FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2017/2018

Trabajo de Fin de Máster

DEEP LEARNING: REDES NEURONALES CONVOLUCIONALES APLICADAS A LA CLASIFICACIÓN DE VIENA

Alumno: José Luis Díaz Soto

Tutor: Javier Portela García-Miguel

Septiembre de 2018

UNIVERSIDAD COMPLUTENSE
MADRID

“Well, the harder I practice, the luckier I get.”,
Gary Player.

RESUMEN

El objetivo de este proyecto es la realización de un estudio para la clasificación de nuevos elementos figurativos de las marcas (logotipos) atendiendo a la Clasificación de Viena, mediante la utilización de redes neuronales profundas (convolucionales).

Para ello se ha dispuesto de datos reales proporcionados por la Agencia Europea de la Oficina de Propiedad Intelectual (EUIPO), con más de 26GB de información correspondiente a las 487.556 marcas registradas durante el periodo comprendido entre los años 2013 y 2017 en dicha organización.

Este estudio pretende servir de preámbulo e impulso para la resolución de la identificación de nuevos logotipos.

Las redes neuronales convolucionales son un sistema habitualmente utilizado en el reconocimiento y clasificación de imágenes, por lo que parece razonable abrir esta vía.

Palabras clave: Aprendizaje profundo, CNN, Redes Neuronales Convolucionales, Clasificación de Viena, Logos, EUIPO, marcas registradas.

ABSTRACT

The goal of this project is the performance of a study for the classification of brands' new figurative elements according to Vienna Classification, through the use of deep neural networks.

To this end, real data has been provided by the European Agency of the Intellectual Property Office (EUIPO), with more than 26GB of information corresponding to the 487.556 trademarks registered during the period between 2013 and 2017 in mentioned organization.

This study aims to serve as a preamble and impulse for the resolution of the identification of new logos.

Convolutional neural networks are a system commonly used in the recognition and classification of images, so it seems reasonable to open this way.

Keywords: Deep learning, CNN, Convolutional Neural Networks, Vienna Classification, Logos, EUIPO, registered trademarks.

AGRADECIMIENTOS

Este proyecto se ha estado fraguando durante casi dos años, entre las ciudades de Alicante y Madrid. Finalizar un máster de este calibre siempre requiere de grandes esfuerzos y sacrificios personales y profesionales. Con cuarenta años se hace si cabe más complicado.

Este es mi segundo título universitario, al igual que en el primero, no habría sido posible sin todos aquellos que de manera incondicional han compartido esos esfuerzos y sacrificios. A todos aquellos que han apartado piedras de mi camino sin pedírselo, incluso antes de que yo mismo fuese consciente de su existencia, o me han levantado cuando me hubiera gustado rendirme… GRACIAS.

Os quiero dedicar y agradecer este éxito, y por supuesto compartirlo con todos vosotros.

Gracias a mi familia, a los que están y a los que ya no, especialmente a mis padres y mi hermano. A los amigos de siempre (Dani, Sergio, Leo), a Laura, a los más recientes (Alejandro, Gabriel, Paco). A tantos y tantos grandes compañeros y profesores que he encontrado no solo en este máster, sino durante toda mi vida académica, y en especial a Javier Portela, por apostar por mí y este proyecto.

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

ÍNDICE DE CUADROS

ACRÓNIMOS

ARIPO	African Regional Intellectual Property Office
CNNs	Convolutional Neural Networks
CPDs	Centros de Procesamiento de Datos
CSV	Comma-Separated Values
EUIPO	European Union Intellectual Property Office
EUTMs	European Union Trade Marks
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
JPG	Joint Photgraphic Group
PIL	Python Imaging Library
OCR	Optical Character Recognition
OMPI	Oficina Mundial de Propiedad Intelectual
OpenCV	Open Source Computer Vision
ReLu	Rectified Linear Units
RGB	Red, Green, Blue
TFM	Trabajo Fin de Máster
TPU	Tensor Processing Unit
VGGNet	Visual Geometry Group Net
XML	Extensible Markup Language

1 INTRODUCCIÓN

1.1. Motivación del proyecto

Tras el acuerdo de Viena en 1973 se establece la clasificación de Viena, también conocida como Clasificación Internacional de los Elementos Figurativos de las Marcas [1].

Esta clasificación se utiliza en más de 60 oficinas nacionales y 3 internacionales, entre las que destaca la Oficina de la Propiedad Intelectual de la Unión Europea (EUIPO). Solamente en el periodo comprendido entre 2013 y 2017 se han registrado casi medio millón de marcas únicamente en EUIPO.

El avance en las técnicas de aprendizaje automático supervisado, y más concretamente en el hardware necesario para poder llevarlo a cabo, ha facilitado que se busquen soluciones automatizadas a problemas que se venían resolviendo de forma manual. A partir de 2012, la aplicación de las redes neuronales convolucionales revolucionó la clasificación automática de imágenes.

Parece razonable, y un reto apasionante y necesario, utilizar la visión artificial para afrontar el reto de buscar una solución automática que resuelva una tarea que históricamente ha sido resuelta por el ser humano.

1.2. Objetivos y enfoque

El objetivo de este estudio es la realización de un clasificador de imágenes capaz de automatizar el proceso de clasificación de los elementos figurativos (logos) incluidos en los registros de marcas de EUIPO.

Figura 1: Oficinas de EUIPO

Alcanzar un nivel de fiabilidad asumible por esta institución para la sustitución del proceso manual que se lleva a cabo en la actualidad es extremadamente ambicioso, y probablemente fuera del alcance con los recursos de los que se dispone para la realización de este TFM.

Se persigue desde un primer momento conseguir un producto fácilmente escalable para facilitar el trabajo de posteriores estudios que pudieran estar dotados de mayores recursos y que consigan mejorar nuestros resultados iniciales.

Para la evaluación de nuestro clasificador se ha procedido a la creación de dos conjuntos de datos diferentes dispuestos de la siguiente forma:

- Datos de entrenamiento y validación, se dispondrá de una base de datos de entrenamiento (90% de conjunto), y otro (10% del conjunto) que no se utilizará para el entrenamiento, sobre el que se realizarán las pruebas de validación (auto-test).
- Datos de test, con la misma composición que los datos de entrenamiento, pero no habiendo sido nunca utilizados para entrenar ni auto-validar la red.

Se procederá a construir los conjuntos de datos a partir de los descargados de la web oficial de EUIPO. Los elementos figurativos están clasificados en 30 clases descritas en la clasificación de Viena.

La principal dificultad del estudio radica en que las clases no son excluyentes, lo que hace que los elementos figurativos puedan pertenecer a un mismo tiempo a más de una clase.

Esto condicionará la estrategia tanto de extracción y transformación de los datos como del posterior entrenamiento de las redes e interpretación de sus resultados.

A continuación mostramos algunos ejemplos del conjunto de datos y una breve descripción de las diferentes categorías.

Figura 2: Ejemplos elementos figurativos

Clase	Detalle	Clase	Detalle
1	Cuerpos celestiales y meteorológicos,..	**16**	Telecomunicaciones, fotografía…
2	Seres humanos	**17**	Horología, joyería…
3	Animales	**18**	Transporte, equipamiento para animales
4	Fantasía, seres no identificables,…	**19**	Contenedores, embalajes, …
5	Plantas	**20**	Material de dibujo, escritura, pintura…
6	Paisajes	**21**	Juegos, juguetes, artículos deportivos…
7	Construcciones, edificios	**22**	Instrumentos musicales, campanas…
8	Productos alimenticios	**23**	Armas, municiones,armerías
9	Textiles, ropa,..	**24**	Heráldica, monedas, emblemas
10	Tabaco, carteras, paraguas, cerillas,..	**25**	Motivos ornamentales
11	Utensilios para el hogar	**26**	Cuerpos y figuras geométricas
12	Muebles, útiles sanitarios,..	**27**	Formas de escritura y números
13	Iluminación, maquinas de secado,..	**28**	Inscripciones en diferentes carácteres
14	Ferretería, herramientas, escaleras	**29**	Colores
15	Maquinaria, motores	**98**	-

Cuadro 1: Breve descripción de categorías

1.3. Alcance

Para la realización de este estudio hemos analizado más de 26GB de información, referentes a 487.556 marcas registradas en EUIPO y 226.584 elementos figurativos asociados a dichas marcas en el periodo comprendido entre 2013 y 2017.

Se han generado más de 100GB de información repartidos entre imágenes (miniaturas de diferentes tamaños de las imágenes originales), archivos de log y errores, archivos de datos de entrada del clasificador, redes neuronales y pesos de las mismas, resultados de los entrenamientos de las diferentes redes generadas y resultados de las predicciones realizadas sobre casi 40.000 imágenes para cada una de las 3 arquitecturas diseñadas.

Para cada arquitectura se han generado 30 redes neuronales convolucionales, una por cada clase correspondiente a la clasificación de Viena.

1.4. Estructura de la memoria

La memoria de este proyecto está dividida en los siguientes capítulos:

- Capítulo 1. Introducción.
- Capítulo 2. Estado del Arte.
- Capítulo 3. Entorno: Hardware y Software.
- Capítulo 4. Estructura de datos.
- Capítulo 5. Redes Neuronales Convolucionales.
- Capítulo 6. Nuestro clasificador.
- Capítulo 7. Experimentos y resultados.
- Capítulo 8. Conclusiones y trabajo futuro.
- Capítulo 9. Código.
- Referencias y anexo.

2
ESTADO DEL ARTE

Las redes neuronales convolucionales nacieron en los años 90 pero en aquel momento no se contaba ni con la capacidad de cálculo ni con los grandes volúmenes de datos de entrenamiento que existen en la actualidad. Aunque en un principio nacieron como una aplicación en el análisis de imágenes en la actualidad se utiliza también para temas relacionados con sonidos.

2.1. Mejoras en el hardware

El crecimiento de las redes neuronales convolucionales se ha dado en el último septenio gracias en parte a los avances producidos en el diseño de los procesadores de las tarjetas gráficas y a la disposición de grandes conjuntos de datos que sirven de materia prima de los estudios realizados. Esto ha hecho que muchas empresas tecnológicas hayan apostado por esta tecnología y hayan desarrollado sus propios entornos de trabajo y librerías, entre ellos Google.

El hardware es realmente la base de la revolución de la inteligencia artificial y de la visión artificial del último septenio. La metodología y la tecnología ya existía a finales de los años 90, sin embargo fueron los avances en las tarjetas gráficas y más concretamente en sus procesadores, denominados GPU (Graphics Processing Unit), los que provocaron una significativa mejora.

Las GPU es donde se realizan todos los cálculos matemáticos destinados a que nuestro ordenador sea capaz de representar y manipular una imagen, por ello su alto desarrollo impulsó en igual medida el desarrollo del "deep learning" y la clasificación de imágenes.

Existen también productos hardware destinados únicamente a aprender: las TPU (Tensor Processing Unit), un producto desarrollado por Google para mejorar el aprendizaje neuronal de computadores.

