región extrae un vector de características de longitud fija, por último se encuentra un conjunto de Máquinas de Vectores de Soporte, en inglés *Support Vector Machine* (SVM), que son específicos de la clase.

La proposición de Red en Red, en inglés *Network In Network* (NIN) Min Lin, Qiang Chen y Shuicheng Yan, surge este mismo año con el fin de mejorar la exclusión de parches

locales dentro del campo receptivo. [LCY13] Para esto construyen micro Redes Neuronales con estructuras más complejas para separar los datos dentro del campo receptivo. La estructura de NIN se basa en capas CNN que utilizan Perceptrón Multicapa, *MultiLayer Perceptron* (MLP), para convertir la entrada, con el objetivo de modelar mejor los parches locales y una capa de agrupación promedio global como reemplazo de las capas completamente conectadas en CNN, para evitar el sobreajuste global. En [LCY13] se demuestra el rendimiento mediante el conjunto de datos en CIFAR-10, CIFAR-100 y SVHN.

Otro hecho relevante a destacar del 2013 fue Overfeat (Sermanet et al.), [SEZ+13], donde proponen un tipo de ventana deslizante de múltiples escalas fácil de implementar en CNN para la clasificación, localización y OD. Demuestran también que se pueden aprender tareas simultáneas con una sola red compartida. Este nuevo enfoque ganó la competición de localización de *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) en 2013.

A principio de 2014 se encuentra MultiBox (Erhan et al.), con el cual se pretende entrenar una CNN que origine las coordenadas de los cuadros delimitadores de objetos directamente [SRE+14]. Multibox es una mejora de Inception. Multibox esta compuesto de una red profunda de 42 capas sobre un campo receptivo de 299 x 299, el cual contiene más de 130 capas. [SRE+14] usan la capa convolucional superior de 8 x 8 x 2048. Demuestran que la mejora superando la Precisión Media Media, en inglés *Mean Average Precision* (mAP), más cercano en más de un 10 %.

Entre el año 2014 y 2015 se encuentra la Combinación de Pirámides Espaciales, en inglés *Spatial Pyramid Pooling* (SPP), para eliminar la restricción de tamaño fijo de la red. [HZRS15c] se agrega una nueva capa SPP encima de la última capa, esta agrupa las características y genera salidas de longitud fija, para evitar la necesidad de recortar o deformar al principio. Esta nueva estructura se conoce como SPP-net. SPP es capaz de usar una salida de longitud fija, usar contenedores espaciales de niveles múltiples características que la agrupación de ventanas deslizantes no pueden. Esto nos permite poder entrenar a nuestra red con imágenes de diferentes tamaños.

Alrededor del 2014 localizamos VGGNet propuesto por Karen Simonyan y Andrew Zisserman en [SZ14]. Compitieron en ILSVRC de 2014, donde quedaron en primer lugar en las pistas de localización y en segundo lugar en las de clasificación. [SZ14] usan filtros pequeños de 3x3 (tamaño mínimo para poder capturar tanto arriba, abajo, izquierda y derecha). Este filtro mencionado anteriormente se mueve cada píxel cubriendo así la imagen entera. La capa final contiene el clasificador softmax, mientras que las capas ocultas están dotadas de la función de activación Unidad Lineal Rectificada, en inglés *Rectified Linear Unit* (ReLU).

Cerca del 2015 se localiza el inicio de GoogLeNet (Szegedy et al.) una red con 22 capas de profundidad, 27 si se cuentan las capas de agrupación. [SLJ+15] adoptan la arquitecturan de Inception pero con una arquitectura más profunda y amplia. La función

de activación utilizada para este modelo es ReLU. Esta red fue diseñada para que se pudiera usar en dispositivos individuales por lo que tuvieron en cuenta las limitaciones de estos, especialmente poniendo atención a la poca memoria de la que solían disponer.

Fast-R-CNN (Girshick) se situa en 2015, [Gir15], en comparación con [GDDM] este nuevo trabajo utiliza nuevas innovaciones para la mejora de la velocidad de entrenamiento. [Gir15] obtiene un mayor mAP más alto en PASCAL VOC2012. Este modelo está desarrollado con Python y C ++. La aquitectura de Fast-R-CNN se basa en varias capas de agrupación convolucionales por donde pasa la imagen inicial para conseguir un mapa de características. Posteriormente se encuentra una capa de Región de Interés, *Region of Interest* (RoI), para extraer un vector de características con longitud fija del mapa de características. [Gir15] Cada uno de estos vectores de características se alimentan de capas conectadas con dos capas al final, una de ellas produce estimaciones de probabilidad softmax sobre K objetos de clase, y la otra genera cuatro valores (cada conjunto codifica posiciones del cuadro de límite para la clase K) para cada una de las clases de K object.

Ese mismo año se desarrolla Faster-R-CNN (Ren et al.) introducciendo la OD en tiempo real con Redes de Propuestas Regionales, *Region Proposal Networks* (RPN). Las RPN toman una imagen, e independientemente de su tamaño se genera un conjunto de objetos rectangulares, cada uno con su respectiva puntuación de objetividad, este proceso se realiza mediante CNN. [RHGS15] consiguen que mediante su método se pueda ejecutar un modelo de OD entre 5 y 17 fps.

Un nuevo enfoque a la OD surge con *"Solo Miras Una Vez"*, en inglés *You Only Look Once* (YOLO) (Redmon et al.). Hasta ahora nos hemos encontrado con métodos que reutilizaban los clasificadores para la OD, en cambio este nuevo enfoque entiende este problema como una regresión a los cuadros delimitadores separados espacialmente y de probabilidades de las clases asociadas.[RDGF15] Los cuadros delimitadores mencionados anteriormente se consiguen mediante una red, en cambio las probabilidades se consiguen mediante imágenes completas en una evaluación. YOLO es capaz de procesar imágenes en tiempo real a 45fps. En comparación con los últimos OD, este nuevo enfoque comete más errores de localización, pero es capaz de conseguir predecir menos falsos positivos.

El concepto de aprendizaje residual surge en 2015, debido a que las redes profundas suelen ser muy difíciles de entrenar y con este método residual consiguen facilitar un poco el entrenamiento de estas [HZRS15a] (He et al.). Se reformulan las capas con fórmulas residuales con referencia a las capas de entrada, en lugar de tener que aprender funciones no referenciadas. En [HZRS15a] demuestran que las Redes Neuronales Residuales, en inglés *Residual neural network* (ResNet), son más fáciles de optimizar y de conseguir un mejor resultado a capas muy profundas. Desarrollan una red residual de 152 capas (8 veces más profundas que CNN) que logra obtener un 3,57 % en el conjunto de pruebas ImageNet, este resultado obtuvo el primer premio en ILSVRC 2015 en la tarea de clasificación.

Detección de Único Disparo, en inglés *Single Shot Detection* (SSD), aparece alrededor

del 2016 (Liu et al.), este nuevo enfoque modera la salida de los límites de cuadros predeterminados, sobre relaciones de aspecto dispares y según la ubicación que posean los escala en el mapa de características. [LAE+16] para las predicciones, la red genera puntuaciones para las presencias de objetos de cada categoría en cada cuadro, produciendo así un ajuste en el cuadro para que se adecue más a la forma del objeto. La diferencia de SSD con otros métodos es que elimina la generación de propuestas y el remuestreo de píxeles o características y realiza todos los cálculos en una misma red. Para entradas de 300 x 300 SSD consigue un mAP de 74,3 % y para 512 x 512 alcanza un 76,9 % de mAP, superando al modelo R-CNN.

Ese mismo año se encuentra las Redes Totalmente Convolucionales basadas en Regiones, en inglés *Region-based Fully Convolutional Networks* (R-FCN) (Dai et al.), un modelo que es convolucional y casi todos los cálculos que realiza se comparten en la imagen entera. Para esto [DLHS16] proponen mapas de puntuación sensibles a la posición para intentar solventar el problema de traducción-invariancia en la clasificación de imágenes y traducción-varianza en la OD. Con la ResNet101 y el conjunto de datos de PASCALVOC 2007 consiguen un mAP de 83.6 %. El resultado que obtienen lo consiguen en un tiempo de 170ms por imágen, 2.5-20 veces más rápido que Faster-R-CNN.

La retroalimentación de una red aparece por primera vez este año, 2016, con [HLW16], donde con apoyando el pensamiento de redes profundas deciden conectar cada capa en forma de retroalimentación, introduciendo DesNet. Una red convolucional tradicional con L capas tiene L conexiones,mientras que DesNet tiene $\frac{L(L+1)}{2}$ conexiones. La entrada de una capa son los mapas de características de todas las capas anteriores. [HLW16] las ventajas de utilizar DesNet son que mitigan el problema de desaparición del gradiente, fortalecen la propagación de características y reducen el número de parámetros.

[RF16] presenta YOLO9000 el cual es capaz de localizar en tiempo real 9000 categorías de objetos. Antes de presentar este, proponen una mejora de YOLO denominada YOLOv2, esta mejora puede realizar un entrenamiento de múltiples escalas. A 67fps YOLOv2 consigue un mAP de 76,8 % en VOC2007 y a 40fps un mAP de 78,6 % mejorando métodos como Faster R-CNN con ResNet y SSD. [RF16] proponen un método para entrenar conjuntamente la detección y clasificación de objetos, y con este entrenan a YOLO9000 que consigue obtener un mAP de 19,7 % en ImageNet teniendo solo datos de detección de 44 de las 200 clases.

Ya en 2017 se encuentra Mask R-CNN (He et al.) capaz de detectar objetos y a la vez generar un máscara de segmentación de alta calidad para cada instancia. [HGDG17] amplían el método de Faster R-CNN agregando una rama para predecir la máscara mencionada en paralelo con la rama que permite reconocer el cuadro delimitador. Mask R-CNN es fácil de generalizar permitiendo estimar diferentes poses humanas.

[LGG+17] en este artículo se estudia el por qué los detectores de una etapa son más rápidos y simples que los detectores de dos etapas. Encuentran un desequilibrio

entre las clases foreground-background. Para afrontar este desiquilibrio [LGG+17] deciden remodelar la pérdida de entropía cruzada, reduciendo la pérdida a ejemplos bien clasificados. Desarrollan el concepto de pérdida focal, Focal Loss, evitando que la gran cantidad de negativos saturen al detector durante el entrenamiento. Para probar efectividad de este desarrollan una red a la que llaman RetinaNet, la cual es capaz de igualar a los detectores de una etapa cuando se entrena con la pérdida focal, superando la precisión de detectores de dos etapas de la última generación.

2.3. Aprendizaje automático

Una red neuronal es capaz modificar sus pesos en función a la entrada recibida. Creando, eliminando y modificando conexiones entre las diferentes neuronas de la red. Para conseguir una de estas conexiones se necesita que el valor del peso no sea nulo, ya que si esto sucede en vez de crearse una conexión se eliminará. Durante todo el proceso de aprendizaje los valores de los pesos se van modificando hasta que llegan a un punto en el que se mantienen estables, entonces es cuando podremos afirmar que hemos terminado el proceso de entrenamiento. Para la modificación de los pesos en *Machine Learning* (ML) se pueden seguir diferentes métodos de aprendizaje: (i) Aprendizaje Supervisado, (ii) Aprendizaje no Supervisado, (iii) Aprendizaje por Refuerzo y (iv) Aprendizaje Profundo.

2.3.1. Aprendizaje supervisado

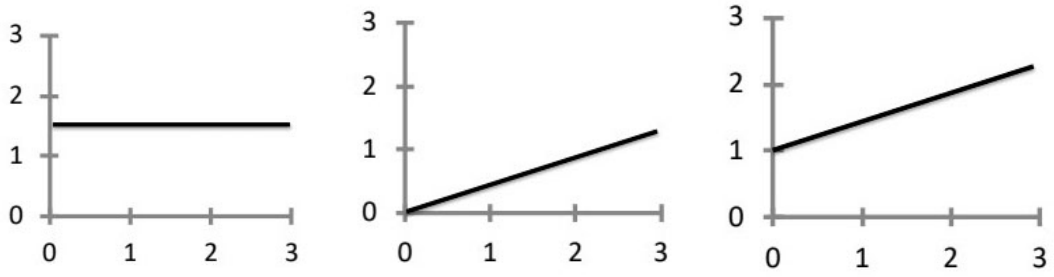
El Aprendizaje Supervisado, *Supervised Learning* (SL), se caracteriza por entrenar a la red con un conjunto de datos etiquetados con la respuesta correcta. Un ejemplo claro de esto sería queremos adivinar el precio de una casa, para esto enviaremos un conjunto de datos sobre casas, las cuales están etiquetadas con su respectivo precio real, así la red será capaz de aprender y podremos enviarle nuevos datos para intentar predecir el precio de la nueva casa. En SL se encuentra diferentes técnicas cómo Regresión lineal, Regresión Logística, Máquina de Vectores de Soporte, Árboles de decisión y Redes Neuronales Artificiales.

2.3.1.1. Regresión Lineal

Para predecir una salida de valor continuo, el ejemplo de predecir el precio de una casa sería un problema de Regresión Lineal. De entrada se tienen los m^2 de la casa con su respectivo precio. Posteriormente se procede a realizar la predicción de nuevos datos. La hipótesis de este algoritmo es la siguiente:

$$h_\Theta(x) = \Theta_0 + \Theta_1 x \tag{2.3}$$

Como se observa en la Figura 2.1, por cada valor que tome Θ hay una recta diferente.



(a) $\Theta_0 = 1.5; \Theta_1 = 0$ (b) $\Theta_0 = 0; \Theta_1 = 0.5$ (c) $\Theta_0 = 1; \Theta_1 = 0.5$

Figura 2.1: Rectas según el valor de Θ

La función de coste $J(\Theta_0, \Theta_1)$ [Wei05] se multiplica por el Error Cuadrático Medio, en inglés *Mean Squared Error* (MSE), es decir, la media de las diferencias entre la predicción y el valor real para cada ejemplo. El principal objetivo de esta función es medir la exactitud de la función de hipótesis.

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{K} (\hat{y}_i - y^{(i)})^2 = \frac{1}{2m} \sum_{i=1}^{K} (h_\Theta(x^{(i)}) - y^{(i)})^2 \tag{2.4}$$

También se encuentra el descenso de gradiente con el cuál modificamos los valores de Θ_0 y Θ_1 de forma iterativa para minimizar la función de coste. Esta iteración se repite hasta que el coste converja hacia un mínimo global establecido previamente por nosotros mismos. Repetir hasta que converja:

$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta) = \Theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - (y^{(i)}))(x_j^{(i)}) \tag{2.5}$$

Hay que tener cuidado al elegir el valor de la tasa de aprendizaje α,ya que si el valor es muy pequeño tardará mucho en converger, mientras que si el valor de α es muy grande el entrenamiento tenderá a desplazarse y a superar el mínimo global.

Hemos visto el caso para una variable, [Wei05] pero si tenemos varias variables de entrada, por ejemplo en el problema mencionado anteriormente de la casa, que a parte de darnos el tamaño de la casa para predecir el precio de esta, nos den el número de habitaciones, el número de plantas y cuantos años tiene la casa. Sería el mismo procedimiento pero teniendo en cuenta que ahora la hipótesis sería de tal forma:

$$h_\Theta(x) = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + ... + \Theta_n x_n \tag{2.6}$$

2.3.1.2. Regresión logística

Es un algorítmo de clasificación que se encarga de elegir si la entrada pertenece a una clase o a otra. Para entender mejor el concepto se muestran varios ejemplos claros, como poder deducir si un email es spam o no, si hay transiciones fraudolentas o no, si el tumor es maligno o no... En estos casos las salidas las podemos representar mediante dos clases binarias $y \in 0, 1$, 0 la clase negativa (no spam) si $h_\Theta(x) < 0.5$ y 1 la clase positiva (spam) si $h_\Theta(x) \geq 0.5$. [KDG+02] También podremos encontrarnos el caso en el que nos den más de dos clases por lo que no podremos representarlas de forma binaria $y \in 0, 1, 2, 3, ..., n$, pero esto se explicará más adelante.

Representación de la hipótesis de regresión logística teniendo en cuenta que g es la función de activación sigmoide.

$$h_\Theta(x) = g(\Theta^T x) \tag{2.7}$$

$$g(z) = \frac{1}{1 + e^{-z}} \tag{2.8}$$

Ecuación del gradiente:

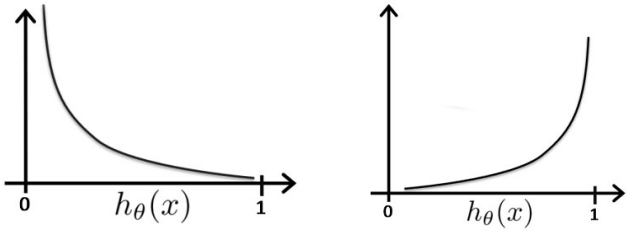
$$\frac{\delta J(\Theta)}{\delta J \Theta_j} = \frac{1}{m} X^T (g(X\Theta) - y) \tag{2.9}$$

La función de coste la podemos representar como:

$$Cost(h_\Theta(x), y) = \begin{cases} -(\log(h_\Theta(x)) & \text{si y} = 1 \\ -(\log(1 - h_\Theta(x)) & \text{si y} = 0 \end{cases} \tag{2.10}$$

Para simplificarlo multiplicamos por y en un caso y por (1-y) en el otro para que siempre uno de estos se haga 0 teniendo en cuenta el valor de y, si y = 1 o y = 0.

$$J(\Theta) = -\frac{1}{m}((\log(g(X\Theta)))^T y + (\log(1 - g(X\Theta)))^T (1 - y)) \tag{2.11}$$



(a) $y = 1$ (b) $y = 0$

Figura 2.2: Valores del coste en función de y

En el caso de que $y = 1$ y $h_\Theta(x) = 1$ tenderá a tener un coste = 0, mientras que si $h_\Theta(x) = 0$ se penalizará con un valor muy alto como se muestra en la Figura 2.2(a).

En el caso contrario la Figura 2.2(b), $y = 0$, y $h_\Theta(x) = 1$ entonces se penaliza con un valor muy alto.

[KDG+02] Podemos encontrarnos con el problema de sobreajuste, es decir tener un modelo con datos muy ajustados evitando que se generalicen y provocando un sobreajuste a los datos de entrada. Para esto se puede añadir el concepto de regularización a nuestro modelo.

Hay dos tipos de regularización:

- **Regularización Lasso (L1)**: Se usa cuando se pretende que la solución sea poco densa, ya que esta regularización tiende a forzar a los coeficientes a tender a 0, excluyendo así a los predictores menos relevantes. El grado de penalización está controlado con λ. A medida que λ vaya aumentando mayor será la penalización y por lo tanto encontraremos más exclusión hacia los predictores.

$$C = \frac{1}{N}\sum_{j=1}^{N}|w_i| \tag{2.12}$$

- **Regularización Ridge (L2)**: Ridge se utiliza cuando tenemos varios atributos correlaciones. Reduce la importancia de cada uno proporcionalmente pero sin llegar a 0 como era el caso de Lasso. Esta reducción a los atributos de entrada permite que minimice la correlación entre ellos y por tanto conseguir una mejora a la hora de generalizar.

$$C = \frac{1}{2N}\sum_{j=1}^{N}w_i^2 \tag{2.13}$$

Por lo tanto añadiendo el concepto de regularización a la función de coste quedaría como se muestra a continuación:

$$J(\Theta) = -\frac{1}{m}((\log(g(X\Theta)))^T y + (\log(1 - g(X\Theta)))^T(1 - y) + \frac{1}{2m}\sum_{j=1}^{N}\Theta_j^2 \tag{2.14}$$

Mientras que la Ecuación 2.15 del gradiente tendría este aspecto:

$$\frac{\delta J(\Theta)}{\delta J\Theta_j} = \frac{1}{m}X^T(g(X\Theta) - y) + \frac{\lambda}{m}\Theta_j \tag{2.15}$$

Como habíamos comentado anteriormente podemos encontrarnos en la situación de que nos den más de dos clases como entrada [KDG+02], evitando la representación de las mismas de forma binaria y representándolo de la siguiente manera $y \in 0, 1, 2, 3, ..., n$. La forma de resolver estos problemas sería poner a 1 una de las clases y las demás a 0, obteniendo así un problema de clasificación binario. Esto se repite hasta que todas las clases hayan obtenido una vez el valor 1. Este método se conoce como uno contra todos, se puede observar un ejemplo en la Figura 2.3.

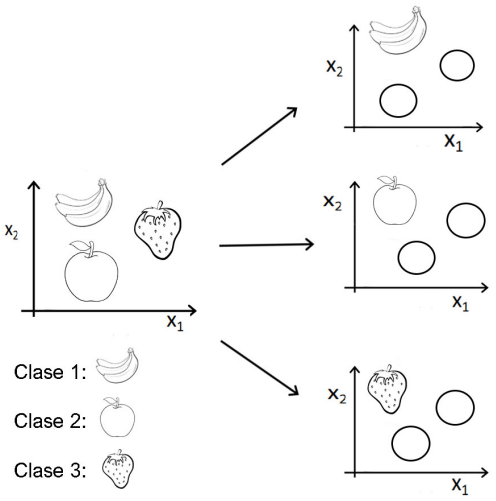


Figura 2.3: Ejemplo de uno contra todos

2.3.1.3. Máquinas de Vectores de Soporte

[NWH03] Las Máquinas de Vectores de Soporte, SVM, es un método de clasificación que se basa en separar ambas clases mediante una recta o un hiperplano con el mayor margen de separación posible. Estas se encargan de encontrar los vectores que dan soporte a la frontera de decisión, el vector Θ es perpendicular a esta. La diferencia de SVM con regresión logística es que intenta buscar el máximo margen posible. El objetivo es minimizar Θ, siempre que se cumpla que la proyección del ejemplo por la norma de theta, $||\Theta||$, es ≥ 1 si $y = 1$ ó ≤ 1 si $y = 0$.

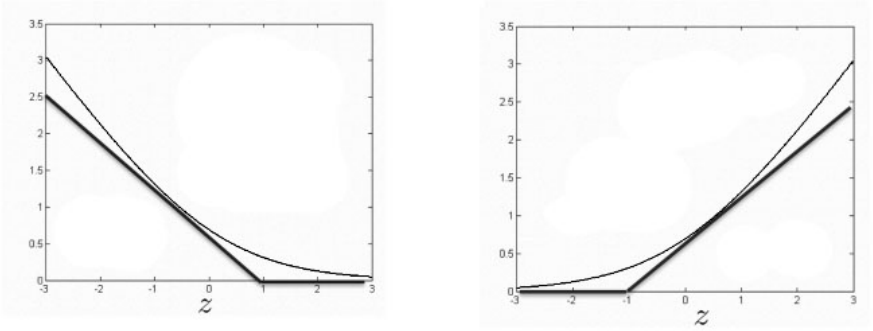
$$min_\Theta \frac{1}{2} \sum_{j=1}^{n} \Theta_j^2 = \frac{1}{2} ||\Theta||^2 \tag{2.16}$$

$$p^{(i)} \cdot ||\Theta|| \geq 1 \quad si \quad y^{(i)} = 1 \tag{2.17}$$

$$p^{(i)} \cdot ||\Theta|| \leq 1 \quad si \quad y^{(i)} = 0 \tag{2.18}$$

$$\Theta^T x(i) = p^{(i)} \cdot ||\Theta|| \tag{2.19}$$

Si $y = 1$, se mantiene el coste en 0 mientras la $\Theta^T x > 1$ y cuando $\Theta^T x < 1$ el coste crece exponecialmente. Se puede observar en la Figura 2.4(a) que el coste de 1 es calcular el máximo entre 0 y 1 - z $cost_1(z)$ $max(0, 1 - z)$. En cambio, si $y = 0$, se mantiene el coste en 0 mientras la $\Theta^T x < -1$ y cuando $\Theta^T x > -1$ el coste crece exponecialmente. Se puede observar en la Figura 2.4(b) que el coste de 0 es calcular el máximo entre 0 y 1 + z $cost_0(z)$ $max(0, 1 + z)$.



(a) $y = 1$ (b) $y = 0$

Figura 2.4: Representación gráfica del coste

La ecuación del coste para minimizar Θ no se divide entre m ,a diferencia como habíamos visto antes, por lo que ya no es el coste medio si no el coste total. [NWH03] A diferencia con regresión logística regularizada, en este caso se quita λ y y multiplicamos por C, su efecto sería inverso al de λ, ya que en vez de estar multiplicando a Θ está multiplicando al coste. Por lo que cuando C es grande, significa que nos ajustamos mucho, y cuando C es pequeño, significa que nos ajustamos poco.

$$min_\Theta C \sum_{i=1}^{m} [cost_1(\Theta^T x^{(i)}) y^i + cost_0(\Theta^T x^{(i)})(1 - y^{(i)})] + \frac{1}{2} \sum_{i=1}^{n} \Theta_j^2 \tag{2.20}$$

Cuando los datos no son linealmente separables usaremos los kernels de SVM, como por ejemplo el kernel gaussiano, que lo que hará será pasar de dimensión n a dimensión m. Se inicializa la C y la σ con los respectivos valores con los que queramos trabajar. La ecuación del kernel gaussiano se puede observar en la Fórmula 2.21.

$$f_i = \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right) \tag{2.21}$$

2.3.1.4. Árboles de decisión

[PARS13] Es un algoritmo donde cada rama representa una decisión y cada nodo hoja representa una decisión. Los árboles de decisiones están divididos en nodos, de los cuales podemos diferenciar:

- **Nodo raíz**: nodo principal donde se encuentra la primera división.
- **Nodos intermedios**: que son los que se encuentran después de la raíz.
- **Nodos hojas o terminales**: son aquellos que se encuentran en las terminaciones del árbol encontrándose la clasificación definitiva.

Otro concepto que se debe tener en conocimiento es la profundidad del árbol, la cual viene definida por el número máximo de nodos de una rama. Algunas de las ventajas de los árboles de decisión entre otras son, su fácil construcción e interpretación, y que solo escogen las variables más importantes.

Para seleccionar que variable tomar podemos utilizar varios métodos como: índice Gini, el error de clasificación, o la entropía.

