Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

Trabajo Fin de Grado

Detección de mascarillas y análisis de la distancia interpersonal en interiores a partir de imágenes

Autor: *Javier Baeza Mas*

Tutor: *Cristina Losada Gutiérrez*

2021

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Electrónica y Automática Industrial

Trabajo Fin de Grado

Detección de mascarillas y análisis de la distancia interpersonal en interiores a partir de imágenes

Autor: Javier Baeza Mas

Tutor: Cristina Losada Gutiérrez

Tribunal:

Presidente: Javier Macías Guarasa

Vocal 1º: Daniel Pizarro Pérez

Vocal 2º: Cristina Losada Gutiérrez

Calificación: ……

Fecha: ……

Agradecimientos

A mi madre, por ser ella.
A mi tutora Cristina Losada, por su enseñanza constante.

Resumen

La inteligencia artificial proporciona una herramienta clave para monitorizar el cumplimiento de normas sanitarias COVID-19: distancia de seguridad, uso correcto de mascarilla y evitar aglomeraciones. Este trabajo tiene como objetivo el diseño, implementación y evaluación de un sistema de detección automática de estas normas.

El modelo de reconocimiento de imágenes generado, basado en la combinación de un detector de caras junto con una CNN, ha logrado una exactitud del 98% distinguiendo entre personas con y sin mascarilla, y un 86% clasificando la mascarilla con, sin, y mal puesta, presentando más dificultad en el reconocimiento de personas con mascarilla mal puesta.

Palabras clave: medidas prevención COVID-19, reconocimiento facial, monitorización de imágenes estáticas, redes neuronales convolucionales.

Abstract

Artificial intelligence provides a key tool to monitor compliance with COVID-19 health standards: safety distance, correct use of mask and avoid agglomerations. This work aims at the design, implementate and evaluate an automatic detection system for these standards. The generated image recognition model, based on the combination of a face detector together with a CNN, has achieved an accuracy of 98% distinguishing between people with and without mask, and 86% classifying the mask with, without, and incorrect wearing, presenting more difficulty in the recognition of people with incorrect wearing mask.

Keywords: COVID-19 prevention measures, facial recognition, static image monitoring, convolutional neural networks.

Índice general

Índice de Figuras

Índice tablas

Lista de acrónimos

AdaGrad	*Adaptative Gradient Algorithm.*
CNN	*Convolucional neuronal network.*
COVID-19	*Corona virus disease* 2019.
FC	Fully Connected.
GD	*Gradient Descent.*
GPU	*Graphics Processing Unit.*
IA	Inteligencia artificial.
ILSVRC	*ImageNet Large Scale Visual Recognition.*
LMS	*Least Mean Square.*
Mini-BGD	*Mini-batch Gradient Descent.*
MNIST	*Modified National Institute of Standards and Technology.*
MSE	*Mean Square Error.*
OMS	Organización Mundial de la Salud.
ReLU	*Rectified Lineal Unit.*
ResNet	*Residual Neural Network.*
RGB	*Red, Green, Blue*.
RMSProp	*Root Mean Square Propagation.*
RNA	Red neuronales artificiales.
ROI	*Region of Interest.*
SARS-CoV-2	*Severe acute respiratory syndrome coronavirus 2.*
SGD	*Stocastic Gradient Descent.*
Tanh	Tangente hiperbólica.
TFG	Trabajo fin de grado.
VGGNet	*Visual Geometry Group Net.*

Capítulo 1

INTRODUCCIÓN AL DOCUMENTO

1.1. Introducción al Trabajo Final de Grado

Una nueva enfermedad ha irrumpido en el panorama mundial. Su origen se sitúa en Wuhan, provincia de Hubei, China, donde, en diciembre de 2019, se detectó un brote de neumonía de origen desconocido. Poco después se conocía el agente causal, un coronavirus denominado por la Organización Mundial de la Salud (OMS) como Coronavirus SARS-CoV-2 (del inglés *severe acute respiratory syndrome coronavirus* 2) y la infección paso a denominarse enfermedad por coronavirus (COVID-19) [1]. En los meses siguientes se asistió, de forma incrédula, a su rápida propagación, hasta que, el 20 de marzo del 2020, la OMS señala el estado de pandemia [1] de una enfermedad grave (un 3% de letalidad, con un 20% de casos graves o críticos) y de rápida propagación. La vía de trasmisión del SARS-CoV-2 más avalada la constituye la transmisión directa persona a persona. Por todo ello, para frenar la trasmisión del virus y salvar vidas, la OMS plantea la necesidad de implementar medidas de contención de la COVID-19 y entre ellas se destacan las dos a las que va dirigido este trabajo: la distancia física, establecida en 1,5 metros y el uso de mascarillas.

Ante esta situación, la Inteligencia artificial (IA), más específicamente, la monitorización de imágenes estáticas, brinda unas posibilidades únicas para monitorizar el cumplimento de estas normas gracias a que: puede estar constantemente activa; su empleo no es invasivo; no requirieren la presencia de personal en todos los lugares y tiempos en los que pudieran estar infligiéndose (lo que haría casi imposible su control) al tiempo que evita exponer a los posibles vigilantes a un riesgo innecesario; su rápida resolución genera, en tiempo real, la información requerida. Así lo han entendido diversos países, que ya han realizada proyectos colaborando a la contención de la enfermedad:

China ha sido pionera en el desarrollo de sistemas de reconocimiento facial aplicados a la contención de COVID-19. En el mismo comienzo de la infección, la empresa *Now Hanwang Technology Ltd* desarrolló el primer sistema para reconocer a personas a pesar de llevar mascarilla. Entre sus objetivos se encontraba detectar y reconocer personas con fiebre que se

encontraban en lugares públicos. No solo era importante identificar al enfermo, también a las personas cercanas que podían haberse contagiado. Afirmaron que podían detectar a una persona mezclada entre 30 personas en menos de un segundo [2].

Baidu, el navegador principal de China, ha desarrollado un modelo de inteligencia artificial en código abierto que puede detectar no llevar puesta la mascarilla o llevarla mal puesta. La red entrenada ha conseguido un porcentaje de acierto de 96,5% [3].

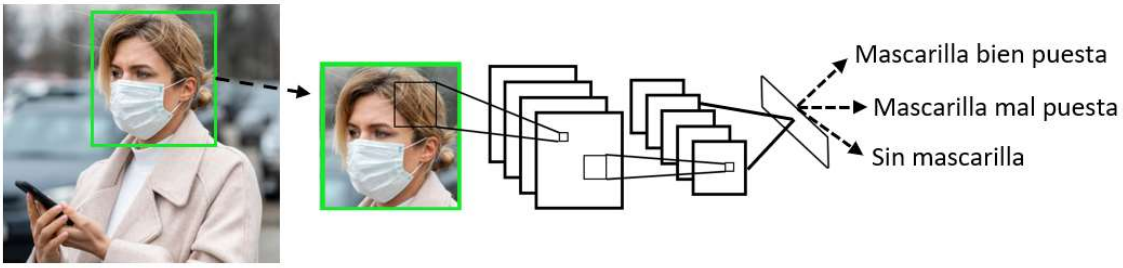
No solo en China, otros países han desarrollado sistemas de este tipo. En Brasil, para el proceso de detección, [4] utiliza el algoritmo *Haar Cascade* para detectar rostros y clasificarlos con y sin mascarilla protectora.

En este contexto, el presente trabajo fin de grado (TFG) surgió con el objetivo de monitorizar, a partir de imágenes estáticas, el uso de las mascarillas, así como la existencia de aglomeraciones de personas. Además, como llevar una mascarilla mal puesta carece de efectos preventivos, se incluyó también la detección de si la colocación es adecuada.

1.2. Sistema propuesto.

Como se ha comentado en la introducción, el objetivo de este TFG es el desarrollo, implementación y evaluación de un sistema automático basado en el análisis de imágenes utilizando inteligencia artificial que detecte la presencia o ausencia de mascarilla en una persona, así como si la mascarilla está en la posición correcta, y analice el cumplimiento de la distancia interpersonal, y la densidad de personas (para la detección de aglomeraciones), en entornos interiores.

Para ello se diseña un sistema que primero, mediante un algoritmo clasificador, detecte a partir de una imagen las caras de las personas presentes en la escena. La región de interés (ROI) alrededor de cada cara detectada se emplea como entrada para una red neuronal convolucional (CNN) que realizará una clasificación sobre ellas de como tienen colocada la mascarilla. El funcionamiento del sistema se muestra en el esquema de la Figura 1.



1º Detección de caras. **2º** Obtención de la región de interés (ROI). **3º** Extracción de características. **4º** Clasificación.

Figura 1. *Esquema de funcionamiento del sistema propuesto en el TFG (fuente propia, imagen [5]).*

1.3. Organización de la memoria

El resto del documento se organiza en 6 capítulos. En el capítulo 2 se presentan los fundamentos teóricos necesarios para el desarrollo del TFG. A continuación, el capítulo 3 recoge el desarrollo del trabajo realizado. Para finalizar, los capítulos 4, 5 y 6 describen los resultados experimentales y las principales conclusiones del TFG.

Capítulo 2

ESTUDIO TEÓRICO

2.1. INTRODUCCIÓN

La Inteligencia artificial es una rama de las ciencias computacionales encargada de estudiar modelos de cómputo capaces de realizar actividades propias de los seres humanos en base a dos de sus características primordiales: el razonamiento y la conducta [6]. Se nutre de los conocimientos de otras ramas de la ciencia a las que, a su vez, facilita el trabajo.

El término IA se acuñó en la Congreso de Dartmouth, en el año 1956 a propuesta de John McCarthy (considerado el padre de esta área), para denominar el hacer que una máquina se comporte como lo haría un ser humano, de tal manera que se la podría llamar inteligente [6].

Desde la antigüedad hacer máquinas que simulen o emulen a los hombres es un anhelo humano, su confección se atribuía, en algunas ocasiones, los propios dioses. Posiblemente, los primeros autómatas se desarrollaron en el antiguo Egipto. Allí construyeron dioses que escupían fuego por los ojos o brazos mecánicos que unían a las estatuas. En los ritos, los sacerdotes movían estos artilugios al tiempo que proclamaban que eran sus dioses quienes lo hacían [7], [8]. De forma similar, los griegos movían las estatuas en los templos gracias a sistemas hidráulicos.

No ha sido fácil definir que es la IA y, a lo largo de los años, se han propuestos diferentes definiciones que se muestran a continuación, por orden cronológico [6], [9]:

- “La automatización de actividades que vinculamos con procesos de pensamiento humano, actividades tales como la toma de decisiones, resolución de problemas, aprendizaje ...” [10]
- “La interesante tarea de lograr que las computadoras piensen ... máquinas con mente, en su amplio sentido literal.” [11]
- “El estudio de las facultades mentales mediante el uso de modelos computacionales.” [12]
- “El arte de crear máquinas con capacidad de realizar funciones que realizadas por personas requieren de inteligencia.” [13]

- “El estudio de cómo lograr que las computadoras realicen tareas que, por el momento, los humanos hacen mejor.” [14]
- “Un campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales.” [15]
- “El estudio de los cálculos que permiten, razonar y actuar.” [16]
- “La rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente.” [17]

Desde hace unos años, la IA se ha basado en el desarrollo de redes neuronales artificiales para que un computador aprenda a resolver problemas de forma similar a un cerebro humano. Ramón y Cajal, Premio Nobel de Medicina en el año 1906, descubrió la neurona, unidad histológica y de funcionamiento cerebral, la interconexión de estas células forma las redes neuronales. Treinta años después, Alan Turing [18] indagó la posibilidad de realizar redes neuronales artificiales (RNA) que se comportaran como un cerebro humano. La evolución de las RNAs ha sido muy rápida, en la Tabla 1 se presenta una breve cronología con los principales hitos relacionados con las RNAs.

Hito
Alan Turing (1936) relaciona procesos cerebrales con la computación.
Warren McCulloch y **Walter Pitts (1943)** ponen los fundamentos de computación neuronal y escriben una teoría a cerca de la forma de trabajar de las neuronas que quedan reflejadas en “*A Logical Calculus of Ideas Immanent in Nervous Activity*”. A través de circuitos eléctricos modelaron una red neuronal simple.
Donald Hebb (1936/1949) conformó que sus trabajos formasen las bases de la Teoría de las Redes Neuronales. Definió la **Regla de Hebb**, que hoy sigue vigente. En “*The Organization of Behavior*" afirma que, si dos neuronas interconectadas se activan a la vez, la fuerza sináptica aumenta.
Karl Lashley (1950) investigó donde se almacena la información (engrama), demostrando que se encontraba distribuida por todo el córtex.
Congreso de Dartmouth (1056), referente del nacimiento de la inteligencia artificial.
Desarrollo del **Perceptrón**, red neuronal artificial desarrollada por **Frank Rosenblatt (1957),** es la red neuronal más antigua, sigue vigente hoy en día para identificar patrones.

Teorema de Convergencia del Perceptrón, Frank Rosenblatt (1959): bajo ciertas condiciones, el aprendizaje del Perceptrón convergía hacia un estado finito.
Bernard Widroff y **Marcian Hoff (1960)** desarrollan el algoritmo de aprendizaje **LMS (*Least mean squares*)** que permitió el futuro desarrollo del *backpropagation* gracias a sus filtros adaptativos.
Paul Werbos (1974) sistematiza el algoritmo de ***backpropagation***, lo que conlleva el resurgimiento de las redes neuronales al superarse las debilidades de Perceptrón basado en funciones lineales.
David E. Rumelhart y **James McClelland (1986)**, simulan procesos neuronales empleando en los computadores el conexionismo (procesamiento distribuido en paralelo).
Se investiga y trabaja a partir de las bases fundamentales previas **(1986 a la actualidad)**.

Tabla 1. Cronología de la evolución de las redes neuronales artificiales, adaptada de *[18]*.

2.2. REDES NEURONALES ARTIFICIALES.

A continuación, se presentan los fundamentos teóricos relacionados con redes neuronales, estudiando su evolución hasta el desarrollo de las redes neuronales convolucionales (CNN).

2.2.1. El perceptrón.

El perceptrón fue el primer algoritmo que simulaba el comportamiento del cerebro humano. Creado en 1957 por el psicólogo estadounidense Frank Rosenblatt, considerado uno de los padres de la Inteligencia Artificial. Este algoritmo conformó la primera **red neuronal artificial**.

Para comprender mejor el funcionamiento de ésta, primero se debe entender el comportamiento de una **neurona biológica.** Las neuronas, principalmente, constan de **dendritas** (las receptoras de los estímulos a la red), el **soma** (donde interpreta las señales que recibe de las dendritas) y el **axón** (transmite la señal interpretada por el soma hasta otras neuronas). La unión entre neuronas se realiza mediante la sinapsis, que estimula o inhibe la comunicación entre ambas, consiguiendo complejas combinaciones de la transmisión de la información.

Imitando el comportamiento y estructura de las neuronas biológicas, se definieron las **neuronas artificiales,** como se muestra en la Figura 2. Las neuronas artificiales son unidades de cálculo computacional que multiplican la entrada de la red por unos coeficientes numéricos llamados **pesos**, también conocidos como **pesos sinápticos**, que son los que otorgan mayor o menor importancia a la información que recibe la neurona en sus entradas.

Inicialmente, el perceptrón, más que una compleja red de neuronas interconectadas entre sí solo constaba de una única neurona, capaz de extraer características de los datos que le entran y hacer predicciones **binarias** sobre estos, obteniendo como resultado **0** ó **1**, lo que equivaldría a **verdadero** o **falso**, **sí** o **no**, **uno** u **otro**, etc. Este resultado es función de la suma total de las operaciones numéricas de los pesos con las entradas. Si esta suma es mayor que un cierto valor, el resultado será 1 y, si es menor, será 0.

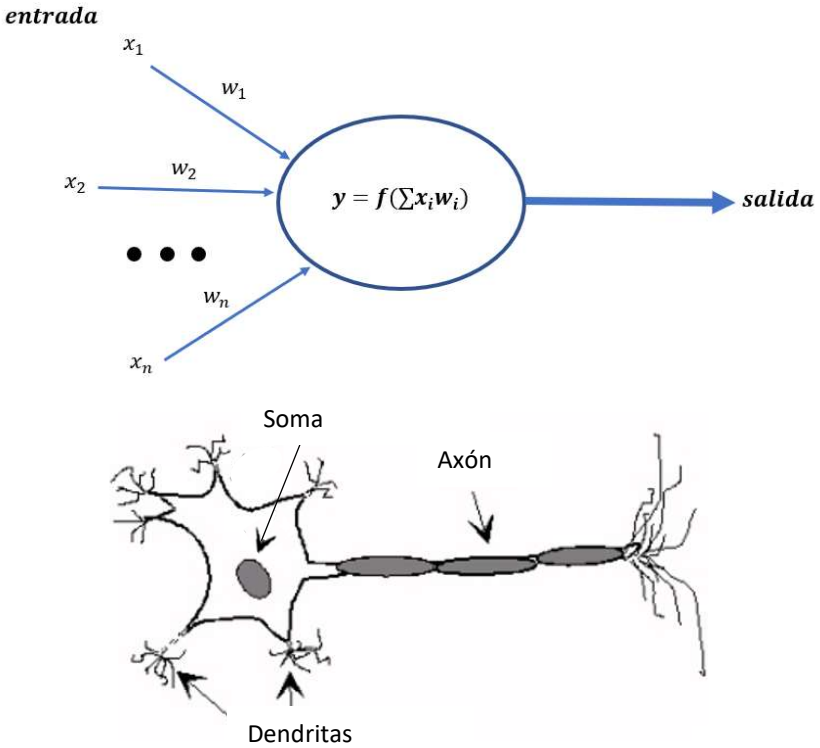


Figura 2. *Comparación de una neurona biológica (abajo, modificada de [19]) y una artificial (arriba, fuente propia).*

Esta primera red solo tenía una fase de operación con pesos, lo que se denomina ***capa***. Su efectividad y capacidad de proceso aumentó cuando, años más tarde, se le aplicaron más capas de forma consecutiva. De esta forma, se le hacía pasar por más fases operacionales de distintos pesos, abstrayendo más la información y consiguiendo una mayor complejidad de las

características que se podían extraer a partir de las entradas a la red, dando lugar al ***perceptrón multicapa*** [20] , [21].