2.2. Principales librerías y entornos de trabajo

En el último lustro son muchas las empresas que se han sumado a la carrera de las CNNs, una de las que más fuerte ha apostado ha sido sin lugar a dudas Google.

No solo ha creado su propia herramienta de trabajo, conocida como TensorFlow, sino que en 2015 la liberó poniéndola en manos de muchos desarrolladores ávidos de esta tecnología. TensorFlow está basada en DistBelief, que fue el primer sistema de aprendizaje automático de Google. La principal diferencia entre ambas radica en que TensorFlow puede ejecutarse en múltiples CPU's mientras que DistBelief (el sistema de referencia) solo puede hacerlo en dispositivos aislados.

En Mayo de 2016 Google anuncia su TPU (Tensor Processing Unit), un circuito integrado, desarrollado exclusivamente para el aprendizaje continuo.

Todo esto ha provocó que, como declaró Jeff Dean (de Google), en Junio de 2016 1,500 repositorios en GitHub mencionasen TensorFlow, de los cuales solo 5 eran de Google.

Además, en 2017 TensorFlow decide apoyar al que en un principio fuese un competidor que inicialmente también desarrollaba su propia librería, Keras, y este pasa a convertirse en un alidado. Keras es actualmente un API que facilita el uso de TensorFlow.

Colaboraty es un laboratorio compartido que ofrece Google y que permite probar en un entorno basado en Jupyter las CNNs que definamos sobre GPUs y llegar incluso a compartirla con otros usuarios así como añadir notas. Además permite la utilización de fórmulas matemáticas ya que se integra con LaTeX.

También ha democratizado el desarrollo de la inteligencia y visión artificial, haciendo que muchas empresas estén desarrollando sus propios productos basados en estas tecnologías.

Existen otros marcos de trabajos como Caffe (el más veterano de todos, bastante robusto pero muy poco flexible) y otras librerías como Theano, muy similar a TensorFlow pero con el inconveniente de que no soporta ejecución múltiple, lo que ha hecho que se quede atrás con respecto a la librería de Google.

2.3. ILSVRC

Imagenet es una base de datos de gran escala, imprescindible en la investigación sobre visión artificial, que contiene más de un millón de imágenes clasificadas en más de 20.000 categorías diferentes.

El equipo de Imagenet organiza anualmente la competición ImageNet Large Scale Visual Recognition Challenge (ILSVRC), que sirve de laboratorio a equipos de investigación de todo el mundo y mide las soluciones desarrolladas enfrentándolas entre sí con un subconjunto de imágenes clasificadas en mil categorías diferentes.

Entre los ganadores de este reto se encuentran algunas de las arquitecturas de clasificación de imágenes más reconocidas del mundo como pueden ser AlexNet que marcó un antes y un después, ya que fue la primera en utilizar redes profundas, desde entonces los resultados han mejorado llegando a superar los resultados obtenidos por el ser humano.

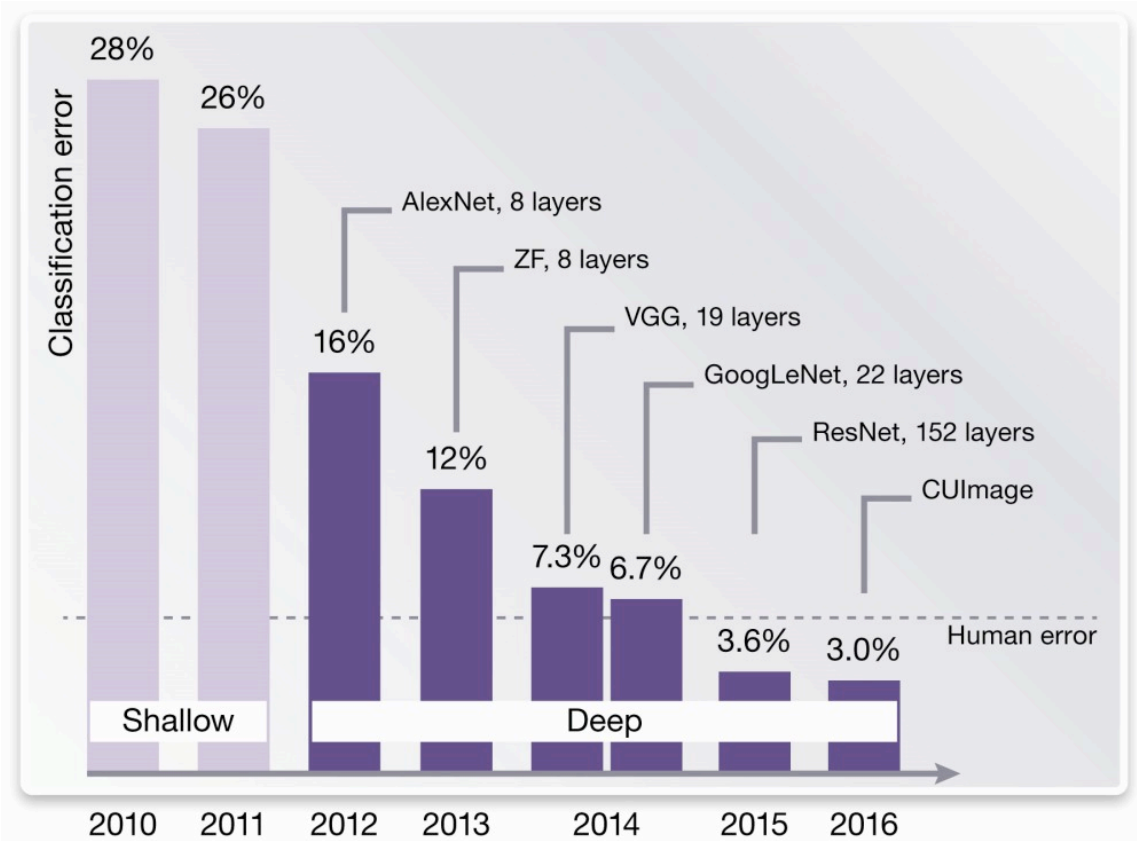


Figura 3: Gráfica de resultados del reto ILSVRC

2.4. Aplicaciones de las CNNs

Las redes neuronales convolucionales tienen infinidad de aplicaciones en el campo del reconocimiento, clasificación y segmentación de imágenes. Veánse algunos ejemplos:

- Generación automática de color, a través del entrenamiento con las CNN y a partir de fotos o películas en blanco y negro podemos obtener formatos en color de esas imágenes.
- Reconstrucción de imágenes, a partir de imágenes pixeladas de baja resolución se consigue regenerar la imagen original.
- Descripción de contenido, partiendo de una imagen somos capaces de describir los elementos que contiene dicha imagen.
- Estimación de pose en tiempo real, basándonos en una secuencia de video podemos identificar las personas que aparecen en ella y describir los movimientos de esas personas en tiempo real.
- Análisis del comportamiento en tiempo real, podemos determinar basado en el movimiento y los gestos la actitud de las personas. Por ejemplo, en la cola de un cajero.
- Traducción de imágenes, esto se utiliza tanto con fines comerciales para traducir las imágenes impresas en envases de productos, como en la traducción de documentos mediante reconocimiento de caracteres (OCR).
- Automóviles autónomos, mediante visión artificial el automóvil es capaz de conducir de forma autónoma.

Recientemente también se ha comenzado a hacer uso de la CNN para temas relacionados con audio y música. Algunos ejemplos son:

- Reconocimiento de voz, se pueden llegar a reconocer órdenes e incluso a clasificar acentos (uso forense).
- Reconocimiento de sonidos, para por ejemplo a partir de ellos detectar incidencias en electrodomésticos, como frigoríficos.
- Síntesis musical, se entrena un conjunto de datos para que sea capaz de crear música de forma autónoma.
- Clasificación géneros musicales, estos sistemas también son capaces de clasificar música atendiendo a su género.

3
ENTORNO: HARDWARE Y SOFTWARE

En este capítulo detallaremos el software y hardware utilizado para la creación de los conjuntos de datos así como para el entrenamiento y evaluación de las redes diseñadas.

3.1. Software utilizado

Para la realización de este estudio nos hemos decantado por software libre y especializado en el tratamiento de imágenes.

Figura 4: Software utilizado

3.1.1 Entorno de desarrollo

Hemos utilizado Anaconda Navigator. Esta aplicación nos permite tener instalados diferentes entornos a la vez en uno solo y cambiar de uno a otro de una forma sencilla.

Se trata de un entorno de trabajo para Python, muy sencillo de instalar y que también facilitará enormemente la instalación de este lenguaje y otros paquetes de trabajo necesarios y específicos para trabajar con redes neuronales en el tratamiento de imágenes.

Figura 5: Entorno de trabajo

La versión inicial incluye Spyder, un entorno de desarrollo integrado (IDE) multiplataforma de código abierto para programación científica con Python. Spyder se integra con una serie de paquetes muy utilizados en la programación científica y necesarios para nuestro estudio, como NumPy, SciPy o Pandas.

Jupyter es otro IDE que viene instalado de serie y muy útil cuando es necesario ejecutar pequeñas porciones de código.

3.1.2 Lenguaje de programación

Figura 6: Python

Python es el lenguaje de programación preferente para el tratamiento de imágenes. Muchas de las herramientas que se han desarrollado para trabajar con inteligencia artificial, machine learning, o deep learning están desarrolladas en este lenguaje de programación que copa el mercado en este sector, junto a R.

3.1.3 Librerías

El último apartado del engranaje del entorno de software son las librerías. Nos hemos apoyado en dos librerías básicas para el tratamiento de imágenes.

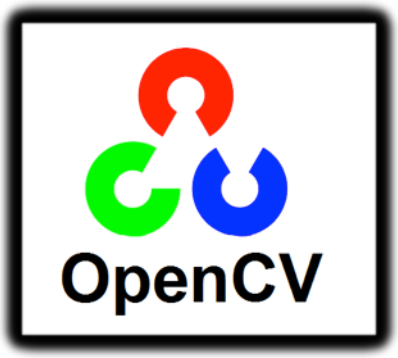


Figura 7: OpenCV

OpenCV (Open Source Computer Vision) es una librería de programación de código abierto dirigida principalmente a la visión por computador en tiempo real.

Python Imaging Library(PIL), es una biblioteca gratuita para el lenguaje de programación Python que facilita la apertura, manipulación y guardado de imágenes.

Figura 8: TensorFlow

La popularidad de TensorFlow se disparó cuando en 2015 Google Brain liberó su código. Unido a Keras han democratizado el desarrollo de la inteligencia y visión artificial.

3.2. Hardware utilizado

Para este estudio no se ha dispuesto de recursos como GPUs o recursos compartidos, lo que sin duda no afecta en ejemplos más sencillos, seguramente ha mermado los resultados obtenidos, por eso se propondrá como medida futura la aplicación sobre recursos técnicos más complejos.

En nuestro caso el hardware utilizado se resume en la siguiente tabla.