- **Índice de Gini**: Es una medida de impureza, la cual nos mide la probabilidad de sacar dos elementos de una población y que sean de la misma clase. La probabilidad de cada clase, p_c, se calcula como el número de dado de clases en cada nodo entre el número de muestras totales por nodo.

$$gini = 1 - \sum_{k=1}^{n} p_c^2 \tag{2.22}$$

- **Entropía**: La teoría de la ganancia de la información expone que un nodo puro necesita menos información para ser descrito que un nodo que sea más impuro. Es una manera de definir el grado de desordenación que tiene un sistema. Si un nodo es puro se representa con una entropía de 0 mientras que un nodo impuro con 1.

$$H = -\sum_{k=1}^{n} p_c^2 * \log_2 p_c \tag{2.23}$$

- **RSS**: Cuando es un problema de regresión se suele utilizar la Suma Residual de Cuadrados, en inglés *Residual Sum of Squares* (RSS). Que sirve para ver la diferencia entre los datos reales y los predichos por el propio modelo.

$$RSS = \sum_{k=1}^{n} (y_c - \hat{y}_c)^2 \tag{2.24}$$

2.3.1.5. K-vecinos más cercanos

A diferencia de los árboles de decisión, los k-Vecinos más Cercanos, en inglés *k-Nearest Neighbors* (kNN), solo guardan las instancias, no creando así un modelo como tal. kNN es local, es decir,[Pet09] asumen que dependen únicamente de los k vecinos más cercanos.

Si se escoge una $k = 1$ tienen más influencia aquellas clases con ruido, con $k > 1$ tienen más importancia los vecinos que el propio ruido, en cambio, si se escoge una k muy alta la percepción de localización se pierde. Si hay dos clases se recomienda usar una k impar para que puede desempatar. El valor de k se suele implantar mediante la validación cruzada de los datos de entrenamiento.

kNN suele ser demasiado lento, pero eliminando las instancias superfluas (no son necesarias para clasificar) se decrementará el tiempo de clasificación, a esta técnica se la conoce como condensación. También nos podemos encontrar con instancias ruidosas que pueden llegar a ser confusas para el clasificador, si las eliminamos conseguiremos mejor porcentaje de aciertos, esta técnica es conocida como *editing*.

2.3.1.6. Redes Neuronales Artificiales

Las Redes Neuronales surgen con el objetivo de imitar el cerebro humano. Están compuestas por varias neuronas conectadas que emiten señales entre sí. [Yeg09] La estructura de una red neuronal se muestra en la Figura 2.5.

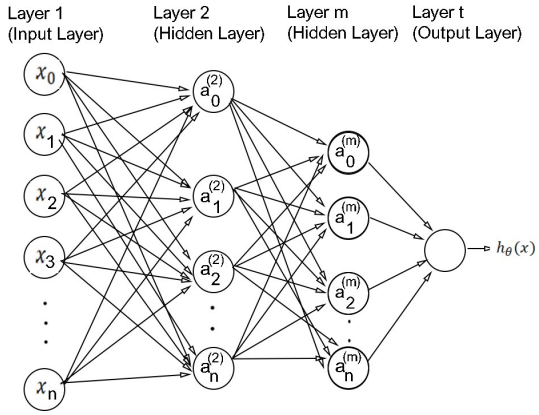


Figura 2.5: Estructura red neuronal

La red está compuesta por m capas con n neuronas en la última capa oculta de (m). El uso de muchas neuronas en las capas ocultas ralentiza el entrenamiento y puede provocar un sobreajuste, *overfitting*, es decir, puede llegar a no generalizar, lo que supone un problema con nuevos datos de entrada, en cambio, si tiene pocas neuronas puede llegar a no resolver el problema, *underfitting*. La capa de entrada tiene tantas neuronas como características tenga nuestro problema, por otro lado, la capa de salida dependerá del vector de salida.

La activación de las n neuronas en la capa m se realiza con la función activación a_n^m que se encarga de calcular estado de actividad de una neurona, para esto cambia la entrada en un valor de activación, el que a menudo se encuentran números entre $(0, 1)$ o entre $(-1, 1)$. La neurona se encuentra totalmente inativa cuando toma los valores de 0 ó -1 o activa 1.

La intensidad de la señal que se traspasa a la siguiente capa se determina con $\Theta^{(m)}$ o matriz de pesos de la red neuronal que se modifican mediante el entrenamiento. Se encuentrarán $m-1$ pesos. El número de filas de Θ viene dado por el número de nodos de la capa a la que llega $(m+1)$, mientras que las columnas serán el número de nodos de la capa donde está m.

Si se inicializa a 0 los pesos al principio, al actualizar seguiría teniendo el mismo valor, y por tanto estaría calculando la misma característica oculta. Para que esto no suceda se inicializan aleatoriamente entre el intervalo de $[-\epsilon, \epsilon]$. Al final se concatenan en un único vector unidimensional para conseguir la generalización del problema, provocando que la propagación hacia atrás *backprop* no sepa cuantas matrices de pesos hay.

Un ejemplo de la inicialización de los pesos en python se puede observar en el Código 2.1.

Listing 2.1: Inicialización de los pesos

```
INIT_EPSILON = 0.12

Theta1 = np.random.random((num_ocultas, num_entradas + 1)) *
        (2 * INIT_EPSILON) - INIT_EPSILON

Theta2 = np.random.random((num_etiquetas, num_ocultas + 1)) *
        (2 * INIT_EPSILON) - INIT_EPSILON
np.concatenate((np.ravel(Theta1),np.ravel(Theta2)))
```

Sobre el modelo neuronal se encuentra una función f, función de transferencia, que está asignada a una neurona tal y como se muestra en la Ecuación 2.25.

$$h_\Theta(x) = f(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + ... + \Theta_n x_n) \tag{2.25}$$

Hay varios tipos de función de transferencia, algunas de ellas son:

- **Función lineal de transferencia**: Se utiliza en la capa de salida de la red neuronal mediante la Ecuación 2.26.

$$f(h_\Theta(x)) = h_\Theta(x)) \tag{2.26}$$

- **Función de transferencia log-sigmoidea**: Es útil en problemas de clasificación ya que devuelve valores comprendidos entre 1 y 0, donde 1 representa una clase y 0

la clase contraria, esto se puede observar en la Ecuación 2.27.

$$f(h_\Theta(x)) = \frac{1}{1 + \exp^{-h_\Theta(x)}} \tag{2.27}$$

- **Función de transferencia sigmoidea tangente hiperbólica**: Se encarga de configurar las neuronas que se encuentran en las capas ocultas y devuelve valores entre -1 y 1, se encuentra la Ecuación 2.28.

$$f(h_\Theta(x)) = \frac{2}{1 + exp^{-2 \cdot h_\Theta(x)}} - 1 \tag{2.28}$$

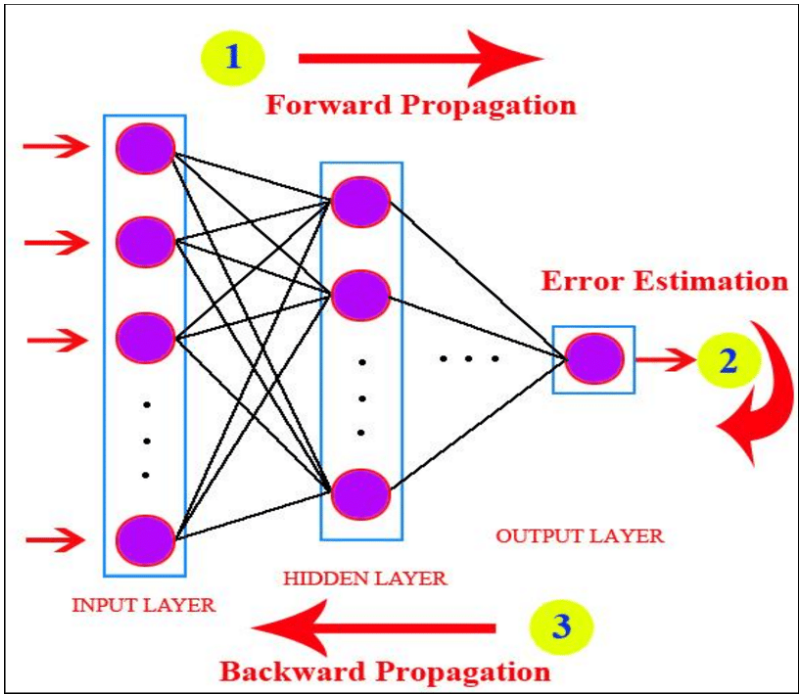


Figura 2.6: Ejemplo ilustrado de propagación hacia delante y hacia atrás. [Res21]

- **Propagación hacia delante**: Sirve para conseguir los valores de activación de cada capa $a^{(m)}$. Para la primera capa el valor de activación $a^{(1)}$ será equivalente a los valores de entrada X, mientras que para las capas posteriores $a^{(m)}$, se realizan los dos siguientes pasos:

  1. Multiplicar la matriz de pesos Θ correspondiente con la activación conseguida en la anterior capa. Como resultado obtendremos el vector $z(i)$.
  2. Realizar la función de activación sigmoide de $z(i)$ para conseguir $a(i)$.

  Un ejemplo sencillo de lo explicado anteriormente:

$$a^{(1)} = X \tag{2.29}$$

$$z^{(2)} = \Theta^{(1)} a^{(1)} \tag{2.30}$$

$$a^{(2)} = g(z^{(2)}) \quad (add \quad a_0^{(2)}) \tag{2.31}$$

$$z^{(3)} = \Theta^{(2)} a^{(2)} \tag{2.32}$$

$$h_\Theta(x) = a^{(3)} = g(z^{(3)}) \quad (add \quad a_0^{(3)}) \tag{2.33}$$

También tendremos que añadir un 1 (por el término independiente para que el número de las dimensiones de las matrices concuerden).

- **Propagación hacia atrás**: La propagación hacia atrás al igual que en otros métodos de ML como la regresión lineal, mediante el descenso de gradiente, sirve para minimizar la función de coste $J(\Theta)$. En la última capa el error será la resta entre el resultado y lo que debería salir, almacenado en Y. En la primera capa no se encuentra error ya que son las entradas, X, de la red directamente. Para calcular el error de las diferentes capas de nuestra red multiplicamos la matriz de pesos de esa capa traspuesta, por el delta de la siguiente capa, y esto lo multiplicamos elemento a elemento (*) por la derivada de la z de esa capa.

$$\delta^{(i)} = (\Theta^{(i)})^T \delta^{(i+1)} \cdot * g'(z^{(i)}) \tag{2.34}$$

No hace falta guardar la z explicada en propagación hacia delante, si no que multiplicamos elemento a elemento el vector a de esa misma capa, por el vector 1 menos a. El vector 1, tiene tantos 1 como elementos tiene a.

$$g'(z^{(l)} = a^{(l)} \cdot * (\vec{1} - a^{(l)}) \tag{2.35}$$

Podemos encontrar otras variantes de este algoritmo [Opt], como por ejemplo:

- *Stochastic Gradient Descent*: *Stochastic Gradient Descent* (SGD), se basa en actualizar los diferentes parámetros Θ en la dirección negativa del gradiente g (para un objetivo de minimización) tomando un subconjunto o"mini-lote" de tamaño de datos (m).
- *Adaptive Gradient Algorithm*: El Algoritmo de Gradiente Adaptativo, en inglés *Adaptive Gradient Algorithm* (AdaGrad), mejora el rendimiento el problemas con gradientes escasos. Tasa de aprendizaje por parámetro.
- *Root Mean Square Propagation*: La Propagación Cuadrática Media, en inglés *Root Mean Square Propagation* (RMSProp), al igual que AdaGrad se encuentra tasa de aprendizaje por parámetro y es capaz de adaptarse según las magnitudes de los gradientes para el peso, es decir, como de rápido están cambiando. Este tipo de algoritmo suele funcionar bien en aquellos problemas que no son fijos.

- *Adam* [KB14]: Adam, es una combinación de AdaGrad y RMSProp. Es muy adecuado para aquellos problemas con gran cantidad de datos y/o parámetros, también puede ser conveniente para aquellos problemas no estacionarios y con gradientes muy ruidosos y/o dispersos. Los hiperparámetros que tiene este optimizador no requieren mucho ajuste porque suele ser intuitivos.

2.3.2. Aprendizaje no supervisado

A diferencia del SL, los datos de entrada del Aprendizaje no Supervisado, *Unsupervised Learning* (UL), [Bar89] se proporcionan sin etiquetas de salida, por lo que permite revelar distintos patrones en los datos que no eran conocidos previamente. Un ejemplo de lenguaje supervisado es cuando se intentan agrupar varios grupos de clientes para realizar una estrategia de marketing. Este tipo de aprendizaje se puede dividir en dos categorías.

- **Análisis de clúster**: Dónde podemos encontrar varios tipos, como el agrupamiento basado en particiones (Partitioning Clustering), un método de clasificación de varios grupos dependiendo de la similitud de los mismos, en el que se necesita la especificación del número de clústers que se van a usar de antemano. Varios ejemplos de algorítmos de este tipo serían: *K-means, K-medoids, CLARA*.

  Por otro lado se encuentra el agrupamiento jerárquico (Hierarchical Clustering) en el que se busca crear jerarquías de grupos, en este tipo de algorítmos no se requiere la especificación del número de clústers de antemano. Ejemplos de este tipo de algorítmos: *agglomerative clustering, divisive clusterig.*

  Para terminar se pueden encontrar algorítmos que combinen los dos anteriores como es el caso de: *hierarchical K-means, fuzzy clustering, model based clustering y density based clustering.*

- **Reducción de la dimensionalidad**: Como su propio nombre indica, su objetivo es conseguir que a la hora de analizar se reduzca el número de variables. Con esto conseguiremos una reducción del ruido en el conjuto de datos que estamos usando y analizaremos los resultaros de una manera más sencilla y clara.

  A grandes rasgos podemos encontrarnos tres formas de ejecutar la reducción de la dimensionalidad. La primera sería mediante la selección de características, donde se seleccionarían carazterísticas originales que representen al conjunto de datos que se va a utilizar. La segunda forma, mediante la derivación de características, donde se crean nuevas características combinando dos o más características de las originales, sustituyendo estas por las nuevas sin producir un pérdida exagerada de información.

2.3.3. Aprendizaje por Refuerzo

El Aprendizaje por Refuerzo, en inglés *Reinforcement Learning* (RL), [KLM96] es un aprendizaje que se basa en maximizar las *recompensas* en función del estado actual y la acción que se ha realizado. En este tipo de aprendizaje no tenemos salidas etiquetadas con su solución real por lo que no se trata de SL, y aprenden de manera autónoma por lo que tampoco se trataría de un UL.

Para este nuevo planteamiento se necesita un agente que será el propio modelo quien se encargará de tomar decisiones, y un entorno donde se mueva el agente con limitaciones y reglas que tiene que seguir en cada momento, a este entorno se le conoce como ambiente. Estos se encuentran interconectados mediante distintas relaciones, donde se encuentran las acciones que puede tomar el agente en un momento determinado, los diferentes estados de los distintos componentes del ambiente en ese momento y las recompensas/castigos que se generan tras la elección de la acción en función de si ha sido buena o no.

En la Figura 2.8 podemos observar los diferentes componentes e interconectores mencionados anteriormente.

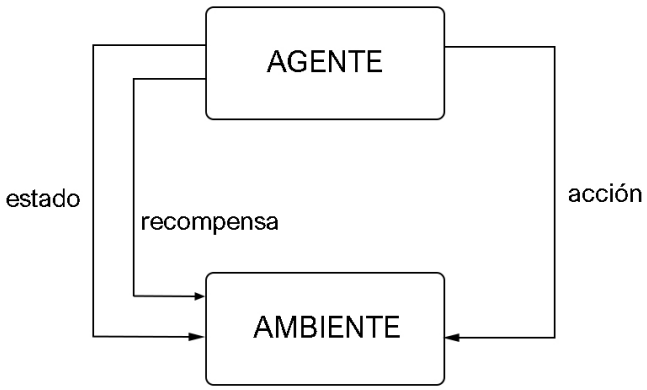


Figura 2.7: Componentes del aprendizaje por refuerzo

2.3.4. Aprendizaje Profundo

El Aprendizaje Profundo, DL, es un aspecto del ML que posibilita el aprendizaje de la experiencia y la comprensión en forma de jerarquía de conceptos, con la que a través de conceptos simples consigue aprender conceptos más complicados. Debido a que la propia computadora reúne el aprendizaje de la experiencia, no necesita la intervención de un humano que indique el conocimiento que ésta necesita.

Dentro del DL podemos encontrar las CNN y las Redes Neuronales Recurrentes, en inglés *Recurrent Neural Networks* (RNN). Ambas se explicarán en secciones posteriores, debido a su importancia en este trabajo.

2.4. Redes Neuronales Convolucionales

Las CNN son principalmente utilizadas para resolver tareas difíciles de reconocimiento de patrones basados en imágenes.

Una de las diferencias claves respecto a las ANN es que las neuronas que forman parte de las CNN se encuentran en tres dimensiones, siendo cada una de estas la altura, el ancho y la profundidad de la entrada, también que éstas se encuentran dentro de cualquier capa de CNN solo estarán conectadas de la capa que les precede mediante una pequeña región.

2.4.1. Estructura

Las CNN se constituyen de tres prototipos de capas, las capas convolucionales, las capas de agrupación y las capas completamente conectadas. [ON15] Al igual que las ANN, los valores de píxeles de la imagen se encuentra en la capa de entrada.

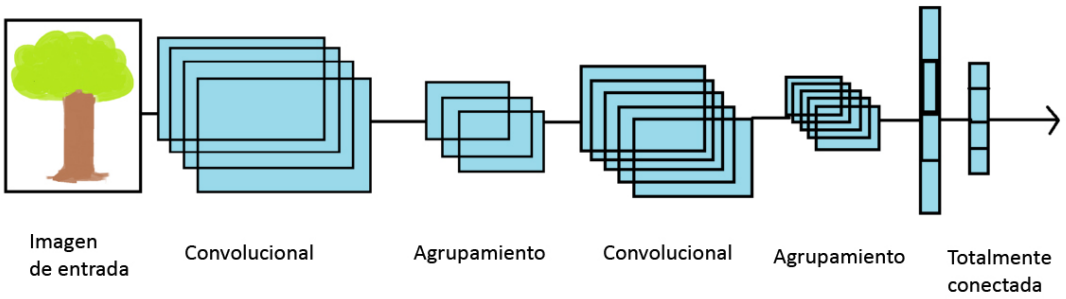


Figura 2.8: Arquitectura de red convolucional [Deb21]

- **Capa convolucional**: Determina la salida de las neuronas que están conectadas a regiones locales de la entrada, por medio del computo del producto escalar entre sus pesos y la región conectada al volumen de entrada. Las capas contienen parámetro que se concentran en el uso de núcleos (*kernels*) aptos para el aprendizaje, estos suelen tener una dimensionalidad pequeña, se puede observar un ejemplo en la Figura 2.9. Cuando llegan los datos de entrada a esta capa, esta convoluciona cada filtro a través de todo el espacio de dimensionalidad produciendo así un mapa de activación 2D. A medida que nos movemos con el filtro por los datos entrada, para cada valor se computa el producto escalar en ese núcleo. Una vez realizado ésto, la red es capaz de aprender cuales de los núcleos se activan viendo las características específicas en una determinada posición de la entrada. Estos son comúnmente conocidos como

activaciones. Cada núcleo tendrá su correspondiente mapa de activación, los cuales se apilarán formando el volumen de salida de la capa convolucional. Entrenar una ANN con imágenes como entradas da como resultado modelos muy grandes siendo una manera poco efectiva, por esta razón, cada neurona de una capa convolucional solo está conectada a una pequeña región del volumen de entrada, la dimensionalidad de esta se conoce habitualmente como el tamaño del campo receptivo de la neurona. Un ejemplo para entender esto mejor sería, imaginemos que tenemos como entrada una imagen de 64 x 64 x 3, es decir una imagen de tamaño 64 x 64 a color, si configuramos un campo receptivo de tamaño 6 x 6 tendremos 108 pesos, mientras que si introducimos esta imagen como entrada a una ANN tendríamos 12.288 pesos.

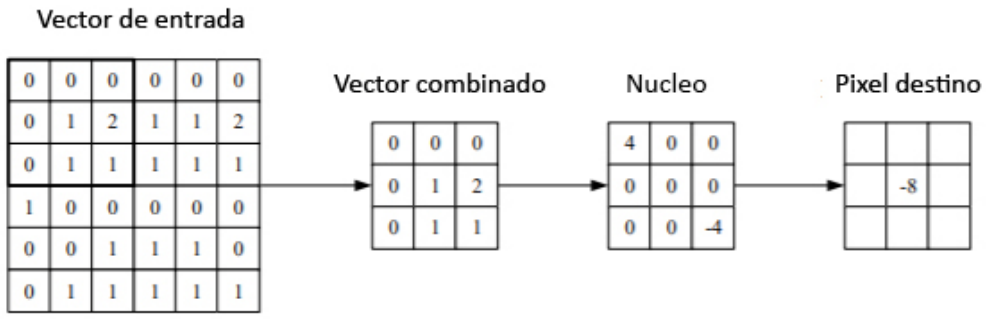


Figura 2.9: Representación de una capa convolucional, imagen recogida de [ON15]

- **Capa de agrupación**: La capa de agrupación, como su propio nombre indica, será la encargada de reducir la dimensionalidad, reduciendo por tanto, los parámetros utilizados y la dificultad del problema. Cada mapa de activación es operado por esta capa utilizando una función de "max", agrupación máxima. Los núcleos superiores a 3 como regla general deteriorarán el rendimiento del modelo, ya que esta capa suele tener una naturaleza destructiva. Aparte de la agrupación máxima, podemos encontrar la agrupación general, las capas con esta agrupación están compuestas por neuronas agrupadas que pueden realizar una varias operaciones habituales, incluida tanto la agrupación promedio como la normalización L1 / L2.

- **Capa completamente conectada**: Esta capa obtendrá puntuaciones de clase mediante las activaciones que posteriormente serán usadas para la clasificación, a parte de realizar funciones parecidas a las de ANN. Para mejorar el rendimiento se sugiere utilizar ReLU. Estas capas contienen neuronas en las que todas están conectadas entre la capa anterior y la capa siguiente. Este tipo de capas suelen encontrarse al final de la red.

Las CNN son capaces de reducir la complejidad del modelo mediante la optimización a través de tres hiper-parámetros, como la profundidad que se puede establecer

manualmente,al reducir este hiper-parámetro se puede minimizar significativamente el número total de neuronas de la red, pero también puede reducir significativamente las capacidades de reconocimiento de patrones del modelo, otro hiper-parámetro como la zancada, en la que si estableciésemos esta a 1, entonces el campo receptivo estaría muy superpuesto y por lo tanto, produciría activaciones extremadamente grandes, sin embargo, si la zancada es un número mayor, reduciría la cantidad de superposición y producirá una salida de dimensiones espaciales más bajas, el último hiper-parámetro sería el relleno de ceros que es un método eficaz para proporcionar un mayor control en cuanto a la dimensionalidad de los volúmenes de salida mediante el simple proceso de rellenar el borde de la entrada y es un método eficaz para proporcionar un mayor control en cuanto a la dimensionalidad de los volúmenes de salida. Otro de los métodos para reducir la complejidad del modelo es utilizar el uso compartido de parámetros, la cual considera que si una característica de región es útil para esta, también lo será para otra región. Si se restringen los mapas de activación a los mismo pesos y sesgo, encontraremos una gran reducción de parámetros en CNN. En la etapa de propagación hacia atrás solo se actualizará un conjunto de pesos, a diferencia de cada uno.

2.5. Redes Neuronales Recurrentes

Las RNN han sido parte de la investigación de Redes Neuronales en 1990. Actualmente se suelen utilizar desde detección de movimiento y la síntesis de música hasta la previsión financiera.

"Una red recurrente es una red neuronal con conexiones de retroalimentación (circuito cerrado)" Fausett, 1994.

Las RNN fueron introducidas para la identificación de caractéres a finales de la década de 1980 por varios investigadores como Rumelhart, Hinton y Williams. Otras aplicaciones más actuales se pueden ver en la Tabla 2.1, donde se pueden ver la amplitud de las aplicaciones recientes de las Redes Neuronales recurrentes.

En las redes recurrentes se encuentran dos tipos de entrenamientos, uno de ellos determinado, aprendizaje recurrente en tiempo real, este fue introducido por Williams y Zipsen en 1989 [WZ89], y el otro modo de entrenamiento se le denomina Retro-propagación A través del Tiempo, en inglés *Backpropagation Through Time* (BPTT), [Wer90] por P.Werbos en 1990.

La función de BPTT es minimizar el error en un periodo de tiempo, este será la diferencia entre el valor de salida $y(t)$ de la neurona menos el valor esperado $d(t)$.

$$E = \int_{t_0}^{t_i} (y(t) - d(t))dt \tag{2.36}$$

Tabla 2.1: Ejemplo de aplicaciones de Redes Neuronales Recurrentes

Tema	Autores	Referencia
Seguimiento predictivo de la cabeza para sistemas de realidad virtual [SCW99]	Saad, Caudell, and Wunsch, II	[Saad, 1999]
Estimación de la potencia de la turbina eólica [SBK07]	Li, Wunsch, O'Hair, and Giesselmann	[Li, 1999]
Predicción financiera utilizando redes neuronales recurrentes [GLT98]	Giles, Lawrence, Tsoi	[Giles, 1997]
Método de síntesis musical para instrumentos chinos de cuerda pulsada [LSL00]	Liang, Su, and Lin	[Liang, 1999]
Previsión de carga eléctrica [PPR99]	Costa, Pasero, Piglione, and Radasanu	[Costa, 1999]
Previsión de entradas de agua natural [SYCZ19]	Coulibaly, Anctil, and Rousselle	[Coulibaly, 1999]

El cálculo de la salida y de una neurona viene dado por:

$$\frac{\partial y_i}{\partial t} = -y_i + \sigma(x_i) + I_i \tag{2.37}$$

$$x_i = \sum_j w_{ji} y_j \tag{2.38}$$

Donde x_i es la entrada que viene dada por otra neurona, I_i es la entrada externa a la neurona i, w_{ji} es la representación de la conexión entre la neurona i a la neurona j, y σ es una función de activación, como por ejemplo sigmoide.