2.2.2. Estructura habitual.

Teniendo como modelo esta primera red, se compusieron las ***redes neuronales clásicas*** (también llamadas ***Fully Connected*** o ***FC***) [22], [23] siguiendo el esquema mostrado en el ejemplo de la Figura 3:

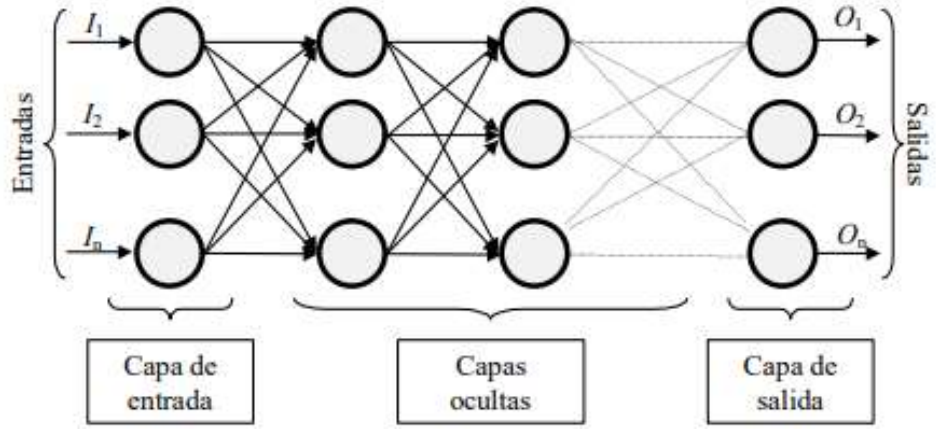


Figura 3. Ejemplo de red FC [24].

Capa de entrada: también denominada *expuesta* o *visible*, permite que ingresen los datos de entrada a la red y pasen a la próxima capa.

Capas ocultas: capas intermedias entre la capa de entrada y la de salida. Se ocupan de la extracción de características de los datos de entrada, aumentando el nivel de abstracción a medida que se incrementa el número de capas ocultas. A esto se le denomina ***profundidad*** de la red**.**

Capa de salida**:** dan significado a los resultados numéricos obtenidos por las capas anteriores para proporcionar la salida de la red.

2.3. Redes neuronales convolucionales (CNN)

2.3.1. Introducción

A medida que se aumentaba el número de capas de las redes clásicas, las limitaciones a las que se enfrentaban, tanto a nivel de hardware como de los algoritmos empleados, no permitían responder a los problemas planteados. El desarrollo de nuevas técnicas, como el *backpropagation*, y los posteriores sistemas con mayor poder de cálculo, como las GPU, lograron superar el *impasse* al que las redes neuronales habían llegado a finales de los años 70.

En este contexto surgen las redes neuronales convolucionales (CNN), que emulan el funcionamiento de la corteza visual primaria del cerebro humano imitando su capacidad de analizar imágenes y han demostrado su eficacia en tareas fundamentales de visión artificial como la clasificación y segmentación de imágenes [25].

Para comprender como se analizan las imágenes, primero se debe entender como son interpretadas de forma digital. Las cámaras fotográficas son capaces de capturar el color que refleja un objeto cuando incide sobre él un haz de luz mediante un **filtro de Bayer**, formado por miles de células fotosensibles que separan un haz luz en la intensidad que contiene de los colores rojo, verde y azul (RGB por sus siglas en ingles). Esta información se almacena en píxeles de 8 bits para la intensidad cada color, con valores de 0 a 255. Si se tratase de una imagen en blanco y negro, los 8 bits denotarían su intensidad en escala de grises. En la Figura 4 se muestra un ejemplo.

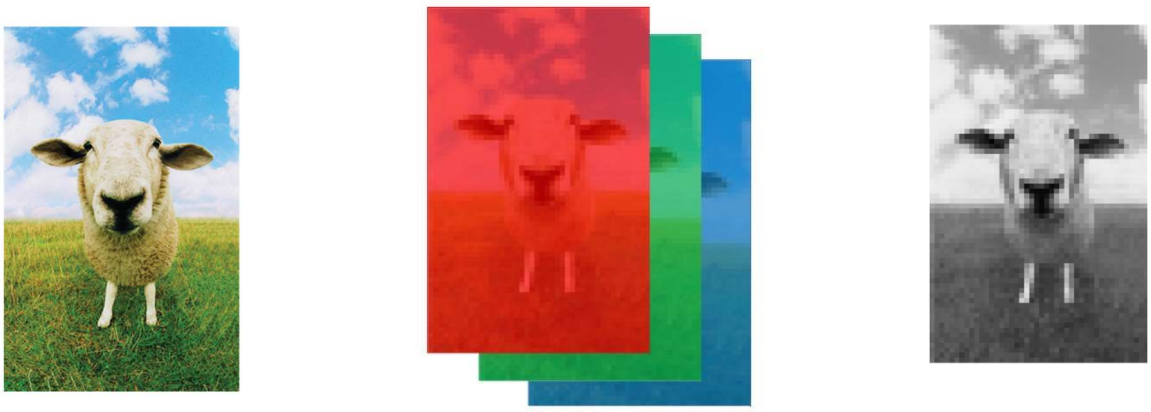


Figura 4. *Imagen natural (izquierda [26]), disgregación en RGB (centro, modificada de [26]) y escala de grises (derecha, modificada de [26]).*

La red recibe una matriz de píxeles si la imagen está compuesta en escala de grises o tres matrices de píxeles si la imagen es RGB. Para poder extraer la información de cada píxel los pesos

actuarán matemáticamente sobre ellos, pero como sus valores son tan altos puede provocar en la red saturaciones internas [27] e inestabilidades [28], pesos excesivos [29], etc. Habitualmente se acotan los datos de entrada a la red para que estos valgan entre 0 y 1. Pero las CNN no solo pueden interpretar imágenes, también pueden recibir datos numéricos o de texto, y, dependiendo del tipo de dato que reciba, el acondicionamiento se lleva de diferentes formas:

- En **imágenes,** que cada píxel puede tener un valor de 0 a 2^8=255, los píxeles son divididos entre 255, el valor máximo, de forma que el valor resultante no sea nunca mayor que 1.

- Con los ***datos numéricos*** es aconsejable obtener la media de toda la base de datos, restársela a cada dato y dividirlo entre la desviación típica. Así se consigue una media de 0 y una desviación típica de 1.

- Para los **datos de texto** típicamente se realiza una codificación de la información en ***One Hot***, que obtiene un vector de ceros y un único uno que representan las palabras, letras, símbolos, etc. que puede interpretar la red. En el caso de un diccionario de 300 palabras, por ejemplo, tendrá un vector de 300 posibles posiciones de entrada codificadas con varios ceros y un uno, siendo la posición del uno el identificador de cada palabra que contiene el diccionario. Cada palabra nueva será una posición más en este vector.

La normalización de los datos facilita el aprendizaje a la red evitando la saturación de gradiente [30], lo que a su vez impide que ocurran saturaciones en la función de activación, explicadas con más detalle en el apartado **2.5**.

A continuación, se explican brevemente las características de las redes convolucionales.

2.3.2. Estructura y capas fundamentales

Las CNN pueden incluir diferentes tipos de capas: la **capa de entrada** de datos a la red, las **capas convolucionales**, las **capas de activación**, las capas ***Pooling*** y las **capas densas**, como se puede ver en el ejemplo de la Figura 5.

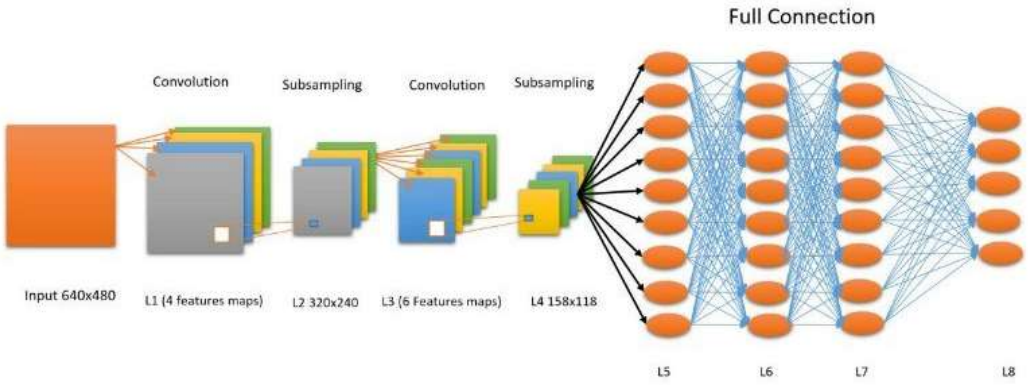


Figura 5. *Ejemplo de una red neuronal convolucional completa [31].*

Las **capas convolucionales** permiten extraer características directamente a partir de la imagen 2D**.** Los pesos de la capa, ordenados en **filtros convolucionales** o ***Kernels*** (matrices), realizan convolucionales sobre los datos que llegan a su entrada, también ordenados en forma matricial. La convolución consiste en que el filtro lea, de la matriz de datos de entrada, grupos de datos cercanos y del mismo tamaño que el filtro, calcule su producto escalar y obtenga un término para rellenar una matriz resultado (la capa convolucionada) con tantos términos como lecturas tenga el filtro por toda la matriz de datos de entrada. Se aporta un ejemplo a continuación representados en las Figuras 6, 7, 8 y 9:

Capa de entrada

0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0

1	0	1
0	1	0
1	0	1

Filtro convolucional

Figura 6. *Capa de entrada y filtro convolucional que opera sobre ella (fuente propia).*

Teniendo una matriz de entrada 7x7 compuesta de ceros y unos (matriz de capa de partida) y un filtro convolucional de 3x3, para obtener el término de la primera fila y columna de la matriz resultante el filtro opera, de forma aislada, con el primer grupo de 3x3 que puede tomar de la matriz de entrada.

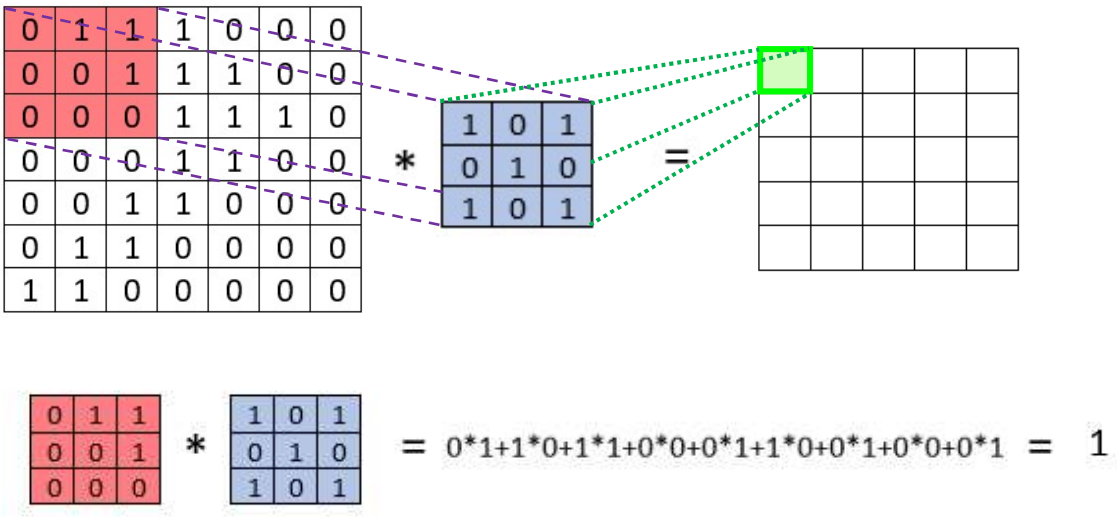


Figura 7. *Cálculo del primer término de la capa convolucionada* (fuente propia).

Para el término de la primera fila y segunda columna el grupo será el de inmediatamente a la derecha del anterior.

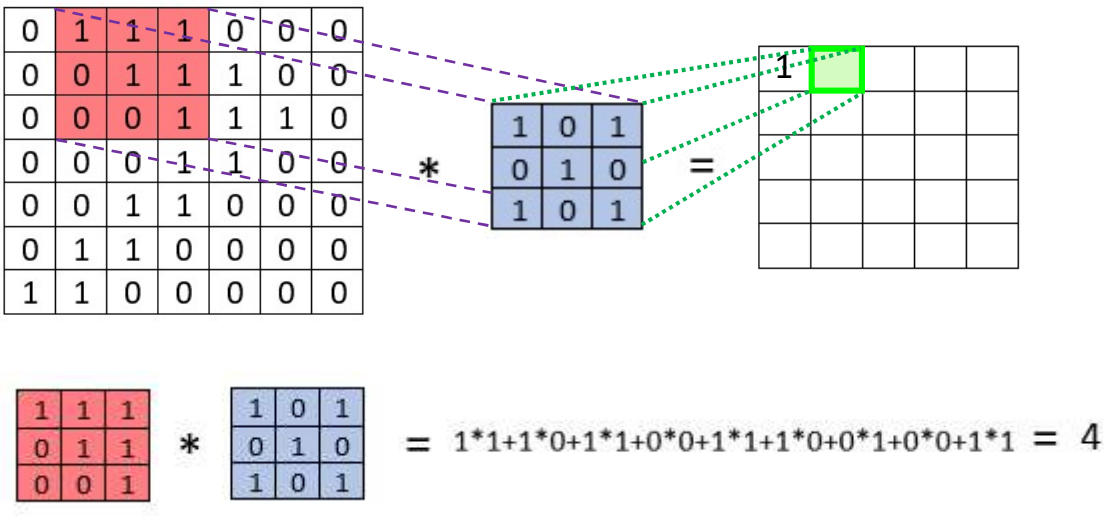


Figura 8. *Cálculo del segundo término de la capa convolucionada (fuente propia).*

Como en las filas se pueden hacer 5 desplazamientos de 3x3, la matriz resultante tendrá 5 columnas. De la misma forma, como en las columnas se pueden hacer otros 5 desplazamientos

de 3x3, el número de columnas de la matriz resultante será 5. Por tanto, la matriz resultante de la operación convolucional de una matriz 7x7 por un filtro de 3x3 es una matriz de 5x5.

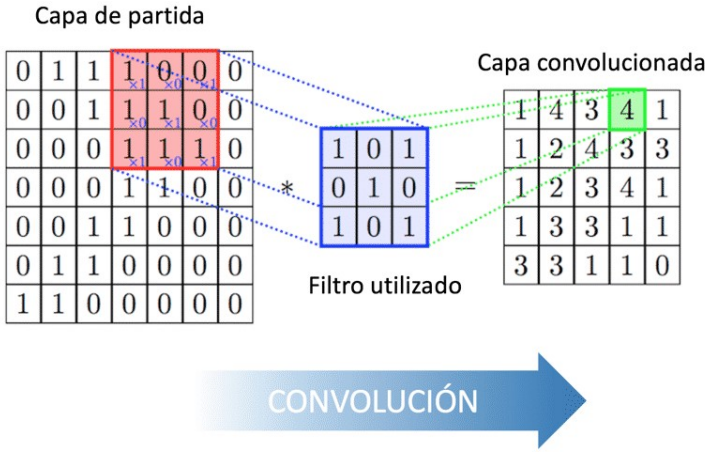


Figura 9. *Capa convolucionada resultado de la convolución con un filtro de 3x3 [32].*

No obstante, este es el resultado que se obtiene de un único filtro. Las capas convolucionales están compuestas de muchos filtros y forman la ***profundidad*** de la capa. Con cada filtro se busca una característica y el total de las características que se buscan forma un **mapeado de características (*feature mapping*),** como se observa en el ejemplo de la Figura 10.