MacBook Pro (Retina, 15 pulgadas, finales de 2013)
Procesador 2 GHz Intel Core i7
Memoria 16 GB 1600 MHz DDR3
Intel Iris Pro 1536 MB

Cuadro 2: Descripción técnica Equipo MAC

NAS Synology DS218Play	
Procesador 4 núcleos de 64 bits a 1,4 GHz	
Memoria	- Memoria del sistema: 1GB DDR4

Cuadro 3: Descripción técnica NAS

4
ESTRUCTURA DE DATOS

En este capítulo trataremos la forma en la que se organizan los datos originariamente y ofreceremos una breve introducción acerca de su fuente. También describiremos el proceso de transformación para llegar a obtener los conjuntos que utilizaremos para el entrenamiento, validación y testeo de nuestro clasificador.

4.1. EUIPO y clasificación de Viena

La clasificación de Viena, también conocida como Clasificación Internacional de los Elementos Figurativos de las Marcas, se estableció en 1973 como criterio de ordenación en los procesos de registro de nuevos elementos figurativos (logos) de las marcas.

Cuando se procede a registrar un nuevo logo, se debe realizar una búsqueda en el inventario para contrastar con los elementos previamente registrados y poder afirmar que no entra en conflicto con ningún ninguno de ellos.

EUIPO ha registrado millones de marcas desde que comenzó su andadura como oficina de armonización del mercado interior de la Unión Europea hace más de 20 años. Es fácil suponer que buscar un elemento de similares características entre una cantidad tan ingente no resulta sencillo.

Por ello, el primer paso para poder llevar a cabo dicha tarea de forma eficaz y eficiente es la clasificación de nuestro inventario y de todo elemento nuevo que vaya a formar parte del mismo.

La clasificación de Viena es utilizada por más de 60 organismos nacionales de otros tantos países y por otros dos organismos internacionales, además de EUIPO, cabe destacar la Organización Regional Africana de la Propiedad Intelectual (ARIPO).

La clasificación de Viena distribuye los elementos figurativos en 29 categorías que constituyen un sistema jerárquico que procede de lo general a lo particular ya que a su vez se dividen en diferentes divisiones y secciones. EUIPO adapta, al igual que el resto de organizaciones, la clasificación a sus necesidades, pero esto no altera el objeto de nuestro estudio ya que no modifica las categorías que es el nivel de clasificación que pretendemos predecir.

La OMPI (Oficina Mundial de Propiedad Intelectual) publica las versiones auténticas de la clasificación de Viena (en inglés y en francés) tanto en formato impreso como digital.

4.2. Datos originales

Los datos originales que han servido como base de los conjuntos de entrenamiento, validación y testeo han sido recabados desde la propia página web de EUIPO [2].

Estos se encuentran organizados por años, comprimidos en formato *'zip'* y distribuidos en dos tipos de archivos:

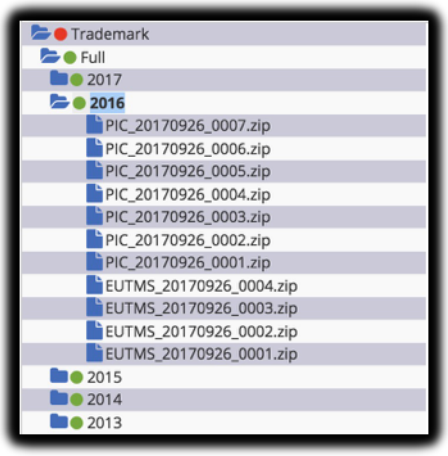


Figura 9: Fuente de los datos

- Archivos EUTMS, los cuales contienen directorios de archivos *'xml'* que guardan la información completa del nuevo registro, incluida la ruta dónde se ha almacenado la imagen del nuevo elemento figurativo, si lo hubiere.

- Archivos de imagen, contienen directorios de imágenes de los elementos figurativos en la ruta descrita en la etiqueta del archivo *'xml'*.

Los archivos pertenecientes al año 2017 aún no han sido terminados de cargar por EUIPO, por lo que al no estar el año completo, preferimos utilizar para el entrenamiento los correspondientes al periodo 2013-2017, pero cabe mencionar que en la página web EUIPO proporciona datos desde el año 1996.

4.3. Preparación de los datos

Hasta obtener un conjunto definitivo con el que poder entrenar nuestro modelo, debemos pulir y transformar los datos originales de los que partimos.

4.3.1 Filtrado de XMLs

El primer paso para la formación de nuestro conjunto de datos es la descarga y descompresión de los datos originales. Recorremos los archivos del árbol de directorios y accedemos a la información de cada archivo *'xml'*. Cada archivo contiene la información sobre el registro de una marca.

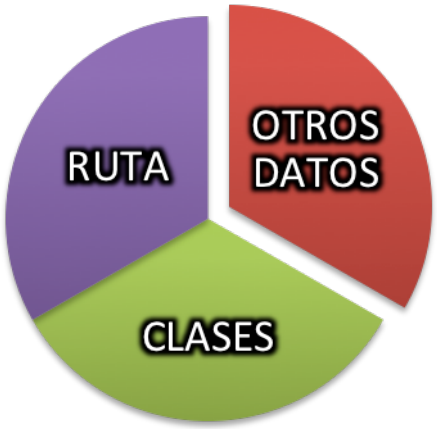


Figura 10: Datos en XMLs

Filtramos aquellos que contienen un elemento figurativo que responda a un formato 'JPG', extrayendo su ruta.

Dentro del árbol de etiquetas también encontramos otra que incluye una lista con las categorías, divisiones y secciones a las que pertenece el elemento figurativo.

Creamos un archivo que denominaremos 'mapa.csv' , a medida que recorremos el árbol de directorios se irán añadiendo filas a ese archivo CSV.

Cada fila corresponde a la información de un elemento figurativo que cumple las condiciones, e incluye:

- Una primera columna con la ruta relativa hasta la imagen.
- Una segunda columna en la que se indican las clases o categorías a las que pertenece el elemento figurativo.

Figura 11: Composición del mapeo de datos

El resultado del archivo es similar a este:

```
1  ruta,clase
2  016/742/016742488/000136312714.JPG,25-27-27-27-29-29
3  016/742/016742736/000136321516.JPG,21
4  016/742/016742496/000136312960.JPG,27
5  016/742/016742553/000136316600.JPG,27
6  016/742/016742629/000136318681.JPG,26-26
7  016/742/016742744/000136321520.JPG,02-07-25-25-25-26-29-29
8  016/742/016742728/000136321426.JPG,05-05-05-26-26-27
9  016/742/016742793/000136323590.JPG,26-26-26-26-26-26
```

Figura 12: Resultado archivo del mapeo

4.3.2 Procesamiento de imágenes

El archivo anterior se reutilizará como entrada para el siguiente procedimiento, de esta forma no es necesario repetir los pasos dados hasta este punto cada vez que se desee aplicar un tamaño diferente de imágenes para entrenar el modelo o describir las imágenes de una manera distinta.

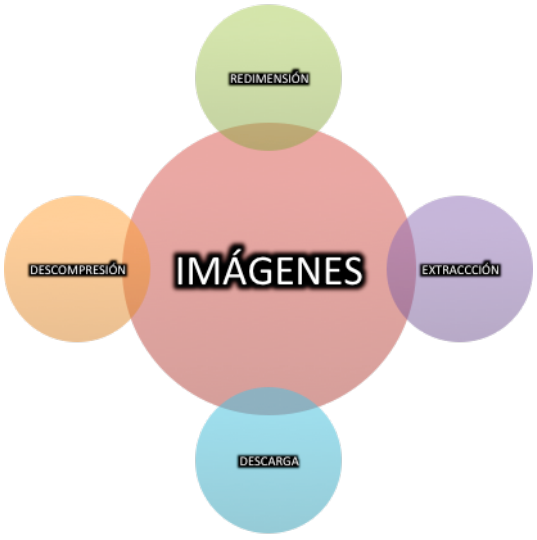


Figura 13: Fases de procesamiento de imágenes

Tras descargar y descomprimir las imágenes el siguiente paso será armonizar el tamaño de las imágenes, escogiendo un formato determinado y redimensionando.

En este estudio hemos redimensionado cada imagen a diferentes tamaños, 28x28, 32x32, 64x64, 128x128 e incluso 256x256.

A continuación se describen las imágenes redimensionadas, para ello se escoge la escala de grises, de esta forma el nivel de profundidad uno, en lugar del nivel tres de haber escogido RGB.

Además, el color solamente tiene importancia en una de las categorías; la 29. Y no es realmente el color lo que marca la pertenencia a esta categoría, sino la distribución del mismo.

Cuando se disponga de un modelo definitivo para nuestra red, este archivo será único. Pero para poder comparar no solamente probando diferentes estructuras sino con diferentes conjuntos de datos crearemos varios, combinando las diferentes opciones de tamaño para la redimensión de las imágenes y metodología para la extracción de las imágenes redimensionadas.

Para cada combinación de tamaño y método de extracción de características se obtendrá un archivo CSV. Cada fila de ese nuevo archivo se corresponde con una de las imágenes redimensionadas.

	clase	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5
1	01-01-02-04-05-17-18	244	244	246	248	250	253
2	27	255	255	255	255	255	255
3	01-27-27-27	254	254	254	254	254	254
4	16-26-26	251	255	255	254	253	255
5	07-26-26	254	254	254	254	254	254
6	27	255	255	255	255	255	255
7	26-27-27	255	255	255	255	255	255

Figura 14: Resultado extracción de características

La primera columna es la variable dependiente u objetivo (que en este caso serán las clases o categorías a las que pertenece el logo), el resto de columnas se corresponden con las variables que describen la imagen. En este estudio cada una de esas columnas indican el valor en escala de grises de un pixel determinado.

Por tanto tendremos un *'Input Shape'* de NxNx1 columnas para describir la imagen.

4.3.3 Binarización de las clases

Tras contemplar diferentes formas de presentar los datos y haber probado con todas ellas, nos decantamos por 'binarizar' los resultados.

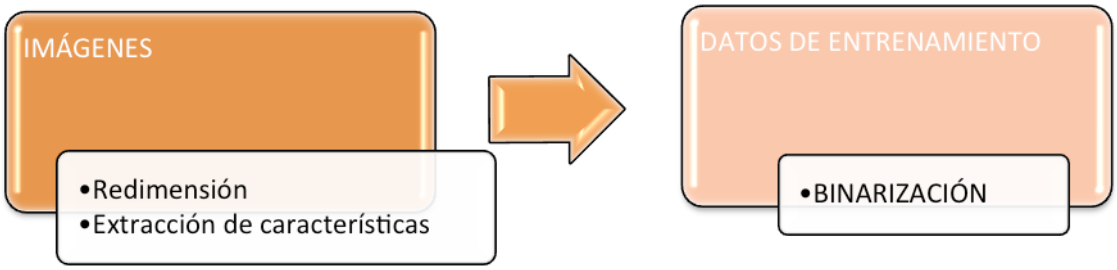


Figura 15: Binarización de las clases

El punto diferenciador de este estudio con otras clasificaciones de imágenes es que los elementos figurativos pueden pertenecer a más de una clase. Sin ser estas excluyentes, lo que complica la disposición de los datos de entrenamiento y de las predicciones. La primera columna no es manejable, la red no podrá interpretarla por lo que optamos por "binarizarla".