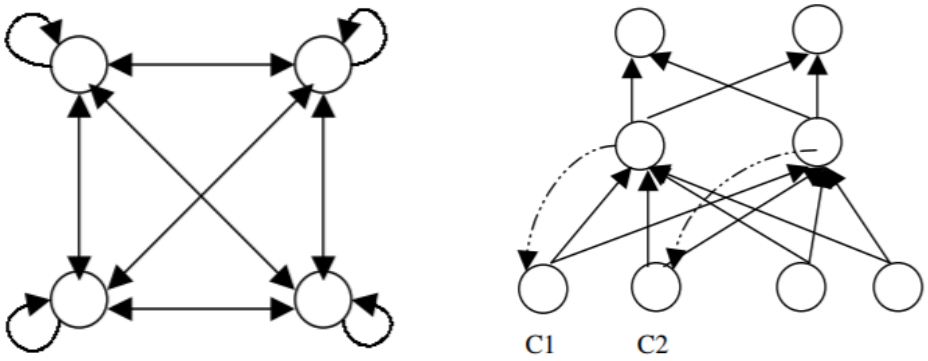
2.5.1. Arquitectura

En la arquitectura de una RNN completamente conectada se diferencia por no tener capa de entrada si no que cada nodo tiene como entrada otros nodos. Un nodo puede tener retroalimentación a sí mismo.

Mientras que la arquitectura de Redes Neuronales simples parcialmente recurrentes se suelen utilizar para aprender cadenas de caracteres. Los pesos de unidades de contexto se procesan como las unidades de entrada, por ejemplo a través de la retropropagación.

Para añadir retroalimentación [MJ01] se pueden utilizar dos formas:

- **Redes de Elman**: Elman introdujo la retroalimentación de la salida de una de las capas ocultas a la parte de contexto de la capa de entrada, lo que proporciona una especie de memoria sobre el estado anterior de dicha capa. En la Figura 2.11(a) se puede observar en una RNN simple, la conexión de la capa oculta a la parte de contexto de entrada. Las líneas discontinuas representan las conexiones entrenables

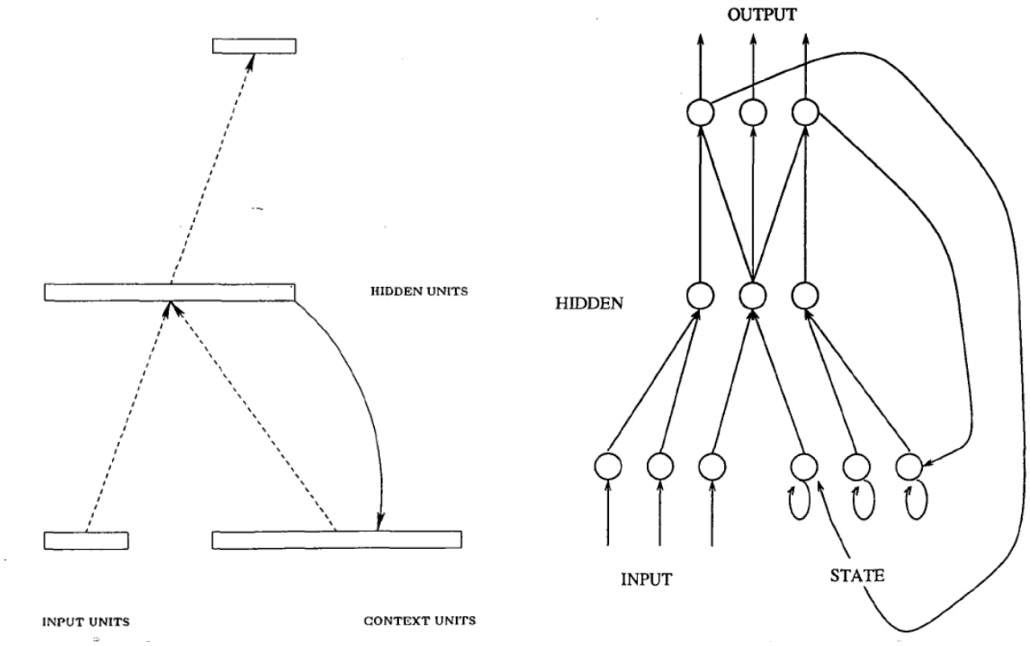


(a) RNN completamente conectada (b) RNN parcialmente conectada

Figura 2.10: Arquitectura de una Red Neuronal Recurrente [MJ01]

de la red.

- **Redes de Jordan**: Estas se basan en la retroalimentación de la capa de salida de la red a los nodos de la capa de entrada y dan más énfasis a la secuencia de valores de salida. Esto se puede observar en la Figura 2.11(b) donde se conecta la salida con las neuronas de estado del inicio. No están representadas todas las conexiones.



(a) Red de Elman (b) Red de Jordan

Figura 2.11: Tipos de retroalimentación [MJ01]

2.6. Aprendizaje por Transferencia

La principal motivación en los inicios del TL, fue discutida en el estudio de NIPS-95 *"Learning to learn"* [TP98], en el que se debatió sobre la necesidad de tener métodos capaces de retener y reutilizar el conocimiento aprendido previamente. Una técnica muy relacionada con TL es la llamada aprendizaje multitarea [Car97], esta técnica intenta aprender varias tareas incluso cuando son diferentes, un planteamiento común es descubrir como beneficiar cada tarea individual.

En TL hay que tener en cuenta lo qué queremos transferir, cómo lo queremos transferir y cuándo lo queremos transferir. *"Qué queremos transferir"* se refiere a la pregunta que se plantea la parte del dominio que puede llegar a ser transferida entre dominios o tareas. Algunos de los conocimientos comentados son típicos tanto para dominios como tareas individuales, mientras que otros conocimientos pueden ser más habituales entre otro tipo de dominios, por tanto, pueden aumentar el rendimiento del dominio o tarea destino. Una vez se ha descubierto qué conocimientos se pueden llegar a transferir, se desarrollan algoritmos para poder transferirlos, en este caso estamos tratando la pregunta de *"Cómo lo queremos transferir"*. Por último la cuestión *"Cuándo lo queremos transferir"* hace referencia a las situaciones en las que se deben transferir, o en caso contrario nos interese más no transferir ningún tipo de conocimiento. A veces, puede ser que la transferencia por fuerza bruta no tenga éxito, como es el caso de cuando no están relacionados el dominio origen con el dominio destino.

Podemos dividir el TL en varios subapartados, como Aprendizaje de Transferencia Inductiva, *Inductive Transfer Learning (ITL)*, Aprendizaje de Transferencia Transductiva, *Transductive Transfer Learning (TTL)*, y por último, Aprendizaje de Transferencia no Supervisado, *Unsupervised Transfer Learning (UTL)*. Se pueden observar las diferentes características de cada uno en la Tabla 2.2 y en la Figura 2.12

Tabla 2.2: Tabla de configuraciones de Transfer Learning

Opciones TL	**Áreas relacionadas**	**Etiquetas de dominio origen**	**Etiquetas de dominio destino**	**Tareas Tareas**
ITL	Aprendizaje multitarea	Disponible	Disponible	Regresión, Clasificación
	Aprendizaje Autodidacta	No disponible	Disponible	Regresión, Clasificación
TTL	Aprendizaje multitarea	Disponible	Disponible	Regresión, Clasificación
UTL		No disponible	No disponible	Agrupamiento, Reducción dimensionalidad

2.6.1. Aprendizaje por Transferencia Inductiva

Como ya hemos visto en la Tabla 2.2, esta configuración de TL tiene dos posibles casos:

1. Cuando los datos etiquetados están disponibles en el dominio de origen.
2. Cuando el dominio origen no dispone de datos etiquetados.

A la hora de la **transferencia en instancias** podemos encontrar que aunque los datos del dominio de origen no se pueden reutilizar directamente, hay ciertas partes de los datos que aún se pueden reutilizar junto con algunos datos etiquetados en el dominio de destino.

Dai et. al [DYXY07] proponen un nuevo algoritmo llamado TrAdaBoost, el cual es una extensión de AdaBoost, dirigido a resolver problemas de ITL. Este asume que el dominio de origen y el de destino usan las mismas características y etiquetas, pero la propia distribución de los datos se encuentra en dos dominios diferentes. TrAdaBoost tiene en cuenta que el dominio de origen y el de destino tienen varias diferencias entre sí, por lo que aunque algunos datos del dominio de origen puedan servir al dominio destino, también pueden ser perjudiciales para el mismo. Para esto itera ponderadamente sobre los datos del dominio de origen para reducir aquellos que puedan ser perjudiciales y fomentar más aquellos que son validos para el dominio destino. Jiang and Zhai [JZ07] proponen un nuevo método para intentar mitigar el problema de la adaptación de dominio en el procesamiento del lenguaje natural debido a la falta de datos etiquetados en dominios. Lia et. al [LXC05] propusieron que con ayuda de los datos etiquetados del dominio de origen, se etiquetaran aquellos datos sin etiquetar del dominio de destino.

Cuando hablamos de transferencia de conocimiento de representación de características, estamos tratando de hallar representaciones de características adecuadas para intentar minimizar la diferencia entre el dominio y el error de clasificación/regresión. Para encontrar las correspondientes hay que tener en cuenta si estamos tratando un problema de datos etiquetados o no. Si tenemos datos etiquetados se pueden usar métodos de aprendizaje supervisado, Argyriou y col. [AEP06] sugirieron un método de aprendizaje de características dispersas para el aprendizaje multitarea(intenta aprender tanto las tareas de origen como las de destino de manera simultánea), por otro lado, si tenemos datos no etiquetados se usa aprendizaje no supervisado, como el método de codificación dispersa [LBRN06] la cual descubre funciones básicas que capturan características de nivel superior en los datos de entrada sin etiquetar.Raina et al. [KB14] propusieron utilizar este método para aprender características de nivel superior para el aprendizaje por transferencia.

La mayoría de enfoques de la transferencia de conocimiento de los parámetros asumen que los modelos individuales para tareas relacionadas deben compartir algunos parámetros o distribuciones previas de hiperparámetros. La Transferencia de conocimiento relacional se encarga de los problemas de transferencia de aprendizaje en dominios relacionales, intenta transferir la relación entre los datos de un dominio de origen a uno de destino.

2.6.2. Aprendizaje por Transferencia Transductiva

El término de TTL fue propuesto por Arnold et al. [ANC07], donde se estudia uno de los casos más desafiantes para el aprendizaje de transferencia transductiva no supervisada, donde no hay datos etiquetados del dominio destino disponibles en el entrenamiento, y donde requerían que las tareas de origen y destino fueran las mismas, aunque los dominios sean diferentes.

En el apartado de transferencia en instancias vemos la importancia del muestreo por importancia. Por otro lado, en transferencia de conocimiento de representación de características Blitzer y col. [BMP06] propuso un algoritmo de aprendizaje por correspondencia estructural (SCL), para hacer uso de los datos no etiquetados del dominio objetivo para extraer algunas características relevantes que pueden reducir la diferencia entre los dominios. [BMP06] es útil cuando nos enfrentamos a nuevos dominios en los que los datos etiquetados son escasos o inexistentes, y por lo tanto, nos interese buscar adaptar los modelos existentes de un dominio origen rico en recursos a un dominio destino pobre en recursos.

2.6.3. Aprendizaje por Transferencia no Supervisada

En este caso no se encuentran datos etiquetados ni el dominio de origen ni en el de destino. Podemos encontrar algoritmos autodidactas de agrupamiento en clústeres [DYXY08], cuyo objetivo es agrupar una pequeña colección de datos de destino sin etiquetar con la ayuda de una gran cantidad de datos auxiliares sin etiquetar, consiguiendo aprender características útiles con la ayuda de los datos auxiliares en los que los datos de destino se pueden agrupar de manera efectiva, también se encuentra el análisis discriminativo transferido [SZ08] para transferir problemas de reducción de dimensionalidad de transferencia.

2.7. Aprendizaje federado

El aprendizaje Federado, (del inglés *Federated Learning*), fue propuesto por Google alrededor del 2016. El aprendizaje federado seguro nació por causa de varios problemas [YLCT19].

Uno de los mayores problemas de la inteligencia artificial viene causado por la falta de privacidad en los datos utilizados, estos últimos años ha crecido el grado de conciencia por parte de las empresas, las cuales buscan que los datos sean seguros y la privacidad del propio usuario. Las noticias sobre filtraciones provocan un cierto rechazo hacia este campo y sobretodo una gran preocupación por los usuarios, por ejemplo la violación de los datos en Facebook. En respuesta a estas noticias se ha reforzado las leyes de protección de seguridad y privacidad de los datos en la RGPD.

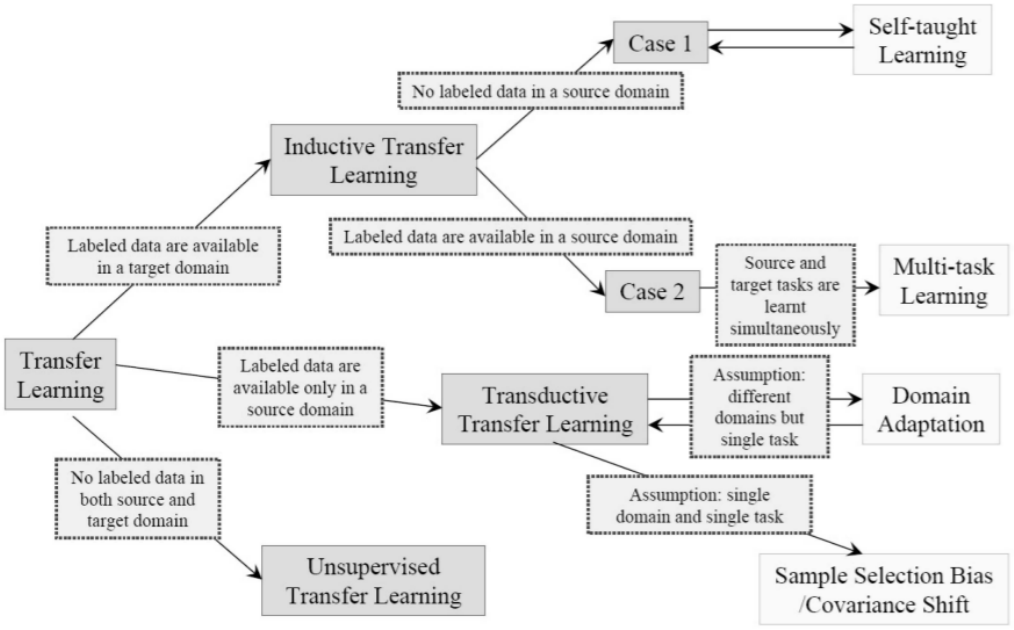


Figura 2.12: Configuraciones del aprendizaje por transferencia [PY09]

Una posible solución a estos problemas es el enfoque de aprendizaje federado, en inglés *Federated Learning*, el cual fue propuesto por Google alrededor del 2016. Su idea principal se basa en construir modelos de aprendizaje automático basados en conjuntos de datos que se distribuyen en múltiples dispositivos mientras se previene la fuga de datos.

La computación segura de múltiples partes, *Secure Multi-party Computation* (SMC), es una de las características del aprendizaje federado. Estos modelos se diferencian por tener varias partes de las que cada una, no sabe nada de las otras excepto su entrada y salida. Otra característica dedicada a agregar ruido a los datos o usar métodos de generalización para ocultar ciertos atributos confidenciales, se denomina privacidad diferencial, la cual imposibilita la restauración de los datos para proteger la privacidad del usuario. También nos podemos encontrar con el cifrado homomórfico [RAD78], que mediante el intercambio de parámetros bajo el mecanismo de cifrado durante el aprendizaje automático, protege la privacidad de los datos del usuario.

Capítulo 3

Estado del Arte

El objetivo de este capítulo consiste en la recopilación de diversos documentos que muestran distintas técnicas relacionadas con la detección de rostros para la estimación de la edad para su posterior revisión, destacando las metodologías más interesantes, así como los resultados obtenidos. De esta manera, se pretenden reunir las mejores conclusiones para, posteriormente, en el siguiente capítulo, tener un marco teórico suficientemente amplio como para poder llevar a cabo las experimentaciones necesarias.

A lo largo de este capítulo, se aporta información sobre tres principales bloques: en el primero se verán los Modelos de Apariencia Activa, con los cuales se estudian las diferentes formas de reconstrucción del rostro. Seguidamente, se encuentran los modelos de la estimación de la edad, donde se ven diferentes formas de estimación, ya sean mediante el cráneo, la superposición de imágenes, mediante imágenes faciales, etc, así como los diferentes problemas relacionados con la estimación, entre los que destaca la influencia del género. Finalmente, se verán los patrones de envejecimiento, a través de los cuales se realizan tanto diferentes funciones relacionadas con la vejez, como sus respectivos problemas.

En la Sección 3.1 se estudian trabajos relacionados con la Apariencia Activa. En la Sección 3.2 se tratan trabajos vinculados con la estimación de la edad facial. Por último, en la Sección 3.3 se tratan aquellas investigaciones relacionadas con los patrones de envejecimiento.

3.1. Modelos de Apariencia Activa

Los Modelos de Apariencia Activa, en inglés (*Active Appearance Model* (AAM)) sirven para la alteración de las expresiones. Los AAM generan puntos en la imagen y se ajustan a ellos, intentando combinar la forma de la cara con la textura (patrón de intensidades o colores dentro de una franja de la imagen).

Uno de los primeros trabajos de ajuste a la imagen fue [BM04], en el que se habla del

algoritmo de composición inversa. En [NMP96] se describe un modelo deformable en 3D del paisaje de intensidad, el cual, se encarga de variar la forma e intensidad del mismo. En este se utiliza un algoritmo de coincidencia de superficie de punto más cercano para realizar el ajuste, que tiende a ser sensible a la inicialización. Lanitis et al. [LTC97] and Edwards et al [ELTC98], usan un algoritmo de búsqueda de límites ("un Modelo de Forma Activo", en inglés *Active Shape Model* (ASM)) para conseguir obtener la mejor forma, posteriormente lo usan para encontrar un modelo de textura de imagen.

Poggio et al. [EP96], [JP04] para conseguir hacer coincidir modelos de forma y textura de forma iterativa utilizan un algoritmo de flujo óptico. En el primer artículo [EP96] hablan de métodos mejorados de modelado que combinan tanto la forma como la textura, por otra parte comentan que debido a diferentes fuentes de variabilidad como pueden ser la persona, la expresión, la pose y la propia iluminación de la imagen deciden aislar los cambios en la apariencia y tratan con métodos no lineales de variación de la forma. En el segundo [JP04] describen un modelo flexible(modelo transformable multidimensional) para representar imágenes de objetos de una determinada clase, conocidos en un principio, como caras, además introducen un nuevo algoritmo para hacer coincidir una imagen nueva y así poder realizar análisis de imágenes.

Vetter [Vet96] utiliza un método de optimización para hacer coincidir modelos fotorrealistas de rostros humanos con imágenes. Se describe una técnica con la cual ser pueden ver a la imágenes desde diferentes puntos de vista cuando sólo se tiene imágenes con un sólo punto de vista. Esta técnica se basa en el conocimiento previo de rostros basados en imágenes de ejemplo de otros rostros vistos en diferentes poses y en un único modelo genérico en 3D de una cabeza humana. Las imágenes que se tienen de ejemplo, son utilizadas para poder aprender una forma invariante de pose y una especificación de la textura de una nueva cara.

Los dos último enfoques comentados anteriormente, tanto de Poggio como de Vetter, son lentos debido a la alta dimensionalidad del problema y al costo de probar la calidad de ajuste del modelo contra la imagen.

Gleicher [Gle97] presenta un nuevo método, el cual se maneja mejor con con las regiones de imágenes pequeñas, este método trata de generalizar los métodos anteriores para manejar mejor las transformaciones no lineales. Hager y Belhumeur [HB98] describen un planteamiento similar al de Gleicher, pero incluyen núcleos robustos y modelos de variación de iluminación. Por otra parte Claroff e Isidro [SI98] amplían este enfoque para rastrear deformaciones de objetos.

El artículo que se describe más a fondo en esta sección es una ampliación de [ETC98].

En [CET01] construyen un modelo de textura separado para cada nivel de la pirámide.Para simplificar, en cada nivel usan los mismos puntos de referencia y la misma forma de modelo. En cambio para ser llegar a ser exactos estos puntos de referencia debería bajar de resolución por cada nivel. La apariencia del modelo en el nivel dado es calculado

dando el modelo global de forma y el modelo particular de textura para ese nivel en concreto.

Cuando llega una nueva imagen hay que tratar el dilema como si fuera un problema de optimización, minimizando la diferencia entre esta y una imagen sintetizada mediante AAM.

$$\delta I = I_i - I_m \tag{3.1}$$

Posteriormente se deberá estudiar la relación entre el error y los parámetros ajustables. El resultado de esto será un modelo lineal simple.

$$\delta c = A\delta I \tag{3.2}$$

Para conseguir **A**, [CET01] realizan una regresión lineal multivariante múltiple en una muestra de desplazamientos de modelos conocidos,δ, y la correspondiente diferencia de las imágenes δ**I**. Podemos generar estos desplazamientos aleatorios perturbando los verdaderos parámetros del modelo para las imágenes conocidas. Pueden ser tanto imágenes de entrenamiento originales o imágenes sintetizadas generadas con AAM (En este caso, conocemos los parámetros exactamente y las imágenes no se corrompen por el ruido), también podemos incluir pequeños desplazamientos en posición, escala y orientación. Por último deberemos registrar la perturbación y la diferencia de la imagen. Es importante considerar un área de la imagen al calcular el error. Donde g_i sería la forma normalizada y g_m sería la normalización de los niveles de grises.

$$\delta g = \delta g_i - \delta g_m \tag{3.3}$$

Por lo tanto, el algoritmo de entrenamiento es simplemente desplazar aleatoriamente el parámetro del modelo en cada imagen de entrenamiento, registrando δ**c** y δ**g**. Finalmente obtenemos la relación lineal actualizada:

$$\delta c = A\delta g \tag{3.4}$$

Idealmente se busca un modelo que tenga un amplio rango de error,δg. En los experimentos que realizaron la perturbación óptima fue cerca de 0,5 desviaciones estándar para cada parámetro. Para entrenar utilizaron 400 imágenes con 68 landmarks cada una. A partir de esto, generon un modelo de apariencia que requiere de 55 parámetros para explicar el 95 por ciento de la variación observada.

[CET01] En la evaluación se puede observar en la Figura 3.1 muestra el error de la intensidad media por cada uno de los 256 pixels que usa la imagen, con un nivel de grises, frente al número de iteracciones que se realizan para la búsqueda. Este modelo al principio fue desplazado un 5 % del ancho de la cara. La linea de puntos es el error de reconstrucción utilizanod los puntos de referencia comentados anteriormente, los cuales son marcados al

principio a mano. Al observar la imagen podemos darnos cuenta que obtienen un buen resulado en la evaluación del modelo.

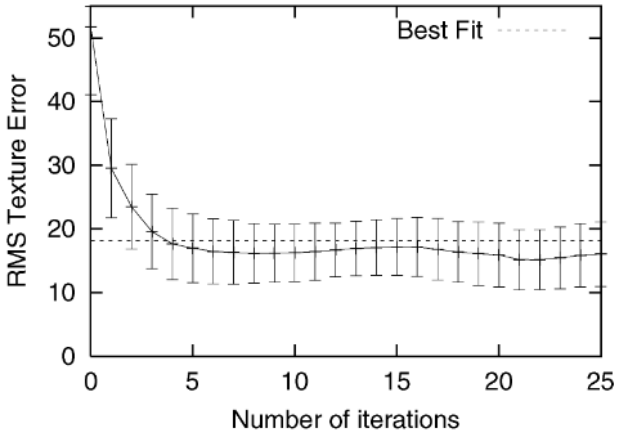


Figura 3.1: Error medio sobre la intensidad a medida que avanza la búsqueda

En la Figura 3.2 podemos observar como de las 100 búsquedas que realizaron, todas ellas convergieron correctamente, dadas varias imágenes que estaban desplazadas hasta 50 píxeles (25 % del ancho de la cara) tanto en el eje x como en el y. Los buenos resultados se pueden contemplar con un desplazamiento inicial de 20 píxeles, es decir un 10 % del ancho de la cara.

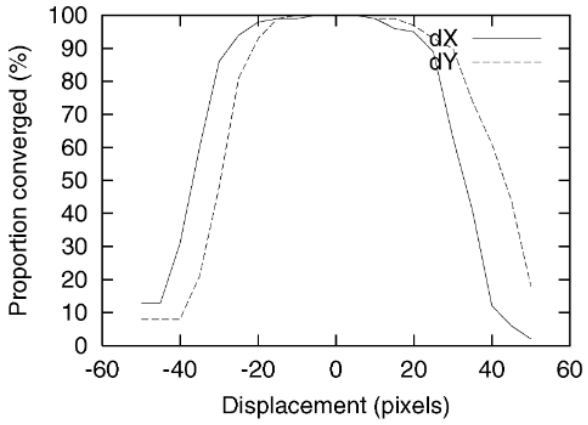


Figura 3.2: Proporción de convergencia de las imágenes con diferentes desplazamientos iniciales

3.1.1. Análisis de Procrustes

Este análisis está inspirado en una mitología griega, donde un posadero llamado Procrustes obligaba a sus huéspedes a acostarse en una de las dos camas que tenía. Si el huésped era bajo le estiraba hasta ajustarle a la longitud de esta, mientras que si el huésped era largo

le cortaba las piernas.

En el estudio [CET01] utilizan análisis de Procrustes para alinear los conjuntos de puntos (cada uno representado como un vector, x) y construir el modelo.Para hacer coincidir los landmarks con la media de los puntos calculados anteriormente se deforma la cara, obteniendo *"shape free patch"*.

El análisis de Procrustes se basa en escalamiento, rotación y traslación. Una vez aplicadas estas propiedades se realiza la media de todas las configuraciones. Esto se repite hasta que el cambio la suma de los cuadrados residuales entre dos pasos consecutivos sea menor que un valor establecido.

3.1.2. Modelo de regresión lineal múltiple

Este modelo es similar al modelo de regresión lineal solo que con el matiz que utiliza más variables explicativas. Se utiliza para poder estudiar la posible relación que existe entre las variables independientes, como puede ser el caso de las predictoras o las explicativas, y otra variable dependiente.

Modelo de regresión lineal:

$$y = b_0 + b_1 x + u \tag{3.5}$$

Modelo de regresión múltiple:

$$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_k x_k + u \tag{3.6}$$

Donde b_0 sería la variable dependiente **y** cuando el valor es 0 para todas las variables predictoras, y los demás coeficientes de **b** van a indicar el incremento de peso de la variable predictora **x** sobre la variable predictora **y**. Por último encontramos **u** que es el residuo o error, es decir, la diferencia entre el valor observado y el estimado por el modelo [Abu].