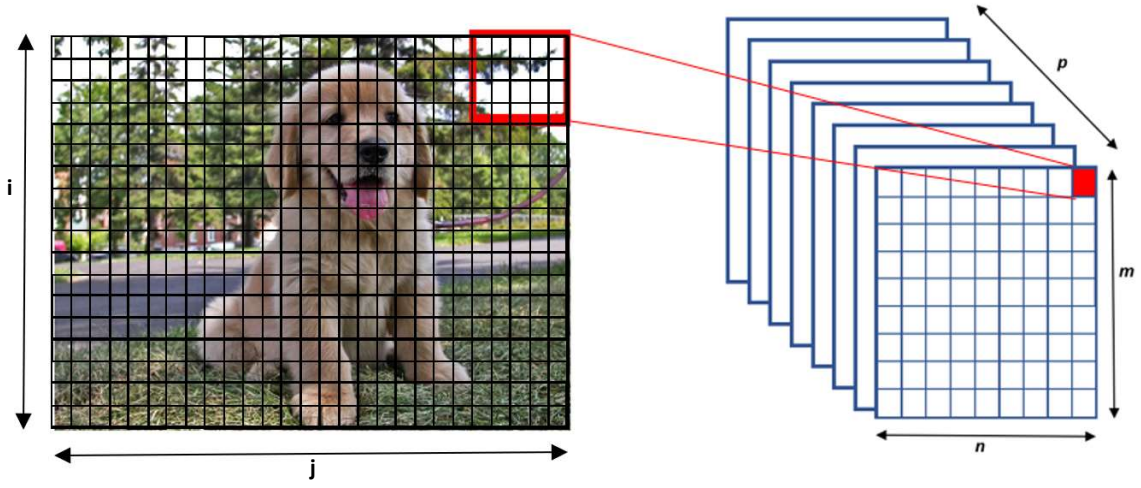


Figura 10. *Mapa de características (fuente propia) obtenidas por una capa convolucional de mxn con p filtros (profundidad) sobre una imagen [33] de ixj píxeles.*

Las primeras capas convolucionales comienzan extrayendo características simples como bordes, esquinas, etc. A medida que nos adentramos por las capas, la búsqueda de información se va

abstrayendo de forma que, lo que las capas pueden reconocer, es cada vez más complejo. Por ejemplo, empieza reconociendo líneas, luego formas y acaba reconociendo objetos, como se aprecia con los filtros de ejemplo en la Figura 11.

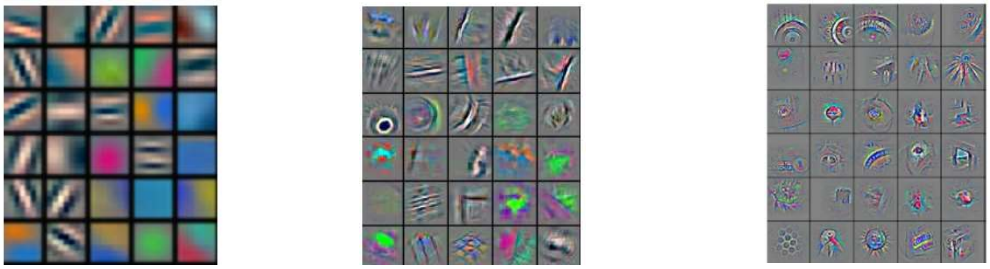


Figura 11. *Ejemplo de algunos de los filtros [34] que componen las capas convolucionales de la red neuronal AlexNet [35]].*

A las salidas de las capas convolucionales es habitual incluir **capas de activación**, que se activan en caso de cumplir una determinada condición. Estas capas de activación operan vectorialmente con todos los elementos de la matriz de resultados que se ha obtenido en la capa precedente de una forma muy rápida. A esto se le llama ***element wise multiplication*** o **producto de *Hadamard*.** Incluir las capas de activación aporta, entre otros beneficios [36] , [37], aumentar la eficiencia de la red sin ralentizarla [35].

La especialización de las capas convolucionales en reconocimientos más abstractos a medida que nos adentramos en la red requiere hacer pasar la información por más filtros y cada vez más pequeños, generándose un volumen de cálculo altísimo. Para evitar el colapso de la red, se aplican las capas ***Pooling***, las cuales reducen el volumen de datos seleccionando la información más destacable del mapa de características obtenido por la capa convolucional precedente (por ejemplo, la capa ***Max Pooling***, mostrada en la Figura 12). El uso de estas capas debe ser muy gradual, para no eliminar demasiada información y perder datos importantes. Estas capas suelen utilizarse para extraer, de un conjunto de datos, cuáles son los más destacados

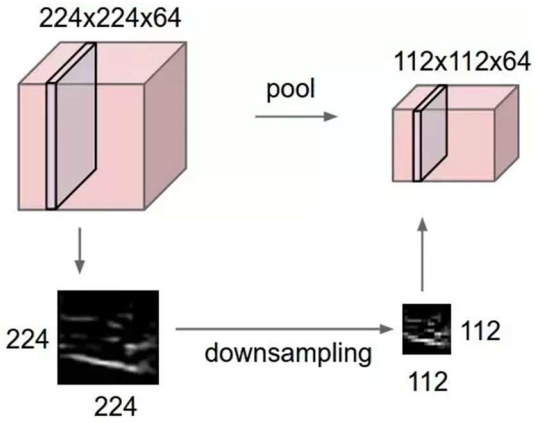


Figura 12. *Ejemplo de operación Max-Pooling que reduce los datos que recibe a la mitad [38].*

Finalmente, antes de salir de la red, es habitual incluir una ***capa densa,*** que está compuesta por neuronas clásicas. La capa densa relaciona todas las características extraídas y realiza una clasificación de los resultados haciendo uso de una función de activación (más información en el apartado **2.5**). Para que la capa densa pueda comprender el resultado de las capas convolucionales, los resultados multidimensionales de estas se convierten a un vector (unidimensional) utilizando una capa intermedia ***flatten***.

2.4. Entrenamiento/aprendizaje.

Las redes neuronales deben aprender a llevar a cabo la labor que queremos que realicen [39], [40]. Esto se consigue sometiéndola a un entrenamiento, con el cual la red ajustará los pesos con los que consigue extraer las características para realizar la labor requerida. El entrenamiento consiste en presentarle a la red iteraciones de datos (llamadas **épocas**) con las que la red aprende mediante prueba y error. El parámetro de ajuste ***learning rate*** (o **tasa de aprendizaje**) estipula en qué medida se modifican los pesos respecto al error cometido (que se calcula con una función determinada para cada tipo de problema y situación) en cada época. El *learning rate* puede mantener durante todo el entrenamiento el mismo valor o irse modificando de forma dinámica cómo se muestra por ejemplo en la Figura 13. Cuando los pesos se estabilizan, el proceso de aprendizaje ha terminado y significa que ya no comete error o éste es muy pequeño.

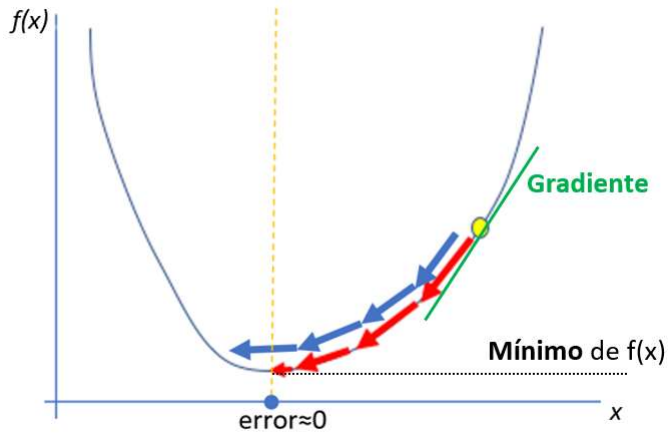


Figura 13. Momentum para un Learning Rate dinámico (rojo) vs constante (azul) sobre función de pérdidas f(x) que dicta como se evalúan los errores de salida en un problema de tipo regresión (modificado de [41]).

La forma en la que se implementa el *learning rate* se lleva a cabo mediante **optimizadores de aprendizaje**. Actualmente el optimizador más utilizado es el llamado **ADAM,** que adopta todo lo favorable de optimizadores anteriores que se muestran en la Tabla 2.

Optimizador	Características
GD	Realiza la modificación de los pesos haciendo *backpropagation* después de haber calculado el error en toda la *DataSet*, con un *learning rate* constante.
SGD	La modificación de los pesos la realiza tras entrenar con cada dato del *DataSet*.
Mini-BGD	Actualiza los pesos después de entrenar con una fracción de datos del *DataSet*.
SGD con *momentun*	Añade el concepto de *momentun* y con él la variación de gradiente depende de todos sus valores anteriores.
AdaGrad	Introduce por primera vez el *learning rate* dinámico que calcula con la suma cuadrática de todos los gradientes anteriores.

AdaDelta	Corrige, de AdaGrad, el cálculo para modificar el *learning rate* y lo hace mediante la suma cuadrática de las últimas variaciones en los gradientes.
RMSProp	Corrige también el AdaGrad calculando el *learning rate* con la media de la suma cuadrática de los últimos gradientes.

Tabla 2. *Evolución de los optimizadores de entrenamiento.*

ADAM propaga el error desde el final al inicio de la red (lo que se denomina *backpropagation*) y modifica los pesos de las neuronas que contribuyan al error de la salida. En cada época se le presenta una fracción de la base de datos de entrenamiento y con ella calcula el error que comete en la función de cálculo de error (función de pérdidas o función de coste, dependiendo del tipo de entrenamiento, explicado en el punto siguiente). Gracias a un parámetro llamado *momentum*, tiene en cuenta las últimas variaciones en el gradiente y su situación actual, con lo que modifica el *learning rate* de forma gradual hasta que, idealmente, llegue a valer 0 y consiga que el error cometido también sea 0. El único inconveniente de este optimizador es que no funciona bien con bases de datos pequeñas, pero en grandes *DataSets* su baja necesidad de procesamiento no requiere mucha carga computacional, lo que acelera el proceso de entrenamiento como se muestra en la Figura 14, comparándole con distintos optimizadores.

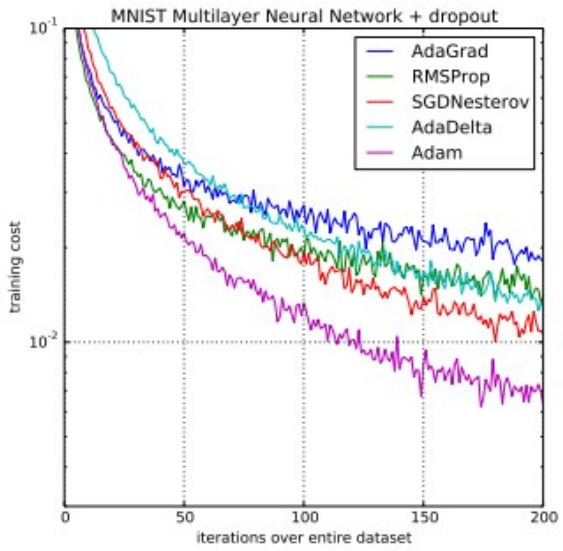


Figura 14. *Comparación de la carga computacional de distintos optimizadores durante el entrenamiento sobre la base de datos MNIST [42].*

Los principales métodos de aprendizaje de redes neuronales pueden dividirse en dos grandes grupos dependiendo de si requieren, o no, datos etiquetados de entrada y salida: el **aprendizaje supervisado** y el **aprendizaje no supervisado**.

2.4.1. Aprendizaje supervisado

Este método compara la salida actual de la red con la salida deseada. Para ello los datos tienen que estar etiquetados, tanto los de entrada como los de salida, de forma que para cada entrada determinada se conozca cuál debe ser su salida correspondiente. Si la red no acierta durante una época, modifica los pesos con el objetivo de acertar eventualmente.

Existen a su vez tres formas de llevarlo a cabo:

-Aprendizaje por corrección de error: mediante lo que se denomina **función de pérdidas**, compara la salida obtenida con la que debería haber obtenido y calcula el error cometido mediante la diferencia entre ambas. La función de pérdidas que se elige depende del objetivo del aprendizaje. Las principales funciones de pérdidas se explican en el apartado **2.6**.

-Aprendizaje por refuerzo: a diferencia del aprendizaje por corrección del error, que calcula el error como la diferencia entre la salida obtenida y la deseada, en este caso, se devuelve una señal de refuerzo que indica si se ha obtenido el resultado deseado o no (éxito = +1 o fracaso = -1). Este proceso es más lento que el anterior y ajusta los pesos basándose en un mecanismo de probabilidades.

-Aprendizaje estocástico: el resultado de la red depende acciones predecibles y de elementos aleatorios. A partir de un objetivo deseado y de distribuciones de probabilidad, se realizan cambios aleatorios en los pesos para evaluar su efecto en el resultado. Se suele utilizar para hacer una analogía en términos termodinámicos de un sólido físico en un estado energético y el efecto que le producen variaciones aleatorias.

2.4.2. Aprendizaje no supervisado

Para este aprendizaje la red no recibe información de si la salida que obtiene es correcta o no, por lo que no requiere contar con información etiquetada acerca de cuál debe ser la salida para cada entrada. En lugar de eso, busca características, correlaciones o categorías que se puedan establecer entre los datos de entrada. Para calcular el error utiliza una **función de coste** [43] en vez de una función de pérdidas mencionada en el apartado anterior. La salida resultante por el aprendizaje no supervisado puede ser:

- La **familiaridad** o **similitud** entre la información que se le presenta a la entrada y las informaciones pasadas que se le han presentado hasta entonces (***algoritmo de aprendizaje hebbiano***);

- Una **clasificación** de los datos de entrada, ajustándose a los patrones que previamente tenía determinados para cada categoría y, en el caso de que no coincida con ninguna, los pesos se ajustarán para reconocer esa nueva clase (***algoritmo de aprendizaje competitivo y comparativo***).

2.5. Funciones de activación

Las funciones de activación contribuyen a la **interpretación** de los resultados y a la **versatilidad** de la red. Las salidas de las neuronas de la capa previa desembocan en la función y, mediante una relación matemática específica para cada una de ellas se obtiene una salida que, dependiendo del tipo de función que sea, cumple con un objetivo determinado.

A continuación, se describen las funciones de activación que se emplean con más frecuencia:

La primera función de activación es la función escalón. Esta función determina que, si la suma de las operaciones con los pesos superaba un cierto umbral, el resultado de la función es 1 y, en caso contrario, es 0, como puede observarse en la Figura 15.

$$f(x) = \begin{cases} 0 & \text{sí } x < 0 \\ 1 & \text{sí } x \geq 0 \end{cases}$$

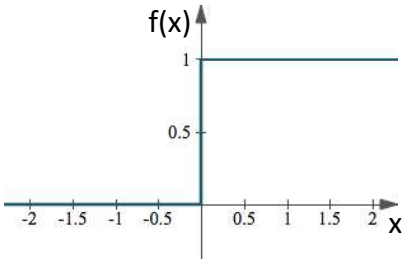


Figura 15. *Fórmula y representación gráfica de una función escalón (modificada de [44]).*

Esta primera función presenta muchas limitaciones de representación y solo es válida para clasificaciones binarias. Por ello, han surgido otras alternativas para la activación.

Actualmente, se diferencian en dos tipos de funciones: **funciones de activación lineales** y **funciones de activación no lineales.**

Las ***funciones lineales*** obtienen un resultado de salida directamente proporcional a su entrada (expresión matemática en Figura 16). A diferencia de la función de escalón, se pueden obtener más de dos resultados lo que le hace más versátil, mas no tienen muchas aplicaciones y solo se utilizan en aplicaciones de optimización [45] o de regresión [46]. Las **regresiones** lineales tratan de predecir el valor de una variable numérica a partir de variables independientes (por ejemplo, predecir el precio de una vivienda dependiendo de la zona en la que se encuentre). Se emplean cuando el resultado que se le requiere a la red es numérico o cuantitativo.

$$linear(x) = m * x$$

siendo ***m*** su pendiente

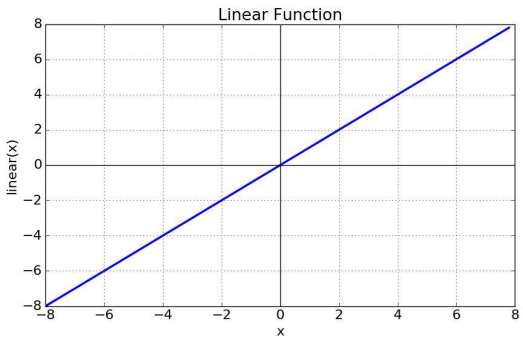


Figura 16. *Fórmula y representación gráfica de la función de activación lineal identidad con pendiente=1 [47].*

El desarrollo de las ***funciones no lineales*** brindó la posibilidad de conseguir representaciones mucho más complejas.

La función de activación ***Sigmoidal*** es muy conocida para la clasificación binaria. A diferencia de la función escalón, el resultado es un valor comprendido entre 0 y 1 que predice la **probabilidad** de ser una de las posibilidades (valor más cerca del 0) o la otra (más cerca del 1).