Se reemplaza la primera columna por 30 columnas nuevas (tantas columnas como clases) y para cada registro damos valor 1 a las columnas de las clases a las que pertenece la imagen y 0 a aquellas a las que no pertenece.

En la imagen solo se muestran las siete primeras. Tras estas nuevas columnas se siguen manteniendo las correspondientes a las características de la

	clase_1	clase_2	clase_3	clase_4	clase_5	clase_6	clase_7
1	1	1	0	1	1	0	0
2	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0

imagen.

Figura 16: Resultado binarización de clases

Para implementar el modelo se construyen tantas redes como clases tenemos y cada una ha sido entrenada para reconocer si el elemento pertenece a esa clase concreta o no. En el momento de la carga de datos para realizar el entrenamiento, el sistema seleccionará la columna correspondiente a la clase que vamos a entrenar, la cual será la variable dependiente sustituyendo a la columna 1 de la figura X.

4.4. Segmentación de los datos

Distinguiremos entre conjunto de datos de entrenamiento y validación, y conjunto de datos de testeo.

Hemos preparado el sistema para poder introducir una "semilla" de forma manual o aleatoria que divida el primer conjunto en dos partes, el sistema escogerá el 90% del archivo como datos de entrenamiento y el otro 10% como datos de validación (auto-testeo).

Para los datos de entrenamiento y validación usaremos los datos referentes a los años del periodo 2013-2016. Mientras que para los datos de testeo se usarán los datos del año 2017. De esta forma nos aseguramos de no sobreajustar la red usando como datos de testeo algunos de los datos del entrenamiento.

Figura 17: Distribución de datos

5
REDES NEURONALES CONVOLUCIONALES

En este capítulo se enumeran las ventajas de las redes convolucionales sobre las redes neuronales con capas densas, se describe brevemente la composición y las operaciones básicas de las redes neuronales convolucionales.

5.1. Ventajas de las redes neuronales convolucionales

Una red neuronal se compone de tres tipos de capas: Capa de entrada, capa de salida y capas ocultas. Para considerarse aprendizaje profundo, deben existir dos o más capas ocultas. [11]

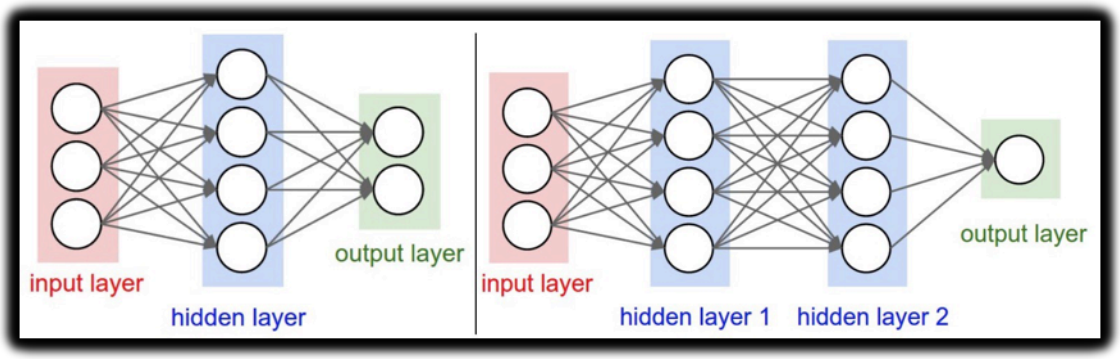


Figura 18: Redes convolucionales

Esas capas ocultas serán las encargadas de detectar características de la imagen de entrada usando diferentes niveles de abstracción, no es necesario hacer ingeniería de características, será la propia red la que aprenda automáticamente a detectar esos patrones que definen la imagen, siendo capaces de modelar relaciones altamente no lineales entre entradas y salidas.

La ventaja de las redes convolucionales es que una vez que aprende a detectar una determinada característica lo hace para toda la imagen y no se limita a la región detectada.

Las redes neuronales convencionales calculan los pesos de las diferentes combinaciones de las neuronas y evalúan los mejores resultados. Cuando se aplican sobre imágenes resulta más complejo, ya que el número de variables aumenta y por tanto el de posibles combinaciones. Debemos tener en cuenta que si tenemos una imagen de 28x28 pixeles y 3 canales (RGB) el número de variables que describen la imagen será de 28x28x3.

Se trata de imágenes, por la naturaleza de este elemento, las variables siguen un orden lógico, cada variable muestra el valor de un pixel. Es posible evaluar combinaciones de variables organizadas en regiones en lugar de relaciones de variables una a una y mediante aplicación de filtros reducir la información y/o transformarla discriminando la menos relevante.

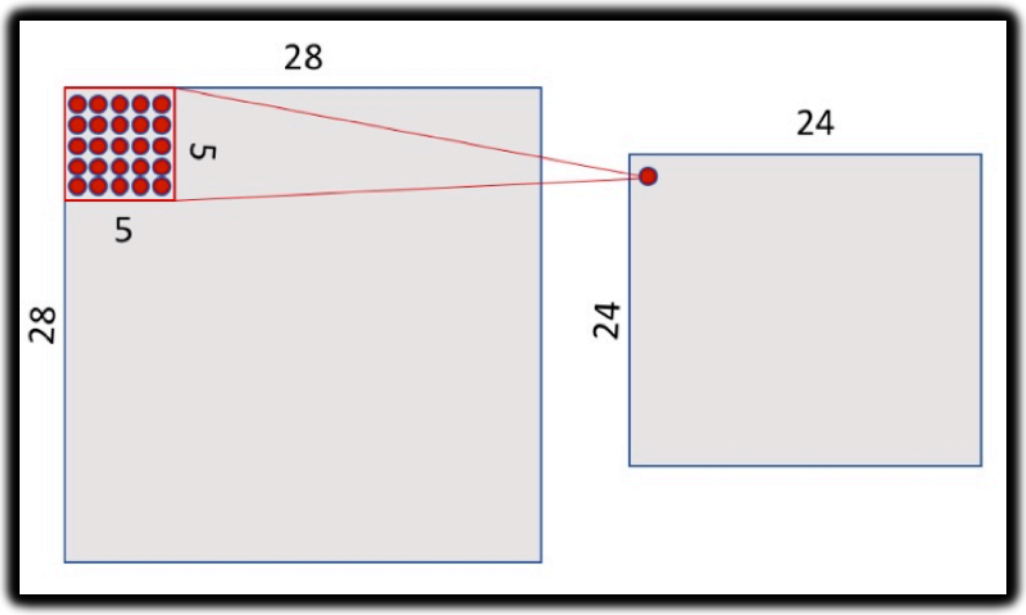


Figura 19: Convolución

A este proceso se le denomina convolución y su aplicación implica estar ante un tipo especial de redes neuronales, las redes neuronales convolucionales.

5.2. Componentes básicos de las CNNs

Convolucionar consiste en iterar sobre la matriz de valores de los pixeles de una imagen, aplicando un filtro y operación matemática para conseguir reducir y/o transformar la información discriminando la menos relevante.

Dependiendo del filtro, su tamaño, su desplazamiento y la operación matemática conseguiremos detectar un tipo u otro de característica de la imagen.

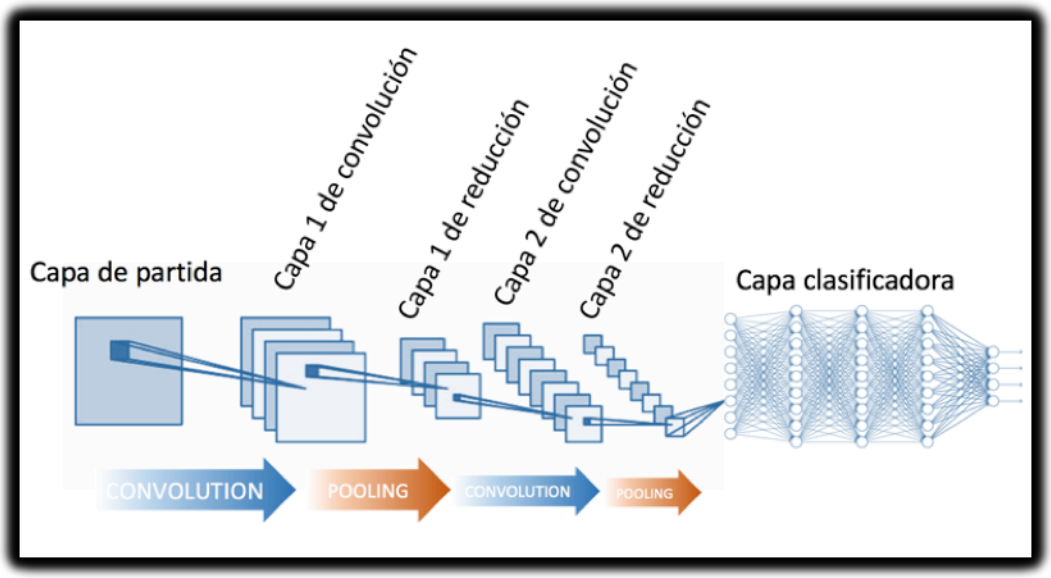


Figura 20: Componentes básicos de las CNNs

El rendimiento de una red depende de muchos factores; datos, infraestructura, pero también en buena parte del talento del arquitecto para diseñar un buen modelo, simplemente variando el orden de una de las redes, modificando la función de activación o incluyendo alguna más los resultados se pueden ver muy afectados. El número de capas no es directamente proporcional al éxito del modelo.

5.2.1 Funciones de activación

La función de activación representa simultáneamente la salida de la neurona y su estado de activación a partir de una entrada.

Los requisitos que vamos a pedir a una función de activación son los siguientes: Monótona creciente, no lineal, acotada y derivable salvo un conjunto finito de puntos.