3.2. Estimación de la edad facial

Kwon y da Vitoria lobo [KdVL99] proponen métodos para clasificar las imágenes en bebés, jóvenes y adultos, utilizando enfoques basados en la antropometría facial y en las arrugas de la piel. En 2006 Ramanathan y Chellappa [RC06] proponen un modelo de crecimiento craneofacial, donde estudian el crecimiento del mismo en rostros humanos durante los años de formación, se puede observar la idea de lo comentado en la Figura 3.3.

En [RC06] no tienen en cuenta las diferencias en la iluminación, la postura de la cabeza, las expresiones faciales, etc. por lo que las puntuaciones de reconocimiento son bajas.

Burt and Perrett [BP95] estudiaron el envejecimiento de las imágenes a través de superponer imágenes que contienen distintas informaciones de formas y colores. El

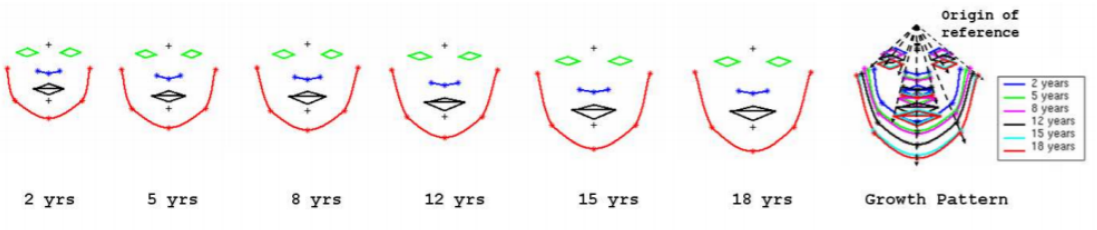


Figura 3.3: Rasgos faciales para las restricciones del modelo de crecimiento craneofacial [RC06]

resultado que obtuvieron se puede observar en la Tabla 3.1. La transformación de color produjo un efecto mayor que la transformación de forma, pero solo para las caras más jóvenes.

Tabla 3.1: Tabla de resultados del preprocesamiento

Edad inicial	Forma	Color	Forma y color
27	+5.9	+8.2	+12.1
40	+3.7	+4.8	+8.2
52	+3.6	+4.3	+5.3

Guo et al [GMF+09] demuestran que la estimación de la edad se ve afectada significativamente por el género, y se pueden lograr reducciones significativas de errores si la estimación de la edad se realiza en grupos que cada uno contiene un pequeño rango de edades con un solo género.

El enfoque F1 consiste en reconocer primero el género de una cara determinada y luego realizar la estimación de la edad correspondiente a ese género. El enfoque F2 consiste en cambio en clasificar el rostro en uno de los grupos de género y edad, y luego hace una estimación de la edad solo en el grupo determinado. El tercer enfoque se basa en clasificar el rostro en uno de los tres grupos, mujer joven, mujer adulta o mujer mayor, en este caso se determina que el rostro es de una mujer, en cambio si se clasifica dentro de uno de los grupos de varón joven, varón adulto o varón mayor, se determina que es varón. Este tercer enfoque puede mejorar la precisión del reconocimiento de género sobre la clasificación de género. Los resultados de cada enfoque se pueden observar en la Tabla 3.2.

En [EEH14], Eidinger et al., abordan uno de los problemas del reconocimiento facial, la falta de datos, para esto presentan un conjunto de datos de imágenes faciales, *Adience*, etiquetadas por edad y sexo, las cuales son obtenidas de Flickr. Por otra parte presentan dropout-SVM para evitar el sobreajuste. Por último explican una nueva técnica eficaz para la alineación facial.

Alrededor de 2018, [RTVG18], propone una nueva solución mediante aprendizaje profundo basada en una CNN VGG16 previamente entrenada en ImageNet. Los datos se entrenaron con IMDB-WIKI que tiene más de medio millón de imágenes de personas

Tabla 3.2: Tabla de resultados del preprocesamiento

Método	MAE	Tasa de reducción de errores
RPK	4.66	0
F1	2.95	36.7 %
F2	2.84	39.1 %
F3	2.87	38.4 %
F2.F	2.66	42.9 %

famosas. Con todo esto consiguen un MAE de 3.252 años. A continuación en la Tabla 3.3 podemos ver las diferentes comparaciones de los MAE en los distintos dataset de MORPH2 y FG-NET.

Tabla 3.3: Comparación de MAE de estimación de la edad

Método	MORPH2	FG-NET
RPK	6.30	4.70
DIF	3.80	4.80
AGES	8.83	6.77
MTWGP	6.28	4.83
CA-SVR	5.88	4.67
SVR	5.77	5.66
OHRank	5.69	4.85
DLA	4.77	4.26
[HFP14]	4.25	N/A
[GM11]	4.18	N/A
[GM14]	3.92	N/A
[LRBS09]	N/A	4.37
[LSS+11]	N/A	4.12
[YLL15]	3.63	N/A
[RTVG16]	3.45	5.01
DEX	3.25	4.63
DEX (IMDB-WIKI)	**2.68**	**3.09**

DeepUAge [ALKS20] utiliza la biblioteca dlib para el preprocesamiento de imágenes, al principio se pensó en utilizar la librería Face++ con la cual se consiguen 1000 puntos de referencia de los cuales 273 para el contorno de la cara, a diferencia de dlib que obtiene 68 puntos de referencia, de los cuales 34 corresponden al contorno de la cara. Con la librería de Face++ se obtuvo un problema con los servicios de la nube ya que no se puede compartir información con terceros. Por esta razón eligieron utilizar dlib. Para el problema de la estimación de la edad se dieron cuenta que la parte superior del rostro no estaba representada con esos 34 puntos de referencia, por lo tanto basandose en el método de Loomis ampliaron estos puntos de referencia para abarcar esta zona.

Para esto siguieron los siguientes pasos:

1. Usar los puntos de referencia marcado por la biblioteca dlib para obtener la coordenada x, y del punto de la nariz (punto 34).

2. Calcular la diferencia entre el punto 34 y la perpendicular de este a los costados. Siendo el cuadrado el doble de la distancia media obtenia.

$$d_1 = \sqrt{(p_{left_x} - p_{34_x})^2 + (p_{left_y} - p_{34_y})^2} \tag{3.7}$$

$$d_2 = \sqrt{(p_{right_x} - p_{34_x})^2 + (p_{right_y} - p_{34_y})^2} \tag{3.8}$$

$$square_{side} = \frac{d_1 + d_2}{2} * 2 \tag{3.9}$$

$$square_{side} = d_1 + d_2 \tag{3.10}$$

3. Localizar los dos vértices v_1, v_2 de la raiz cuadrada y usar esos valores para dibujar la forma dentro de la figura.

4. Desde los puntos medios c_x, c_y, dibujar un Isodecágono (N = 20).

En la Figura 3.4 podemos observar visualmente los pasos descritos anteriormente.

Figura 3.4: Reconstrucción del contorno mediante dlib

3.2.1. Alineación del rostro

Para alinear el rostro se utilizan los landmarks correspondientes al ojo izquierdo *(42, 48)* como al ojo derecho *(36, 42)*, por lo tanto se cogerán de la posición 42 a la 48 las coordenadas calculadas anteriormente mediante el predictor, así mismo se realizará con las coordenadas del ojo derecho. Posteriormente se cogerá la media de las coordenadas correspondientes del ojo izquierdo y del ojo derecho, haciendo la media de las x por una parte y de las y por otra.

Para conseguir el ángulo de rotación, se restarán ambos valores de los dos ojos por sus respectivas coordenadas, y se hará la arcotangente de estas coordenadas calculadas, posteriormente se pasará a radianes mediante la función de *degrees* de la librería de numpy.

Para la matriz de rotación utilizaremos la función *getRotationMatrix2D* de cv2 pasándole como parámetros, las coordenadas del centro de los dos ojos, el ángulo calculado previamente y el factor de escala, en este caso 1.

Por último la función de *warpAffine* de cv2, nos permitirá transformar la imagen de entrada mediante la matriz de rotación. Este resultado nos dará la imagen alineada de la Figura 3.5.

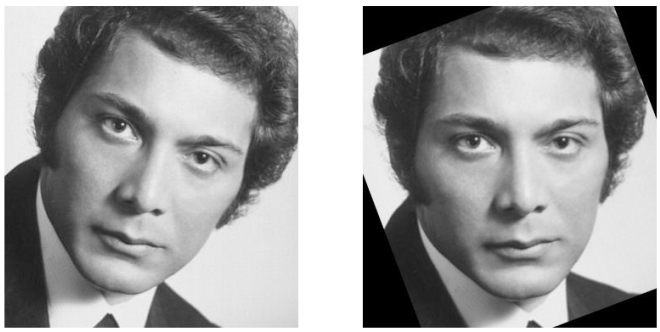


Figura 3.5: Ejemplo de alineación del rostro [MPS+17]

3.2.2. Recorte de la cara

Para el recorte de la cara se utilizará la función *convexHull* de la librería cv2, con la cual se conseguirá un remap del rostro, esta función se define en matemáticas como envolvente convexa, la cuál dado un conjunto de puntos X realiza la intersección de todos los conjuntos convexos que contienen a X.

$$C(X) = \left\{ \sum_{i=1}^{k} \alpha_i x_i | x_i \in X, \alpha_i \in \mathbb{R}, \alpha_i \geq 0, \sum_{i=1}^{k} \alpha_i = 1 \right\} \tag{3.11}$$

Posteriormente se realizará un polígono convexo con la función de *fillConvexPoly* de la librería cv2 pasándole el resultado de *convexHull* y los primeros 27 puntos de este. Todos los resultados que nos den 0 se ignorarán, se coge el argumento mínimo de las coordenadas x que representará el punto de la zona izquierda por donde se tiene que recortar, mientras que el máximo argumento de las coordenadas x representará el punto de la zona derecha. El mínimo argumento de las coordenadas y representará la parte de arriba y el máximo la de abajo. El resultado de estas configuraciones se puede observar en la Figura 3.6.

A continuación, en la Tabla 3.4 se muestran los resultados obtenidos por [ALKS20] en comparación con otras técnicas de preprocesamiento.

Figura 3.6: Ejemplo de imagen recortada [MPS+17]

Tabla 3.4: Tabla de resultados del preprocesamiento

Propuesta	MAE
DCA	2.73
Face++ contour	2.79
Dlib contour aligned	4.01
Dlib contour non-aligned	4.28
Dlib cropped	5.31
Non-processed	5.71

3.3. Patrón de envejecimiento

El primer algorítmo de estimación de la edad fue propuesto en por Latnis et al. en [LTC02], en este trata el patrón de envejecimiento mediante la fórmula de $age = f(b)$ donde **b** es vector que representa los parámetros del modelo facial y f se define como la función cuadrática. Para el proceso de entrenamiento se ajusta esta función cuadrática para cada entrada como su función de envejecimiento. Para esto propusieron 4 formas diferentes, de las cuales sobresalió con mejores resultados el método del Apariencia Específica Ponderado, en inglés *Weighted Appearance Specific* (WAS).

- **Función de Envejecimiento Global**: En este enfoque se asume que todas las personas envejecen por igual y que por tanto se puede establecer una función de envejecimiento global. No obstante no todos los individuos envejecen por igual por lo que una sola función no valdría para las posibles variaciones de envejecimiento.
- **Apariencia Específica**: Para este enfoque se tiene en cuenta que aquellas personas con parecidos físicos envejecerán por igual. Realizan una investigación para ver si estas dos variables tienen correlación entre sí, para lo que definen funciones cuadráticas de envejecimiento para personas que proporcionaron imágenes de progresión de la edad. Se calcula la distancia de Mahalanobis entre las representaciones de los modelos de dos imágenes de diferentes individuos, y la distancia Euclidiana entre los pesos de las funciones de envejecimiento entre los dos sujetos. La distancia de Mahalanobis se utilizó para ver la similitud entre caras,

mientras que la segunda distancia, la distancia Euclidiana, se utilizó para ver la similitud entre los diferentes patrones de envejecimiento. Los resultados de estos experimentos, fueron de un 0,55 de correlación entre las dos cantidades.

- **Función de Envejecimiento Ponderado**: el anterior enfoque funciona bien cuando se tiene un gran número de imágenes para el entrenamiento, en caso contrario no conseguirá dar buenos resultados. Con este nuevo enfoque se intenta solventar este problema calculando una nueva función de envecimiento ponderada que se puede observar en la Fórmula 3.12, donde p_i es la probabilidad de que el rostro actual esté en la Base de Datos y f_i es la función de envejecimiento del individuo.

$$f_{nueva} = \sum p_i f_i \tag{3.12}$$

- **Función de Envejecimiento Ponderado de una Persona Específica**: Los enfoques anteriores se basan en la apariencia de la cara, mientras que este enfoque pretende incorporar el estilo de vida del sujeto ya que es razonable que influya en el envejecimiento de la persona. A fin de ver si esto era cierto, realizaron un experimento para poder comprobar que la apariencia facial y el estilo de vida se correlacionan ambos con el patrón de envejecimiento del individuo. Para este experimento, cogieron las imágenes de la Base de Datos y les añadieron a cada una un estilo de vida diferente, donde entran vario aspectos como: *"el género, la salud, el nivel de vida, la situación económica, el nivel de estrés, las condiciones laborales, etc"*. Los datos fueron recogidos de encuestas que realizaron con anterioridad a varias personas.

  El estilo de vida se codificó en un vector de características que más adelante fue combinado con las apariencias del rostro para tener un único vector.

Estos nuevos enfoques fueron probados con 5.500 casos, utilizaron la distancia de Mahalanobis para calcular la diferencia entre la apariencia facial predicha y la real. Los resultados obtenidos de este experimento se pueden observar en la Tabla 3.5, donde se especifica el resultado de la distancia de Mahalanobis en los diferentes enfoques presentados, el que mejor resultado dió fue el enfoque de Persona Específica Ponderada, en inglés *Weighted Person Specific* (WPS), con un 5.91.

Tabla 3.5: Resultados del preprocesamiento

Método	Después de la simulación	Antes de la simulación
Función de Envejecimiento Global	6.18	6.12
Apariencia Específica	6.39	6.12
Función de Envejecimiento Ponderado	6.00	6.12
Función de Envejecimiento Ponderado de una Persona Específica	5.91	6.12

Más tarde estos mismos, Latnis et al. [LDC04] compararon la función de envejecimiento cuadrático f con otros métodos habituales de clasificación en el ámbito de la estimación

de la edad. Se compara, entre otros, con mapas auto-organizados, en inglés *Self Organizing Map (SOM)*, un tipo de ANN que están entrenados mediante aprendizaje no supervisado. En la Tabla 3.6 se puede observar que el método más eficaz es el de apariencia y edad específica, el cual es una combinación de los dos anteriores.

Tabla 3.6: Resultados obtenidos en [LDC04]

Método	Cuadrático	Distancia más corta	MLP	SOM
Global	5.04	5.65	4.78	4.9
Edad específica	4.87	5.02	4.52	N/A
Apariencia específica	4.61	5.58	4.64	N/A
Apariencia y edad específica	3.82	4.92	4.38	N/A

Sin embargo, siguen existiendo varios problemas sin resolver, como por ejemplo que la fórmula de función de envejecimiento se calcula empíricamente, no hay ninguna evidencia que relacione el rostro con la edad mediante una función cuadrática. Además, nos encontramos que esta función de envejecimiento, no maneja bien la característica temporal, por ejemplo, la relación que tienen los rostros envejecidos es monodireccional, es decir, el estado de un rostro en concreto solo afecta a rostros envejecidos. Sin embargo, la relación que tiene la función de envejecimiento es bidireccional, ya que, cualquier cambio en un rostro concreto cambiará la función de envejecimiento, y por tanto, afectará a los demás rostros.

Otro de los problemas es que el patrón de envejecimiento usa únicamente rostros de una misma persona. Y por último, la función de envejecimiento para un rostro desconocido, es una combinación lineal de funciones de envejecimiento conocidas, en vez de generarse a partir de un determinado modelo de patrones de envejecimiento. Todos estos problemas se consiguen solventar gracias a la aportación del algoritmo conocido como AGES, explicado más adelante.

En [ZGZC05] se presenta un algoritmo de regresión basado en imágenes. Este regresor propuesto aprende mediante el método de refuerzo. Utilizan salidas múltiples para evitar el sobreajuste y admitir una solución analítica. Los enfoques más populares basados en datos incluyen tanto *Non-Parameteric kernel Regression* (NPR), *Kernel Ridge Regression* (KRR) y *Support Vector Regression* (SVR).

- **NPR**: Este método es parecido a kNN, en cambio este toma la siguiente Fórmula 3.13.

$$g(x) = \frac{\sum_{n=1}^{N} h_\sigma(x; x_n) y(x_n)}{\sum_{n=1}^{N} h_\sigma(x; x_n)} \tag{3.13}$$

donde, $h_\sigma(x; x_n)$ es la función del kernel, la función más usada de núcleo es el núcleo RBF.

$$h_\sigma(x; x_n) = rbf_\sigma(x; x_n) = exp(-\frac{\|x - x_n\|^2}{2\sigma^2}) \tag{3.14}$$

NPR suele sobreajustarse a los datos, es decir, produce un sesgo bajo y una varianza alta.

- KRR: Este asume que la función de regresión de salida múltiple toma una forma lineal:

$$g(x) = \sum_{n=1}^{N} \alpha_n k(x : x_n) \tag{3.15}$$

donde, $k(x : x_n)$ es una función de núcleo (RBF, núcleo polinomial...) y α_n un vector de $q \quad x \quad 1$ que pondera a la función de núcleo ya mencionada. Cuando se usa un kernel lineal, KRR tiende a no ajustarse a los datos, produciendo una varianza baja y un sesgo alto. A menudo el rendimiento suele mejor si se utiliza una función de kernel no lineal.

- SVR: Este método se trata de una regresión robusta con una formulación que funciona para datos de salida única, es decir, $q = 1$. Este minimiza la siguiente función de coste:

$$\frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}|y(x_n) - g(x_n)|_\epsilon \tag{3.16}$$

SVR logra conseguir un equilibrio entre sesgo y varianza a diferencia de los dos anteriores, sin embargo, es difícil de aplicar directamente al problema de regresión con múltiples salidas.

Al testear el algorítmo propuesto en [ZGZC05] con los mencionados anteriormente, se consiguen los resultados de la Tabla 3.7. Como se puede observar consiguen una mejora muy diferenciada respecto al tiempo de ejecución.

Este algorítmo es entrenado con las imágenes de la base de datos FG-NET, la cual está compuesta por 1002 imagenes faciales, de las cuales 800 se han usado para entrenar y el resto para las pruebas. Las imágenes de entrada x están todas a una dimensión de 60 x 60, la salida y es la normalización de la edad. Convierten la salida de la edad a $y = \log(y + 1)$ para evitar salidas negativas en el regresor.

Tabla 3.7: Tabla de comparación de regresores para la estimación de la edad

	NPR	**KRR**	**SVR**	**IBR**
Error medio	8.44	13.56	6.60	5.81
25 % de error	2.54	3.99	1.38	1.26
Mediana de error	5.50	10.80	4.39	3.15
75 % de error	10.87	17.99	9.04	7.79
Tiempo de ejecución	3.6s	3.6	3.3s	0.016s

En el documento [GhZM+07] realizado por Xin Geng, Zhi-Hua Zhou, y Kate Smith-Miles realizan una estimación automática de la edad mediante patrones de envejecimiento. Según estos patrones cada imagen ***(I)*** debería tener una etiqueta extra,

*id**(I)***, a parte de la etiqueta de la edad, *age**(I)***. Para el problema de estimar la edad utilizan Análisis Discriminante Lineal ó *Linear Discrimiant Analysis* (LDA), por lo que tienen que lidiar con el problema de etiquetas múltiples. La definición que dan al patrón de envejecimiento es: *"Un patrón de envejecimiento es una secuencia de imágenes de caras ordenadas por tiempo"*.

Todas las imágenes que contiene un patrón de envejecimiento deben de ser de una misma persona. Para este patrón, se debe crear un vector del tamaño de la edad máxima hasta la que se desea estimar, supongamos que queremos estimar la edad de las personas hasta 18 años, inicializaremos un vector vacio de 18 espacios. Cuando nos lleguen las imagenes de una persona, que supongamos que tenemos las imágenes de 3,6,7,9,14 años de esta, los tendremos que introducir en el vector teniendo como índices la edad. Por lo tanto, las posiciones 3,6,7,9 y 14 del vector contendrán vectores de las imágenes según su edad, mientras que las demás posiciones se mantendrán vacias. Si todas las posiciones de vector están llenas se considera un patrón de envejecimiento completo de lo contrario se denominará patrón de envejecimiento incompleto.

Al representar los patrones de envejecimiento de esta manera se consigue acoplar las dos etiquetas ya mencionadas, *age**(I)*** y *id**(I)***. Gracias a esto se podría llevar a cabo un proceso de aprendizaje no supervisado, pero esto conllevaría con dos nuevos problemas, el primero que el algorítmo aplicado para el entrenamiento tendría que poder tratar con muestras incompletas, y segundo, durante la estimación de la edad de los datos de test se debe escoger un patrón de envejecimiento correcto y conseguir una posición adecuada en este para una imagen desconocida. Para abordar estos problemas, [GhZM+07] presentaron estas dos soluciones:

1. **Subespacio de patrón de envejecimiento**: En este caso se aplica *Principal Component Analysis* (PCA) para construir un subespacio que captura la variación de un conjunto de datos. El algoritmo de aprendizaje está constituido por un proceso de interacción entre el modelo de patrón de envejecimiento global y los patrones de envejecimiento personalizados.Aunque las personas envejecen de diferentes maneras, la similitud (modelada por el subespacio) de todos los patrones de envejecimiento también es crucial para la edad estimación, principalmente cuando los patrones de envejecimiento son muy incompletos. En cada iteración, la parte faltante del patrón de envejecimiento personal se estima mediante el modelo de patrón de envejecimiento global actual. Por lo tanto, el modelo global se refina aún más con el envejecimiento personal actualizado patrones. De esta manera, la similitud y la personalidad del los patrones de envejecimiento se utilizan alternativamente para aprender el subespacio final.

2. **Estimación de la edad**: La estimación de la edad es un modelo global de patrones de envejecimiento. En este caso cuando te llega una nueva imagen lo primero que

se hace es detectar el vector de características ya comentado anteriormente. Al no saber el posicionamiento de la imagen en el patrón de envejecimiento no se puede calcular el error de reconstrución. Por lo tanto se debe colocar en todas las posibles soluciones del patrón de la edad, obteniendo p vectores de patrón de envejecimiento.

$$\varepsilon^a(j) = (b - \mu_{(j)} - W_{(j)y_j})^T (b - \mu_{(j)} - W_{(j)y_j}) \tag{3.17}$$

Donde $\mu_{(j)}$ es la parte de μ y $W_{(j)}$ es la parte de W que corresponde a la posición j. Finalmente la proyección y_r puede ser reconstruida como el mínimo error de reconstrución en todas las posiciones posibles de p.

$$r = argmin(\varepsilon^a(j)) \tag{3.18}$$

La evaluación de [GhZM+07] se puede observar en la Tabla 3.8, donde se ve claramente que han obtenido una mejora respecto otros modelos, y que con el dataset de FG-NET obtenido mejores resultados que con el de Morph.

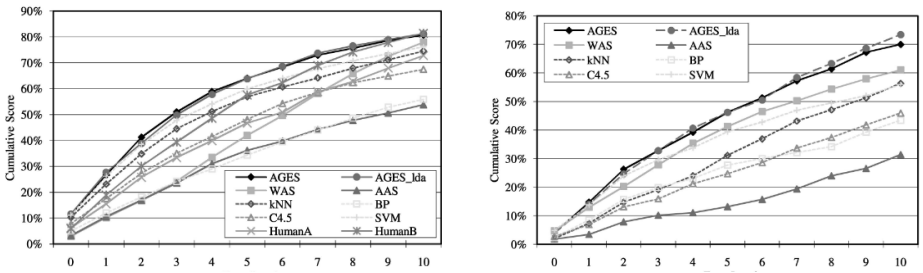
Tabla 3.8: Tabla de resultados de AGEs

Método	**Dataset**	
	FG-NET (LOPO)	**Morph (Test Set)**
AGES	6.77	8.83
AGESIda	6.22	8.07
WAS	8.06	9.32
AAS	14.83	20.93
KNN	8.24	11.30
BP	11.85	13.84
C4.5	9.34	12.69
SVM	7.25	9.23
HumanA	8.13	-
HumanB	6.23	-

En las Figuras 3.7(a) y 3.7(b) se contempla en el eje de la x la puntuación acumulada respecto al eje de la y que representa las edades de 0 a 10 años, en los niveles de error más altos no se muestra nada ya que no suele ser aceptable la estimación de la edad con un error tan alto. [GhZM+07] divide los resultados en los obtenidos mediante FG-NET Figura 3.7(a) y los obtenidos mediante el dataset de Morph Figura 3.7(b).

En la Figura 3.7(a) los mejores algorítmos respecto los niveles de error serían Patrones de Envejecimiento en Subespacio, en inglés *Aging Patterns Subspace* (AGES), AGESIda, SVM y kNN el peor HumanA a pesar de que el *Mean Absolute Error* (MAE) de WAS es menos que este sus puntuaciones acumulativas son mayores.

En la Figura 3.7(b) se puede observar una situación parecida a la anterior figura con la excepción de que WAS se desempeña notablemente mejor, por otra parte sus niveles acumulativos son peores que las de AGES y AGESIda.



(a) FGNET (b) Morph

Figura 3.7: Puntuaciones acumulativas de la estimación de la edad estándar

- **Análisis discriminante lineal**: LDA es una manera de tratar la clasificación supervisada en el que se modela la distribución de predictores X para cada clase k por separado $P(Y = k|X = x)$. Para poder evaluar esta probabilidad se utiliza el Teorema de Bayes lo que nos va a permitir es que en vez de calcular la probabilidad de que dado X nos de k, vamos a calcular la probabilidad de que dada la clase k obtenga X. Las distribuciones para la probabilidad $P(Y = k|X = x)$ vienen dadas por una distribución normal:

$$f_k(X) = P(Y = k|X = x) \sim N(\mu_i, \sigma) \tag{3.19}$$

  Este método se suele utilizar cuando las clases tienen una clara separación (en este caso la regresión logística no es estable), también cuando tenemos poco datos y los predictores estén distribuidos en cada una de las clases, por último este método es popular cuando se quiera hacer una clasificación cuando haya más de una clase.