Una gran desventaja de esta función, a parte de su alto coste computacional, es que debido a que el salto de 0 a 1 es muy pequeño (ver Figura 17), durante el entrenamiento, cuando el resultado de la red se acerca a sus extremos (lo que le estaría propiciando seguir mejorando su precisión de predicción) el error que propaga a sus pesos es prácticamente nulo y la red deja de aprender, a lo que se le llama **desvanecimiento de gradiente** [48]. Este problema lo trata de solucionar la tangente hiperbólica empleando un salto mayor, de -1 a 1.

$$S(x) = \frac{1}{1 + e^{-x}}$$

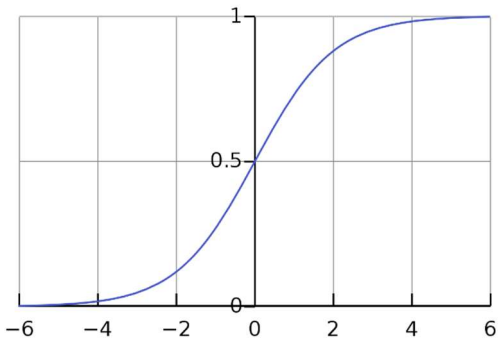


Figura 17. *Fórmula y representación gráfica de la función de activación Sigmoidal [49].*

Una extensión de la función Sigmoidal es la función **Softmax**, mostrada en la Figura 18, que se utiliza para la clasificación multiclase. Para cada clase se obtiene una probabilidad normalizada, de esta forma la suma de todas las probabilidades dadas es igual a 1 (100%). Devuelve así, de entre todas las clases, cual tiene mayor probabilidad de ser.

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

j=0,1,2…k

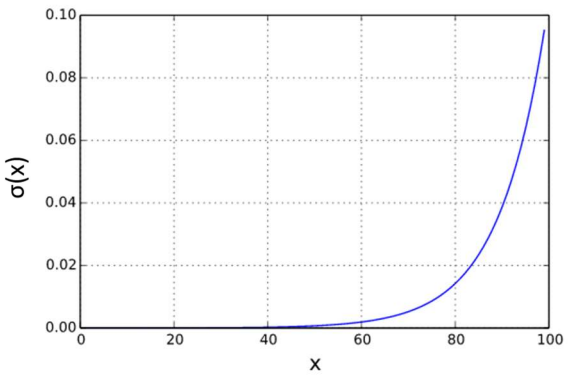


Figura 18. *Fórmula y ejemplo de representación gráfica de la función de activación Softmax [50].*

La ***Unidad Lineal Rectificada***, o ***ReLU***, es muy utilizada en las capas de activación intermedias. Su forma es la de una función lineal en la zona positiva y cero para valores negativos (representación y fórmula Figura 19). Es invariable a la escala, pues cualquier modificación en los parámetros de la red solo supondría un cambio en su pendiente. Gracias a esto no se producen **saturaciones**, uno de los grandes problemas en las redes neuronales, responsables del desvanecimiento de gradiente durante el entrenamiento. Es decir, evita llegar al punto en el que la red deja de aprender durante el entrenamiento, cosa que si puede ocurrir con otras funciones como la Sigmoidal.

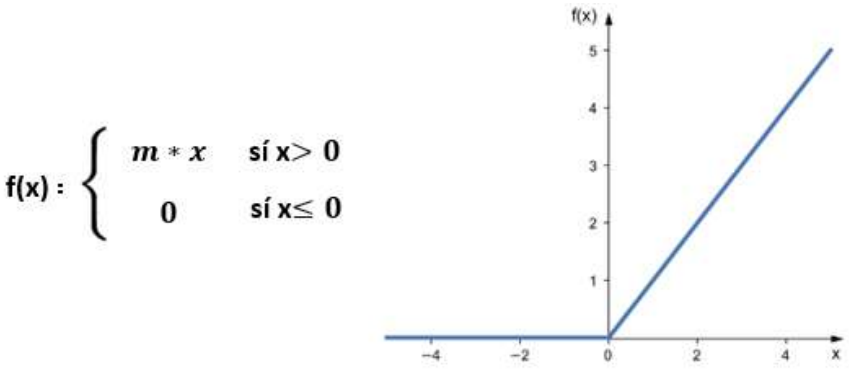


Figura 19. *Fórmula y ejemplo de representación gráfica de función de activación ReLU con una pendiente=1 [51].*

La ***Tangente hiperbólica*** (tanh) es una función clásica de estructura similar a la función sigmoidal. Siguiendo la Figura 20, varía entre 1 y -1, por lo que el salto es mayor que en la función sigmoidal. El tener un rango más grande para cuantificar el error durante el entrenamiento la hace más eficiente, aunque sigue provocando saturaciones. Además, su alto coste computacional ralentiza en gran medida el proceso ya que se utilizaba en las capas de activación intermedias. Por todo ello, la aparición de la función ReLU la desbancó.

$$\mathbf{f(x) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}}$$

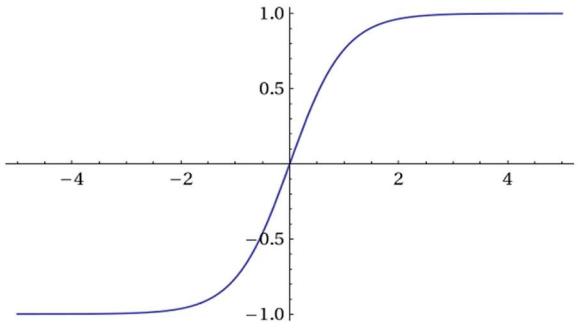


Figura 20. *Fórmula y ejemplo de representación gráfica de la función de activación tanh [52].*

2.6. Funciones de pérdidas

Las funciones de activación que se aplican al final de la red para interpretar el resultado van de la mano con las funciones de pérdidas que se utilizan durante el entrenamiento. La función de pérdidas que se utilice debe ser capaz de adaptarse al problema en situación y apreciar todos los aspectos del modelo (red) para poder cuantificar los errores que cometa.

En el caso de los problemas de clasificación, se calcula el error que se comete en las distribuciones de probabilidad (*softmax, sigmoid* y tanh) utilizando **entropía cruzada** (*cross entropy*): si la **entropía** es la incertidumbre de **una única** distribución de probabilidad, la **entropía cruzada** contempla la incertidumbre de **varias** distribuciones de probabilidad.

Por ejemplo, siguiendo la Figura 21: Para una clasificación multiclase, *softmax* calcularía un vector en el que cada término se corresponde a la probabilidad de ser cada clase, siendo la suma de todas las probabilidades obtenidas igual a 1 (100%). El error que se ha cometido se consigue comparando el resultado de la función *softmax* y el de un vector *One Hot*, siendo todos sus términos ceros salvo un uno que se corresponde con la clase que debería haberse obtenido.

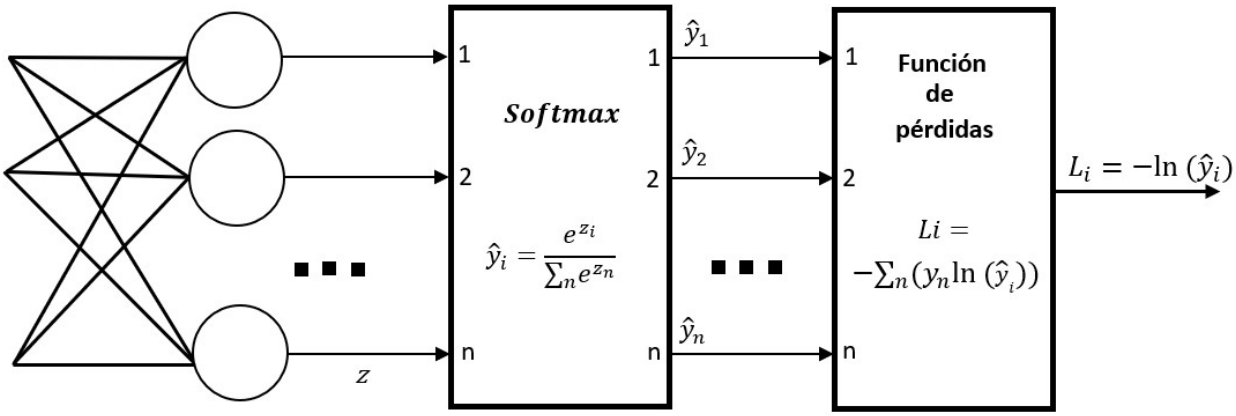


Figura 21. *Representación gráfica de las salidas calculadas $\hat{y}_i$ por una función softmax a partir del resultado obtenido por la última capa densa y el error L_i calculado con el que se realiza el backpropagation (fuente propia).*

De esta forma, la única probabilidad que permanece es la probabilidad de la clase que debería haberse obtenido y el error se contempla como el logaritmo de esta probabilidad. Como se aprecia en la Figura 22, cuanto menor sea el porcentaje obtenido, mayor es el error cometido. Todo esto comprende a lo conoce como ***categorical crossentropy***, empleado en entrenamientos de clasificación multiclase.

$$L_i = -\ln(\hat{y}_i)$$

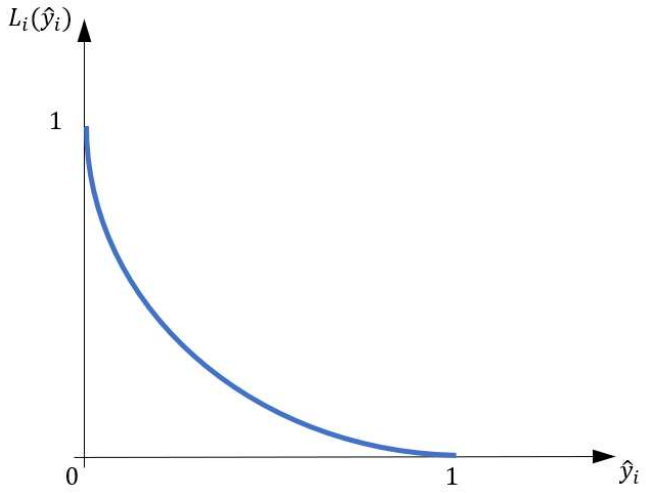


Figura 22. *Fórmula y gráfica de función de pérdidas categorical crossentropy, visualizando el error sobre la predicción de una clase i (fuente propia).*

De igual manera, en una clasificación binaria el resultado de la probabilidad de una clase es la probabilidad opuesta de la otra, siendo esta la ***binary crossentropy***. Ver Figura 23.

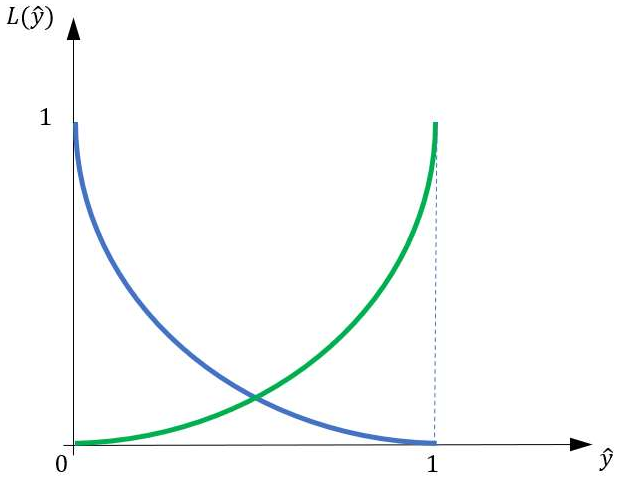


Figura 23. *Gráfica de función de pérdidas binary crossentropy, visualizando el error sobre la predicción de una clase (azul) y la predicción sobre la alternativa (verde) (fuente propia).*

Para los problemas de regresión, que como el resultado obtienen valor numérico, el método más común es el uso de la función de pérdidas ***MSE (Mean Square Error)***. Esta función calcula la suma del cuadrado de las diferencias de los valores obtenidos frente a los que tendría que haber conseguido y lo divide entre el número de resultados para obtener la media, como se muestra en la Figura 24. El cuadrado sirve para que, si la diferencia es muy grande, el error se vuelve aún más grande.

$$\boldsymbol{MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

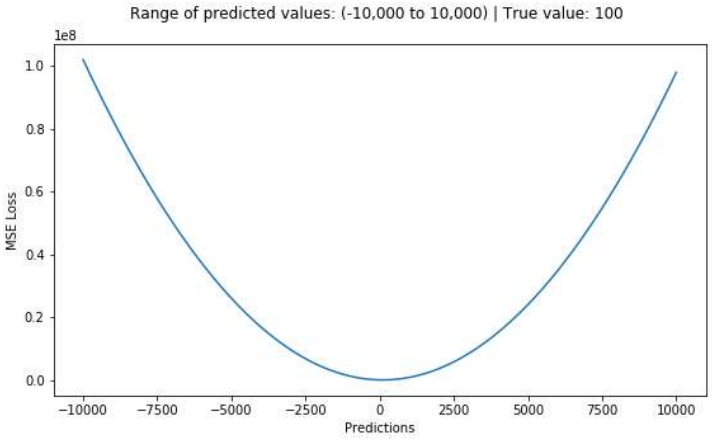


Figura 24. *Fórmula y gráfica de función de pérdidas MSE para una regresión cuando el valor a obtener es 100 y los resultados pueden oscilar desde -10 mil a 10 mil [53].*

2.7. Arquitecturas típicas

El desarrollo de los modelos que han marcado el estado del arte en cada momento del *Deep Learning* fueron producto del desafío anual ***ImageNet Large Scale Visual Recognition (ILSVRC)***, donde trabajos de todo el mundo compiten por conseguir los mejores resultados sobre la base de datos *ImageNet*, que comprende más de 14 millones de imágenes y más de 20 mil categorías etiquetadas.

AlexNet [35] fue la primera arquitectura que ganó el reto estando basada en CNNs, en el año 2012. Marcó un antes y un después y, desde entonces, todas las arquitecturas ganadoras estuvieron basabas en CNNs.

Está compuesta por 5 capas convolucionales y 3 capas FC, esquematizado en la Figura 25. La primera capa convolucional filtra la imagen de entrada, de 224x224x3, con 96 filtros de 11x11x3 resultando un tensor de 55x55x96. Seguidamente, pasa por otra capa convolucional más reducida y profunda que la anterior, de 27x27x256, con filtros de 5x5x96. A ambas capas se les reduce la dimensionalidad de sus resultados con una capa intermedia de *max pooling*, adentrándose así más al detalle de lo que busca. Las tres capas convolucionales siguientes reciben directamente los resultados de la capa anterior. El resultado se hace pasar por un último max pooling y lo recibe la FC.

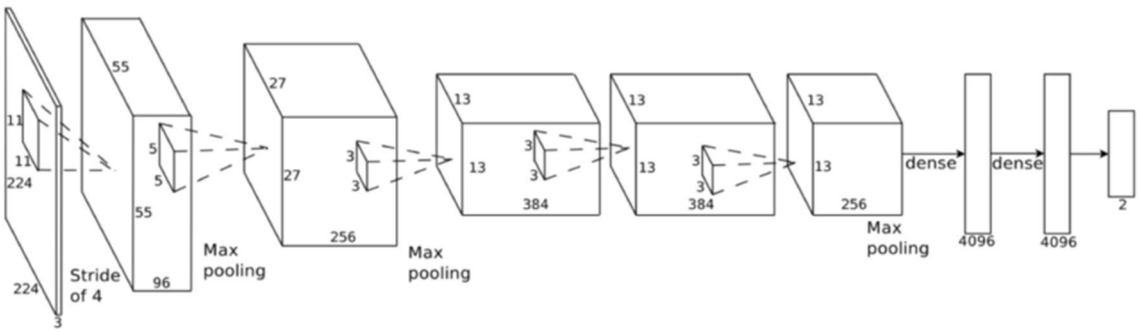


Figura 25. Representación de la arquitectura de la red neuronal AlexNet [54].

Aunque no se muestre en la figura, esta red fue de las primeras que utilizó ReLUs a la salida de sus capas convolucionales, lo que aceleró el entrenamiento hasta hacerlo 6 veces más rápido que la misma red utilizando tanh intermedias (utilizadas comúnmente hasta entonces [35], en

la Figura 26 se muestra un ejemplo comparativo de ambas funciones), sin perder precisión y reduciendo el error de su salida.

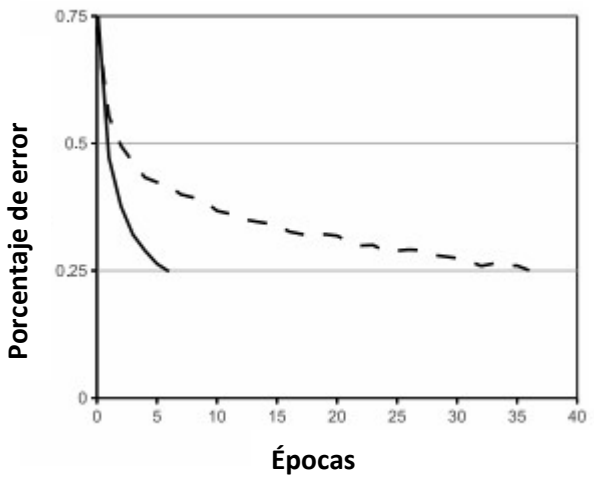


Figura 26. *Comparación del tiempo que tarda el entrenamiento de una red de 4 capas convolucionales en alcanzar un error del 0.25% con ReLUs intermedias (línea contínua) y con tanh (línea rayada) sobre la base de datos CIFAR-10 [35].*

Para el entrenamiento, además, utilizó dos técnicas muy extendidas actualmente: el ***data augmentation***, que altera levemente las imágenes de la base de datos de forma que la red las vea como si fuesen imágenes nuevas, consiguiendo así más datos para entrenar; y el ***dropout***, que desconecta neuronas aleatorias, lo que favorece la generalización de lo que busca la red, la no dependencia entre las neuronas y evita que se produzca sobreentrenamiento [55].