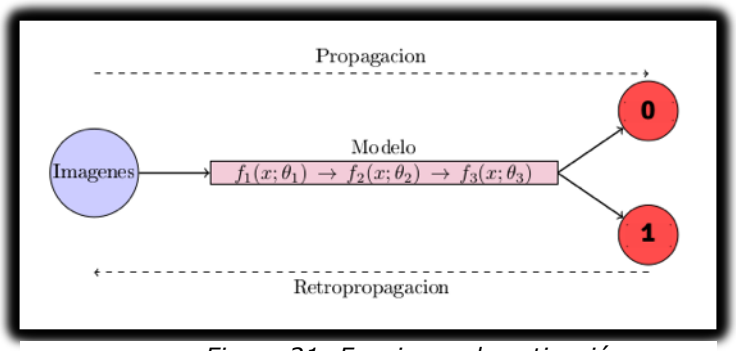


Figura 21: Funciones de activación

Estos son algunos ejemplos:

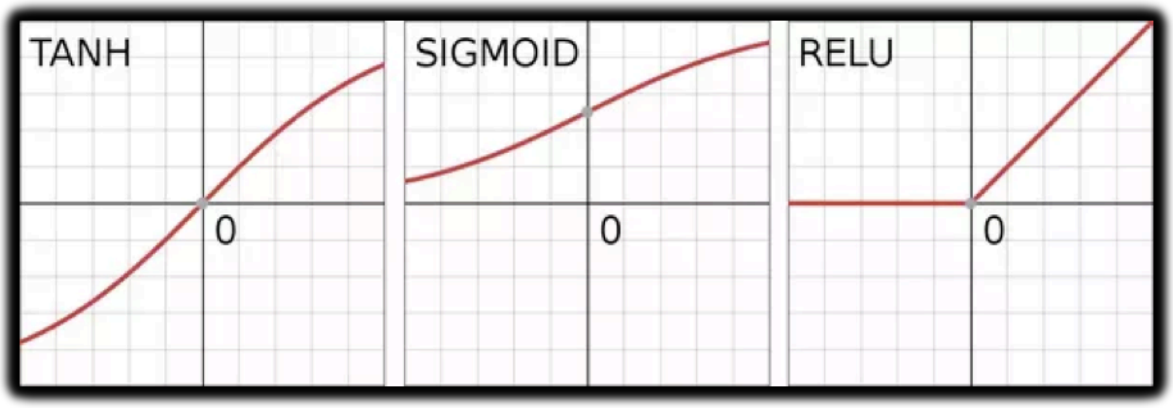


Figura 22: Ejemplos funciones de activación

5.2.2 Hiper-parámetros de una capa

1. Filtro (Kernel)

Se define por un tamaño de la ventana (ancho y alto), una profundidad (número de canales) y unos coeficientes para aplicar sobre cada pixel. La red convolucional calcula los coeficientes de estos filtros de forma automática. Todo el proceso de entrenamiento esta orientado precisamente a buscar estos coeficientes.

2. Desplazamiento (Stride)

Al tamaño del desplazamiento del filtro sobre la matriz de valores en cada iteración se le denomina "stride". Cuanto mayor sea ese desplazamiento más se reducirá el tamaño de la imagen original.

3. Relleno (Padding)

Permite obtener una matriz del mismo tamaño de la imagen original pero destacando las regiones más significativas. Se consigue añadiendo un marco de ceros alrededor de la matriz antes de comenzar a operar, con lo que "agrandamos" la matriz para después mediante la convolución reducirla a su tamaño original pero destacando los valores más significativos ya que realmente estamos añadiendo contraste a la imagen.

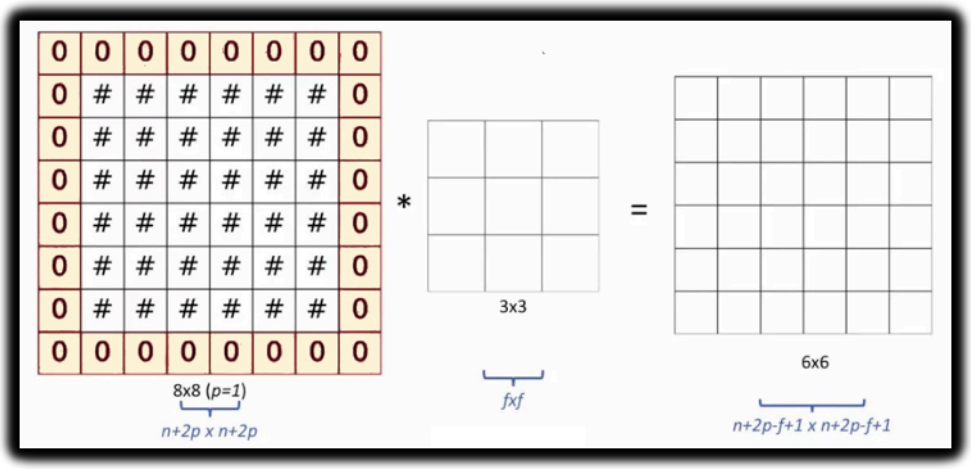


Figura 23: Padding

5.2.3 Tipos de operaciones

1. Operación de convolución

Esta operación consiste, como se menciona anteriormente, en iterar un filtro sobre la imagen original. El resultado de la salida se obtiene del sumatorio de los productos entre los elementos de las mismas posiciones en la ventana de la iteración correspondiente y del filtro.

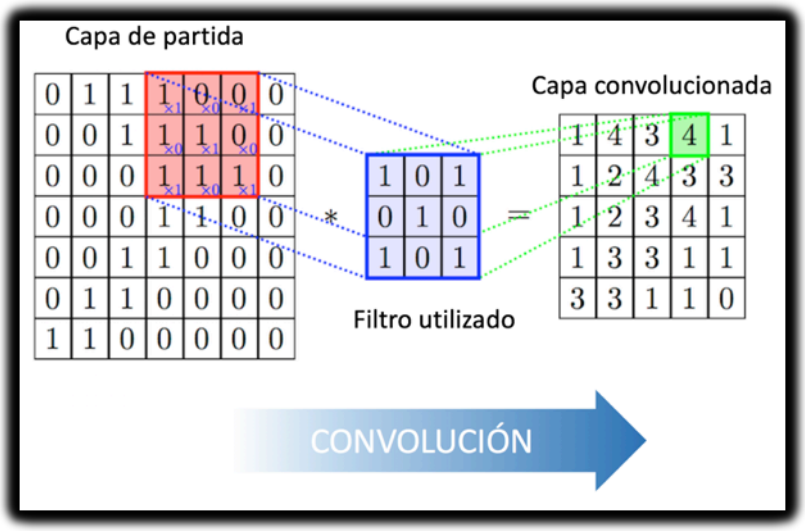


Figura 24: Operación de convolución

De esta forma, reducimos el tamaño de la salida y resaltamos dependiendo del filtro utilizado una u otra característica.

2. Operación Pooling (Reducción)

Tenemos diferentes operaciones de reducción o condensación de la información, la más habitual es la búsqueda del máximo dentro de la ventana. A continuación vemos un ejemplo práctico de cómo partiendo de una imagen de 4x4, con un filtro de 2x2 y un "stride" de 2, conseguimos reducir el tamaño de la salida y además preservamos los elementos más relevantes. [11]

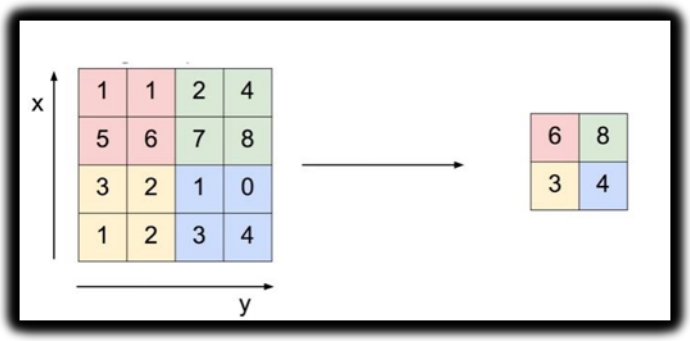


Figura 25: Operación de Pooling

La salida muestra el máximo de la ventana de cada iteración, pero hay otras opciones, como podría ser la media de la ventana (average pooling).

Estas operaciones son también utilizadas en los algoritmos para la compresión de imágenes. A continuación vemos un ejemplo más práctico de lo que hace el algoritmo con un dígito. [7]

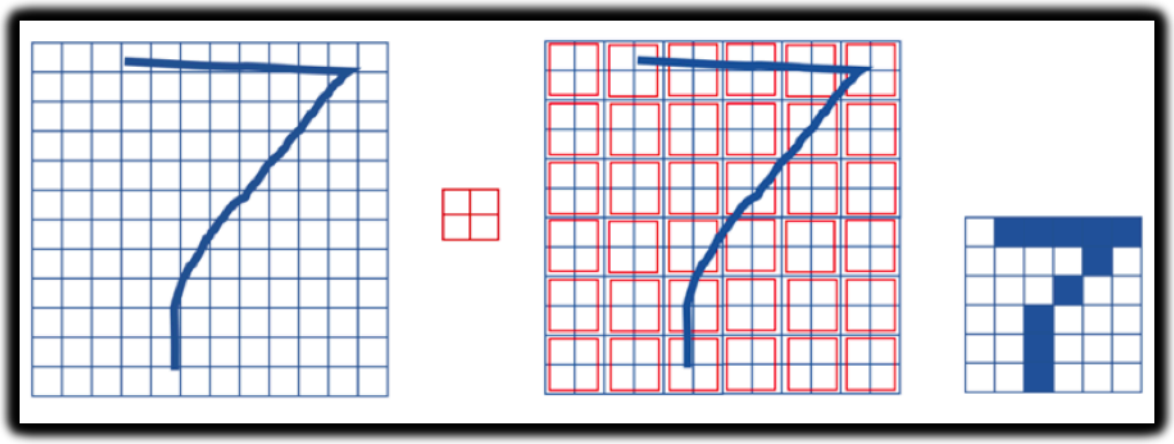


Figura 26: Ejemplo operación de pooling

5.3. Redes reconocidas

Existen arquitecturas de redes neuronales convolucionales que tienen nombre propio dentro de la comunidad de Deep Learning.

5.3.1 LeNet

Una de ellas es LeNet, creada por Yann LeCun en los años noventa, con ella consiguió el primer éxito de redes neuronales convolucionales en la lectura de códigos postales y dígitos.

Los datos de entrada eran imágenes de 32x32 y la arquitectura de su red extremadamente sencilla, se componía de dos etapas de convolución-pooling, una capa densamente conectada y una capa softmax final que nos permite reconocer los números.

5.3.2 AlexNet

AlexNet de Alex Krizhevsky, ganadora del ILSVRC del año 2012, marcó un antes y un después en la historia de las redes neuronales convolucionales y de ImageNet. Incluida en el top-5 de redes con menor error de la historia de este reto, AlexNet sigue una arquitectura muy similar a LeNet pero incluye capas convolucionales sucesivas por primera vez en la historia de los ganadores de esta competición. Hasta ese momento a una capa convolucional siempre le seguía una de reducción.

5.3.3 VGGNet

La ganadora de competición en 2014 contribuyó a demostrar que la profundidad de las capas es una componente crítica para la obtención de buenos resultados. Aquí podemos ver su código en Python utilizando Keras.

```
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten
from keras.layers import Conv2D
from keras.layers import MaxPooling2D

input_shape = (224, 224, 3)

model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=input_shape, padding='same',activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same',))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(256, (3, 3), activation='relu', padding='same',))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same',))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same',))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(512, (3, 3), activation='relu', padding='same',))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same',))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same',))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(512, (3, 3), activation='relu', padding='same',))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same',))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same',))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dense(4096, activation='relu'))
model.add(Dense(1000, activation='softmax'))
```

Figura 27: Código de VGGNet

6
NUESTRO CLASIFICADOR

En este capítulo se describe la arquitectura del modelo diseñado y el proceso de entrenamiento que se han llevado a cabo sobre el mismo.