- **Teorema de Bayes**: Es utilizado para calcular las diferentes probabilidades de los acontecimientos dados. Se puede representar con la siguiente Ecuación 3.3.

$$P(Y = k|X = x) = \frac{P(X = x|Y = k)P(Y = k)}{P(X = x)} \tag{3.20}$$

$$= \frac{f_k(x)\pi_k}{\sum_{i=1}^{K} \pi_i f_i(x)} \tag{3.21}$$

  donde $P(X = x|Y = k)$ está normalmente distribuido. La función $f_k(x)$ nos indica la densidad de X en cada clase k y π_k nos indica la probabilidad de la clase. Para $P(X = x)$ se usa el teorema de la probabilidad total que es $\sum_{i=1}^{K} \pi_i f_i(x)$ en el que la K son las clases que tenemos (tamaño de espacio muestral).

- **Análisis de componentes principales**: Es un tipo de aprendizaje no supervisado, para conseguir una reducción de dimensionalidad (Jolliffe 1986), para esto hay que intentar encontrar la mejor proyección para no perder mucha información. Toma

un conjunto amplio de variables correlacionadas y lo transforma en un conjunto nuevo de nuevas variables independientes (componentes principales) que expliquen la variación existente, estas últimas son combinaciones lineales de las anteriores y se construyen según el orden de importancia que tengan. Lo primero que hay que hacer es centrar los datos, es decir, mandar el centro de masa al origen de coordenadas, posteriormente calcular la matriz de covarianza para ver como de dispersos están los datos, de esta matriz sacaremos los vectores propios que nos dan la dirección y los valores propios que nos dan la magnitud de la transformación lineal, posteriormente realizaremos una transformación lineal para .estirar el plano". Escoger el vector propio correspondiente al mayor valor propio, ya que este capturará mayor información. Conseguimos el negativo del vector y mandamos los puntos a este plano. Lo que habremos conseguido es haber proyectado los datos en la dimensión donde mejor se expande.

Capítulo 4

Algoritmo de Detección de la Edad

En este apartado se describen cada una de las partes que han influido a la hora de la composición de este trabajo, explicando tantas las tecnologías utilizadas, es decir, las librerías y las herramientas usadas, como las Bases de Datos utilizadas, el preprocesamiento de las imágenes y la creación de los modelos teniendo en cuenta las métricas existentes.

El algorítmo está conformado por lo siguientes procesos, (i) Preprocesamiento, (ii) detección del rostro, (iii) recorte del rostro y (iv) estimación de la edad.

En la Sección 4.1 se explicará una breve descripción del algoritmo, mostrando por encima el funcionamiento de este. En la Sección 4.2 se ahondará en los conceptos del preprocesamiento, explicando más a fondo la detección del rostro como la corrección de luz. En la Sección 4.3 se explicarán las técnicas utilizadas para la clasificación de las imágenes, explicando tanto el modelo usado como los parámetros utilizados para su creación. En la Sección 4.4 se explicará como se han procesado los videos para su posterior clasificación. En la Sección 4.5 se hará un breve resumen de la selección de las métricas de rendimiento viendo la utilidad de cada una. En la Sección 4.6 se mostrarán las tecnologías utilizadas en este trabajo, desde las librerías más utilizadas, hasta las herramientas y plataformas que han servido de ayuda en la realización del proyecto. Por último la Sección 4.7 muestra que Base de Datos se ha utilizado, comentando las características de la misma.

4.1. Descripción técnica del algoritmo

En esta Sección se mostrará una visión global del algoritmo presentado, desde el preprocesamiento hasta el entrenamiento.

La imagen de entrada se pasa por un filtro de iluminación, la cual será alterada siempre y cuando sea necesario, subiendo la exposición si está muy oscura y bajando la exposición en caso contrario. Una vez pasada por el filtro, se realizará una detección del rostro obteniendo los *landmarks* correspondientes. Posteriormente con ayuda del algoritmo presentado por [ALKS20] se añade la parte de la frente a los *landmarks* conseguidos con

anterioridad. Por último, recortaremos el rostro por los *landmarks* obtenidos.

En caso de que no se detecte el rostro, se pasará por otro detector de CNN, y si este si consigue detectar el rostro se procederá al recorte del rostro directamente.

Tras realizar el preprocesamiento completo de las imágenes se introducirá en el modelo ResNet50 creado mediante Keras. En este modelo se entrenará con imágenes del dataset de [ALKS20]. Se compila mediante el optimizador de SGD, una pérdida de entropía cruzada categórica y una métrica de MAE. A la hora de evaluar se introducirán los datos guardados anteriormente para la parte del test. Por último se verá la diferencia entre los datos predichos y los reales para ver la eficiencia del modelo.

En caso de que la entrada sea un video, se separará en frames y se preprocesarán igual que las imágenes.

4.2. Técnicas de Preprocesamiento

En esta Sección se explican las técnicas de preprocesamiento aportadas para este trabajo. En la Figura 4.1 se puede observar mediante un diagrama de flujo una visión general del preprocesamiento presentado.

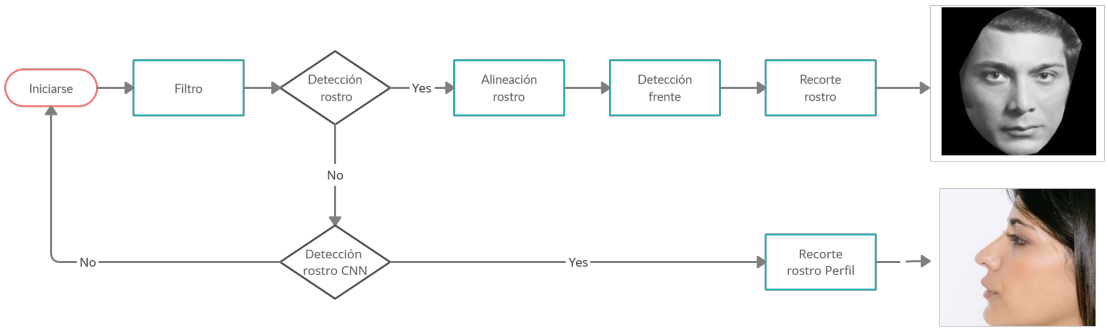


Figura 4.1: Diagrama de flujo preprocesamiento

4.2.1. Reconocimiento facial

Para la detección del rostro, se utiliza la librería *dlib*, esta contiene algoritmos de ML, con la función *get_frontal_face_detector* a la que una vez conseguido el detector se pasa por parámetro la imagen en escala de grises, previamente editada mediante la función de *cvtColor*, a la que se le pasan como parámetros, la imagen a cambiar y la conversión del color que se quiere conseguir, en este caso *COLOR_BGR2GRAY*. El Código 4.1 muestra la función de detección facial realizada.

Si no detecta el rostro termina la ejecución, mientras que si lo detecta utiliza el predictor con el modelo pre-entrenado shape_predictor_68_face_landmarks, este permite crear el objeto de los landmarks, posteriormente con la librería de *face_utils* y la función

Listing 4.1: Detección de rostro

```
def face_detection(self):
    detector = dlib.get_frontal_face_detector()

    image = cv2.imread(self.file)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    return detector(gray, 1), detector
```

de *shape_to_np* consiguen las coordenadas (x, y), iterando el objeto creado anteriormente y convirtiendo en tuplas de dos para las coordenadas (x, y).

Listing 4.2: Creación de *landmarks*

```
predictor = dlib.shape_predictor(SHAPE_PREDICTOR)

for (i, rect) in enumerate(rects):
    shape = predictor(gray, rect)  # Create landmark object
    shape = face_utils.shape_to_np(shape)
```

El primer problema que se identifica es que no es capaz de detectar rostros de perfil. Para probar esto, introducimos una imagen de perfil al detector actual. Como se esperaba, el resultado fue que no detectaba ningún rostro. Por lo que se empieza a investigar como poder detectar rostros de perfil. Tras varias pruebas mmod_human_face_detector.dat resulta vencedor, este modelo entrenado se basa en una red convolucional CNN. Al procesar la foto de perfil en el nuevo modelo, conseguimos que detecte el rostro de la persona, tal y como se ve en la Figura 4.2(a).

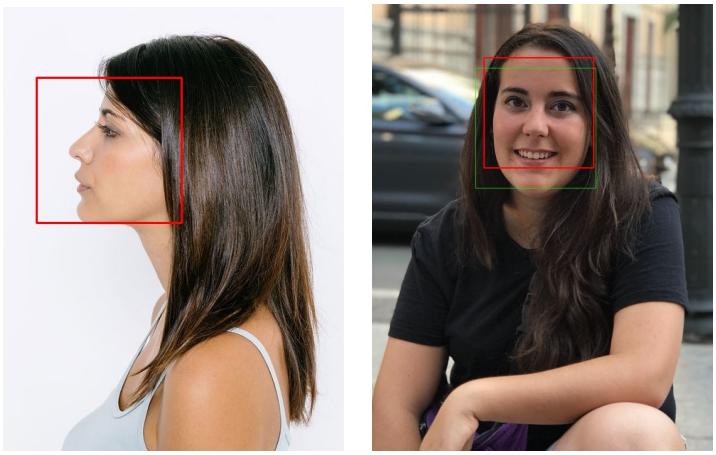
Listing 4.3: Detección del rostro con CNN

```
def face_detection_with_cnn(self):
    cnn_face_detector = dlib.cnn_face_detection_model_v1(NMOD_HUMAN)

    image = cv2.imread(self.file)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    return cnn_face_detector(gray, 1)
```

Para ver claramente las diferencias se muestra un ejemplo de dos resultados en la Figura, en las que si el modelo pre-entrenado shape_predictor_68_face_landmarks.dat detecta el rostro lo remarca con un cuadrado verde, mientras que si lo hace el modelo pre-entrenado de la CNN se remarcará con uno rojo.



(a) Detectada con CNN (b) Imagen aclarada

Figura 4.2: Detectada por ambos

4.2.2. Corrección de luz

Posteriormente se decide investigar sobre la identificación de rostros con poca limunación. Para esto se baja la iluminación a la Figura 4.2(a), la cual al meterla en el modelo dca de [ALKS20] no se consigue detectar el rostro. La imagen con poca iluminación se puede visualizar en la Figura 4.4(a), para poder conseguir detectar el rostro se pasa por un filtro para conseguir aumentar la iluminación.

El filtro utilizado para corregir la iluminación se basa en una función con parámetro la propia **imagen** a modificar el brillo, en función de esta se eligirá el **gamma** que será el encargada de aumentar/decrementar el brillo de dicha imagen. Dentro de esta función recorremos los 256 píxeles y mediante la ayuda de la librería de numpy utilizamos la función de clip, esta se encarga de acortar el valor introducido al intervalo dado, por lo cual recibe tres parámetros, el primero es el número que se desea acortar, en nuestro caso ese elemento será $pow(i/255.0, gamma) * 255.0$, el segundo es el valor mínimo a ser reducido, en este caso 0, y el último es el valor máximo a ser reducido, que será 255. Es decir si el número introducido es un número mayor que 255, se acortará a 255, mientras que si el valor introducido es menor que 0, se acortará a este mismo.

Listing 4.4: Filtro de brillo

```
gamma = get_gamma(img)
lookUpTable = np.empty((1,256), np.uint8)
for i in range(256):
    lookUpTable[0,i] = np.clip(pow(i / 255.0, gamma) * 255.0, 0, 255)
res = cv2.LUT(img, lookUpTable)
    return cnn_face_detector(gray, 1)
```

Cuando finalice el bucle se ejecutará el método LUT, este se encuentra en la librería

de cv2. LUT recibe dos parámetros, la propia imagen que se desea modificar y la tabla de consulta calculada anteriormente de 256 elementos.

Para este filtro se necesita indicar una gamma, la cual será la encargada de ajustar la iluminación a la imagen. La corrección de gamma se puede utilizar mediante una transformación lineal para conseguir reparar el brillo en la imagen. Los diferentes valores de gamma con su respectivo valor de entrada y salida se pueden observar en la Figura 4.3, cuanto más bajo sea el gamma más conseguiremos aumentar la iluminación, y cuanto más alto sea menor iluminación aplicaremos.

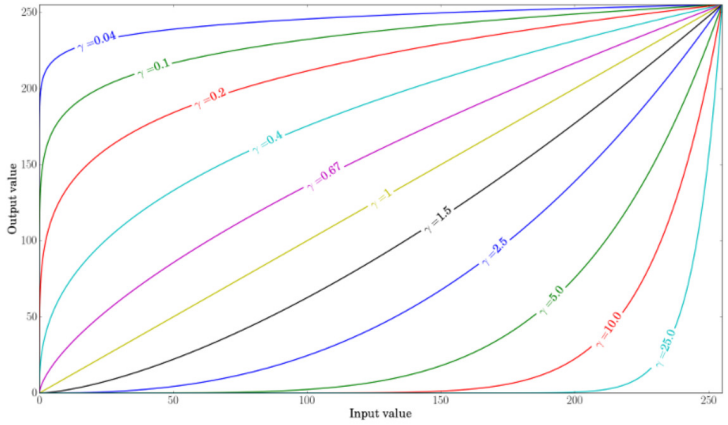


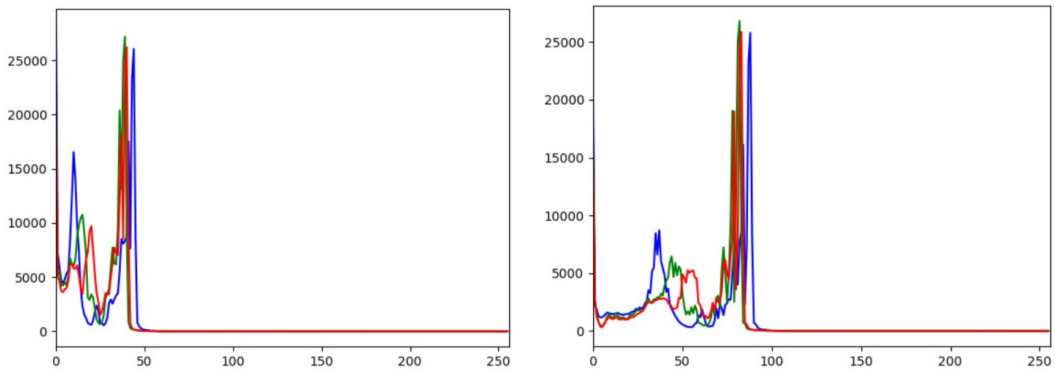
Figura 4.3: Diferentes valores de gamma

Tras aplicar este filtro se consigue la siguiente imagen de la Figura 4.4(b).

(a) Imagen oscura (b) Imagen aclarada

Figura 4.4: Diferencia al aplicar filtro iluminosidad

En la Figura 4.5 se puede observar los diferentes histogramas de las Figuras 4.4(a) y 4.4(b). Siendo el eje vertical el número de píxeles que contiene la imagen y el horizontal los valores de luminosidad que puede tomar la imagen, en este eje podemos encontrar como tres franjas imaginarias, la primera nos informa de las sombras de la imagen, en el primer caso se ve como la imagen es todo sombras, por lo tanto nos encontramos con una imagen subexpuesta donde la información de la imagen se acumula a la izquierda en forma de picos, la segunda franja nos informa sobre zonas de luminosidad media, que es en donde podemos encontrar la segunda imagen, por último, la tercera franja nos informa de las luces de la imagen.



(a) Imagen oscura (b) corrección de gamma

Figura 4.5: Histogramas de imagen oscura y corrección de gamma

4.3. Técnicas de clasificación

Para la parte de clasificación usamos transfer learning mediante la plataforma de TensorFlow Hub y buscamos un clasificador adecuado, en este caso se elige ResNet50, ResNet es un tipo de red neuronal que fue publicada por Kaiming He et al. en [HZRS15b].

Para entrenar el modelo ya cargado con nuestras respectivas imágenes bastará con invocar a la clase de Keras Sequential, en la cual podremos apilar las diferentes capas del modelo.

Para esto cargamos las capas de la ResNet50 y le añadimos al final una capa Densa con dos neuronas, las cuales serán las salidas respectivas de menores y mayores de edad, la activación elegida será sigmoide ya que restringe la salida a un número entre 0 y 1, lo cuál para el caso de nuestra clasificación binaria es lo necesario.

Listing 4.5: ResNet50 TL

```
model = tf.keras.Sequential([
    hub.KerasLayer("https://tfhub.dev/tensorflow/resnet_50/classifi
                   cation/1"),
    tf.keras.layers.Dense(2)
    tf.keras.layers.Flatten()
])
```

Posteriormente elegimos los parámetros adecuados para poder compilar el modelo.

- **Función de pérdida**: Este es un método para evaluar si el algoritmo está yendo correctamente, si las predicciones se desvían demasiado de los resultados reales, la función de pérdida mostraría un gran número. Esta consigue reducir el error de la predicción mediante ayuda de determinadas funciones de optimización. En este caso se decide utilizar la función de pérdida "binary_crossentropy", la entropía cruzada binaria calculará la perdida de entropía cruzada entre las predicciones y verdaderas.
- **Optimizador**: Actualiza los valores de los parámetros, respecto al learning rate, dando como resultado la medición del error cometido por la red. Para el optimizador escogemos SGD ya explicado en el capítulo dos, una breve explicación sería que se basa en actualizar los diferentes parámetros Θ en la dirección negativa del gradiente g (para un objetivo de minimización).
- **Tasa de aprendizaje**: Es la tasa a la que estamos ajustando los pesos de nuestro modelo con respecto al gradiente de pérdida, cuanto más pequeño el valor, más lento irá. En este caso se ha seleccionado una tasa de aprendizaje del 0.002.
- **Métrica**: La métrica elegida para este modelo es MAE para que calcule las diferencias absolutas entre los valores objetivo y las predicciones.

Listing 4.6: Compilar modelo ResNet50 TL

```
model.compile(
        loss='binary_crossentropy',
        optimizer=SGD(lr=LEARNING_RATE, momentum=MOMENTUM, nesterov=True),
        metrics=['mae'])
```

Para cargar los datos en el modelo llamamos a la clase *LoadData* en la que utilizamos la librería de pandas para poder leer el *Comma-Separated Values* (CSV) con las direcciones de las imágenes y sus respectivas etiquetas. Una vez cargadas las imágenes mediante la función *map_fn* y la ayuda de Open-CV, se separan los datos en entrenamiento y test, mediante *train_test_split()* una función especifica de sklearn.preprocessing. Estos los dividimos en 90 % y 10 % respectivamente.

Listing 4.7: Separaración de datos

```
self.train_X, self.test_X, self.train_y, self.test_y =
train_test_split(features,labels,test_size=0.10,random_state=42)
```

Posteriormente del 90 % de datos para el entrenamiento volvemos a separar esta vez en 20 % para conseguir los datos para validación. Por lo tanto, tendríamos una distribución de 70 % entrenamiento, 20 %validación y un 10 % test.

Una vez que tenemos los datos ya repartidos, entrenaremos el modelo mediante *fit_generator*. Una vez entrenado guardaremos el modelo para guardar el proceso y para posteriormente agilizar el proceso al realizar pruebas.

Calculamos las predicciones del modelo con *predict_generator*. Por último guardaremos en un CSV el resultados de las predicciones comparadas con los datos objetivos, para así poder ver las diferencias entre estos visualmente.

También se ha utilizado el modelo de ResNet-50 de la librería de keras.applications. Como capa de agrupamiento se usa *"avg'*, es decir capas de agrupamiento promedio. Este tipo de capas nos va a permitir reducir las dimensiones de entrada de la matriz, reduciendo por tanto la varianza y la complejidad de los cálculos. Un ejemplo de este tipo de capa de agrupación promedio se observa en la Figura 4.6.

Listing 4.8: ResNet50 con Keras

```
base_model = ResNet50(include_top=False, weights='imagenet',
                      input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3), pooling="avg")
```

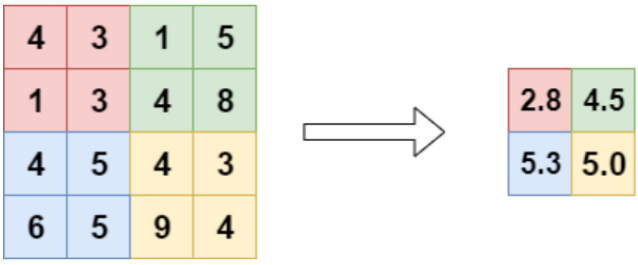


Figura 4.6: Ejemplo de capa de agrupamiento promedio [Mac21]

Para acelerar el proceso de ejecución se guardan en archivos los modelos entrenados, preservando aquellos con los que se obtiene mejores resultados. Estos archivos contienen tanto la arquitectura del modelo, como los pesos que se aprendiendo durante el entrenamiento, la configuración del modelo y su estado, permitiendo así ejecutar el modelo sin tener que volver a entrenarlo.

Para volver a cargar el modelo guardado, valdría con llamarlo de la siguiente manera.

Listing 4.9: Guardar modelo

```
model.save('path_to_my_model.h5', save_format="tf")
```

Listing 4.10: Cargar modelo

```
model = load_model('path_to_my_model.h5', custom_objects={'KerasLayer':
                   hub.KerasLayer})
```

4.4. Procesado de vídeos

Para procesar los vídeos se hace uso de la librería de *Open Source Computer Vision* (OpenCV) con la función VideoCapture, a la que se pasará por parámetro la ubicación del vídeo, y se irá leyendo frame a frame mediante la función de read, que devolverá True si el video se ha leido bien, y false si no se ha podido leer.

Listing 4.11: Leer frames

```
ret, frame = cap.read()
if not ret:
    break
frames.append(frame)
```

Una vez guardados los frames, se tratarán igual que en el caso anterior de las imágenes.

4.5. Selección de métricas de rendimiento

Poniendo como ejemplo la detección del rostro, las predicciones se pueden clasificar según:

- **Verdadero Positivo:** Se produce cuando el resultado predicho encuentra el rostro que se encuentra en la imagen en un principio.
- **Verdadero Negativo:** Sucede cuando la imagen original no tiene ningún rostro y el modelo no detecta ningún rostro.
- **Falso Positivo:** Se produce cuando el modelo encuentra un rostro pero en verdad en la imagen original no existe ningún rostro.
- **Falso Negativo:** Sucede cuando la imagen contiene un rostro pero el modelo no es capaz de detectarlo.

La matriz de confusión se utiliza para poder medir la precisión que tiene el clasificador, se puede observar un ejemplo en la Figura 4.7.

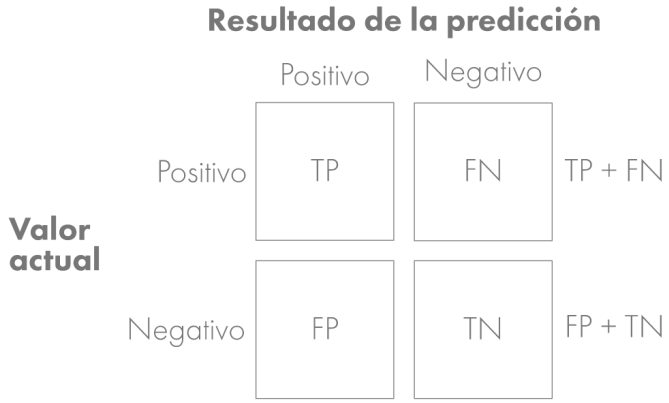


Figura 4.7: Matriz de confusión [Mac21]

- **Métrica de exactitud:** esta métrica es una de las más usadas, se basa en identificar el número de elementos clasificados correctamente en comparación con el total de elementos.

$$Exactitud = \frac{VP + VN}{VP + FP + VN + FN} \tag{4.1}$$

- **Métrica de exhaustividad/sensibilidad:** esta métrica muestra el número de verdaderos positivos que ha detectado nuestro modelo en función del total de positivos.

$$Exhaustividad = \frac{VP}{VP + FN} \tag{4.2}$$

- **Métrica de precisión:** esta muestra el número de verdaderos positivos que existen realmente comparándolos con los predichos.

$$Precisión = \frac{VP}{VP + FP} \tag{4.3}$$

- **Puntuación F1:** esta métrica es una combinación de las dos últimas métricas (exhaustividad y precisión). La mejor puntuación se representa con 1 y la peor con el valor 0.

$$F1 = 2 * \frac{precisión * exhaustividad}{precisión + exhaustividad} \tag{4.4}$$

4.6. Tecnologías

A lo largo del proyecto se han usado distintas librerías para la ayuda de detección de rostros, tratamiento de datos, mejorar el rendimiento, etc. A continuación se nombran

varias librería utilizadas, comentando el uso que se ha realizado con cada una. A parte, también se muestran las herramientas y plataformas que se han ido usando a lo largo de todo el trabajo, desde la utilización de modelos ya pre entrenados hasta la creación de máquinas virtuales para mejor el proceso de ejecución.

4.6.1. Librerías

A continuación se explican algunas de las librerías más relevantes usadas a lo largo del proyecto.

- **Dlib**: Dlib se trata de una biblioteca de software multiplataforma escrita en C ++, de código abierto. Su diseño está mayoritariamente influenciado por ingeniería de software basada en componentes y las ideas del diseño por contrato. Esta contiene muchos algoritmos de ML, compresión, análisis de imágenes, etc.

  En este trabajo se ha utilizado para conseguir los landmarks de los rostros.

- **Numpy**: Esta librería de código abierto, está especializada en el cálculo numérico y el análisis de datos, sobretodo para grandes conjuntos de datos. En el trabajo se utiliza numpy en vez de listas propias que ofrece Python debido a que ocupa menos memoria y es mucho más rápido en términos de ejecución.

- **OpenCV**: OpenCV, fue creada por Intel, la primera versión se lanzó en 2006 y actualmente cuenta con más de 2500 algoritmos. Esta librería ofrece una interfaz gráfica tanto para Java, C++ y Python. Abrir, guardar, dibujar formas simples y escribir en imágenes, son unas de las operaciones simples que ofrece OpenCV para tratar las imágenes. Las oportunidades que ofrece OpenCV son enormes, desde la creación de modelos de realidad aumentada, la detección de rostros con su posterior clasificación de género o detección de objetos. En este trabajo se ha explotado el potencial de esta librería sobretodo a la hora de procesar imágenes para su posterior estudio.

- **Matplotlib**: Esta librería se utiliza por su gran potencial a la hora de producir gráficas a partir de datos contenidos en diversas variables del lenguaje de programación python. Con esta librería se han realizado las gráficas para poder seguir el proceso de entrenamiento de la red neuronal, tanto para la pérdida como para la métrica utilizada.