VGGNet [56] fue una de las arquitecturas finalistas en el año 2014 y, aunque no ganó, ha sido muy influyente. Con una estructura similar a AlexNet, como se ve en la Figura 27, consiguió mejorar los resultados aumentando su profundidad. Es un modelo simple, eficiente e intuitivo que permite entender lo que ocurre en cada sección. Todo esto lo convierte en un referente educativo para familiarizarse con las CNNs.

Su fundamental diferencia respecto a AlexNet está en el tamaño reducido de sus filtros, de 3x3. Esta innovación aporta múltiples mejoras. En vez de realizar una única convolución grande con una ReLU a su salida, emplea más convoluciones y más pequeñas, con ReLUs intermedias que hacen el resultado conjunto mucho más discriminativo [56]. Además, esta aplicación de más convoluciones pequeñas en lugar de una grande reduce considerablemente el número de

parámetros de las capas. Por ejemplo, si se compara una convolución grande de 7x7 con las tres pequeñas de 3x3 que servirían para hacer su equivalente, las pequeñas tienen en total un 81% menos de parámetros que la grande. Así la red se dota de mayor profundidad con un menor número de pesos.

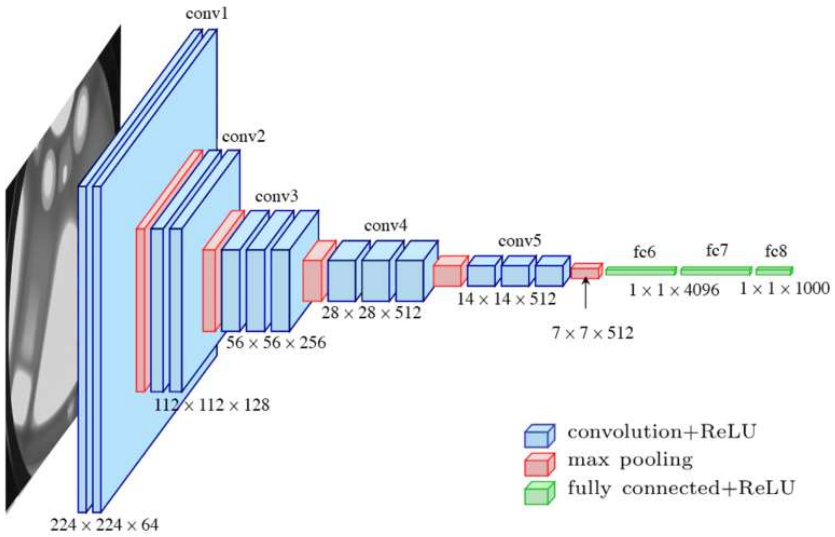


Figura 27. *Representación de la arquitectura red neuronal VGGNet, modelo VGG19 [57].*

En vez de *VGGNet*, en el año 2014 ganó la red *GoogLeNet* [58]. Lo interesante de esta red fue la aplicación de una estructura que se había desarrollado para la red neuronal ***Inception*** [58], que pretendía reducir la ineficiencia computacional de todas las redes anteriores. *Inception* estaba basada en el uso de ***módulos Inception***, bloques que concatenan información de muchas capas que actúan simultáneamente, enriqueciendo la salida. En su primera versión (mostrada en la Figura 28), los datos que recibía el *módulo Inception* pasaban por separado por distintas convoluciones (cada una buscando algo en concreto) y por un *max pooling* que destacaba lo más relevante. Combinando toda esta información de los datos de entrada a la salida en vez de analizarla secuencialmente, *Inception* logra que el análisis sea sobre el conjunto.

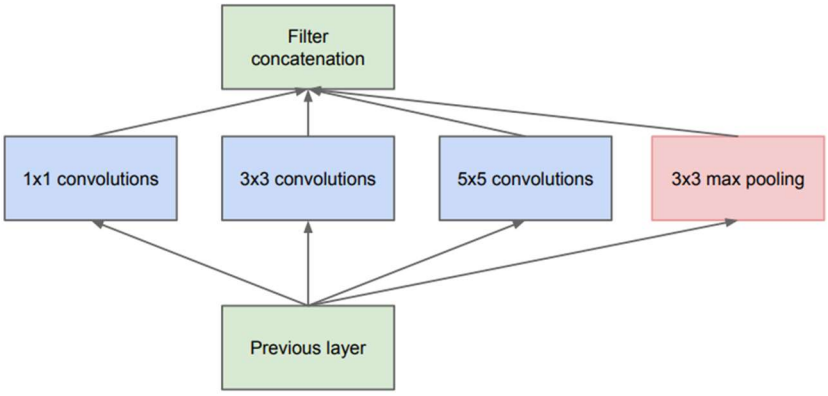


Figura 28. *Esquema del módulo Inception V1 [59].*

El único inconveniente de la red *Inception* fue la cantidad tan alta de operaciones que tenía que realizar, asique en siguientes versiones, como la ***Inception V3*** [60], se embeben las operaciones en las convoluciones añadiendo filtros de 1x1 previos a cada convolución, así como ReLUs intermedias, teniendo así el filtro una doble funcionalidad (esquema del bloque en la Figura 29). Estos módulos actualizados fueron los empleados en la red ganadora *GoogLeNet.*

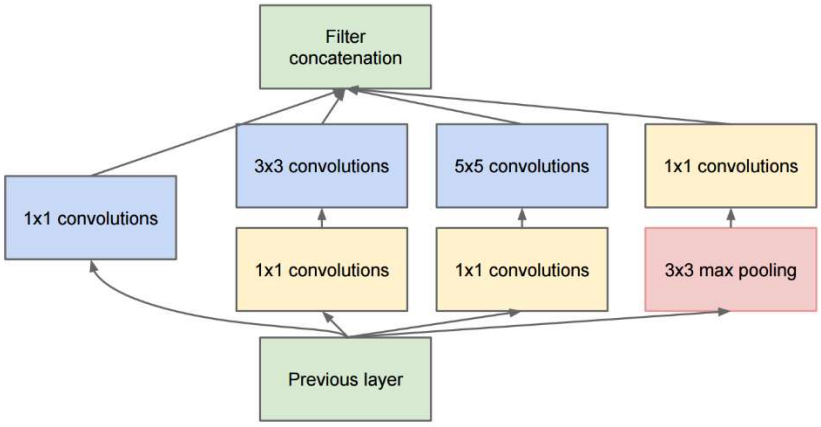


Figura 29. *Esquema del módulo Inception V3, usado en la GoogLeNet [61].*

En 2015 la red ganadora fue una CNN basada en *bloques residuales*, la ***ResNet*** [62]. Consiguió unos resultados increíbles y además consiguió aumentar la profundidad de su red hasta las ***152 capas***, algo nunca visto debido a que en las redes anteriores aumentar la profundidad no significaba siempre conseguir mejores resultados. Introdujo los ***bloques residuales***, bloques que recirculan la información **sumándola** de la entrada a la salida para que no se pierda información por el camino (esquematizada en la Figura 30). A medida que avanzan en profundidad disminuye

el error que se comete y por eso consigue aumentar la profundidad de forma tan efectiva, a diferencia de lo que pasaba con redes anteriores.

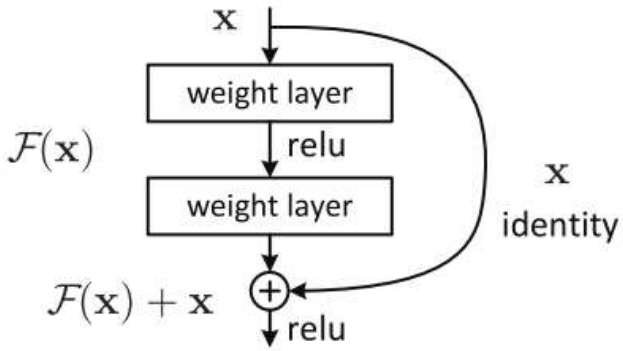


Figura 30. *Esquema del bloque residual [63].*

Lo mejor de las redes *Inception* y *ResNet* se unió para desarrollar la ***Inception Resnet*** [64], mejorando los resultados anteriores aplicando la compleja estructura mostrada en la Figura 31.

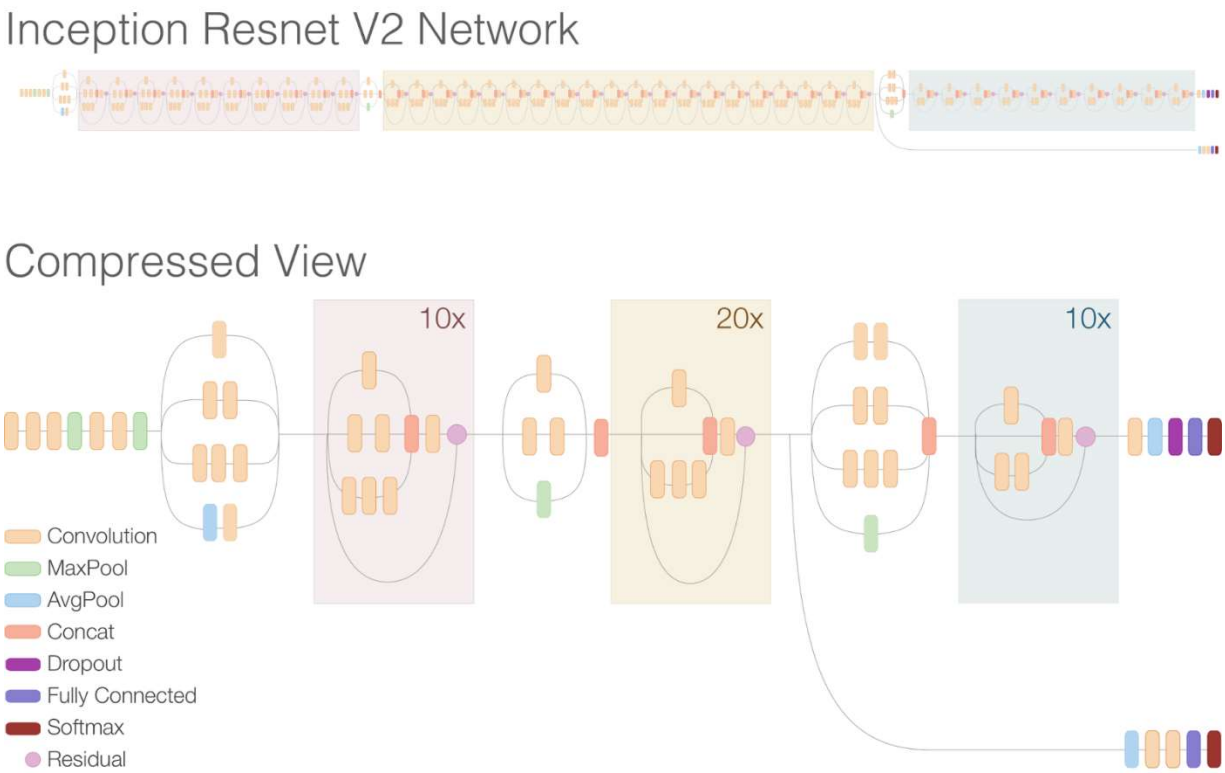


Figura 31. *Representación de la arquitectura red neuronal Inception ResNet V2 [65].*

2.8. Clasificador en cascada para la detección de caras

Para finalizar este capítulo, se describen los fundamentos teóricos del clasificador en cascada *Haar-cascade Classifier* [66] elegido para la detección de caras.

La **detección de objetos** en *machine learning* consiste en identificar objetos determinados que se encuentran dentro una imagen. Esto puede llevarse a cabo fijándose en el color del objeto [67] pero ante cualquier cambio de luz el resultado de la clasificación puede cambiar drásticamente. Un método más sofisticado consiste en fijarse en características propias del objeto a detectar. Este método es el que utiliza el *Haar-cascade Classifier*, un clasificador robusto y de alta velocidad basado en *machine learning* desarrollado por Paul Viola y Michael Jones [68].

Este clasificador recorre toda la imagen analizando, por **secciones rectangulares**, una primera característica. Si la encuentra, se mantiene en esa sección buscando en ella y en secciones vecinas la siguiente característica a detectar, de forma que hace una búsqueda secuencial de todas las características a detectar, siguiendo un **árbol de decisiones** como el que se muestra en la Figura 32. Si falla una de esas detecciones, el clasificador reinicia la búsqueda y prosigue en una nueva sección de la imagen buscando nuevamente la primera característica.

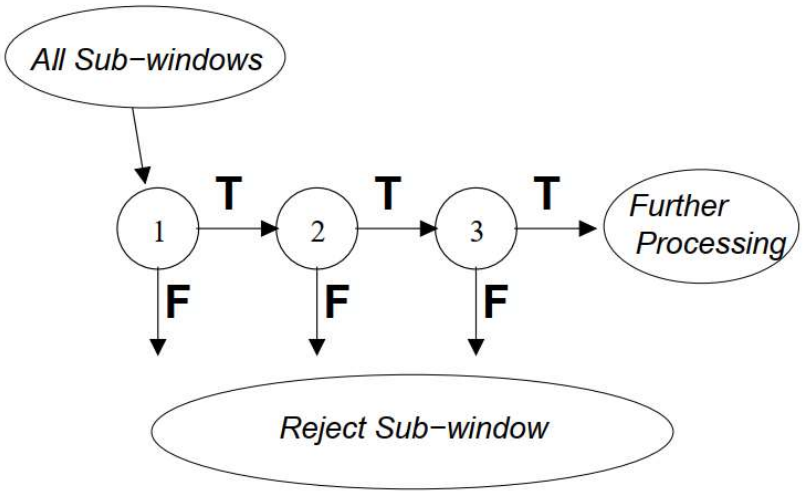


Figura 32. *Gráfico de las fases del árbol de decisiones del cascade classifier [66].*

Cada fase del árbol de decisiones debe ser muy precisa y cometer pocos falsos negativos, es decir, que no confunda una característica presente como no presente y no detecte el objeto por

este error. Para evitarlo, la selección de características a detectar se lleva a cabo mediante el algoritmo de aprendizaje AdaBoost [69], que para la detección de caras elige, de entre 180,000 potenciales características, solo 38, en las que se centrará con gran precisión. En la Figura 33 se muestra la primera y la segunda característica que ha determinado AdaBoost para fijarse: en el sombreado de los ojos y en el caballete nasal.

Figura 33. *Primeras 2 características de la secuencia de detección de caras del haar-cascade [66].*

Todos estos rasgos hacen que este clasificador sea uno de los más rápidos que se hayan desarrollado y por eso se utiliza para aplicaciones en tiempo real [70].

Capítulo 3

DESARROLLO

3.1. Planteamiento / Introducción.

Cómo se introdujo al principio del documento, el objetivo del presente trabajo es el diseño, implementación y evaluación de un sistema para la detección automática de mascarillas faciales y el análisis de la distancia interpersonal, así como la densidad de persona, a partir de secuencias de imágenes.

El sistema desarrollado se rige por el esquema que se muestra en la Figura 34.

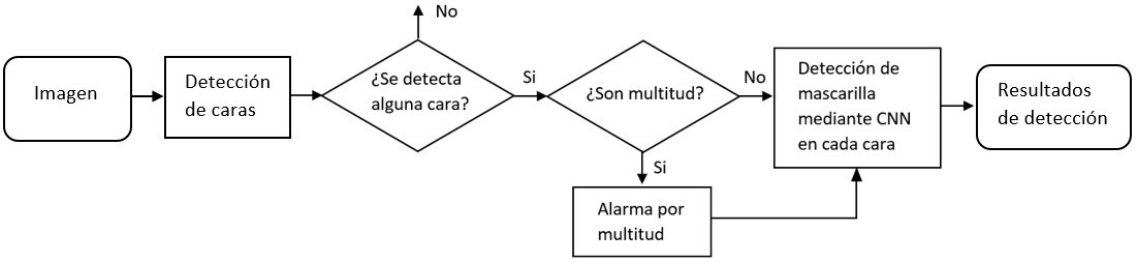


Figura 34. *Esquema funcionamiento del sistema implementado (fuente propia).*

3.2. Entorno de desarrollo

El trabajo se ha llevado a cabo en el entorno de desarrollo proporcionado por **Google Colab**, que permite ejecutar código *Python* a través de la nube de Google y permite hacer uso de una de sus GPUs [71] o una de sus TPUs [72], solventando el problema de muchos usuarios por no contar en su ordenador con componentes necesarios para llevar a cabo el altísimo cómputo que demanda este tipo de aplicaciones.

Se hace uso de las siguientes bibliotecas de código abierto para *Python* de visión e inteligencia artificial:

- ***TensorFlow:*** plataforma *open source* de *machine learning* con bibliotecas, herramientas y recursos fundamentales para el desarrollo en este ámbito [73].
- ***Keras:*** biblioteca de redes neuronales con multitud de herramientas [74].

- ***Numpy:*** biblioteca para cálculo matemático con vectores y matrices, entes matemáticos empleados ampliamente para IA [75].
- ***OpenCV:*** biblioteca de herramientas *Computer Vision* [76].
- ***Matplotlib:*** biblioteca que permite representar gráfico, visualizar imágenes, etc [77].
- ***Scipy.spatial***: módulo para el cálculo de distancias en imágenes 2D [78].