6.1. Arquitectura del modelo

El tipo de capas de un modelo, el número de ellas, su orden, los hiper-parámetros, funciones de activación y operaciones aplicadas: todos estos elementos influirán en el rendimiento del modelo.

Se componen como un lego, se pueden combinar de innumerables formas y del talento del arquitecto depende en parte su éxito. Al igual que en una receta de cocina estos elementos no se incluyen aleatoriamente, tienen un sentido y al igual que en la alta cocina, mayor cantidad de elementos no implica un mejor resultado.

Existen arquitecturas reconocidas para la resolución de problemas relacionados con la clasificación de imágenes, entre ellas algunas de las expuestas en el apartado cinco, que manejan estructuras muy complejas que no están al alcance de los recursos e infraestructuras con que se cuentan en este estudio.

El diseño de nuestra red responde al siguiente esquema

Layer (type)	Output Shape	Param #
conv2d_29 (Conv2D)	(None, 30, 30, 32)	320
conv2d_30 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d_12 (MaxPooling	(None, 14, 14, 64)	0
dropout_3 (Dropout)	(None, 14, 14, 64)	0
flatten_4 (Flatten)	(None, 12544)	0
dense_9 (Dense)	(None, 128)	1605760
dropout_4 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 2)	258

Total params: 1,624,834
Trainable params: 1,624,834
Non-trainable params: 0

None

Figura 28: Esquema del clasificador

Se analizan más de un millón y medio de parámetros y para su composición se han utilizado: Dos capas convolucionales con filtros de 3x3 sin 'strides', una capa de reducción (max-pooling) con ventanas de 2x2 y sin 'strides' y dos capas densas.

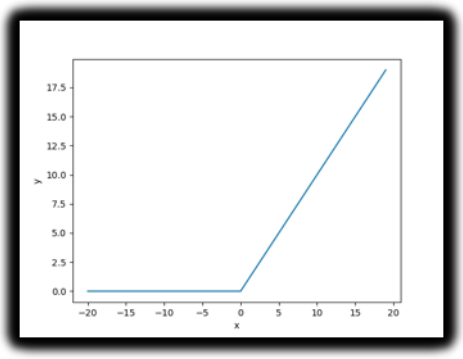


Figura 29: Función ReLu

La primera de las funciones de activación utilizadas ha sido la función *ReLu*, para las dos capas convolucionales y la primera de las capas densas.

Esta función obtiene tiempos de ejecución mucho más rápidos por ello se está utilizando en la mayoría de los modelos para clasificación de imágenes. Además, entornos como TensorFlow facilita el uso de las mismas de forma que no es necesario implementar su algoritmo.

$$f(x) \quad \max(x, 0)$$

La segunda de las funciones utilizadas ha sido la función *softmax,* esta función realiza una distribución categórica de la probabilidad, de forma que la suma de todas ellas sume un 100%. Esta función se utiliza en la capa clasificadora, es decir, la capa densa que ofrece los resultados finales.

A continuación, se muestra la codificación de nuestro modelo utilizando Python.

```
def estructura_A(input_shape, num_classes):

    model = Sequential()

    #Dos capas consecuitvas de convolución
    #Filtros de 3x3
    model.add(Conv2D(32, kernel_size=(3, 3),
                             activation='relu',
                             input_shape=input_shape))
    model.add(Conv2D(64, (3, 3), activation='relu'))

    #capa de reducción por 'max-pooling'
    #ventanas de 2x2, sin 'stride'.
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))

    #obtención de resultados en capa densa
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                          optimizer=keras.optimizers.Adadelta(),
                          metrics=['accuracy'])

    print(model.summary())
    return model
```

Figura 30: Código del clasificador

6.2. Entrenamiento del modelo

El entrenamiento de este modelo se ha llevado a cabo sobre todos los elementos figurativos del intervalo de años 2013-2016, sumando un total de 149.275 imágenes.

El entrenamiento del modelo para el conjunto de datos descrito ha llevado un total de 36 horas ininterrumpidos de ejecución. A este tiempo habría que añadir las más de dos horas de construcción del conjunto de entrenamiento.

Aquí observamos los resultados de ejecución durante el entrenamiento de una de las 30 clases.

```
Train on 149275 samples, validate on 37319 samples
Epoch 1/10
149275/149275 [==============================] - 405s 3ms/step - loss: 0.0844 - acc: 0.9845 - val_loss: 0.0736 - val_acc: 0.9866
Epoch 2/10
149275/149275 [==============================] - 412s 3ms/step - loss: 0.0805 - acc: 0.9850 - val_loss: 0.0733 - val_acc: 0.9866
Epoch 3/10
149275/149275 [==============================] - 420s 3ms/step - loss: 0.0796 - acc: 0.9850 - val_loss: 0.0703 - val_acc: 0.9866
Epoch 4/10
149275/149275 [==============================] - 418s 3ms/step - loss: 0.0786 - acc: 0.9850 - val_loss: 0.0702 - val_acc: 0.9866
Epoch 5/10
149275/149275 [==============================] - 412s 3ms/step - loss: 0.0781 - acc: 0.9850 - val_loss: 0.0699 - val_acc: 0.9866
Epoch 6/10
149275/149275 [==============================] - 403s 3ms/step - loss: 0.0775 - acc: 0.9850 - val_loss: 0.0698 - val_acc: 0.9866
Epoch 7/10
149275/149275 [==============================] - 404s 3ms/step - loss: 0.0772 - acc: 0.9850 - val_loss: 0.0704 - val_acc: 0.9866
Epoch 8/10
149275/149275 [==============================] - 405s 3ms/step - loss: 0.0768 - acc: 0.9850 - val_loss: 0.0699 - val_acc: 0.9866
Epoch 9/10
149275/149275 [==============================] - 404s 3ms/step - loss: 0.0764 - acc: 0.9850 - val_loss: 0.0702 - val_acc: 0.9866
Epoch 10/10
149275/149275 [==============================] - 410s 3ms/step - loss: 0.0758 - acc: 0.9850 - val_loss: 0.0699 - val_acc: 0.9866
RED_9 Validation loss: 0.06991385160437631
RED_9 Validation accuracy: 0.9866019989833491
Modelo guardado en disco
```

Figura 31: Resultados entrenamiento de la clase 12

Al finalizar el entrenamiento se almacenan los metadatos generados y se guardan los resultados para poder reanudarlo con un conjunto de datos diferente en otro momento.

También se almacenan las gráficas que ayudan a la posterior interpretación. En este ejemplo de entrenamiento de las clases 26 y 24 se observan respectivamente tanto como aumenta el acierto como cómo desciende el error a medida que avanza el entrenamiento.

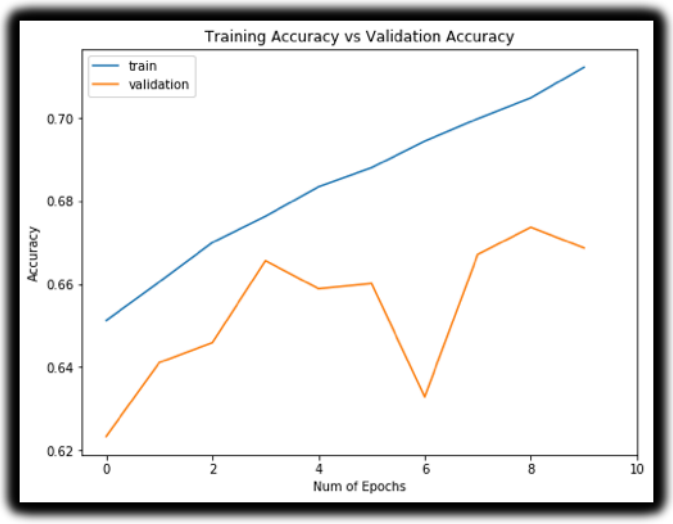


Figura 32: Aciertos de la clase 26

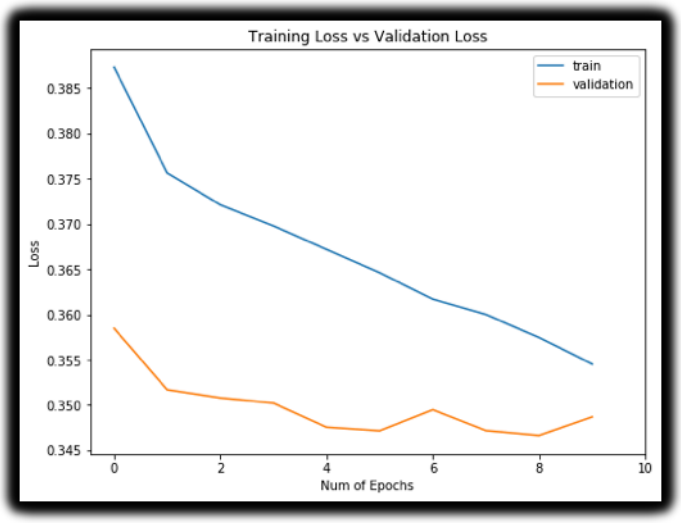


Figura 33: Error de la clase 24

7
EXPERIMENTOS Y RESULTADOS

Han sido innumerables los experimentos que se han realizado desde que comenzó el estudio, estos se han basado en iterar sobre tres variables: la formación de los datos de entrenamiento y validación, las diferentes arquitecturas que se pueden crear y diferentes formas de medir los resultados. A continuación se detallan algunos de ellos.

7.1. Experimentando con los datos

La materia prima de un estudio como este son los datos, los cientos de miles de imágenes y las clases a las que pertenecen, pero podemos transformarlos siguiendo cuatro criterios y obteniendo numerosas combinaciones posibles.

Hemos diseñado y encapsulado nuestro código de forma que podamos combinar rápidamente todas ellas.

7.1.1 Diferentes tamaños de las imágenes

Cada imagen original responde a un tamaño propio, es por tanto necesario establecer un estándar, para clarificar si los resultados variarían en función de cual se eligiese.

Parte del procedimiento de transformación de los datos, como se explica en el capítulo 4, trata sobre la redimensión de las imágenes. Se experimenta redimensionando las imágenes a tamaños de 28x28, 32x32, 64x64, 128x128, 256x256 e incluso 512x512.

El proceso de redimensión en sí no revierte mayor problema, aunque obviamente a mayor dimensión de la imagen mayor será el espacio necesario para su almacenamiento y mayor el tiempo para su posterior procesamiento. Se han llegado a obtener archivos que ocupaban decenas de GB de memoria.

El principal problema de una dimensión mayor es que obtendremos mayor número de características y por tanto de parámetros finales en nuestra red.

Teniendo en cuenta que 2016 tiene 40.000 imágenes, con un tamaño de 32x32 y un canal de color, el input-shape sería de (40.000x32x32x1), es decir un CSV de 40.000 filas y al menos 1024 columnas. Si aumentamos la dimensión de las imágenes a 256x256 el input-shape pasaría a ser de (40.000x256x256x1), tendríamos 65536 columnas.

Al convolucionar lo que se hace es combinar estas características, por lo que es fácil intuir la dificultad que entrañará su procesamiento.