- **Multiprocessing**: Este paquete se utiliza para exprimir al máximo los múltiples procesadores de una máquina determinada. En este trabajo se hace uso del objeto Pool, con el que se consigue mediante múltiples valores de entrada, paralelizar la ejecución de una función repartiendo los datos de entrada a través de procesos. Gracias al uso de estos multiprocesos se consigue bajar el tiempo de ejecución muy

considerablemente. En la Tabla 5.7, se puede observar los diferentes resultados de ejecutar el preprocesamiento con y sin multiprocesamiento. Los núcleos con los cuales se han realizado tales pruebas son un total de cuatro.

Tabla 4.1: Tabla de resultados de tiempo de ejecución para el preprocesamiento

Número imágenes	Tiempo de preprocesamiento	Tiempo de preprocesamiento con multiproceso
100	4 min 37 seg	1 min 51 seg
200	10 min 58 seg	3 min 42 seg
300	16 min 12 seg	5 min 23 seg

- **Keras**: Keras es una biblioteca de código abierto (con licencia MIT). El 28 de marzo de 2015, se lanza la primera versión de esta multiplataforma. El objetivo principal era conseguir mejorar la fluidez a la hora de crear de redes neuronales: para ello, Keras no actúa como framework independiente, sino como una interfaz de uso intuitivo (API) que autoriza el acceso a diferentes frameworks de ML para así poder desarrollarlos. En este trabajo Keras ha sido una herramienta fundamental tanto para la creación de redes neuronales.

4.6.2. Herramientas y Plataformas

En esta sección se mostrarán tanto las herramientas como las plataformas utilizadas a lo largo del proyecto. Estas son (i) *TensorFlow Hub*, (ii) *TensorBoard*, (iii) *A Neural Network Playground* y (iv) *Google Cloud*.

- **TensorFlow Hub:** Esta herramienta sirve como repositorio (tfhub.dev) de modelos ML entrenados. La librería de TensorFlow Hub te permite descargar estos modelos y reutilizarlos con unas mínimas líneas de código. Se pueden observar modelos diferentes tanto para la clasificación de imágenes, como para la clasificación de texto, de transferencias de estilos y detección de objetos.

  Este trabajo se ha centrado en la investigación de modelos de la parte de clasificación de imágenes.

- **TensorBoard:** Con su simple interfaz y gracias a sus gráficos representativos, esta herramienta se ha convertido en una herramienta muy potente a la hora de la visualización de los modelos. Con esta herramienta se consigue obtener las diferentes métricas después de realizar el entrenamiento de los diferentes modelos, gracias a esto se puede contrastar la información de éstos, para conseguir un resultado mejorado al finalizar el proceso.

- **A Neural Network Playground:** Esta herramienta es del estilo de TensorBoard, con la que se puede ver el proceso de las redes neuronales durante su entrenamiento,

pudiendo elegir la tasa de aprendizaje, la activación, la regularización, las neuronas por capa, las capas de la red y el tipo del problema clasificación ó regresión. También podemos encontrar 4 tipos de datasets con los que se puede experimentar distintos problemas de ML.

- **Google Cloud:** Es una plataforma que contiene un espacio virtual, por el cual varias tareas que anteriormente requerían hardware o software pueden ser ejecutadas y recientemente se puede utilizar la nube de Google para el almacenamiento, acceso y gestión de datos. Por cuestiones de optimización se decide utilizar la plataforma de Google Cloud ya que el ordenador utilizado para este trabajo no disponía de GPU y para la utilización de tensorflow es necesario una para conseguir una buena optimización.

4.7. Base de Datos

La base de datos utilizada para este trabajo es la de AgeDB [MPS+17], este dataset destaca porque evita el problema del ruido que tienen muchos otros al ser procesados semi-automáticamente. Las imágenes contenidas en AgeDB son recopiladas manualmente, obteniendo etiquetas precisas del año y del género correspondiente sin ruido.

Para conseguir las respectivas etiquetas se buscaron imágenes en Google imágenes, y se cogieron sólo aquellas que tuvieran la fecha de nacimiento expresamente debajo de éstas. Además hay que tener en cuenta que esta base de datos contiene imágenes en diferentes poses, con ruido, etc.

AgeDB contiene un total 16,488 imágenes que contienen tanto personas famosas, como actores/actrices, escritores, científicos, políticos, etc. El número medio de imágenes por cada persona es de 29 fotografías, siendo la edad mínima 1 y la edad máxima de 101 años. En la Figura 4.8, se puede observar las cantidad de imágenes por años, encontrándose un pico alrededor de los 30 y 40 años.

En la Figura 4.9, se puede observar las imágenes con las etiquetas con su nombre, edad, género, todas éstas separadas mediante un guión bajo. También se puede observar como de un mismo personaje se tienen varias fotos con varias edades.

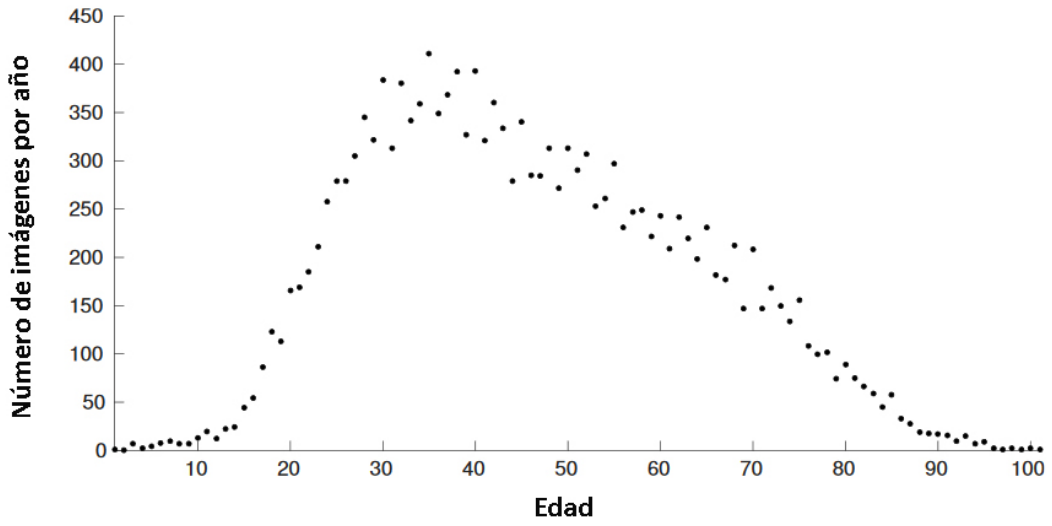


Figura 4.8: Cantidad de imágenes por Edad [MPS+17]

Figura 4.9: Ejemplo de porción aleatoria de AgeDB

Capítulo 5

Experimentos y Resultados

En este apartado se muestran los objetivos obtenidos, la parte más complicada y la que más tiempo requirió, aparte de la investigación de métodos potentes para la estimación de la edad, fue la búsqueda de un buen equipo para poder procesar tal cantidad de datos y poder trabajar fluidamente con TensorFlow ya que el equipo actual no posee de una GPU, por lo tanto todo el tema del modelo se complicaba, ya que podía llegar a tardar un simple entrenamiento de una red mínimamente sencilla alrededor de 100 horas, cosa que no era asequible de ninguna manera.

Por lo que los experimentos realmente comienzan con una etapa de empezar a poder tener un equipo decente para poder ejecutar las cosas. Una vez conseguidas las herramientas para poder empezar a trabajar, se empezó a ejecución de programas que se tenían, tanto filtros, como preprocesamiento de todas las imágenes del dataset y de la comprobación de estas mediante un modelo.

En la Sección 5.1 se explicará como se consiguió el equipo necesario para poder realizar el trabajo aquí presente. En la Sección 5.2 se muestran los experimentos relacionados con el preprocesamiento de las imágenes. En la Sección 5.3 se realizan las pruebas con estas imágenes mejoradas y se comparan con imágenes no preprocesadas. Por último, en la Sección 5.4 se exponen los experimentos realizados a la hora de procesar los videos de entrada.

5.1. Equipo necesario

Como ya se ha comentado en la parte introductoria del capítulo, una gran parte del tiempo se gastó en poder encontrar un equipo adecuado para la ejecución del modelo y el procesamiento de todas las imágenes del dataset. A continuación se muestran las características del ordenador que se posee:

- **Procesador:** Intel(R) Core(TM) i5-7200U CPU 2.50GHz 2.71 GHz

- **RAM instalado:** 8,00 GB
- **Tipo de sistema:** 64-bit operating system, x64-based processor
- **GPU:** No

Se empieza probando con una tarjeta gráfica, la cual la pide el tutor responsable del TFG para poder conectarla al equipo actual y poder ejecutar tirando de ésta. Una vez llega esta tarjeta gráfica y se prueba a conectar al ordenador actual no realiza ningún tipo de respuesta por lo que se piensa en un plan B para no perder mucho más tiempo con este método.

Posteriormente se decide construir una máquina virtual en Google Cloud con un presupuesto limitado. Se instala un CUDA 11.1 para poder disponer de una GPU. Cuando se finaliza el proceso de instalación y se prueba a ejecutar el código, tensorflow empieza a dar problemas con el tipo de versión de CUDA, ya que resulta que había que tener una versión de CUDA anterior, se intenta trabajar sobre lo ya instalado, intentando descargar una versión inferior a la actual. Tras varios intentos fallidos se decide borrar todo lo realizado hasta el momento y empezar de nuevo.

En esta ocasión se busca un método para levantar una máquina virtual directamente con todo lo necesario. Se empieza a investigar y se encuentra una forma de levantar una instancia de tensorflow a través de Google Cloud. La instancia levantada tiene las siguientes características:

- **GPU:** 1 x NVIDIA Tesla P4
- **Tipo de máquina:** n1-highmem-4 (4 CPU virtuales, 26 GB de memoria)
- **Plataforma de CPU:** Intel Skylake
- **Tamaño**: 100GB
- **Zona:** northamerica-northeast1-b
- **Framework:** TensorFlow:2.4

Cuando se prueba a ejecutar el código en esta instancia comienza a funcionar, por lo que ya se procede a realizar los experimentos correspondientes.

5.2. Preprocesamiento de las imágenes

Para intentar mejorar el resultado, se pretende mejorar las imágenes de entrada para reducir tanto el ruido, poca iluminación, etc. Para conseguir esto se prueban varios filtros, que se desarrollan a continuación.

El filtro de la mediana tiene como objetivo principal eliminar el posible ruido que haya en la fotografía, mediante una máscara de 3x3, iremos recorriendo los píxeles de la matriz, pondremos los 9 píxeles correspondientes al pasar el núcleo en un array, lo ordenaremos, y cogeremos el elemento del medio, remplazándolo en el píxel correspondiente.

Los resultados obtenidos se pueden observar en la Figura 5.1, donde al pasar la primera imagen con ruido por el algoritmo se obtiene la segunda imagen.

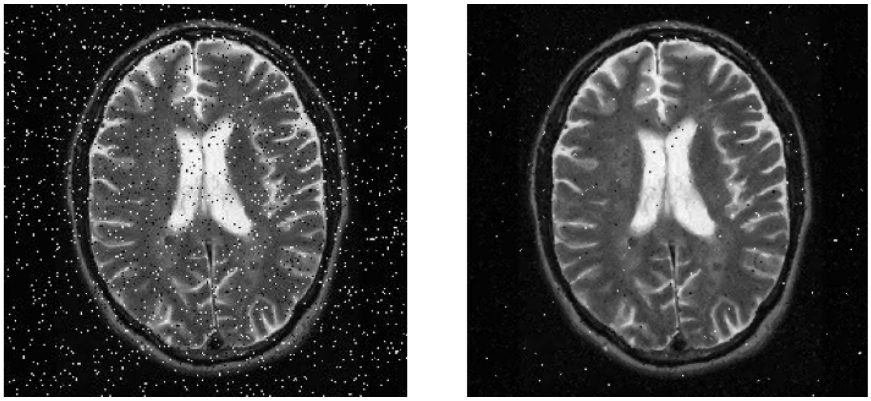


Figura 5.1: Filtro de la mediana

Posteriormente se tiene la idea de potenciar los bordes para poder potenciar los rasgos característicos del envejecimiento, tal y como pueden ser las arrugas, para esto necesitamos localizar los bordes de las imágenes, por lo que escogemos el filtro de Laplaciano.

Este filtro consiste en pasar un máscara (en este caso la máscara elegida es de $3x3$) por cada píxel de la imagen, multiplicando esta por una máscara predefinida anteriormente.

Las máscaras predefinidas anteriormente con la respectiva salida de la imagen son:

$$mascara1 = [[-1,-1,-1],[-1,8,-1],[-1,-1,-1]] \tag{5.1}$$

$$mascara2 = [[1,1,1],[1,-8,1],[1,1,1]] \tag{5.2}$$

$$mascara3 = [[0,-1,0],[-1,4,-1],[0,-1,0]] \tag{5.3}$$

$$mascara4 = [[0,1,0],[1,-4,1],[0,1,0]] \tag{5.4}$$

$$mascara5 = [[1,1,1],[1,1,1],[1,1,1]] \tag{5.5}$$

Como se puede observar, los mejores resultados se encuentran al utilizar la máscara 2, correspondiente a la Figura 5.2(b) y la máscara 4 que corresponde a la Figura 5.3(b). En cambio

Como en la parte de detección de bordes no se obtuvo el resultado deseando se siguió buscando investigando otro tipo de filtros para detectar bordes, obteniendo así los filtros de Prewitt, Sobel y Robert.

El operador de Prewitt se basa en pasar una máscara por toda la imagen obteniendo

(a) Máscara 1 (b) Máscara 2

Figura 5.2: Filtro Laplaciano máscaras 1 y 2.

(a) Máscara 3 (b) Máscara 4 (c) Máscara 5

Figura 5.3: Filtro Laplaciano máscaras 3, 4 y 5.

así la magnitud del gradiente de esta, para esto están involucrados tanto las filas como las columnas adyacentes para poder conseguir una mayor inmunidad al ruido. Sobel en cambio aplica otra máscara, este filtro es más sensible a los bordes diagonales aunque hay poca diferencia entre ellos.Por último, Robert, es el filtro que mejores resultados nos ha dado obteniendo los bordes de las imágenes más claros.

En las Figuras 5.4 y 5.5 se puede ver el resultado de ejecutar Prewitt, Sobel y Robert en este orden, y como Robert arroja un mejor resultado.

La luminosidad de una imagen es la que muchas veces otorga el sentido de la misma, ya que si esta se encuentra en malas condiciones no se podrá realizar un estudio sobre la imagen en cuestión. Por este motivo tan importante, se empieza a investigar como se podría mejorar en este aspecto la fotografía. Se encuentra una operación no lineal, que es capaz de ajustar el brillo de la imagen, llamada corrección de gamma o gamma. Esta operación se puede definir con la siguiente fórmula.

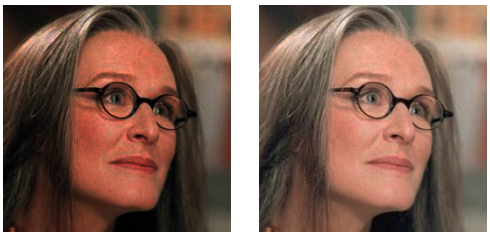
$$Img_{output} = \frac{Img_{input}}{255}^{\gamma} * 255 \tag{5.6}$$

Dónde gamma $\gamma < 1$ se utiliza cuando hay una imagen oscura y se quiere dar

Figura 5.4: Filtros de Prewitt, Sobel y Robert

Figura 5.5: Filtros de Prewitt, Sobel y Robert

luminosidad, mientras que $\gamma > 1$ se utiliza en caso contrario, cuando es una imagen iluminada y se quiere bajar el brillo de la misma. Una vez ya ha quedado claro el uso de gamma se realiza un pequeño programa que será ejecutado cuando se procesen todas las imágenes. El gamma seleccionado para bajar la exposición de las imágenes oscuras es un $\gamma = 0,6$. El resultado de probar esto se muestra en las siguientes Figuras 5.6, 5.7 y 5.12.



(a) Imagen entrada (b) Imagen salida

Figura 5.6: Ejemplo de filtrado por luminosidad [MPS+17]

Como se puede observar la Figura 5.6, se obtiene una mejora considerable en la imagen procesada, debido a el aumento de la exposición. Por otro lado, en las siguientes Figuras, se observa un deterioro de la imagen ya que éstas no necesitaban el mismo valor de gamma

(a) Imagen entrada (b) Imagen salida

Figura 5.7: Ejemplo de filtrado por luminosidad [MPS+17]

(a) Imagen entrada (b) Imagen salida

Figura 5.8: Ejemplo de filtrado por luminosidad [MPS+17]

que la anterior Figura. Como es obvio no todas las imágenes necesitan el mismo grado de exposición que otras, ya que una imagen sobreexpuesta no necesita un aumento en la exposición, si no todo lo contrario, una disminución de ésta. En consecuencia, se piensa una forma para detectar si la imagen necesita un aumento de la exposición o una reducción.

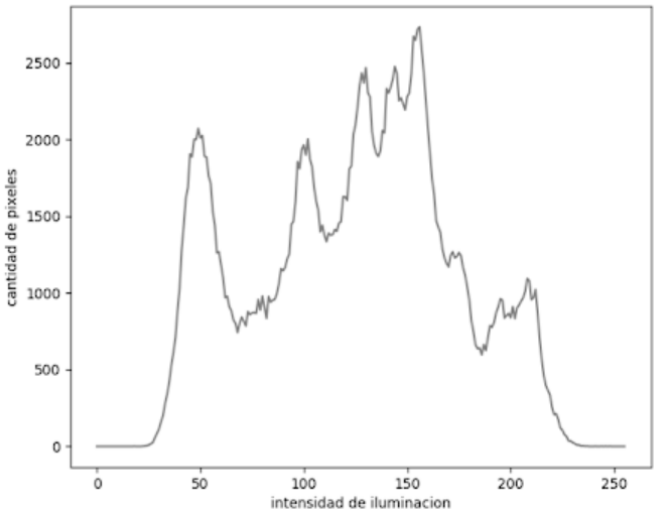


Figura 5.9: Ejemplos de histograma

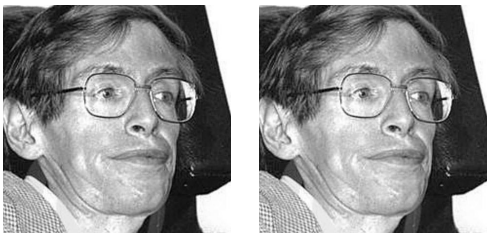
Tras varias pruebas e indagaciones, se decide utilizar el histograma para saber el nivel de intensidad de iluminación por cada píxel. Un ejemplo de lo comentado, se puede observar

en la Figura 5.9. Se localizan dos casos diferenciados entre sí, pero en ambos se tiene en cuenta si pertenecen a la parte izquierda o derecha del histograma.

- Contar los niveles de saturación que están vacíos.
- Contar los niveles de saturación que tienen más de 800 píxeles.

Una vez contados los puntos anteriores, se procederá a su interpretación, dando más importancia a aquellos que tengan más niveles de exposición vacíos. Si el contador de ceros es mayor en la franja izquierda que en la derecha, se corregirá bajando la exposición, en caso contrario se le subirá. Si estamos tratando con el otro contador, tendremos que tener en cuenta que si la franja izquierda obtiene un valor mayor que la derecha, será porque la imagen está más oscura y por lo tanto tendremos que aumentar la exposición, en caso contrario se la bajaremos. Para cada caso se prueba con varias imágenes y se decide bajar o subir el gamma en proporción a varias imágenes ya probadas con anterioridad.

Se prueba a ejecutar lo comentado previamente en el dataset de AgeDB. A continuación se muestran algunos resultados, incluyendo la mejora de la Figura 5.12 mediante la Figura 5.10, que aunque el cambio es mínimo se puede ver como hay zonas que se han aclarado.



(a) Imagen entrada (b) Imagen salida

Figura 5.10: Ejemplo de filtrado por luminosidad automatizado [MPS+17]

(a) Imagen entrada (b) Imagen salida

Figura 5.11: Ejemplo de filtrado por luminosidad automatizado [MPS+17]

Tiempo después, se añade una condición al principio con la que solo se va a realizar el ajuste si la diferencia de los parámetros calculados anteriormente es un valor mayor que

(a) Imagen entrada (b) Imagen salida

Figura 5.12: Ejemplo mejorado de filtrado por luminosidad automatizado [MPS+17]

5, debido a que, tras varias pruebas se detecta que estas imágenes no necesitan un cambio de exposición.

A la hora de la detección se realizan varias pruebas para poder detectar aquellas fotos de perfil. Se puede observar en las imágenes de la Figura 5.13 la diferencia de realizar la detección mediante CNN representados con los cuadrados rojos y dlib get_frontal_face_detector representado con cuadrados verdes.

Tabla 5.1: Detección imágenes de perfil

Path	CNN	Frontal Detector
imagenDePerfil.jpg	SI	NO
imagenNormal	SI	SI

Las dos primeras imágenes solo son detectadas por la red convolucional mientras que la última imagen detecta todos los rostros con ambos detectores.

Tras aplicar las técnicas restantes se consigue como resultado un dataset con rostros recortados de 224x244 de dimensión por cada imagen. De 16.488 imágenes que contiene el dataset principal se consiguen detectar 16.345. Estas imágenes se guardarán para más adelante, obteniendo un CSV para almacenar el path de cada imagen con su respectiva edad.

Tabla 5.2: Tabla de ejemplo de CSV

Id	Path	Age
0	databaseAge/0_MariaCallas_35_f_0.jpg	mayores
1	databaseAge/10000_GlennClose_62_f_0.jpg	mayores
2	databaseAge/10001_GoldieHawn_23_f_0.jpg	mayores
3	databaseAge/10002_GoldieHawn_24_f_0.jpg	mayores
4	databaseAge/10003_GoldieHawn_24_f_0.jpg	mayores

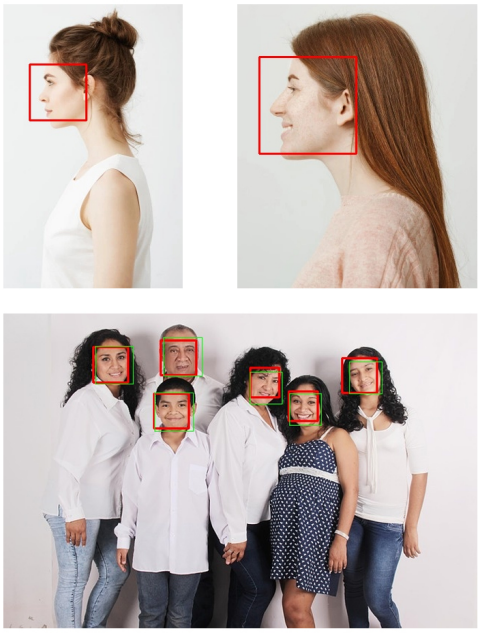


Figura 5.13: Ejemplos de detección de rostros CNN vs frontal dlib

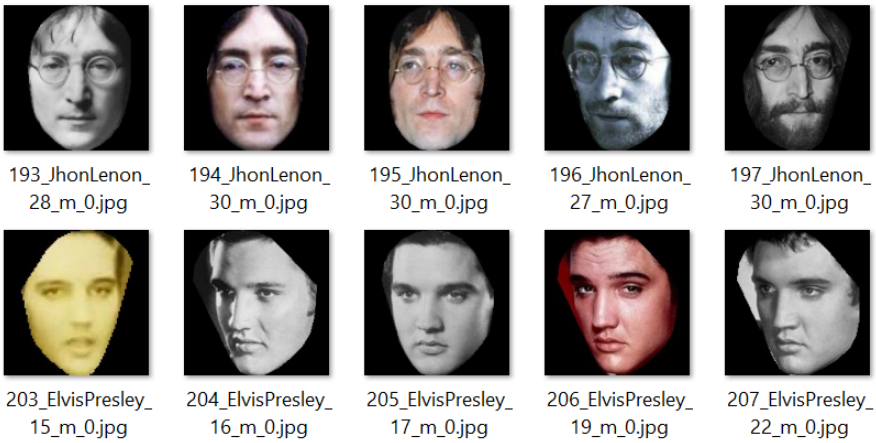


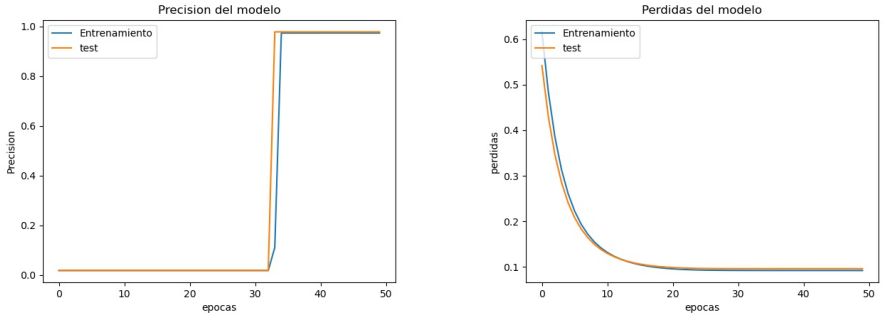
Figura 5.14: Dataset preprocesado

5.3. Prueba de las imágenes mejoradas

Para probar que estas imágenes procesadas producen una mejora se empieza utilizando mediante TL un modelo entrenado de resnet-50 `https://tfhub.dev/tensorflow/resnet_50/classification/1`. Se empieza clasificando entre menores y mayores de 18. Por lo que como salida se añade una capa con una neurona y una activación sigmoide.

Se vuelve a entrenar el modelo con las imágenes del dataset de AgeDB. Únicamente se realizaron 50 vueltas debido a que el tiempo de ejecución se prolongaba a 2 horas. El resultado de entrenar el modelo se puede observar en la Figura 5.15. Cuando termina la ejecución, se realizan las predicciones sobre los datos de test y se guardan en un CSV.

Como se puede observar en las gráficas se ve un sobreajuste de los datos de test con los de entrenamiento. Tras indagar sobre este problema se detecta que las imágenes de menores del dataset AgeDB es un total de 315 fotografías respecto a las imágenes de mayores que es de 16030 imágenes, por lo que las imágenes de menores son irrelevantes respecto a los mayores.