3.3. Detección y localización de caras

Puesto que lo que se quiere detectar si las personas en la imagen llevan mascarilla, en primer lugar, es necesario detectar las caras de las personas presentes en la imagen. Para ello se ha elegido emplear un clasificador de alta velocidad, el *Haar-cascade Classifier,* combinado con una detección en múltiples escalas, que genera una región de interés (ROI) alrededor de cada cara, que será procesada posteriormente para determinar la presencia de mascarilla. El código empleado se adjunta a continuación seguido de la definición de sus parámetros en la Figura 35:

```
Import cv2
clasif=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
imagen = cv2.imread('fotografía.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces=clasif.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=4)
```

Figura 35. Código empleado para Haar cascade (fuente propia).

La entrada de datos para este clasificador debe ser en escala de grises, por lo que se realiza un preprocesado de las imágenes.

El **factor de escala** (***scaleFactor***) indica el porcentaje de reducción de escala que se le aplicará a la imagen sobre la que se realiza la detección con el objetivo de detectar caras de distintos tamaños. Para detectar caras más grandes interesa un valor de escala más grande, de forma que la imagen se reduzca lo suficiente para contemplarla entera mientras que para detectar caras pequeñas no se debería reducir demasiado para que no perderla. En experimentos se ha observado que a factores de escala menores a 1.1 le resulta más fácil al clasificador detectar caras en multitudes, aunque también comete más fallos. Cuando no se trata de muchas caras o caras muy pequeñas, el clasificador funciona correctamente con un valor de 1.1.

Por último, se destaca el parámetro ***minNeigbors*** que denota el número de rectángulos (que identifican que se ha detectado una cara) que hacen falta para definir una cara como detectada. En la figura 36 se aprecia un ejemplo. Si se pone este valor a 1, significa que tantas veces detecte la misma cara la identifica como una cara nueva pudiendo posicionar sobre ella más de un rectángulo. Si se pone a un número mayor (para este trabajo se puso a 4), aunque la identifique muchas veces el clasificador, al final la identificará como una sola cara.

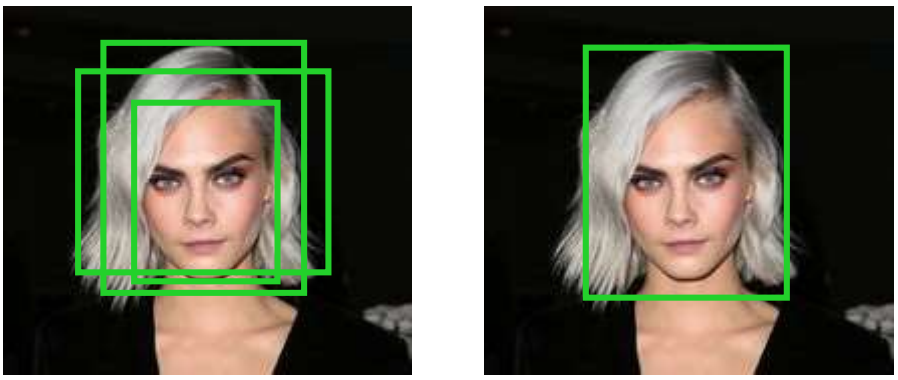


Figura 36. *Imagen con varias detecciones sobre una misma cara (izquierda) frente a una única detección (derecha) (fuente propia sobre [79]).*

El detector devuelve como resultado una tupla (x, y, w, h) que identifican las coordenadas de la esquina inferior izquierda del rectángulo identificador de la cara (x, y), su anchura (w) y altura (h).

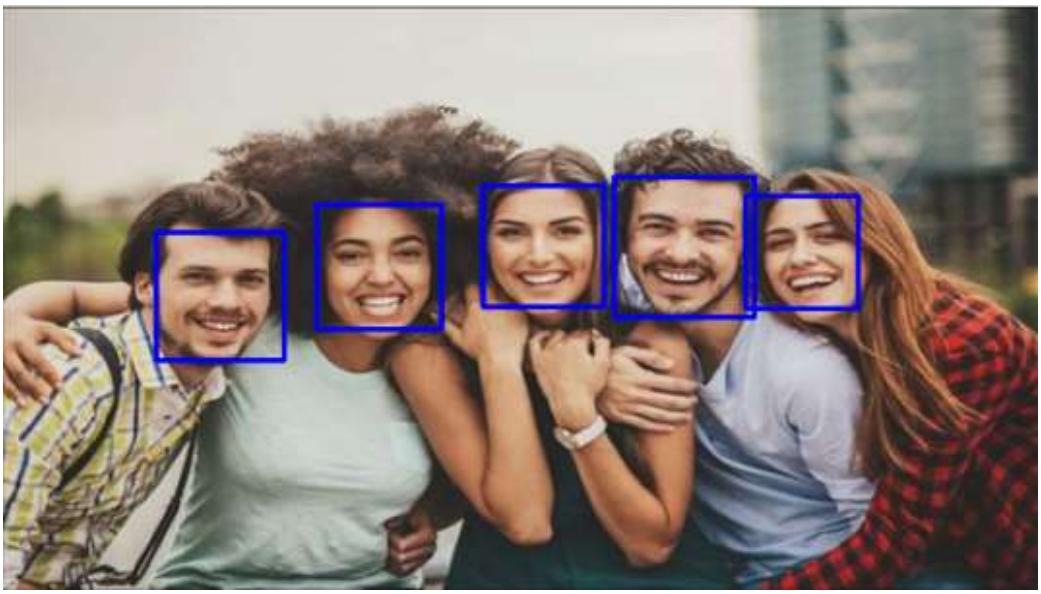


Figura 37. *Ejemplo de detección múltiple de caras usando el clasificador parametrizado con scalefactor=1.1 y minNeigbors=4 (fuente propia sobre [80]).*

Se ha añadido que, si en la imagen se detecta más de 6 personas, se notifique una alarma por alta densidad de personas.

3.4. Detección de distancia de seguridad

El cálculo se pretende implementar en imágenes 2D sin hacer uso de 2 cámaras a una distancia conocida que puedan hacer el cálculo por triangulación [81] ni con sistemas de tiempo de vuelo [82]. Se tiene que realizar sobre personas, de tamaños variados, por lo que no se puede hacer un cálculo por referencia de una medida conocida [83], por lo que se decide hacer el cálculo mediante **distancia euclídea**: mediante el teorema de Pitágoras, se calcula la distancia entre las esquinas inferiores izquierdas en los pares de caras.

Se resalta de rojo aquellas personas que violan que estén muy cerca y de verde aquellas que estén suficientemente separadas, como se muestra en el ejemplo a continuación de la Figura 38, con resultados de la detección de caras obtenidas.

Figura 38. *Ejemplos de detección de distancia: a la izquierda (detección sobre [84]), 2 personas que no cumplen una mínima distancia y quedan remarcados en rojo; a la derecha (detección sobre [85]), dos personas remarcadas en verde porque cumplen esa mínima distancia establecida.*

Este método, sin embargo, no aporta la distancia real de la medición, sino la distancia en píxeles, por lo que entre los posibles trabajos futuros se propone una alternativa a implementar, con más medios de los que se contaron para este proyecto.

3.5. Detección de mascarillas

Como se indicaba en el esquema, la cara detectada se recorta de la imagen para que esta sea la entrada a la CNN y se analice solo la cara.

3.5.1. Red neuronal

Para la clasificación de imágenes, es habitual tomar como punto de partida una red neuronal previamente definida (como las explicadas en el apartado **2.7**.), y adaptarla para la aplicación de interés, reentrenándola en caso necesario.

En el caso de este TFG, se escogió, por su alta eficacia y relación coste-beneficio, una *VGGNet* de 19 capas de profundidad, la ***VGG19*** [56], que se modificó para adaptarla a la detección de personas con mascarillas. En concreto, se modificaron las capas más próximas a la salida. Se comenzó con una **capa *flatten*** que vectoriza los resultados de la convolución previa y se finaliza con una función de activación para el resultado de la clasificación.

A lo largo del desarrollo del trabajo, se planteó inicialmente una clasificación binaria que permitiera discriminar si la persona lleva a o no mascarilla, y posteriormente se incluyó una tercera clase, de forma que la red pudiera clasificar: personas sin mascarilla, personas con la mascarilla mal colocada, y personas con ella correctamente colocada. Así, en el caso de la clasificación binaria se incluyó una capa de activación ***sigmoidal*** tras la capa densa de salida, mientras que para las tres clases se empleó una activación ***softmax***, como se puede ver en la Tabla 3.

Tipo de capa	Filtros	Parámetros
Input (224x224x3)		
Conv3x3	64	1.7K
Conv3x3	64	36K
Max Pool		
Conv3x3	128	73K
Conv3x3	128	147K
Max Pool		
Conv3x3	256	300K
Conv3x3	256	600K
Conv3x3	256	600K
Conv3x3	256	600K
Max Pool		
Conv3x3	512	1.1M
Conv3x3	512	2.3M
Conv3x3	512	2.3M
Conv3x3	512	2.3M
Max Pool		
Conv3x3	512	2.3M
Conv3x3	512	2.3M
Conv3x3	512	2.3M
Conv3x3	512	2.3M
Max Pool		
Flatten		
Dense		*

Tabla 3. *Arquitectura de la red a implementar, con sus capas, filtros y parámetros.*

(*) Los parámetros de la última capa densa son 16 mil para la clasificación binaria y 24 mil para la clasificación multiclase (modificado de [86]*).*

3.5.2. Base de datos

La calidad de las imágenes a las que expondremos a la red durante el entrenamiento determinará, en gran medida, los resultados que consiga. Deben ser claras y focalizadas en lo que se busca. Por tanto, se señala la importancia de seleccionarlas exhaustivamente para que la red aprenda exactamente lo que se le quiere enseñar.

Para la primera clasificación binaria se empleó la base de datos ***Face Mask Detection - 12K Images Dataset***, de *Ashish Jangra* [87], que contiene las dos clases muy bien definidas y una cantidad considerablemente alta de imágenes, con 5 mil imágenes por clase para el entrenamiento, 400 para validación y 500 para comprobar los resultados finales de la red. Las imágenes de este *DataSet* están compuestas por caras delimitadas al igual que lo haría el

clasificador. Las caras con mascarilla constan de mascarillas muy distintas (de tela, quirúrgicas y variadas) sobre personas de todo rango de edad y raza, como se muestra en la Figura 39.

Figura 39. *Ejemplos de imágenes de las dos clases del DataSet [87] .*

Para el segundo planteamiento con las tres clases (mascarilla bien puesta, mascarilla mal puesta y sin mascarilla puesta) se presenta un inconveniente: la falta de bases de datos etiquetados de personas con mascarilla mal puesta. Ante esta escasez de imágenes de personas llevando mal puesta la mascarilla, en el proyecto ***MaskedFace-Net*** [88] generaron mascarillas por ordenador sobre fotografías de personas sin mascarilla, como se muestran en la Figura 40. Estas son las imágenes que se emplearán.

Figura 40. *Ejemplo de las imágenes generadas para el DataSet [88].*

Dentro de esta categoría de mascarilla mal puesta en *MaskedFace-Net*, se diferencian tres subclases: personas con la nariz por fuera de la mascarilla; con nariz y boca fuera; barbilla descubierta. Sobre esta base de datos se eliminan las imágenes que haya más de una persona y, en la categoría de "mascarillas mal puestas", se dejan solo las subclases más posibles de encontrar en una situación real (con la nariz por fuera de la mascarilla y con nariz y boca no cubiertas).

La nueva base de datos de entrenamiento para el proyecto con 3 clases se construye de forma equitativa, con 3 mil 200 imágenes por clase para el entrenamiento, 430 para validación y alrededor de 400 para la comprobación final. En este caso, se revisaron 14 mil imágenes y de ellas se eliminaron más de 1.500 por presentar elementos que podían inducir a error, como puede ser la presencia de más de una cara junto a una persona con mascarilla. Se adjuntan ejemplos de las imágenes eliminadas en la Figura 41.

Figura 41. *Ejemplos de imágenes descartadas del DataSet [88].*

Las imágenes admitidas estaban formadas por personas de todo rango de edad y raza, aunque a diferencia de la anterior *DataSet*, las imágenes se encuentran un poco más alejadas de la cara sobre la que detectar la mascarilla. Ejemplos de estas imágenes se muestran a continuación, en la Figura 42.

Figura 42. *Ejemplos de imágenes de las tres clases del DataSet [88].*

3.5.3. Entrenamiento

En el entrenamiento se ha empleado la técnica de ***transfer learnning***, que consiste en entrenar la red manteniendo invariables los pesos, de un entrenamiento previo, en las capas convolucionales y someterla a que aprenda con las capas nuevas que hemos incluido. De esta forma, la red no está aprendiendo de cero y se consigue reducir el tiempo de entrenamiento considerablemente. Se parte de los pesos que ha obtenido VGG19 en el entrenamiento con la amplísima base de imágenes *ImageNet*, lo que asegura que la extracción de características se realiza de forma correcta y, posteriormente, se adapta el resultado de la interpretación de la red al nuevo objetivo mediante el entrenamiento de las capas añadidas. De esta forma, del total de más de 20 millones de parámetros que tiene la red, solo 16 mil son susceptibles de entrenamiento para en el caso de la clasificación binaria y 24 mil para la clasificación multiclase.

Para ajustar el *learning rate* de forma dinámica se aplica el optimizador ADAM. En él, para ambas clasificaciones se ha comprobado empíricamente que, con aplicarle un valor inicial por defecto de *momentum* de 0,001, la evolución del entrenamiento es favorable.

La *función de pérdidas* con la que evaluará el entrenamiento debe detectar si está haciendo bien o no la clasificación. Así, para la clasificación binaria se ha elegido la función **crosentropía binaria** [89] y para la clasificación multiclase, la función **crosentropía categórica** [90].

El entrenamiento de ambas clasificaciones, gracias al alto volumen de imágenes de los dos *DataSets*, se realizaron en 20 épocas por entrenamiento con una duración aproximada de 20 minutos cada uno.

Capítulo 4

RESULTADOS EXPERIMENTALES

4.1. Resultados de entrenamiento.

Graficando los resultados que se obtienen durante el entrenamiento ayuda a supervisar si está aprendiendo la red y validar su grado de acierto gracias a las pruebas que se realizan con el conjunto de imágenes para validación [91] y revisar que no ocurra, por ejemplo, subentrenamiento [92]. Mas esto no demuestra que la red haya aprendido correctamente y, por ejemplo, haya experimentado sobreentrenamiento debido a que la red solamente haya aprendido a reconocer un único tipo de mascarillas. Esto será comprobado experimentalmente en los apartados siguientes.

De los resultados de entrenamiento se hablará únicamente de la red de detección de tres clases, ya que los resultados son iguales en ambas redes y resultaría redundante explicar lo mismo para ambas. En la Figura 43 los resultados muestran que la red aprende rápidamente. En 10 épocas se completa el entrenamiento y con oscilaciones mínimas de los datos de validación del entrenamiento, se mantiene constante y próximo al 100% de exactitud hasta el final de las épocas.

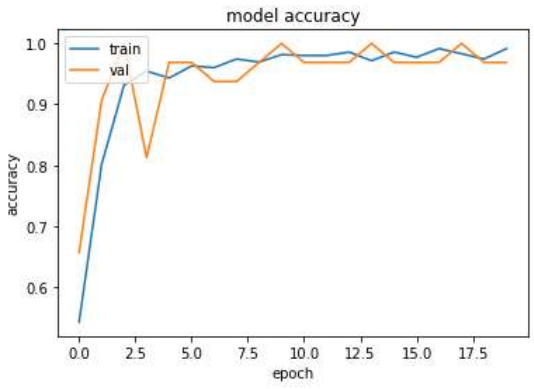


Figura 43. *Gráfica de entrenamiento con la precisión del entrenamiento y de validación.*

Por su parte y de forma complementaria, en la Figura 44 la función de pérdidas muestra como los errores decaen abruptamente en las 4 primeras épocas, manteniendo desde entonces una reducción moderada y con ligeras oscilaciones que no superan el 20% de error hasta el final del entrenamiento.

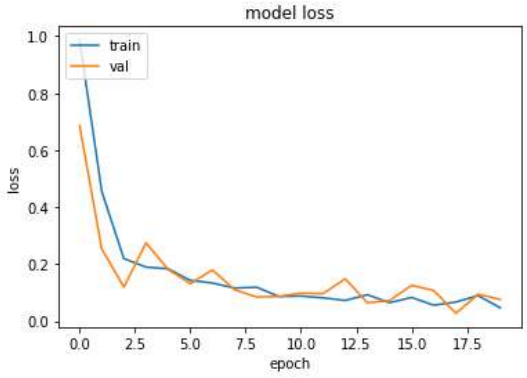


Figura 44. *Gráfica de entrenamiento con el error del entrenamiento y de validación.*

No se considera pertinente añadir más épocas ya que, en aproximadamente 10 épocas, completa el objetivo a conseguir.