Se han creado conjuntos de datos de entrenamiento para todos los tamaños anteriormente mencionados, y se ha entrenado el clasificador elegido con algunos de ellos.

7.1.2 Diferentes extracciones de características

Originariamente se dispone de imágenes, se pueden elegir muchas formas de describir esas imágenes, pero es necesario que todas sean descritas bajo el mismo criterio en un mismo conjunto de datos de entrenamiento.

Inicialmente, al realizar el estudio del arte y decidir que técnicas y metodologías serían más convenientes para realizar este estudio, se valoraron otros algoritmos propios de la visión artificial como SIFT (Scale-invariant feature transform) e incluso se codificó un algoritmo para obtener descriptores y formar conjuntos de entrenamiento. Finalmente se descartó esta vía.

Se describen las imágenes mediante vectores con la escala de grises de los pixeles que las conforman. Esto ya reduce bastante la información necesaria (objetivo de las convoluciones) pues el color se entiende que no afecta al objetivo del estudio, tan solo hay una categoría que podría tocar tangencialmente y no es realmente el color lo importante sino su disposición en la imagen.

Por este motivo, se descarta representar la imagen con RGB (3 canales de información) y se opta por la escala de grises (1 canal de información).

7.1.3 Diferentes lotes de imágenes

En la fuente de datos (web de EUIPO) las imágenes se organizan por años de registro de las marcas.

Inicialmente se decide estudiar el año 2016. El año 2017 en la fecha en la que iniciamos nuestro estudio no se había terminado de poner a disposición de los usuarios de la web, y se entiende que si alguien quisiese reproducir este estudio debía disponer de un conjunto de datos que no fluctuase con respecto al utilizado.

Tras evaluar más de 110.000 registros de marcas se obtienen más de 40.000 imágenes clasificadas. Se entiende que la muestra sería suficiente para formar un conjunto de datos de entrenamiento.

Sin embargo, al hacer un estudio de la distribución de las imágenes en función de las clases a las que pertenecían, se observa que hay algunas clases muy representativas y otras que no lo son tanto.

Por ello decidimos obtener los datos para los años 2013, 2014, 2015 y 2016, para posteriormente transformarlos y obtener conjuntos de entrenamiento para todos ellos y las posibles combinaciones de los mismos. Se dispone de un conjunto de datos para el intervalo total de estos años de 150.000 imágenes, que es el finalmente utilizado para entrenar el modelo.

Se decide también utilizar el año 2017 como conjunto de datos de testeo, para la posterior evaluación del modelo.

7.1.4 Diferentes estructuras de variable objetivo

El principal problema al que se hace frente en este estudio es que las categorías o clases en que se engloban las imágenes no son excluyentes y de hecho es bastante habitual que una imagen pertenezca a más de una.

Planteamos hasta tres formas de resolver este problema.

1. **Monolabel**

Si una misma imagen pertenece a dos categorías diferentes, no se debe asignar un valor de la variable objetivo a su descripción de la imagen para posteriormente a la misma descripción de la imagen asignarle un valor diferente. Estaríamos confundiendo a nuestra red y no se obtendrían resultados concluyentes.

Se valora incluir en el conjunto de datos solamente aquellas imágenes "puras", es decir, que solo pertenezcan a una única clase. Esto resolvería el problema, pero merma aún más las frecuencias de algunas clases y obtendríamos resultados que indicarían solamente la clase predominante de los nuevos elementos figurativos que evaluásemos.

Esta fue la primera fase de nuestro estudio y se llegaron a resultados aceptables con aciertos de más del 98% para las clases más representativas.

clase	COUNT	PERCENT
1	623	4,77
2	474	3,63
3	231	1,77
4	142	1,09
5	186	1,42
6	83	0,64
7	210	1,61
8	63	0,48
9	119	0,91
10	33	0,25
11	50	0,38
12	16	0,12
13	33	0,25
14	80	0,61
15	157	1,20
16	71	0,54
17	101	0,77
18	103	0,79
19	142	1,09
20	124	0,95
21	65	0,50
22	20	0,15
23	25	0,19
24	673	5,15
25	459	3,51
26	1435	10,98
27	6886	52,69
28	463	3,54
98	2	0,02

Cuadro 4: Frecuencias de red 'monolabel'

2. Multiclase

Una vez alcanzado el objetivo del apartado anterior se decide profundizar un poco más, pues no resolvía el problema inicial que motivó este estudio. La mayoría de elementos figurativos se clasifica en más de una categoría y la estrategia anterior no resuelve ese problema.

Se nos ocurre experimentar teniendo en cuenta solamente las dos clases más representativas del año 2016 (clase 26 y clase 27). Creamos un nuevo conjunto de datos de entrenamiento teniendo en cuenta las cuatro posibles combinaciones, y asignamos todas las imágenes de ese año a esas cuatro nuevas categorías.

Nueva Categoría	Pertenecen a ella todas las imágenes que
0	No pertenecen a la clase 26 ni 27
1	Pertenecen al menos a la clase 26 y no a la 27
2	Pertenecen al menos a la clase 27 y no a la 26
3	Pertenecen al menos a las clases 26 y 27

Cuadro 5: Recategorización multiclase

Funciona, tiene una probabilidad de acierto muy alta (más de un 98%) pero hay que tener en cuenta que se ha estudiado sobre las dos clases más representativas.

Además, se observa un problema. El número de neuronas de salida para poder resolver el problema del estudio sería de 2^n siendo n el número de clases iniciales. Por tanto, sería de 2^{30} ó (dicho de otra forma) 1.073.741.824 neuronas de salida, un conjunto inmanejable.

3. Binarización

Finalmente está ha sido la opción escogida, se detalla en el capítulo 4, supone la creación de 30 redes y 30 conjuntos de datos de entrenamiento para cada lote de imágenes.

Responde a la siguiente estructura para cada clase:

Nueva Categoría	Pertenecen a ella todas las imágenes que
0	No pertenecen a la clase de estudio
1	Pertenecen a la clase de estudio

Cuadro 6: binarización

Los resultados son más difíciles de interpretar, al testear se evaluará cada nuevo elemento sobre las 30 redes y la probabilidad de cada logo de pertenecer a cada una de las clases, por tanto la suma de probabilidades de todas ellas no será 100%.

7.1.5 Estructura de conjuntos de datos

Durante este estudio, se han combinado los criterios anteriores dando como resultado la generación de numerosos conjuntos de datos de entrenamiento.
Se han organizado estos archivos dentro de esta estructura local:

'\outputs\datos\entrenamiento\{año}\{tipo}\{método}_{tamaño}_{tamaño}.csv'

Por ejemplo, para el conjunto de datos de entrenamiento de los elementos figurativos del año 2016 con un tamaño de 32x32, siguiendo el método de extracción de características de escala de grises y un criterio de categorización binaria la ruta donde encontraríamos el archivo sería la siguiente:

'\outputs\datos\entrenamiento\2016\binaria\EG_32_32.csv'

7.2. Experimentando con las redes

Además de con la red descrita en el capítulo 6, se ha utilizado la misma arquitectura con otros conjuntos de datos, por ejemplo habiendo usado imágenes de mayor tamaño. Esto da como resultado dos macro-redes diferentes, entrenadas con conjuntos diferentes.

También se prueba a modificar la arquitectura de la red y usar el mismo conjunto de datos.

7.3. Diferentes formas de organizar los resultados

Llegados a este punto diseñamos y codificamos métodos para poder comparar los ensayos que estábamos evaluando y llegar a puntos concluyentes.

Para cada entrenamiento hemos almacenado los resultados:

- En un archivo CSV;
  - Fecha y hora del comienzo de la ejecución y número de épocas.
  - Parámetros de formación del archivo de entrenamiento (año, método de extracción de características, % de distribución para entrenamiento y validación, tamaño de las imágenes y tipo de categorización).
  - Red utilizada y arquitectura asociada a la red.
  - Resultados de probabilidad de acierto final de la red y de perdida.

- En imágenes;
  - Gráfico con curvas del avance de resultados de entrenamiento y validación a través de las distintas épocas.

- En un archivo de texto;
  - Hemos incluido información sobre la ejecución de los procesos (logs) y los errores que se producían durante el mismo.

A medida que hemos evaluado las redes entrenadas hemos ido generando y almacenando más datos.

7.4. Valoración del rendimiento del clasificador

Para poder valorar el rendimiento del clasificador diseñado se necesita un conjunto de datos de testeo. Se procederá a evaluar la probabilidad de cada elemento figurativo del conjunto de pertenecer a cada una de clases de la clasificación, es decir, estamos evaluando el conjunto global de las 30 redes convolucionales que conforman nuestro clasificador, de ahí la dificultad de la estrategia de "binarización".

7.4.1 Resultados del proceso de testeo

El proceso de testeo consta de 4 fases.

1. Formación del conjunto de testeo.
    - Se han utilizado los elementos figurativos registrados en el año 2017 como conjunto de testeo, un total de 37.897 elementos.
    - Estos datos obviamente no han formado parte nunca del entrenamiento de ninguna de las redes. Son extraídos en momentos diferentes, y almacenados en directorios distintos.

2. Cálculo de la matriz de resultados.
    - Se ha generado una matriz con las probabilidades de cada uno de los 37.897 elementos de pertenecer a cada clase de la clasificación.

3. Interpretación de la matriz de resultados.
    - A partir de la matriz de resultados anterior se genera otro archivo con las clases reales a las que sabemos que pertenece cada elemento y las clases predichas por el clasificador.
    - Se utilizan distintos criterios para entender una clase como clase predicha por el clasificador. El primer criterio es escoger la clase con mayor acierto.

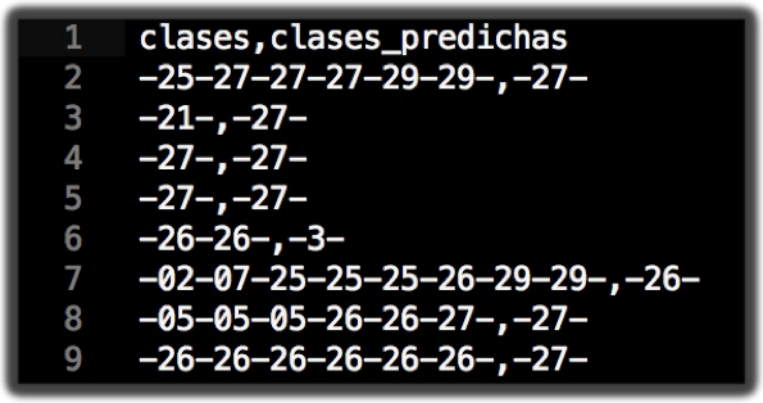


Figura 34: Matriz de resultados

4. Cálculo de la matriz de confusión.

   Esta sería la matriz de confusión para una clase (ejemplo 19).

La observación es positiva si el elemento evaluado pertenece a la clase 19 y negativa si no lo hace. La predicción es positiva si la clase 19 está entre las que hemos predicho para el elemento.