(a) Precisión modelo (b) Perdidas modelo

Figura 5.15: Gráfica resultante al entrenar el modelo

Se confirma el resultado del CSV generado y se observa que lo que detecta el modelo todo el rato son imágenes de menores. Las primeras 10 lineas de este CSV se muestran en la Tabla 5.3, todos los demás ejemplos dan el mismo valor predicho. 0 representa la clase de los mayores, mientras que 1 representa la clase de menores.

Tabla 5.3: Resultado Resnet-50 de TL

Valor predicho	Valor real
0	0
0	0
0	0
0	0
0	0
0	1
0	0
0	0
0	0
0	1

Se prueban a cambiar la métrica utilizada, el ratio de aprendizaje, la función de aprendizaje, etc, pero ninguno cambia el resultado. El problema se centraba en la diferencia de imágenes por clase.

La siguiente solución planteada fue igualar el número de imágenes entre menores y mayores, para esto se utiliza otro dataset llamado *appa-real-release* [CBT+18]. Leemos los datos del CSV que contienen tanto el path de las imágenes como sus respectivas edades. Para separar los datos de menores y mayores se utiliza la librería de pandas mediante las

lineas escritas a continuación.

Listing 5.1: Separación de menores y mayores

```
train_dataset = pd.read_csv(filepath_or_buffer="database_age.csv",
                                    names=column_names, header=0)

df3 = train_dataset[["path", "age"]]
menores = df3.loc[df3.age == "menores"]
mayores = df3.loc[df3.age == "mayores"]
```

Posteriormente se concatenarán los menores de appa-real-release con los datos del dataset de AgeDB. Sin embargo, aunque los menores conseguido de este dataset fueron mayores (854 imágenes de menores), no fueron suficientes, ya que ahora serían un total de 1169 menores sobre más de 15.000 imágenes de mayores. Para ver si se conseguía eliminar el sobreajuste, se cogieron estas imágenes de menores y con el mismo tamaño se cogieron las de mayores, es decir, se tratan de 1169 imágenes de menores y 1169 de mayores. Se vuelve a entrenar el modelo con 50 vueltas. El resultado fue el mismo, cambiaban unas décimas sobre los valores de predicción pero seguían perteneciendo a la misma clase 0 de mayores.

10 de las lineas del CSV se pueden observar en la siguiente Tabla 5.4, donde se observa una misma detección.

Tabla 5.4: Resultado Resnet-50 de TL con 1169 imágenes de menos y 1169 de mayores

Valor predicho	Valor real
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	1
0	1
0	1

Antes de buscar otra solución se busca otro modelo entrenado en Tensorflow Hub, en este caso se encuentra "https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/5". Se vuelve a entrenar con 50 vueltas y cuando se realizan las predicciones se ven variaciones pero no se consiguen buenos resultados.

Como las imágenes de menores son insuficientes se utilizan rangos de edad, es decir, se va a realizar una clasificación multiclase para intentar ajustar los datos de entrada. Los rangos de edad seleccionados son los siguientes:

- De 0 a 29 años: adolescentes.

- De 30 a 49 años: adultos.
- De 51 a más: mayores.

Las clases pertenecientes a cada uno serán, 0, 1 y 2 respectivamente. Antes de empezar se crea un CSV en el que se almacenarán tanto el path, si pertenecen a la clase adolescentes, adultos o mayores, y la clase en valor numérico a la que pertenecen. En este caso tendremos un CSV como la Tabla 5.5.

Tabla 5.5: Tabla de ejemplo de CSV con rangos de edad

Id	Path	Age	Categorical
0	databaseAge/0_MariaCallas_35_f_0.jpg	adultos	1
1	databaseAge/10000_GlennClose_62_f_0.jpg	mayores	2
2	databaseAge/10001_GoldieHawn_23_f_0.jpg	adolescentes	0
3	databaseAge/10002_GoldieHawn_24_f_0.jpg	adolescentes	0
4	databaseAge/10003_GoldieHawn_24_f_0.jpg	adolescentes	0

Posteriormente al coger los datos, se utiliza para las etiquetas la función de *to_categorical* de la librería de utils de keras. Esta nos permite definir las clases de la siguiente manera.

Tabla 5.6: Tabla de ejemplo de CSV

Clase predicha	Clase real
0	[1,0,0]
1	[0,1,0]
2	[0,0,1]

Después se prueba con el modelo preentrenado de TL, `"https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/5"`. Los resultados se muestran en la tabla siguiente, donde se muestran algunas lineas del CSV. También se puede observar los valores resultantes de predecir.

Tabla 5.7: Tabla de resultados con MobileNet

Clase predicha	Clase real	Valor de la predicción
2	[0,1,0]	[0.0000000e+00, 2.9596832e-09, 1.0000000e+00]
2	[0,0,1]	[0.0000000e+00, 3.1118234e-05, 9.9996889e-01]
1	[0,0,1]	[0. , 0.99696034, 0.00303965]
2	[0,0,1]	[0.0000000e+00, 1.06807936e-20, 1.0000000e+00]
1	[0,0,1]	[0.0000000e+00, 1.0000000e+00, 6.1985296e-19]
1	[0,1,0]	[0.0000000e+00, 1.0000000e+00, 1.0852366e-31]
1	[0,1,0]	[0.000000e+00, 1.000000e+00, 8.236708e-15]
1	[0,0,1]	[0.000000e+00, 9.99778e-01, 2.217434e-05]
1	[0,1,0]	[0.0000000e+00, 1.0000000e+00, 4.5968102e-10]
1	[0,0,1]	[0.0000000e+00, 1.0000000e+00, 9.4488066e-21]
1	[0,1,0]	[0.0000000e+00, 1.0000000e+00, 1.7571465e-28]

Como podemos observar, los valores predichos varian, cosa que no sucedían antes al clasificar entre menores y mayores. Aún así no se consigue un buen resultado, por lo que se decide crear un modelo desde 0 con Keras.

Para este modelo vamos a utilizar una ResNet-50, a esta red se le quita la última capa y se le añade en su lugar, una capa con 3 neuronas con una activación softmax. El tamaño de entrada será el tamaño de las imágenes, en este caso 224x224x3. Para inicializar los pesos de la última capa densa, utilizaremos el atributo *kernel_initializer* con una distribución normal.

Listing 5.2: Creación modelo ResNet50 con Keras

```
base_model = ResNet50(include_top=False, weights='imagenet',
                      input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3),
                      pooling="avg")

prediction = Dense(units=3, kernel_initializer="he_normal", use_bias=False,
                   activation='softmax',name="pred_age")
                   (base_model.layers[-1].output)

model = Model(inputs=base_model.input, outputs=prediction)
```

Para compilar este modelo se decide utilizar un optimizador SGD, una pérdida de categorical_crossentropy y una métrica de MAE.

Listing 5.3: Compilar modelo ResNet50 con Keras

```
sgd = SGD(lr=LEARNING_RATE, momentum=MOMENTUM, nesterov=True)

model.compile(optimizer=sgd, loss="categorical_crossentropy", metrics='mae')
```

Una vez entrenado el modelo con 50 vueltas, se procede a las predicciones, un ejemplo de algunas se pueden observar en la Tabla 5.8.

Tabla 5.8: Tabla de resultados con ResNet-50

Clase predicha	Clase real	Valor de la predicción
2	[0,0,1]	[1.0264375e-06, 1.8513334e-01, 8.1486559e-01]
2	[0,0,1]	[1.7351347e-25, 8.9680810e-08, 9.666666883-01]
2	[0,1,0]	[1.1360840e-15, 3.8202133e-02, 9.6179978e-01]
1	[0,1,0]	[1.6065695e-06, 8.6513853e-01, 1.3485979e-01]
1	[0,0,1]	[2.3987856e-10, 9.999906e-01, 9.5465989e-06]
1	[0,1,0]	[5.23554975e-05, 9.9994683e-01, 8.7916288e-07]
2	[0,0,1]	[2.125005e-30, 2.398706e-06, 9.999976e-01]
1	[0,0,1]	[1.9220511e-07, 9.8709667e-01, 1.2903138e-02]
2	[0,0,1]	[4.3756824e-22, 4.6763611e-07, 9.9999952e-01]
2	[0,0,1]	[3.0347889e-14, 3.4708813e-05, 9.9996531e-01]
2	[0,1,0]	[1.9449652e-10, 5.0299717e-03, 9.9497002e-01]

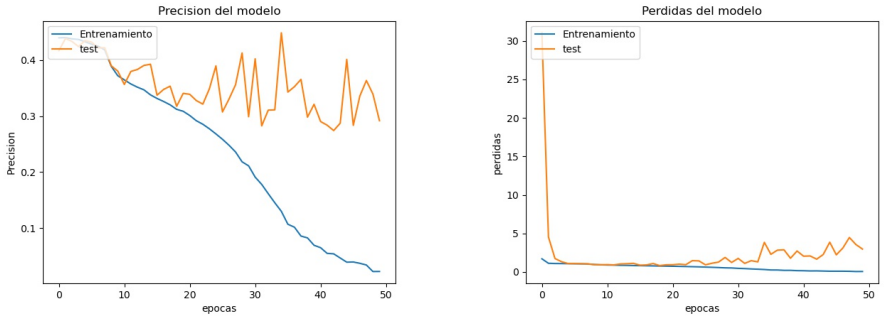
Al ver los resultados se observa una mejora por lo que se decide realizar un *evaluate* sobre los datos de test.

Listing 5.4: Evaluar modelo

```
1 results = model.evaluate(data.test_X, data.test_y)
```

El resultado de esta evaluación es de un MAE de 0.236345514 y una pérdida de 2.6246. Los resultados ya empiezan a mejorar y por lo tanto se quiere ver la diferencia de entrenar un modelo con imágenes procesadas a un modelo con imágenes no procesadas.

Por tanto, esta vez se entrena el modelo pero cogiendo las imágenes directamente de AgeDB, sin alinear el rostro, ni aplicar ningún recorte. Tras entrenarlo y realizar un evaluate sobre los datos de test se obtiene un MAE de 0.27751219 y una pérdida de 2.55655. Como vemos el resultado es algo peor que el modelo anterior entrenado con imágenes procesadas con el alineamiento y el corte. Las gráficas se pueden ver en la Figura 5.16



(a) Precisión modelo (b) Perdidas modelo

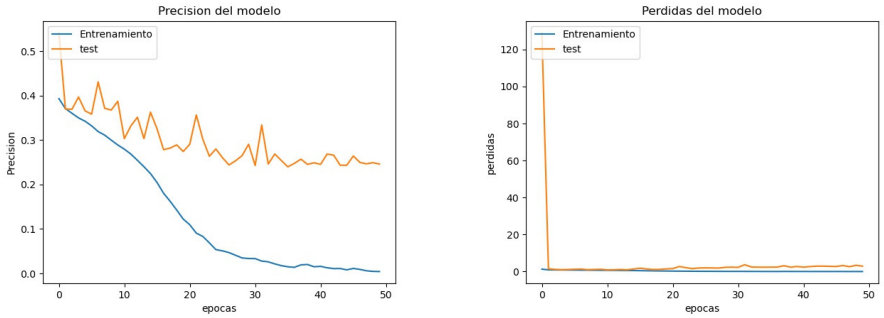
Figura 5.16: Gráfica resultante al entrenar el modelo con imágenes de AgeDB sin preprocesar

Para probar diferencias entre cada uno se prueba a introducir a la hora de evaluar, en el modelo preprocesado imágenes no preprocesadas, y en el modelo con imágenes no preprocesadas imágenes preprocesadas. El resultado de esto se puede observar en la Tabla 5.9. El modelo 1 hace referencia al modelo entrenado con imágenes preprocesadas, mientras que el modelo 2 hace referencia al modelo entrenado con imágenes no preprocesadas.

Tabla 5.9: Tabla de ejemplo de CSV

Modelo escogido	Pérdida	MAE
Modelo 1 con imágenes preprocesadas	2.6246	0.2600745514
Modelo 2 con imágenes no preprocesadas	2.55655	0.29751219
Modelo 1 con imágenes no preprocesadas	4.9013	0.2985
Modelo 2 con imágenes preprocesadas	2.9501	0.3031842

Como se puede observar hay una mejora entre los dos modelos, donde el primero tiene un mejor porcentaje de acierto.Una vez que se ha definido el modelo con el que se va a trabajar y tras la vista de que las imágenes procesadas dan un mejor resultado que aquellas no procesadas, se decide realizar el mismo proceso con la imágenes filtradas con la función de exposición. Tras el entrenamiento con 50 vueltas se devuelven las siguientes imágenes. Las gráficas resultantes de este entrenamiento se pueden observar en la Figura 5.17.



(a) Precisión modelo (b) Perdidas modelo

Figura 5.17: Gráfica resultante al entrenar el modelo con imágenes con preprocesamiento + filtrado

También se realizó un evaluación, consiguiendo los resultados de la siguiente Tabla 5.10

Tabla 5.10: Resultados modelo con imágenes preprocesadas + filtrado

Pérdida	MAE
2.821057	0.24501931

Algunas líneas del CSV resultante se pueden observar en la siguiente Tabla 5.11

Tabla 5.11: Tabla de resultados con ResNet-50 imágenes preprocesadas + filtro

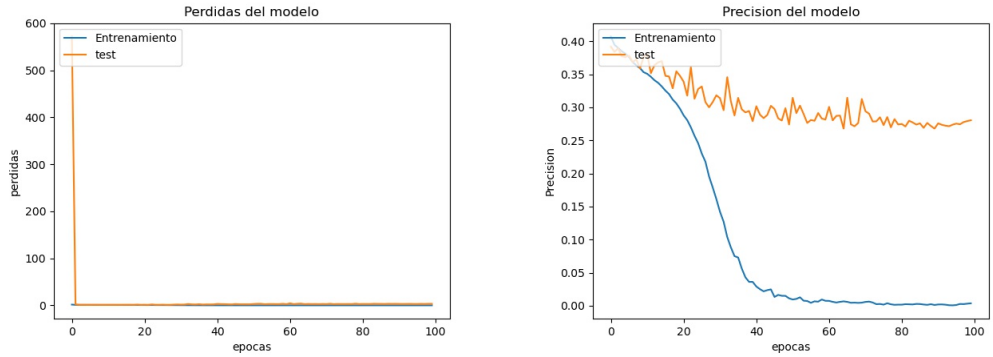
Clase predicha	Clase real	Valor de la predicción
0	[0,1,0]	[6.1506194e-01 3.8493812e-01 3.1521803e-11]
1	[1,0,0]	[1.3866599e-09 9.9947196e-01 5.2809023e-04]
1	[1,0,0]	[1.2323041e-03 9.9876773e-01 1.1073955e-09]
1	[0,1,0]	[2.2177470e-01 7.7822524e-01 6.1334852e-08]
1	[0,1,0]	[5.2338517e-03 9.9476612e-01 1.9554738e-08]
2	[0,0,1]	[4.7915443e-12 1.1795022e-01 8.8204974e-01]
1	[0,1,0]	[3.0277354e-11 9.8702347e-01 1.2976542e-02]
0	[1,0,0]	[9.9928361e-01 7.1643858e-04 3.5386127e-20]
2	[0,0,1]	[0.0000000e+00 4.4179713e-13 1.0000000e+00]
1	[0,1,0]	[2.2793576e-05 9.9997687e-01 4.0948848e-07]
2	[0,0,1]	[3.7379954e-17 1.3530060e-03 9.9864703e-01]

Como se sospechaba, el modelo con las imágenes mejoradas gracias al filtro da un mejor resultado frente a los otros dos modelos, este último no solo cuenta con la mejora del filtro, si no también cabe recordar, que cuenta con la mejora de la detección de fotos de perfil.

Posteriormente, se repara en la realización de un procedimiento erróneo basado en la extracción de gamma, consistente en la ejecución de un proceso cuya base reside en la proporción de la misma, consiguiéndose así únicamente que ésta fuese proporcional al dato usado en la regla de tres. De esta manera, se identificó dicho error, y tras la realización de múltiples pruebas se trató de mejorar, optimizando así la eficacia del preprocesamiento. La mejor solución hallada consistió en la traza de una recta entre dos puntos, los cuales se obtuvieron a través de la ejecución de sucesivos intentos con diferentes imágenes afectadas para detectar aquellos puntos más adecuados para la realización de la recta. Tras cumplir este proceso, se consigue no sólo solucionar el error detectado, sino también, una proporción más razonable con respecto a los nuevos datos de entrada, incrementándose así la eficacia del proceso. Las rectas resultantes se consiguen mediante la Fórmula 5.7, donde y es el valor resultante de sustituir x (el dato de entrada) en la Ecuación. Siendo el punto 1 $(x1, y1)$ y el punto2 $(x2, y2)$.

$$\frac{x - x1}{x2 - x1} = \frac{y - y1}{y2 - y1} \tag{5.7}$$

Para probar que esto funciona se vuelve a hacer un preprocesado completo de la Base de Datos AgeDB [MPS+17], una vez realizado, se entrena el modelo correspondiente. Las gráficas resultantes tras entrenar este modelo con las imágenes preprocesadas con el nuevo método son las siguientes, en la Figura 5.18 se pueden observar tanto las pérdidas Subfigura 5.18(a), como la precisión del modelo Subfigura 5.18(b).



(a) Pérdidas (b) Precisión

Figura 5.18: Diferencia recorte de frame

Se introducen varias imágenes para el testeo, se puede observar la diferencia entre pasarlas por el filtro o no en la Figura 5.19. También se observa que con el nuevo método de preprocesiento, de las 31 imágenes introducidas, se detectan con éxito el rostro de las 31 personas. Mientras que con el método de preprocesamiento de DeepuAge [ALKS20] sólo se consiguen detectar 27. Un ejemplo de las imágenes de testeo se puede observar en la Figura 5.19, donde se muestra tanto la imagen real introducida, como las diferencias entre el modelo presentado en este trabajo y el de [ALKS20].

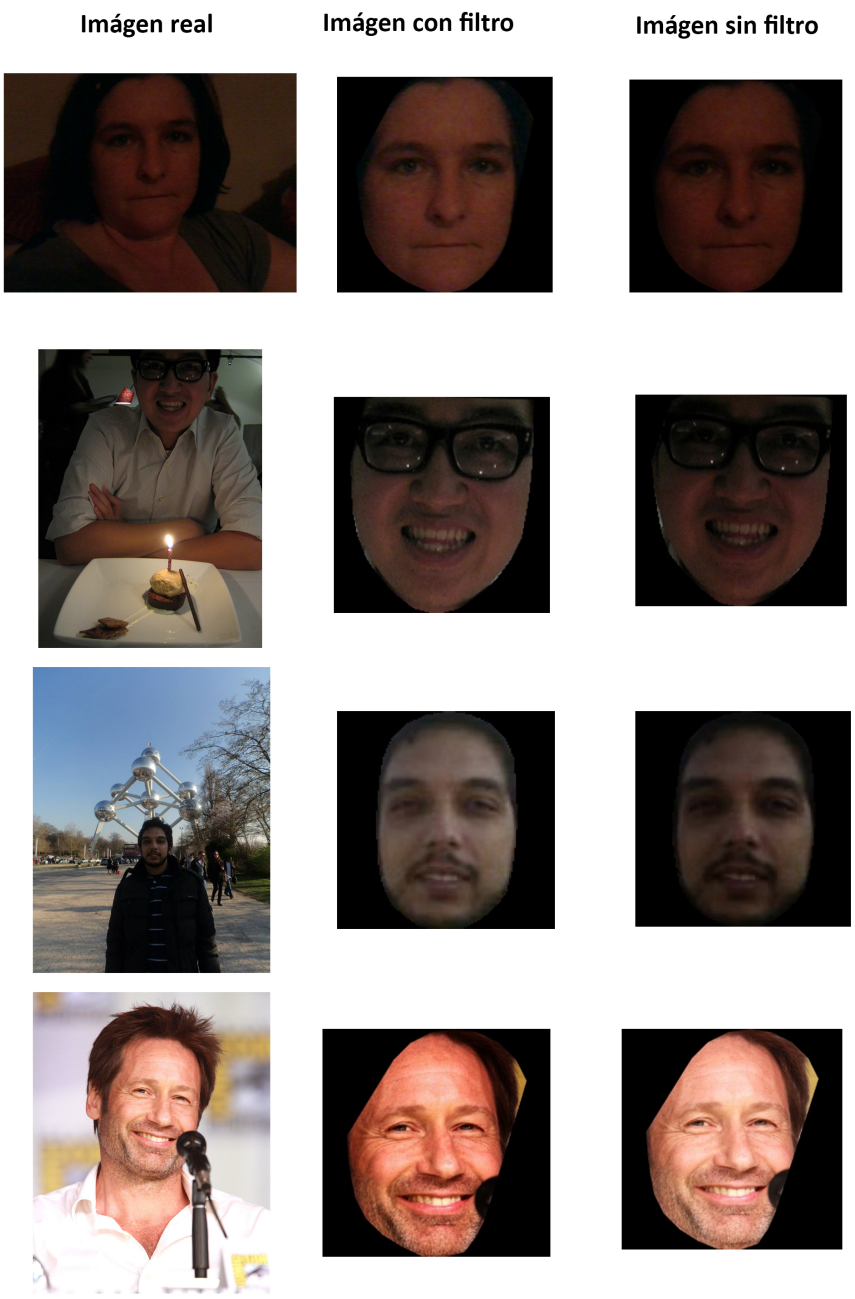


Figura 5.19: Imágenes de testeo

Dos de las imágenes que no se consiguieron detectar con el trabajo de [ALKS20] se pueden observar en la Figura 5.20. Donde las imágenes de la izquierda hacen alusión a las imágenes de entrada y las de la derecha al resultado de pasarlas por el preprocesamiento mostrado en este trabajo.

Figura 5.20: Imágenes no detectadas por [ALKS20]

Se usa el modelo guardado para ver la diferencia con este trabajo y se consiguen las siguientes diferencias. Para la primera imagen de la Figura 5.19 se detecta con el trabajo de DeeUAge [ALKS20] un rango de edad de entre [0-29] con una edad real de 35, mientras que el modelo creado en este trabajo consigue detectar un rango correcto de entre [30-49], en la Tabla 5.12 se pueden ver las diferencias con los modelos mencionados.

Tabla 5.12: Resultado de Modelo3 con vídeo

Edad real	Modelo DeepUAge	Modelo propio
35	[51 - >]	[30 - 49]
62	[51 - >]	[51 - >]
10	[51 - >]	[30 - 49]
90	[30 - 49]	[51 - >]
69	[30 - 49]	[51 - >]
24	[51 - >]	[30 - 49]
16	[51 - >]	[0-29]

Se puede observar (aunque con resultados no exactos) que se consigue una mejora tanto en imágenes detectadas, como en detección de la edad. Gracias a la mejora de la iluminación en las imágenes se consigue enfatizar/descubrir rasgos potentes para la detección de la edad, así como para la propia detección de rostros como ya se ha demostrado con anterioridad.

5.4. Detección de la edad mediante vídeos

Para esto simplemente se separan los frames del vídeo y posteriormente se tratan igual que si fueran un conjunto de imágenes. Para separar los frames del vídeo, se utiliza la función *load_video*, donde se realiza un resize a todos éstos para poder seguir tratando con imágenes de 224x224, tal y como se viene haciendo desde el principio del trabajo.

Para coger las capturas del vídeo, utilizaremos la función de *VideoCapture* de la librería Open-CV. A continuación, se leerá el frame y se recortará para que posteriormente cuando se haga el cambio de tamaño a 224x224 no se altere la imagen, es decir, en el siguiente ejemplo de la Figura 5.21, aparece un frame de un vídeo de ejemplo.

Figura 5.21: Ejemplo de frame de vídeo aleatorio

Si se cambiara de tamaño directamente a 224x244, daría como resultado la Figura 5.22(a), mientras que la Figura 5.22(b), muestra como quedaría la imagen si antes de proceder a realizar el cambio de tamaño, se realizará el recorte del frame. Como se ve, en la Figura 5.22(a), se contrae para hacer el cambio de tamaño oportuno, mientras que en la otra se ve perfectamente la región recortada.

(a) Cambio de tamaño a 224x224

(b) Corte y cambio de tamb año a 224x224

Figura 5.22: Diferencia recorte de frame

Se prueba a introducir un vídeo al modelo, para ver que resultados arroja. Lo primero

de todo es procesar el vídeo, obteniendo los frames y almacenándolos en su correspondiente carpeta y creando un CSV que servirá para la posterior lectura de los frames. Un ejemplo tras preprocesar con el filtro las imágenes del vídeo, se pueden observar en la Figura 5.23. Los frames totales de este vídeo son un total de 44 imágenes.

Figura 5.23: Ejemplo de preprocesamiento del vídeo

Una vez se han separado los frames del vídeo en imágenes, ya se puede tratar como con anterioridad. Los metemos al modelo 3 (modelo entrenado con las imágenes preprocesadas + filtro). El resultado obtenido de las predicciones son las clases [1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 2, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0], también se puede observar parte del CSV generado con las predicciones en la Tabla 5.14. La edad de la persona del vídeo es de 21 años, así que pertenecería a la clase 0, del rango de edad [0 - 29] adolescentes.

Tabla 5.13: Resultado de modelo presentado con vídeo

Valor predicho	Valor real
1	0
1	0
1	0
1	0
0	0
0	0
1	0
1	0
0	0
0	0
0	0
0	0

Posteriormente se introducen estos datos al modelo 1 (modelo preprocesado con

imágenes con alineación y recorte), a continuación se muestran los resultados de la predicción [2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1]

Tabla 5.14: Resultado de modelo [ALKS20] con vídeo

Valor predicho	Valor real
2	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0

En este último caso se puede observar como no es capaz de acertar la edad correcta de ningún frame.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

En este trabajo se han presentado sistemáticamente un conjunto de técnicas con el objetivo de mejorar el preprocesamiento de imágenes, siendo éste un proceso fundamental para poder clasificar dichas imágenes de la forma más óptima y eficaz posible. Una de las características de las imágenes fundamentales para poder obtener la eficacia deseada, es la exposición, siendo una de las principales responsables de los problemas generados a la hora de detectar y clasificar la edad en caso de que sea una exposición que se encuentre por debajo de los índices que se requieren. Así, es fundamental que se tenga en cuenta dicha característica antes de empezar a construir o probar un modelo de preprocesamiento.