4.2. Métricas

Para calificar los resultados de la red se realiza una matriz de confusión (Tabla 4) en la que, para cada clase, en la que se recogen el número total de observaciones junto a los aciertos y los errores en la predicción siguiendo el siguiente esquema:

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Tabla 4. *Aclaración de valores para realizar las métricas en dos clases [93].*

- **VP**: cantidad de positivos clasificados correctamente como positivos por el modelo.
- **FP**: cantidad de negativos clasificados incorrectamente como positivos.
- **FN**: cantidad de positivos clasificados incorrectamente como negativos.
- **VN**: cantidad de negativos clasificados correctamente como negativos.

Con estos resultados se determina diferentes métricas para evaluar el sistema implementado:

Exactitud: porcentaje de clasificación correcto sobre el total.

$$\boldsymbol{Exactitud = \frac{VP + VN}{Total}}$$

Tasa de error: porcentaje de clasificación incorrecto sobre el total.

$$\boldsymbol{Tasa\ de\ error = \frac{FP + FN}{Total}}$$

Tasa de verdaderos positivos: porcentaje de acierto en los positivos.

$$\boldsymbol{Tasa\ de\ verdaderos\ positivos = \frac{VP}{Total\ Positivos}}$$

Tasa de verdaderos negativos: porcentaje de acierto en los negativos.

$$\boldsymbol{Tasa\ de\ verdaderos\ negativos = \frac{VN}{Total\ Negativos}}$$

Valor de predicción de positivos: porcentaje de positivos correctos sobre todos los clasificados como positivos.

$$\boldsymbol{Tasa\ de\ predicción\ positivos = \frac{VP}{Total\ clasificados\ Positivos}}$$

Valor de predicción de negativos: porcentaje de negativos correctos sobre todos los clasificados como negativos.

$$\boldsymbol{Tasa\ de\ predicción\ negativos = \frac{VN}{Total\ clasificados\ Negativos}}$$

Para medir los resultados de la segunda red con tres categorías, se adaptan las métricas anteriores, resultando lo siguiente, Tabla 5:

		Predicción		
		Clase 1	**Clase 2**	**Clase 3**
Observación	**Clase 1**	Verdaderos clase 1 (VC1)	Falso clase 1 (FC1)	Falso clase 1 (FC1)
	Clase 2	Falso clase 2 (FC2)	Verdaderos clase 2 (VC2)	Falso clase 2 (FC2)
	Clase 3	Falso clase 3 (FC3)	Falso clase 3 (FC3)	Verdaderos clase 3 (VC3)

Tabla 5. *Aclaración de valores para realizar las métricas en tres clases.*

Exactitud: porcentaje de clasificación correcto sobre el total.

$$\boldsymbol{Exactitud = \frac{VC1 + VC2 + VC3}{Total}}$$

Tasa de error: porcentaje de clasificación incorrecto sobre el total.

$$\boldsymbol{Tasa\ de\ error = \frac{FC1 + FC2 + FC3}{Total}}$$

Tasa de verdaderos clase 1: porcentaje de acierto en la clase 1.

$$\boldsymbol{Tasa\ de\ verdaderos\ clase\ 1 = \frac{VC1}{Total\ Clase\ 1}}$$

Tasa de verdaderos clase 2: porcentaje de acierto en la clase 2.

$$\boldsymbol{Tasa\ de\ verdaderos\ clase\ 2 = \frac{VC2}{Total\ Clase\ 2}}$$

Tasa de verdaderos clase 3: porcentaje de acierto en la clase 3.

$$\boldsymbol{Tasa\ de\ verdaderos\ clase\ 3 = \frac{VC3}{Total\ Clase\ 3}}$$

Valor de predicción clase 1: porcentaje de clase 1 correctos sobre todos los clasificados como clase 1.

$$\boldsymbol{Tasa\ de\ predicción\ clase\ 1 = \frac{VC1}{Total\ clasificados\ clase\ 1}}$$

Valor de predicción clase 2: porcentaje de clase 2 correctos sobre todos los clasificados como clase 2.

$$\boldsymbol{Tasa\ de\ predicción\ clase\ 2 = \frac{VC2}{Total\ clasificados\ clase\ 2}}$$

Valor de predicción clase 3: porcentaje de clase 3 correctos sobre todos los clasificados como clase 3.

$$\textbf{Tasa de predicción clase 3} = \frac{\textbf{VC3}}{\textbf{Total clasificados clase 3}}$$

4.3. Resultados

Los resultados obtenidos muestran que la red es capaz de diferenciar en ambas tareas la presencia o ausencia de la mascarilla en la cara de la persona, así como de detectar además en la segunda tarea la mascarilla mal puesta.

En la primera tarea, los resultados obtenidos muestran que: de 483 imágenes presentadas de personas con mascarilla, el modelo acierta en el reconocimiento de 470 imágenes y falla en 13; y de 509 imágenes presentadas de personas sin mascarilla, el modelo acierta en el reconocimiento de 503 imágenes y falla en 6. Ver Figura 45 y Tabla 6.

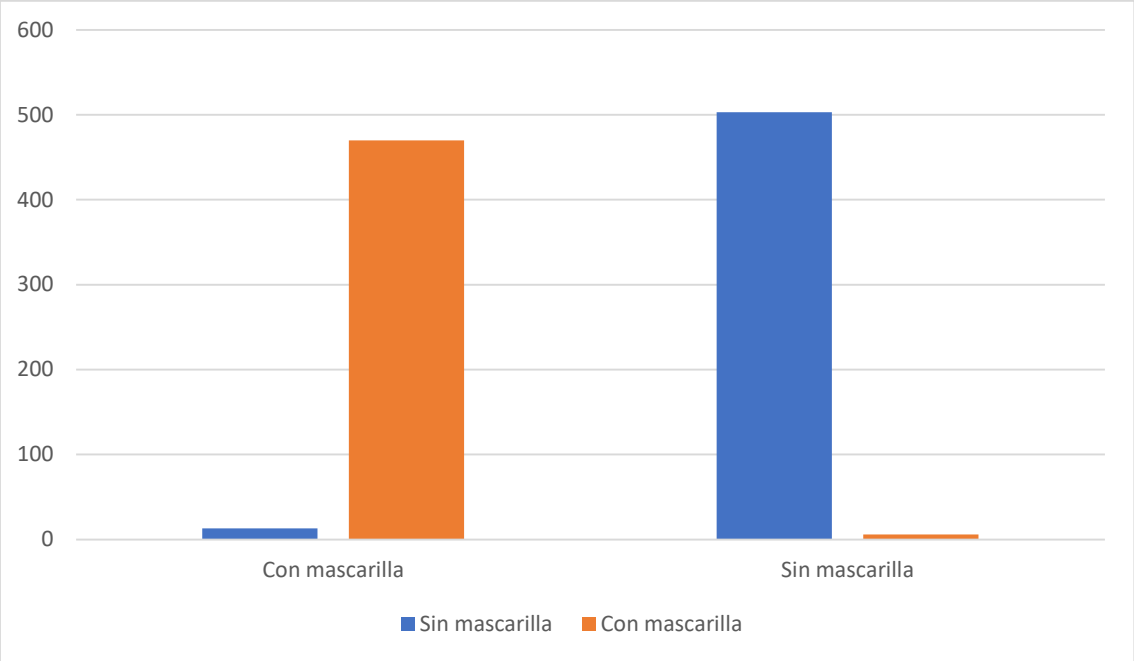


Figura 45. *Aciertos y errores del modelo en el reconocimiento de personas con y sin mascarilla.*

	Mascarilla puesta (%)	Sin mascarilla (%)
Mascarilla puesta (%)	97.3	1.12
Sin mascarilla (%)	2.7	98.8

Tabla 6. *Matriz de confusión obtenida en la evaluación del sistema de detección de mascarillas binario.*

Estos datos aportan los siguientes valores para las métricas descritas anteriormente, donde los positivos corresponde a la clase con mascarilla y negativo a la clase sin mascarilla:

Medida	Porcentaje (%)
Exactitud	98.1
Tasa de error	1.9
Tasa de verdaderos positivos	97.3
Tasa de verdaderos negativos	98.8
Valor de predicción positivos	98.7
Valor de predicción negativos	97.5

Tabla 7. *Métricas sobre el ensayo de detección 2 clases.*

La red es capaz de detectar con muchísima precisión ambas categorías sobre mascarillas de todo tipo: tela, quirúrgicas, etc. Lo convierte en un detector muy fiable y robusto. Se muestra un ejemplo de detección en la Figura 46.

Figura 46. *Resultado de detección de dos personas muy próximas cada una correspondiente a una clase (detectada sobre [94]).*

En la segunda tarea, esta vez con tres clases a detectar, los resultados obtenidos muestran que: de 401 imágenes presentadas de personas con mascarilla, el modelo acierta en el reconocimiento de 343 imágenes y falla en 58, de las cuales 55 las clasificó como sin mascarilla y 3 como mal puestas; de 466 imágenes presentadas de personas sin mascarilla, el modelo acierta todas y no falla en ninguna; y por último, de 413 imágenes de personas con la mascarilla mal puesta, el modelo acierta en el reconocimiento de 286 imágenes y falla en 127, de las cuales cataloga 49 como mascarilla puesta y 78 como sin mascarilla. Los resultados quedan reflejados en la Figura 47 y Tabla 8.

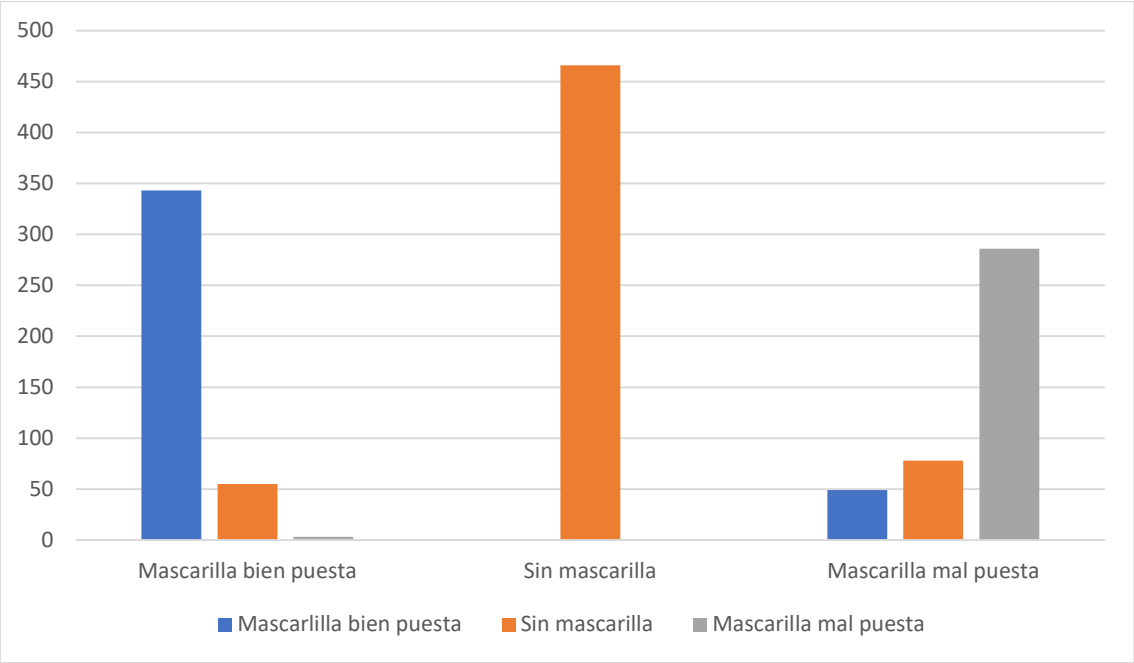


Figura 47. *Aciertos y errores del modelo en el reconocimiento de personas con, sin mascarilla y con la mascarilla mal puesta.*

	Mascarilla bien puesta (%)	Sin mascarilla (%)	Mascarilla mal puesta (%)
Mascarilla bien puesta (%)	85.5	0	11.9
Sin mascarilla (%)	13.7	100	18.9
Mascarilla mal puesta (%)	0.8	0	69.2

Tabla 8. *Datos obtenidos del ensayo de detección 3 clases.*

Dadas las características de la presente tarea en la que se quiere detectar tres clases de imágenes, en vez de denominar con positivo y negativo se denomina con clase 1 a la clase con mascarilla, clase 2 a la clase sin mascarilla y clase 3 a la clase con la mascarilla mal puesta. Estas nuevas métricas son calculadas de la siguiente forma en la Tabla 9:

Medida	Porcentaje (%)
Exactitud	85.6
Tasa de error	14.5
Tasa de verdaderos clase 1	85.5
Tasa de verdaderos clase 2	100
Tasa de verdaderos clase 3	69.3
Valor de predicción clase 1	87.5
Valor de predicción clase 2	77.8
Valor de predicción clase 3	99

Tabla 9. *Métricas sobre del ensayo de detección 3 clases.*

Los resultados son buenos, con una exactitud de casi un 86%. Aunque, son inferiores a los obtenidos en el caso anterior. En esta ocasión, se ha dependido de una base de datos menor para el entrenamiento que en el entrenamiento de la tarea anterior, con tres mil doscientas imágenes para cada clase, frente a las cinco mil de antes. Aunque los resultados son buenos, dejan ver que más datos para la clase de "mascarilla mal puesta" quizás habría conseguido menos fallos, pues esta clase contiene a su vez dos subclases (con nariz fuera y con nariz y boca fuera).

En la subclase "con nariz y boca fuera", se ha comprobado que el error por el que el sistema la clasifica como "sin mascarilla" en vez de "mascarilla mal puesta", es debido a la región que le envía el clasificador de caras, pues éste puede que, en algunas ocasiones, recorte parte de la barbilla dejando fuera del campo visual la mascarilla, reduciendo la probabilidad de que la red la detecte. Se muestra un ejemplo en la Figura 48.

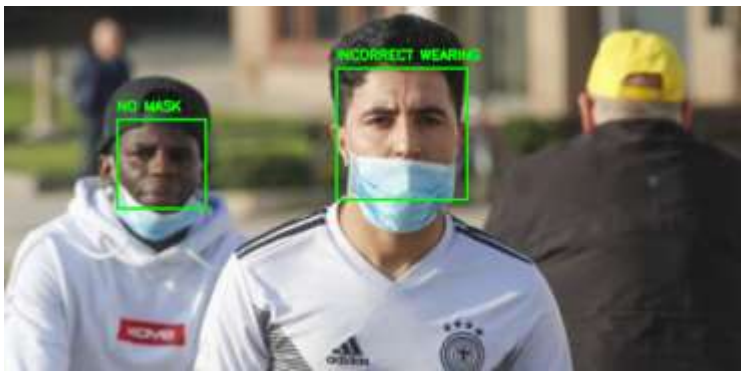


Figura 48. *Error cometido por la red en la detección de mascarilla mal puesta con la nariz y boca fuera (detección sobre [95]).*

Una debilidad detectada en la base de datos utilizada en el entrenamiento es que todas las mascarillas son quirúrgicas. La red queda sobreentrenada a detectar este tipo de mascarillas (o parecidas) obviando los demás tipos.

Se aprecia que no ha habido ningún error en el reconocimiento de personas sin mascarilla. A ello también ha contribuido que para esta clase se integró, en la base de datos de entrenamiento, una vista de las mismas personas desde distintos ángulos consiguiendo una imagen completa de las caras.

Capítulo 5

DISCUSIONES Y LÍNEAS FUTURAS

Los parámetros del clasificador (*scaleFactor y minNeigbors*) influyen considerablemente en la efectividad del sistema, pues si falla en la clasificación la CNN no recibe ninguna imagen que analizar.

Una limitación del clasificador es que identifica mayormente caras frontales, lo que dificulta detectar rostros que no miren directamente a la cámara. Sería conveniente explorar otras alternativas de *Deep Learning* si se contase con suficientes datos etiquetados para reconocer el rostro y la mascarilla independientemente del ángulo desde el que se le vea.

Con una única imagen 2D solo se podría medir una distancia real si se tuviese una medida de referencia, como, por ejemplo, una medida estándar de rostros. Pero esto no es posible tanto porque los rostros son muy variados como porque el clasificador no siempre toma la misma región con la que detecta la cara. Por tanto, la detección de distancias podría ser mejorada haciendo uso de dos cámaras que mediante triangulación detectasen la distancia real de la imagen.

Capítulo 6

CONCLUSIONES

El sistema desarrollado es válido para ayudar al cumplimiento de las medidas sanitarias para prevenir la transmisión del *SARS-CoV-2*. Ha demostrado la más alta precisión para detectar a personas sin mascarilla, siendo sus resultados también muy buenos en la detección de mascarillas y en su modo de uso. Añade la posibilidad de detectar el cumplimiento de la distancia de seguridad y la presencia de aglomeraciones. Es fácilmente implementable en dispositivos de seguridad con una cámara única, facilitando información sobre la disposición o no de mascarillas y su correcta utilización, la distancia de seguridad y la densidad de personas.

Como propuesta de futuro, se propone explorar el uso de varias cámaras y otras alternativas de *Deep Learning*.

Referencias

[1] «Organización Mundial de la Salud,» [En línea]. Available: https://www.who.int/es/news/item/27-04-2020-who-timeline---covid-19.

[2] R. Pequín, «La Vanguardia Tecnología,» 9 Marzo 2020. [En línea]. Available: http://qoo.ly/34tj3t#https://www.lavanguardia.com/tecnologia/20200309/474044526830/empresa-china-desarrolla-sistema-camaras-mascarillas-chinacoronavirus.html. [Último acceso: 20 Junio 2021].

[3] A. Payo, «muy interesante,» 20 Febrero 2020. [En línea]. Available: https://www.muyinteresante.es/tecnologia/articulo/baidu-lanza-una-ia-para-identificar-a-personas-que-no-lleven-mascarilla-para-protegerse-del-coronavirus-721582126860. [Último acceso: 21 Junio 2021].

[4] A. P. Junior, T. P. D. Homem y F. O. Teixeira, «Aplicación de inteligencia artificial para monitorear el uso de mascarillas de protección,» *Revista Científica General José,* vol. 19, nº 33, pp. 205-222, 2021.

[5] «Actualidad Sanitaria,» 27 Abril 2020. [En línea]. Available: https://actualidadsanitaria.com/ciencia/como-saber-si-una-mascarilla-es-efectiva/. [Último acceso: 6 Agosto 2021].

[6] B. López, «Introducción a la inteligencia artificial,» [En línea]. Available: http://itnuevolaredo.edu.mx/takeyas/Articulos/Inteligencia%20Artificial/ARTICULO%20Introduccion%20a%20la%20Inteligencia%20Artificial.pdf.

[7] RottenFlech, «Robotica linea del tiempo Timetoast,» [En línea]. Available: https://www.timetoast.com/timelines/robotica-linea-de-tiempo.

[8] «Muy Historia,» [En línea]. Available: https://www.muyhistoria.es/curiosidades/preguntas-respuestas/ihabia-estatuas-con-movimiento-en-el-antiguo-egipto.

[9] A. Serna, E. Serna y A. E.M., «Principios de la Inteligencia Artificial en las Ciencias Computacionales.,» de *Desarrollo e Innovación en Ingeniería*, Medellín, Instituto Antioqueño de Investigación, 2017, pp. 161-173.

[10] R. E. Bellman, An introduction to Artificial Intelligence: Can Computer think?, San Francisco: Boyd & Fraser Publishing Company, 1978.

[11] J. (. Haugeland, Artificial Inteligence: The very idea., Massachusetts: MIT, Press Cambridge, 1985.

[12] E. a. M. D. Chraniak, Introduction to Artificial Intelligence., Massachusetts: Addison-Wesley, Reading, 1985.

[13] R. Kurzweil, The art of Intelligent Machines., Massachusetts: MIT Press, Cambridge, 1990.

[14] E. a. K. K. Rich, Artificial Inteligence (Second Edition)., New York.: McGraw-Hill, 1991.

[15] R. J. Schalkoff, Artificial Intelligence: An Engineering Approach, New York: McGraw-Hill, 1990.

[16] P. Wiston, Artificial Intelligence (Third edition)., Massachusetts: Adddison-Wesley, Reading,, 1992.

[17] G. a. S. W. Luger, Structures and Strategies for Complex Problem Solving, España: Redwood City, Benjamin Cummings, 1993.

[18] E. M. Acevedo, A. A. Serna y E. M. Serna, «Principios y características de las redes neuronales artificiales,» de *Desarrollo e Innovación en Ingeniería 2a Edición*, Medellín, Instituto Antioqueño de Investigación, 2017, pp. 173-183.

[19] «CONCEPTOS BÁSICOS SOBRE REDES NEURONALES,» [En línea]. Available: https://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm. [Último acceso: 01 julio 2021].

[20] S. M. Sankar K. Pal, «Multilayer Perceptron, Fuzzy Sets, and Classification,» *Transactions on Neural Networks,* vol. 3, nº 5, pp. 683-697, 1992.

[21] E. A. Rubira, «Introducción al reconocimiento de patrones,» ACADEMIA.

[22] M. D. GIRALDO y J. G. HOYOS G, «CONTROL POR REDES NEURONALES DE BASE RADIAL Y PLANOS DESLIZANTES,» *Scientia Et Technica,* vol. X, nº 26, pp. 43-46, 2004.

[23] G. A. Toro Bayona y I. A. Lizarazo Salcedo, «Evaluación de las Redes Neuronales Artificiales Perceptron Multicapa y Fuzzy-Artmap en la Clasificación de Imágenes Satelitales,» *Ingeniería,* vol. 17, nº 1, pp. 61-72, 2012.

[24] D. J. Matich, «Redes Neuronales: Conceptos Básicos y Aplicaciones,» *ACADEMIA,* 2001.

[25] G. W. e. al., «Interactive Medical Image Segmentation Using Deep Learning With Image-Specific Fine Tuning,» *IEEE Transactions on Medical Imaging,* vol. 37, nº 7, pp. 1562-1573, 2018.

[26] «PNG EGG,» [En línea]. Available: https://www.pngegg.com/en/png-ezpye. [Último acceso: 20 Agosto 2021].

[27] J. Zeng, S. B. Lin, Y. Yao y D. X. Zhou, «On ADMM in Deep Learning: Convergence and Saturation-Avoidance,» *Journal of Machine Learning Research,* vol. 22, pp. 1-67, 2020.

[28] K. Gpalsamy, «Stability of artificial neural networks with impulses,» *Applied Mathematics and Computation,* vol. 154, nº 3, pp. 783-813, 2004.

[29] G. W. B. e. al., «Experimental Demonstration and Tolerancing of a Large-Scale Neural Network (165 000 Synapses) Using Phase-Change Memory as the Synaptic Weight Element,» *IEEE Transactions on Electron Devices,* vol. 62, nº 11, pp. 3498-3507, 2015.

[30] S. Ioffe y C. Szegedy, «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariative Shift,» *Proceedings of Machine Learning Research,* vol. 37, pp. 448-456, 2015.

[31] F. Rosa, «ResearchGate,» [En línea]. Available: https://www.researchgate.net/figure/Red-neuronal-convolucional-4_fig7_308783857. [Último acceso: 20 Julio 2021].

[32] D. Calvo, «Diego Calvo,» [En línea]. Available: https://www.diegocalvo.es/red-neuronal-convolucional/. [Último acceso: 20 Julio 2021].

[33] «Country Living,» 15 Julio 2021. [En línea]. Available: https://www.countryliving.com/uk/wildlife/dog-breeds/a37019720/labrador-retrievers-higher-risk-health-problems/. [Último acceso: 5 Agosto 2021].

[34] «StackExchange,» [En línea]. Available: https://stats.stackexchange.com/questions/362988/in-cnn-do-we-have-learn-kernel-values-at-every-convolution-layer. [Último acceso: 25 Julio 2021].

[35] A. Krizhevsky, I. Sutskever y G. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks,» 2012.

[36] A. Khellal, H. Ma y Q. Fei, «Convolutional Neural Network Based on Extreme Learning Machine for Maritime Ships Recognition in Infrared Images,» *Sensers,* vol. 18, 2018.

[37] D. Mack, «Incorporating element-wise multiplication can out-perform dense layers in neural networks,» 11 Novienbre 2018. [En línea]. Available: https://medium.com/octavian-ai/incorporating-element-wise-multiplication-can-out-perform-dense-layers-in-neural-networks-c2d807f9fdc2. [Último acceso: 30 Julio 2021].

[38] «Computer Science Wiki,» [En línea]. Available: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling. [Último acceso: 30 Julio 2021].

[39] E. T. Ocampo, D. M. Giraldo y H. S. Isaza, «PRONÓSTICO DE VENTAS USANDO REDES NEURONALES,» *Scientia Et Technica,* vol. X, nº 26, pp. 25-30, 2004.

[40] G. Ovando, M. Bocco y S. Sayago, «REDES NEURONALES PARA MODELAR PREDICCIÓN DE HELADAS,» *Agricultura Técnica,* pp. 65-73, 2005.

[41] J. Jeremy, «Jeremy Jordan,» 1 Marzo 2018. [En línea]. Available: https://www.jeremyjordan.me/nn-learning-rate/. [Último acceso: 26 Julio 2021].

[42] J. Brownlee, «Machine Learning Mastery,» 3 Julio 2017. [En línea]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. [Último acceso: 27 Julio 2021].

[43] I. C. Pérez y L. A. García, «Una revisión sobre aprendizaje no supervisado de métricas de distancia,» *Revista Cubana de Ciencias Informáticas,* vol. 10, nº 4, pp. 2227-1899, 2016.

[44] «Medium,» 29 Nobiembre 2017. [En línea]. Available: https://medium.com/ai-learners/deep-learning-un-enfoque-pr%C3%A1ctico-parte-1-8a1b4d377e98. [Último acceso: 5 Agosto 2021].

[45] N. G. N. C. W. H. Sanjeev Arora, «A CONVERGENCE ANALYSIS OF GRADIENT DESCENT FOR DEEP LINEAR NEURAL NETWORKS,» 2019.

[46] C. Menacho, «Modelos de regresión lineal con redes neuronales,» *Anales Científicos,* vol. 75, nº 2, pp. 253-260, 2014.

[47] «Dudalia,» [En línea]. Available: https://dudalia.com/tarea/1l4k. [Último acceso: 21 Julio 2021].

[48] S. Sánchez, «PREDICCIÓN DE TERREMOTOS BASADOS EN EXPLORACIÓN DE DATOS ESPACIO-TEMPORALES,» 2019.

[49] «Lonutlang,» [En línea]. Available: https://medium.com/@ionutlang31/how-to-play-around-with-sigmoid-function-to-increase-its-y-max-and-shift-to-the-right-for-positive-ede40daf1fa2. [Último acceso: 21 Julio 2021].

[50] «github,» [En línea]. Available: https://adl1995.github.io/an-overview-of-activation-functions-used-in-neural-networks.html. [Último acceso: 21 Julio 2021].

[51] «sebastian raschka,» [En línea]. Available: https://sebastianraschka.com/faq/docs/relu-derivative.html. [Último acceso: 21 Julio 2021].

[52] «vidyasheela,» [En línea]. Available: https://vidyasheela.com/post/hyperbolic-tangent-tanh-activation-function-with-python-code. [Último acceso: 21 Julio 2021].

[53] «HEARTBEAT,» [En línea]. Available: https://heartbeat.comet.ml/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0. [Último acceso: 21 Julio 2021].

[54] «ResearchGate,» [En línea]. Available: https://www.researchgate.net/figure/An-illustration-of-the-architecture-of-AlexNet-deep-convolutional-neural-network_fig3_308880040. [Último acceso: 22 Julio 2021].

[55] «Neural network studies. 1. Comparison of overfitting and overtraining,» *Journal of Chemical Information and Computer Sciences,* vol. 35, nº 5, pp. 826-833, 1995.

[56] K. Simonyan y A. Zisserman, «VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,» 2014.

[57] «Springer Link,» [En línea]. Available: https://link.springer.com/chapter/10.1007/978-981-15-8221-9_180. [Último acceso: 5 Agosto 2021].

[58] P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke y A. Rabionovich, «Going deeper with convolutions,» 2014.

[59] «sitiobigdata,» [En línea]. Available: https://sitiobigdata.com/2019/05/01/innovaciones-arquitectonicas-redes-neuronales-clasificacion-imagenes/. [Último acceso: 5 Agosto 2021].

[60] X. Xiaoling, X. Cui y N. Bing, «Inception-v3 for flower classification,» de *2nd International Conference on Image, Vision and Computing*, 2017.

[61] «Cross Validated,» [En línea]. Available: https://stats.stackexchange.com/questions/320663/inception-module-backpropagation-how-to-get-the-input-error. [Último acceso: 5 Agosto 2021].

[62] K. He, X. Zhang, S. Ren y J. Sun, «Deep Residual Learning for Image Recognition,» 2015.

[63] «medium,» 14 Mayo 2018. [En línea]. Available: https://medium.com/datos-y-ciencia/modelos-cnn-en-la-clasificaci%C3%B3n-de-im%C3%A1genes-cl%C3%A1sicas-y-modernas-d072a6718689. [Último acceso: 29 Julio 2021].

[64] C. Szegedy, S. Ioffe y V. Vanhoucke, «Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,» 2016.

[65] A. Alemi, «Google AI blog,» 31 Agosto 2016. [En línea]. Available: https://ai.googleblog.com/2016/08/improving-inception-and-image.html. [Último acceso: 1 Agosto 2021].

[66] P. Viola y M. Jones, «Rapid Object Detection using a Boosted Cascade of Simple Features,» de *CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION* , 2001.

[67] V. E.-C. Nathan Lovell, «Color Classification and Object Recognition for Robot Soccer Under Variable Illumination,» 2007.

[68] M. J. Paul Vioal, «Rapid Object Detection using a Boosted Cascade of Simple Features,» 2001.

[69] R. E. Schapire, «Springer Link,» 9 Octubre 2013. [En línea]. Available: https://link.springer.com/chapter/10.1007/978-3-642-41136-6_5. [Último acceso: 8 Agosto 2021].

[70] S. Soo, «Object detection using Haar-cascade,» ACADEMIA, 2014.

[71] J. López, «Hard Zone,» 14 Julio 2021. [En línea]. Available: https://hardzone.es/reportajes/que-es/gpu-caracteristicas-especificaciones/. [Último acceso: 10 Agosto 2021].

[72] «Google Cloud,» [En línea]. Available: https://cloud.google.com/tpu. [Último acceso: 20 mayo 2021].

[73] «Tensor Flow,» [En línea]. Available: https://www.tensorflow.org/?hl=es-419. [Último acceso: 30 mayo 2021].

[74] «Keras,» [En línea]. Available: https://keras.io/. [Último acceso: 30 mayo 2021].

[75] «NumPy,» [En línea]. Available: https://numpy.org/. [Último acceso: 30 Mayo 2021].

[76] «opencv,» [En línea]. Available: https://opencv.org/. [Último acceso: 30 mayo 2021].

[77] «Matplotlib,» [En línea]. Available: https://matplotlib.org/. [Último acceso: 30 Mayo 2021].

[78] «SciPy,» [En línea]. Available: https://www.scipy.org/. [Último acceso: 30 Mayo 2021].

[79] «ElPais,» [En línea]. Available: https://elpais.com/elpais/2017/05/02/gente/1493719395_964043.html. [Último acceso: 27 Agosto 2021].

[80] «Freepik,» [En línea]. Available: https://www.freepik.es/fotos-premium/felices-jovenes-turistas-tienen-viaje-tomar-foto_5203486.htm#page=1&query=amigos&position=13. [Último acceso: 1 septiembre 2021].

[81] A. E. Cheli, Introducción a la fotogrametría y su evolución, La Plata: 1a, 2011.

[82] L. Lorenti y J. Giacomantone, «Segmentación espectral de imágenes utilizando camaras de tiempo de vuelo,» de *Congreso Argentino de Ciencias de la Computación*, 2013.

[83] A. Rosebrock, «pyimagesearch,» 4 Abril 2016. [En línea]. Available: https://www.pyimagesearch.com/2016/04/04/measuring-distance-between-objects-in-an-image-with-opencv/. [Último acceso: 15 julio 2021].

[84] «Freepik,» [En línea]. Available: https://www.freepik.es/foto-gratis/hombres-jovenes-alegres-camisas-azules-cuadros-camisetas-blancas-pantalones-coloridos-posan-pared-naranja-buen-humor-sonrien_15788693.htm#page=1&query=amigos&position=24. [Último acceso: 1 septiembre 2021].

[85] «elCorreoGallego,» [En línea]. Available: https://www.elcorreogallego.es/primer-plano/sanchez-hara-cuerentena-tras-el-positivo-en-covid-de-macron-con-el-que-estuvo-el-lunes-FY5706412. [Último acceso: 1 septiembre 2021].

[86] «Research Gate,» [En línea]. Available: https://www.researchgate.net/figure/Details-on-the-VGG19-architecture-For-each-layer-number-of-filters-parameters-and_tbl1_314237915. [Último acceso: 2 septiembre 2021].

[87] A. Jangra, «Kaggle,» [En línea]. Available: https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset.

[88] K. H. H. B. M. M. J. C. Adnane Cabani, «github,» [En línea]. Available: https://github.com/cabani/MaskedFace-Net.

[89] A. Ramos, J. F. Pedroso y A. L. Diez, «Deconstructing Cross-Entropy for Probabilistic Binary Classifiers,» *MDPI,* vol. 20, nº 3, 2018.

[90] Z. Zhang y M. R. Sabuncu, «Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels».

[91] T. Shah, «towards data science,» 6 diciembre 2017. [En línea]. Available: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7. [Último acceso: 20 Julio 2021].

[92] Na8, «Aprende Machine Learning,» 12 diciembre 2017. [En línea]. Available: https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/. [Último acceso: 23 julio 2021].

[93] C. Zelada, «RPubs,» 10 5 2017. [En línea]. Available: https://rpubs.com/chzelada/275494. [Último acceso: 23 julio 2021].

[94] «BBC,» [En línea]. Available: https://www.bbc.com/mundo/noticias-53399101. [Último acceso: 20 agosto 2021].

[95] «el comercio,» [En línea]. Available: https://static1.elcomercio.es/www/multimedia/202005/20/media/cortadas/arienza%20(4)-kdR-U110243605634X6G-1248x770@El%20Comercio.JPG. [Último acceso: 30 agosto 2021].