Durante la realización del trabajo se ha reparado en el descubrimiento de que numerosas imágenes disponían de problemas de iluminación, generando como se ha expresado anteriormente, una detección incorrecta del rostro, que repercute a su vez en la detección de la edad, puesto que la percepción de dicha imagen resultaba difícil. Como se puede observar en el Capítulo 5 de experimentos, se realizan pruebas con varios filtros, obteniendo con mejores resultados el filtro de mejora de iluminación. De esta manera, se ha demostrado en este trabajo que la aplicación de un filtro de mejora de la exposición de la imagen, permite detectar un rostro que, sin dicho filtro, resultaba indetectable. La importancia de este filtro, es la misma para el proceso de estimación de la edad, puesto que, en ambos, un rostro sobreexpuesto/subexpuesto genera peores resultados al no poder observarse con claridad rasgos potenciales del rostro. Así se demuestra que, con un buen preprocesamiento de las imágenes, generado a través de la implantación del filtro nombrado, se consiguen mejores resultados que sin la realización de dicho preprocesamiento.

Es importante también destacar la introducción de un detector de rostros de perfil, ya que normalmente las imágenes que encontramos en las diferentes Bases de Datos para la detección de la edad son seleccionadas de situaciones cotidianas, con lo que se encuentran distintas poses, iluminaciones, entornos, etc. Este aspecto genera la pérdida de múltiples

imágenes para el entreno del modelo en caso de no introducir el detector de rostros de perfil, negándolo así de un mayor número de imágenes para aprender a detectar la edad. Tras los resultados obtenidos, se puede concluir una mejora en comparación con el modelo de DeepUAge, modelo que destaca por unos resultados buenos en el proceso de detección de edad 2,73 %. La mejora conseguida es evidente, cuya evidencia ha sido comentada y argumentada anteriormente. De esta manera, se puede concluir que el preprocesamiento de las imágenes es un proceso crucial para resolver los problemas que se han ido presentando a lo largo del trabajo, como la pérdida de imágenes cruciales para el entrenamiento, la pérdida de rasgos potenciales para la detección de la edad entre otros, los cuales han sido solucionados en este trabajo consiguiendo así la evidente mejora mencionada.

Finalmente, cabe destacar que no se han conseguido los resultados esperados, y entre las posibles causas debidas a una pérdida de eficacia, se ha intuido que la falta de vueltas para entrenar al modelo ha sido una de las más importantes. La falta de vueltas ha sido generada por problemas relacionados con el equipo técnico como potencia del ordenador como se explicaba en el Capítulo 5. Por otro lado, la red neuronal propuesta en el trabajo se puede mejorar modificando los parámetros de la misma como la tasa de aprendizaje, añadiendo dropout,etc, consiguiéndose así mejores resultados.

6.2. Trabajo futuro

Para el trabajo futuro se plantea el uso de las redes generativas adversarias. La idea principal es utilizarlas para ampliar datasets para menores, ya que actualmente hay una gran falta de imágenes etiquetadas para gente menor. Por esto se pretende realizar un dataset artificial de rostros artificiales.

Para esto tendremos que disponer de una red neuronal que será nuestro generador, con el cuál crearemos imágenes ficticias y una red neuronal llamada discriminador que se encargará de detectar si el rostro es real o artificial, por lo tanto estamos hablando de una red que en la última capa va a tener un única neurona, que mediante la ayuda de una activación sigmoide, vamos a poder detectar si pertenece a la clase real en caso de que se acerque a 1 o pertenezca a la clase de imágenes falsas si se acerca a 0.

El objetivo al entrenar el generador será que mediante una entrada aleatoria consiga sacar una imagen de un rostro.

La idea es entrenarlos simultáneamente para que el generador sea el vencedor en esta competencia. Para esto vamos a analizar el valor del discriminador tanto con imágenes falsas como imágenes reales obtenidas con el generador. Al principio los rostros obtenidos por el generador no se parecerán mucho a un rostro humano, por lo que al discriminador le resultará muy fácil poder detectar entre falsas y reales, y por tanto el error será muy pequeño. Pero a medida que va avanzando el entrenamiento, el generador empezará a aprender a realizar rostros más parecidos a la realidad y por tanto al discriminador le

costará más detectar si esa imagen es real o no, ampliando así el error de éste.

Por otra parte, se pretende usar *Long Short Term Memory* (LSTM), para preservar datos importantes a la hora de la detección de la edad. Esta celda LSTM tiene una entrada y una salida adicional llamada celda de estado, la cual es la clave del funcionamiento de las redes LSTM, ésta es como una banda transportadora a la que se pueden tanto añadir como eliminar datos, para esto se usan varias puertas que dependiendo de si están abiertas o cerradas permiten traspasar información o no, para saber si está abierta o cerrada se realiza con la función de sigmoide.

- **Forget gate:** permite eliminar datos a la memoria. Para esto toma el estado oculto anterior y la entrada anterior, los pasa por la función de sigmoide, si la salida es cercana a 0 la LSTM eliminará esa porción de información, en caso contrario la información se mantendrá y llegará a la celda de estado.
- **Update gate:** permite crear datos a la memoria. Para esto se realiza como en la compuerta anterior, si la salida de la función sigmoide es cercana a 1 preservaremos los datos.
- **Output gate:** permite crear el estado oculto actualizado. Para esta compuerta se utiliza la función tangente hiperbólica.

Cuando se obtiene tanto la salida de forget gate y update gate simplemente se tendrá que actualizar la celda de estado para poder tener una memoria actualizado.

Por último como se comentaba en el Capítulo de conclusiones 5, la mejora de la red neuronal utilizada en este trabajo es fundamental, tanto con un aumento de vueltas como modificando los parámetros de la red. Con esto se conseguirá un aumento de aciertos a la hora de la estimación de la edad. También se podrían probar otros rangos de edad más correctos que se ajusten mejor a las condiciones de entrada.

Resumen del Trabajo en Inglés

Capítulo 7

Introduction

In recent years, it can be seen that Internet access is practically total for the youngest age group analyzed (14 and 16 years old), being 99.9 % of men and 99.6 % of women. women, while with increasing age access to it decreases. The age group that uses it the least is the age group from 65 to 74 years, with 70.5 % of men and 68.9 % of women.

These data commented on the youngest, lead to a large consumption of inappropriate content, this added to the current situation of the pandemic with COVID-19, has led to a much greater increase according to several studies, among which is the global study developed by the cybersecurity company Avast, where 90 % of parents comment the involuntary exposure to adult material as one of their biggest concerns about Internet use by their children. 67 % of the children admitted having bad experiences online during confinement, being exposed to insulting, rude and inappropriate content.

The exhibition of both adult material and sexting is one of the biggest concerns for parents, apart from the fact that children can also endanger devices and homes since they can accidentally access or download false pages, having as a consequence the embedding a virus in the operating system, malware and even problems with information theft.

Very current news makes us consider the importance of estimating age in the field of Artificial Intelligence. Instagram, a multiplatform social network has announced that it is working on several improvements to ensure the protection of the youngest users, especially with an emphasis on the concept of grooming. For this, Instagram intends to restrict direct messages between adolescents and adults that are not followed, and therefore, are not within their range of acquaintances. To determine the age of the user, not only will the age indicated when registering on the platform be taken into account, but also technology related to machine learning will be used to determine the age of the user, in case they falsified their age when registering.

The estimation of age can be used to suggest different types of products based on it. The Center for Industrial Innovation in Artificial Intelligence (CII.IA) wanted to incorporate face detection to make product suggestions to the customer in a chain with different

types of stores since depending on age there is a certain predisposition to acquire certain products, in Based on all this information, you can carry out promotions focused on this type of customer profile.

On the other hand, it is worth mentioning the importance of estimating age in the forensic field, where a good method of estimating age is very important, either through age patterns, dental X-rays, anthropometric measurements, X-rays of the cervical region. , X-rays of the left hand to estimate chronological age based on the bone age parameter, etc. These techniques are usually decisive for those people who are not identified and who need to know their age, either for various matters of crime, sexual abuse, etc. Human intervention in these cases is a very slow task and therefore would slow down the entire previous process.

For all these reasons, it is intended to address in this work, the improvement of existing methods of age estimation, trying to increase the safety of the youngest, to try to prevent current grooming problems, inappropriate content for minors, etc.

Finally, the estimation of age is becoming a weighting factor for the product recommendations in the different companies, since in this way suitable products can be suggested depending on the age group of the user.

7.1. Context

This Final Degree Project has been carried out within the Analysis, Security and Systems Group (GASS Group, https://gass.ucm.es/, Group 910,623 from the catalog of groups recognized by the UCM) as part of the activities of the research project THEIA (Techniques for Integrity and Authentication of Multimedia Files of Mobile Devices) with reference FEI-EU-19-04.

7.2. Object of the investigation

This work is going to be divided into two very different sections:

Previous study of other similar works, trying to find the method that best suits our study, bearing in mind the different implementations used for it. Once the study is done, we will keep those that have given the best results, and they will be used as a starting point to try to make improvements that produce a result with less margin for error. For this, we will incorporate both image preprocessing and variations to the different parameters of the neural network used in this study.

7.3. The work plan

This work has been developed through the activities of the Table 7.1.

Tabla 7.1: Planificación del trabajo

Activities	Date made
Project structure and deadlines	2ª fortnight of september
Research of previous projects	2ª fortnight of october
Summaries of the research carried out	1ª fortnight of november
Conceptual framework chapter	1ª fortnight of december
State of the art chapter	2ª fortnight of february
Code execution	2ª fortnight of march
Analysis of data. Drafting of results and conclusions	fortnight of april
Check current memory	1ª fortnight of may
Finish retouching memory	2ª fortnight of may

To carry out a better routine when carrying out the project, Trello was used, an application with which you can easily and clearly view pending, ongoing and completed tasks. An example of how the organization has been carried out in Trello can be seen in Figure 7.1

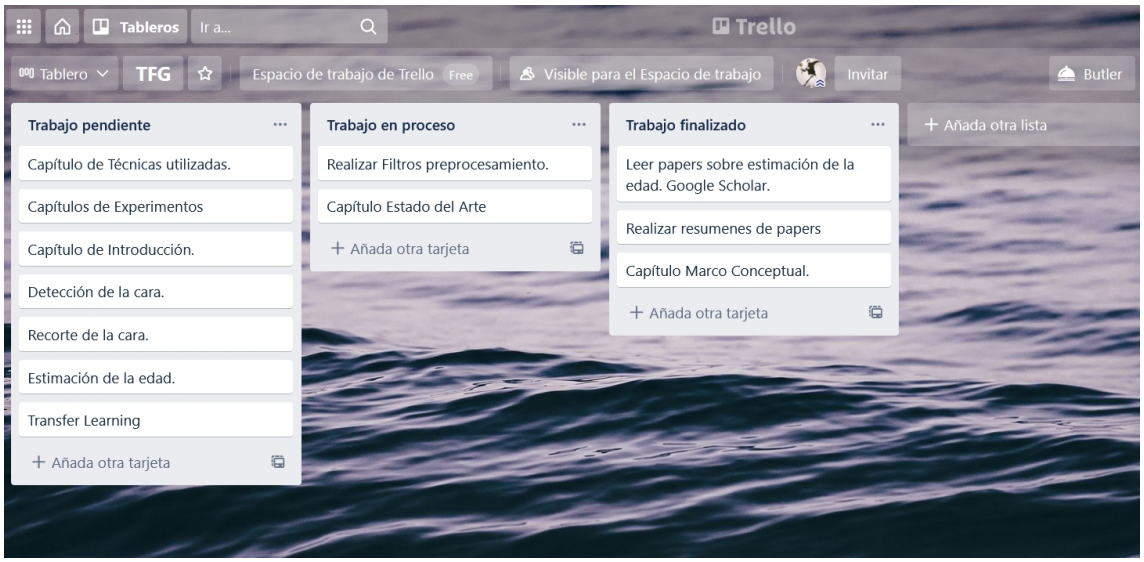


Figura 7.1: Uso de Trello en el proyecto

7.4. Project structure

The structure of this work, excluding this introductory chapter, is based on 4 chapters with a specific structure discussed below.

In Chapter 2 it is intended to address the different theoretical bases that will be used

in the rest of subsequent chapters. Both the birth of Artificial Intelligence and its growth over the years will be discussed. Later, we will delve into the different methods of object detection. Finally, after going through the advances in recent years, the main types of neural networks will be explained, delving into their architecture and use. Other topics such as Transfer Learning and Artificial Intelligence problems are also discussed in this chapter.

In the next Chapter, 3, you will find the state of the art, where a critical reflection is made on various documents, gathering relevant information for further study. From each article, both the method used and the results obtained will be taken. Subsequently, a comparison of the results will be made to find effective methods for the purpose of the work present here.

Chapter 4 addresses the different possible metrics together with the libraries used in this document. The different contributions of the work will also be commented, explaining step by step the contributions entered.

Later in Chapter 5, the experiments carried out using Tables and Figures will be shown. You will also find comparisons of the different models and of the work presented with other works.

Finally, in Chapter 6, the conclusions and future work of this project will be presented.

Capítulo 8

Conclusions and Future Work

8.1. Conclusions

In this work, a set of techniques have been systematically presented with the aim of improving the pre-processing of images, this being a fundamental process to be able to classify said images in the most optimal and efficient way possible. One of the characteristics of the fundamental images to be able to obtain the desired efficiency is the exposure, being one of the main responsible for the problems generated when detecting and classifying the age in case it is an exposure that is below of the indices that are required. Thus, it is essential that this characteristic is taken into account before starting to build or test a preprocessing model.

During the completion of the work, it was noted that numerous images had lighting problems, generating, as previously expressed, an incorrect detection of the face, which in turn affects the detection of age, since the perception This image was difficult. As can be seen in Chapter 5 of experiments, tests are carried out with several filters, obtaining the lighting enhancement filter with better results. In this way, it has been shown in this work that the application of an image exposure enhancement filter makes it possible to detect a face that, without said filter, was undetectable. The importance of this filter is the same for the age estimation process, since, in both cases, an overexposed / underexposed face generates worse results as it is not possible to clearly observe potential facial features. Thus it is shown that, with a good preprocessing of the images, generated through the implementation of the named filter, better results are achieved than without carrying out said preprocessing.

It is also important to highlight the introduction of a profile face detector, since normally the images that we find in the different databases for the detection of age are selected from everyday situations, with which there are different poses, lighting, environments , etc. This aspect generates the loss of multiple images for the training of the model in case of not introducing the profile face detector, thus denying it a greater

number of images to learn to detect age. After the results obtained, an improvement can be concluded compared to the DeepUAge model, a model that stands out for its good results in the age detection process 2,73 %. The improvement achieved is evident, the evidence of which has been discussed and argued previously. In this way, it can be concluded that the preprocessing of the images is a crucial process to solve the problems that have been presented throughout the work, such as the loss of images crucial for training, the loss of potential features for detection age among others, which have been solved in this work, thus achieving the evident improvement mentioned.

Finally, it should be noted that the expected results have not been achieved, and among the possible causes due to a loss of efficiency, it has been intuited that the lack of laps to train the model has been one of the most important. The lack of laps has been generated by problems related to the technical equipment such as computer power as explained in Chapter 5. On the other hand, the neural network proposed at work can be improved by modifying its parameters such as the learning rate, adding dropout, etc., thus achieving better results.

8.2. Future work

For future work, the use of adversarial generative networks is proposed. The main idea is to use them to expand datasets for minors, as there is currently a great lack of images tagged for minors. For this reason, it is intended to make an artificial dataset of artificial faces.

For this we will have to have a neural network that will be our generator, with which we will create fictitious images and a neural network called a discriminator that will be in charge of detecting if the face is real or artificial, therefore we are talking about a network that in the The last layer will have a single neuron, which by means of a sigmoid activation, we will be able to detect if it belongs to the real class if it approaches 1 or belongs to the false images class if it approaches 0.

The objective when training the generator will be that by means of a random input it will be able to take an image of a face.

The idea is to train them simultaneously so that the generator is the winner in this competition. For this we are going to analyze the value of the discriminator with both false images and real images obtained with the generator. At first the faces obtained by the generator will not look much like a human face, so the discriminator will find it very easy to detect between false and real, and therefore the error will be very small. But as the training progresses, the generator will begin to learn to make faces more similar to reality and therefore it will be more difficult for the discriminator to detect whether that image is real or not, thus expanding its error.

On the other hand, it is intended to use LSTM, to preserve important data at the time of age detection. This cell LSTM has an additional input and output called a status cell, which is the key to the operation of networks. LSTM, this is like a conveyor belt to which you can add as well as remove data, for this several doors are used that depending on whether they are open or closed allow information to be transferred or not, to know if it is open or closed is done with the sigmoid function.

- **Forget gate:** allows to delete data to memory. For this, it takes the previous hidden state and the previous input, passes them through the sigmoid function, if the output is close to 0 the LSTM will eliminate that portion of information, otherwise the information will be kept and will reach the state cell.
- **Update gate:** allows to create data to memory. For this, it is done as in the previous gate, if the output of the sigmoid function is close to 1 we will preserve the data.
- **Output gate:** Create the updated hidden state. For this gate the hyperbolic tangent function is used.

When you get both the output of forget gate and update gate, you will simply have to update the status cell in order to have an updated memory.

Finally, as discussed in Conclusions Chapter 5, the improvement of the neural network used in this work is fundamental, both with an increase in turns and modifying the parameters of the network. With this, an increase in correct answers will be achieved when estimating age. Other more correct age ranges that better fit the input conditions could also be tested.

Bibliografía

[Abu] JM Rojo Abuín. Regresión lineal múltiple.

[AEP06] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. pages 41–48, 01 2006.

[ALKS20] Felix Anda, Nhien-An Le-Khac, and Mark Scanlon. Deepuage: Improving underage age estimation accuracy to aid csem investigation. *Forensic Science International: Digital Investigation*, 32:300921, 2020.

[ANC07] Andrew Arnold, Ramesh Nallapati, and William W. Cohen. A comparative study of methods for transductive transfer learning. In *Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, ICDMW '07, page 77–82, USA, 2007. IEEE Computer Society.

[Bar89] Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[BLPL06] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. NIPS'06, page 153–160, Cambridge, MA, USA, 2006. MIT Press.

[BM04] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[BMP06] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, page 120–128, USA, 2006. Association for Computational Linguistics.

[BP95] D. Michael Burt and David I. Perrett. Perception of age in adult caucasian male faces: Computer graphic manipulation of shape and colour information. *Proceedings: Biological Sciences*, 259(1355):137–143, 1995.

[Car97] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[CBT+18] Albert Clapés, Ozan Bilici, Dariia Temirova, Egils Avots, Gholamreza Anbarjafari, and Sergio Escalera. From apparent to real age: gender, age, ethnic, makeup, and expression bias analysis in real age estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2373–2382, 2018.

[CET01] T.F. Cootes, G.J. Edwards, and Christopher Taylor. Active appearance models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23:681 – 685, 07 2001.

[Deb21] Debugger Cafe. Machine Learning and Deep Learning. https://debuggercafe.com, May 2021.

[DLHS16] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[DYXY07] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 193–200, New York, NY, USA, 2007. Association for Computing Machinery.

[DYXY08] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Self-taught clustering. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 200–207, New York, NY, USA, 2008. Association for Computing Machinery.

[EEH14] Eran Eidinger, Roee Enbar, and Tal Hassner. Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, 9(12):2170–2179, 2014.

[ELTC98] G.J. Edwards, A. Lanitis, C.J. Taylor, and T.F. Cootes. Statistical models of face images — improving specificity. *Image and Vision Computing*, 16(3):203–211, 1998.

[EP96] T. Ezzat and T. Poggio. Facial analysis and synthesis using image-based models. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 116–121, 1996.

[ETC98] G.J. Edwards, C.J. Taylor, and T.F. Cootes. Interpreting face images using active appearance models. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 300–305, 1998.

[GDDM] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158.

[GhZM+07] Xin Geng, Zhi hua Zhou, Senior Member, Kate Smith-miles, and Senior Member. Automatic age estimation based on facial aging patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:2234–2240, 2007.

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[Gle97] M. Gleicher. Projective registration with difference decomposition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 331–337, 1997.

[GLT98] C. Giles, Steve Lawrence, and Ah Tsoi. Rule inference for financial prediction using recurrent neural networks. 01 1998.

[GM11] Guodong Guo and Guowang Mu. Simultaneous dimensionality reduction and human age estimation via kernel partial least squares regression. In *CVPR 2011*, pages 657–664, 2011.

[GM14] Guodong Guo and Guowang Mu. A framework for joint estimation of age, gender and ethnicity on a large database. *Image and Vision Computing*, 32, 10 2014.

[GMF+09] Guodong Guo, Guowang Mu, Yun Fu, Charles Dyer, and Thomas Huang. A study on automatic age estimation using a large database. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1986–1991, 2009.

[HB98] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[HFP14] Ivan Huerta, Carles Fernández, and Andrea Prati. Facial age estimation through the fusion of texture and local appearance descriptors. 09 2014.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[HLW16] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[HZRS15a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[HZRS15b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[HZRS15c] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[JP04] M. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and matching object classes. *International Journal of Computer Vision*, 29:107–131, 2004.

[JZ07] Jing Jiang and Chengxiang Zhai. Instance weighting for domain adaptation in nlp. In *ACL 2007 - Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, ACL 2007 - Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, pages 264–271, December 2007. 45th Annual Meeting of the Association for Computational Linguistics, ACL 2007 ; Conference date: 23-06-2007 Through 30-06-2007.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[KDG+02] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[KdVL99] Young H Kwon and Niels da Vitoria Lobo. Age classification from facial images. *Computer vision and image understanding*, 74(1):1–21, 1999.

[KLM96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[LAE+16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A.C. Berg. SSD: Single Shot Multibox Detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[LBRN06] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 801–808, Cambridge, MA, USA, 2006. MIT Press.

[LCY13] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[LDC04] A. Lanitis, C. Draganova, and C. Christodoulou. Comparing different classifiers for automatic age estimation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):621–628, 2004.

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[LRBS09] Khoa Luu, Karl Ricanek, T.D. Bui, and Ching Suen. Age estimation using active appearance models and support vector machine regression. pages 1 – 5, 10 2009.

[LSL00] Sheng-Fu Liang, Alvin Su, and Chin-Teng Lin. Model-based synthesis of plucked string instruments by using a class of scattering recurrent networks. *Neural Networks, IEEE Transactions on*, 11:171 – 185, 02 2000.

[LSS+11] Khoa Luu, Keshav Seshadri, Marios Savvides, T.D. Bui, and Ching Suen. Contourlet appearance model for facial age estimation. 10 2011.

[LTC97] Andreas Lanitis, Chris J. Taylor, and Timothy F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):743–756, July 1997.

[LTC02] A. Lanitis, C.J. Taylor, and T.F. Cootes. Toward automatic simulation of aging effects on face images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):442–455, 2002.

[LXC05] Xuejun Liao, Ya Xue, and Lawrence Carin. Logistic regression with an auxiliary data source. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 505–512, New York, NY, USA, 2005. Association for Computing Machinery.

[Mac21] Machine Learning Metric. Capa de agrupamiento promedio. https://fayrix.com/machine-learning-metrics_es, May 2021.

[MJ01] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5, 2001.

[MPS+17] Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop*, volume 2, page 5, 2017.

[NMP96] Chahab Nastar, Baback Moghaddam, and Alex Pentland. Generalized image matching: Statistical learning of physically-based deformations, 1996.

[NWH03] Anto Satriyo Nugroho, Arief Budi Witarto, and Dwi Handoko. Support vector machine. *Proceeding Indones. Sci. Meeiting Cent. Japan*, 2003.

[ON15] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 11 2015.

[Opt] Neural Network Optimization Algorithms. https://towardsdatascience.com/neural-network-optimization-algorithms-1a44c282f61d.

[PARS13] Anuja Priyam, GR Abhijeeta, Anju Rathee, and Saurabh Srivastava. Comparative analysis of decision tree classification algorithms. *International Journal of current engineering and technology*, 3(2):334–337, 2013.

[Pet09] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[PPR99] Eros Pasero, Federico Piglione, and Daniela Radasanu. Short term load forecasting using a synchronously operated recurrent neural network. volume 5, pages 3478 – 3482 vol.5, 02 1999.

[PY09] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[RBL07] Marc' Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1185–1192, Red Hook, NY, USA, 2007. Curran Associates Inc.

[RC06] N. Ramanathan and R. Chellappa. Modeling age progression in young faces. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 387–394, 2006.

[RDGF15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[Res21] Research Gate. Estructura de Red Neuronal. https://www.researchgate.net, May 2021.

[RF16] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[ROS58] F ROSENBLATT. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386—408, November 1958.

[RTVG16] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Some like it hot — visual guidance for preference prediction. 06 2016.

[RTVG18] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, 126, 04 2018.

[SBK07] Shikha Singh, T. Bhatti, and D.P. Kothari. Wind power estimation using artificial neural network. *Journal of Energy Engineering-asce - J ENERG ENG-ASCE*, 133, 03 2007.

[SCW99] E.W. Saad, Thomas Caudell, and Donald Wunsch. Predictive head tracking for virtual reality. volume 6, pages 3933 – 3936 vol.6, 08 1999.

[SEZ+13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[SI98] S. Sclaroff and J. Isidoro. Active blobs. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 1146–1153, 1998.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[SRE+14] Christian Szegedy, Scott Reed, Dumitru Erhan, Dragomir Anguelov, and Sergey Ioffe. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014.

[STE13] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. 2013.

[SYCZ19] Yang Shuyu, Dawen Yang, Jinsong Chen, and Baoxu Zhao. Real-time reservoir operation using recurrent neural networks and inflow forecast from a distributed hydrological model. *Journal of Hydrology*, 579:124229, 10 2019.

[SZ08] Yangqiu Song and Changshui Zhang. Transferred dimensionality reduction. pages 550–565, 09 2008.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[TP98] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[Tur50a] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[TUR50b] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950.

[Vet96] T. Vetter. Learning novel views to a single face image. *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 22–27, 1996.

[Wei05] Sanford Weisberg. *Applied linear regression*, volume 528. John Wiley & Sons, 2005.

[Wer90] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[WZ89] Ronald J Williams and David Zipser. Experimental analysis of the real-time recurrent learning algorithm. *Connection science*, 1(1):87–111, 1989.

[Yeg09] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[YLCT19] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), January 2019.

[YLL15] Dong Yi, Zhen Lei, and Stan Li. Age estimation by multi-scale convolutional network. volume 9005, pages 144–158, 04 2015.

[ZGZC05] Shaohua Kevin Zhou, B. Georgescu, Xiang Sean Zhou, and D. Comaniciu. Image based regression using boosting method. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 541–548 Vol. 1, 2005.