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

Figura 35: Matriz de confusión

- **VP,** nº de *positivos* que fueron *clasificados c*omo positivos (acierto).
- **VN**, nº de *negativos* que fueron *clasificados* como negativos (acierto).
- **FN,** nº de *positivos* que fueron *clasificados* como negativos (error).
- **FP,** nº de *negativos* que fueron *clasificados* como positivos (error).

De la matriz de confusión podemos extraer otros cuatro valores que nos ayudan a evaluar nuestro clasificador.

Exactitud: porcentaje de elementos clasificados correctamente.

(VP+VN)/TOTAL

Tasa de error: o porcentaje clasificado incorrectamente.

(FP+FN)/TOTAL

Precisión: porcentaje de observaciones positivas logra clasificadas correctamente.

VP/TOTAL POSITIVOS

Especificidad: porcentaje de observaciones negativas clasifica correctamente.

VN/ TOTAL NEGATIVOS

7.4.2 Criterios de interpretación de los resultados

El modelo ofrece como resultados las probabilidad de cada imagen de pertenecer a cada clase. Se debe concluir un criterio que interprete ese resultado.

Como se ha mencionado en el apartado anterior se han estudiado diferentes criterios para a raíz de la probabilidad de pertenencia de un elemento a una clase determinada describir esa predicción como positiva. Se han descrito tres criterios:

1. Valor máximo.
Se considerará una predicción positiva a la clase con la mayor probabilidad.
Este criterio resulta poco flexible. Una imagen podría pertenecer a varias clases y este criterio no nos ofrece esa posibilidad sin penalización. Aumentar el número de valores tampoco resuelve el problema, pues no hay un número concreto de clases a las que pueda pertenecer.

2. Umbral fijo.
Se considerará una predicción positiva para una clase cuando la probabilidad de pertenecer a dicha clase sea mayor que un umbral definido (por ejemplo del 50%).Se observa que variando ese umbral se consiguen mejores resultados con unas clases que con otras. Tiene sentido utilizar un umbral para cada clase.

3. Umbral variable
Se considerará una predicción positiva para una clase cuando la probabilidad de pertenecer a dicha clase sea mayor que un umbral definido. Planteamos la posibilidad de definir ese umbral como la mediana de los resultados para cada clase. También se plantea estudiar los resultados usando la media.

En el siguiente capítulo se menciona la posibilidad de apoyar la utilización de estos criterios en otras técnicas como el análisis de sentimiento para mejorar las predicciones. A continuación se muestran los valores más interesantes de la matriz de confusión usando el criterio del mayor valor.

Clase	VP	VN	FP	FN	Acierto	Error	Precisión	Especificidad
19	79	37234	46	538	0,98	0,02	0,63	0,99
24	397	32693	507	4300	0,87	0,13	0,44	0,88
25	114	30558	107	7118	0,81	0,19	0,52	0,81
26	5764	18507	3349	10277	0,64	0,36	0,63	0,64
27	13769	8532	12515	3081	0,59	0,41	0,52	0,73
29	310	29164	410	8013	0,78	0,22	0,43	0,78

Cuadro 7: Clases destacadas de la matriz de confusión

Se muestra también la matriz completa de confusión de nuestro clasificador para el criterio de la probabilidad mayor.

Se considerará como predicción positiva la clase con la mayor probabilidad.

Algunos valores que inicialmente pueden parecer buenos pueden resultar engañosos. Hay clases con tasas de acierto muy altas pero que no están proporcionalmente representadas en el conjunto de testeo. Por esa razón no se destacaron en la matriz del *cuadro 7.*

Clase	VP	VN	FP	FN	Acierto	Error	Precisión	Especificidad
1	15	34036	21	3825	0,9	0,1	0,42	0,9
2	32	34693	53	3119	0,92	0,08	0,38	0,92
3	101	34785	159	2852	0,92	0,08	0,39	0,92
4	0	37278	0	619	0,98	0,02	0	0,98
5	22	34678	59	3138	0,92	0,08	0,27	0,92
6	0	37338	0	559	0,99	0,01	0	0,99
7	0	36904	0	993	0,97	0,03	0	0,97
8	0	37482	0	415	0,99	0,01	0	0,99
9	0	37450	0	447	0,99	0,01	0	0,99
10	0	37731	0	166	1	0	0	1
11	0	37402	0	495	0,99	0,01	0	0,99
12	0	37821	0	76	1	0	0	1
13	0	37766	0	131	1	0	0	1
14	0	37524	0	373	0,99	0,01	0	0,99
15	0	37339	0	558	0,99	0,01	0	0,99
16	0	37511	0	386	0,99	0,01	0	0,99
17	0	37567	0	330	0,99	0,01	0	0,99
18	0	37131	0	766	0,98	0,02	0	0,98
19	79	37234	46	538	0,98	0,02	0,63	0,99
20	0	37521	0	376	0,99	0,01	0	0,99
21	0	37454	0	443	0,99	0,01	0	0,99
22	0	37781	0	116	1	0	0	1
23	0	37748	0	149	1	0	0	1
24	397	32693	507	4300	0,87	0,13	0,44	0,88
25	114	30558	107	7118	0,81	0,19	0,52	0,81
26	5764	18507	3349	10277	0,64	0,36	0,63	0,64
27	13769	8532	12515	3081	0,59	0,41	0,52	0,73
28	0	36985	0	912	0,98	0,02	0	0,98
29	310	29164	410	8013	0,78	0,22	0,43	0,78
98	0	37705	0	192	0,99	0,01	0	0,99

Cuadro 8: Matriz de confusión para el criterio del mayor valor

7.4.3 Casos de interés

Sorprende observar la gran facilidad para clasificar botellas dentro de la categoría 19 que clasifica todo tipo de embalajes.

Figura 36: Clase 19 - Casos de interés -

La clase 24 es otro ejemplo, se han aprendido a clasificar emblemas, escudos y sellos con una alta tasa de acierto de 0,83% y 971 casos verdaderos positivos.

Figura 37: Clase 24 - Casos de interés -

8
CONCLUSIONES Y TRABAJO FUTURO

8.1. Conclusiones

Se han conseguido unos resultados mejores que los esperados inicialmente. Al aumentar el conjunto de entrenamiento del año 2016 al intervalo 2013-2016 se ha notado un aumento significativo de los elementos clasificados correctamente.

Existen datos desde el año 1996, por lo que se pretende seguir entrenando el modelo una vez generados los nuevos conjuntos de entrenamiento.

Ha sorprendido la capacidad del modelo para detectar no solo los elementos esperados (como los pertenecientes a las clases 26 y 27, las clases más pobladas), sino elementos de las clases 19, 24 y 29 de una forma óptima. Habiendo detectado 97 elementos de estas clases con una probabilidad de acierto de más del 90%.

Se ha mencionado en diferentes ocasiones que el reciente éxito de este tipo de algoritmos que desde los años 90 apenas había gozado de notoriedad se debe al desarrollo de las capacidades de cálculo de los procesadores de las tarjetas gráficas y a la gran cantidad de datos disponibles actualmente.

Durante los últimos 20 años, la industria tecnológica se ha dedicado básicamente a recabar y organizar datos. Para este estudio se disponen de los suficientes como para obtener resultados cada vez más alentadores, pero se debe tener en cuenta que la dificultad y complejidad de su propia naturaleza hace necesaria una gran cantidad de los mismos en comparación con otros problemas de clasificación de imágenes.

Hay otros cuatro aspectos fundamentales para conseguir un buen rendimiento del clasificador diseñado:

1. Mejorar la infraestructura.
   El más importante, de él dependen en buena medida el resto de aspectos.
   Acaba de mencionarse, en comparación con los años 90 las capacidades de cálculo del hardware son muy altas. Sin embargo, ni siquiera grandes empresas disponen de esa capacidad de cálculo y es por ello necesario compartir los recursos en la nube. Tan sólo dos o tres empresas en el mundo (Google, Amazon) tienen la capacidad de desarrollar esos CPD's gigantes compartidos y ponerlos a disposición de los demás.

2. La construcción de los conjuntos de datos.
   Es un aspecto en el que se ha incidido desde el comienzo del estudio. Las combinaciones son innumerables, atendiendo a las variables mencionadas en el capítulo cuatro. Sería conveniente probar el modelo para imágenes de mayor tamaño una vez resuelto el problema del punto uno.

3. La arquitectura del modelo.
   El modelo construido no es un modelo demasiado complejo en comparación con otras arquitecturas mencionadas en el estudio, de nuevo resolver el punto uno nos ayudaría a crecer en el desarrollo del modelo, permitiendo aplicar arquitecturas más complejas.

4. El análisis de los datos obtenidos.
   Atendiendo a todos los aspectos mencionados obtendremos unos resultados, mientras estos no dispongan de tasas de acierto cercanas al 100% no será factible su utilización como vía única de clasificación. Mientras esto no suceda puede ser interesante aprender a interpretar lo que el clasificador nos está transmitiendo, desarrollando más criterios para interpretar como predicha una clase a tenor de los resultados obtenidos.

8.2. Trabajo futuro

Se debe canalizar el esfuerzo por seguir algunas de las vías abiertas y seleccionar cuidadosamente que otras nuevas vías explorar con las estrategias ya mencionadas, pero a raíz de las pruebas realizadas y las conclusiones alcanzadas se han detectado posibles mejoras al estudio que no se han podido realizar en esta fase.

Pasamos a enumerar algunas.

1. Análisis de sentimiento de los registros de las marcas.

   En esos archivos hay una descripción del logo con lenguaje natural.
   Descartamos inicialmente estos datos que sin embargo podrían ayudar a afinar el resultado en aquellos casos en los que nos enfrentamos a elementos figurativos muy abstractos. También ayudaría a corregir la subjetividad del factor humano.

2. Transfer training

   Se ha probado a ejecutar la red VGGNet e intentar entrenarla con nuestros datos de entrenamiento, pero empezando de cero. Sin embargo la previsión de tiempos inmanejables de ejecución hizo desistir de esta vía.

   La técnica del "Transfer training", permite no tener que "reinventar la rueda" y acoplando nuestros conjuntos de datos a cualquier red ya entrenada previamente, incluidas las redes más contrastadas, seguir entrenándola y/o evaluarla.

3. Ejecutar en la nube

   Colaboratory podría ser un primer paso, se trata de un entorno de Google que trabaja con Keras y nos ofrece la posibilidad de ejecutar nuestro modelo en la nube. De esta forma minimizamos nuestra dependencia de recursos ya que se ejecuta sobre GPUs de gran capacidad.

   Además permite compartir los entornos con más colaboradores, añadir notas y almacenar resultados de las ejecuciones.

9
CÓDIGO

En este capítulo veremos partes significativas del código y procederemos a explicar los detalles más relevantes del mismo. Una exposición más detallada del código será incluida en el anexo.

9.1. Entrenamiento de la Red.

Se procede a entrenar una red que pasaremos como parámetro, le indicaremos también el año de los datos que utilizaremos y las clases que queremos entrenar. Para entrenar todas las clases con los mismos datos basta con dejar el último parámetro en blanco. La función que hemos utilizado es la siguiente.

```
def entrenar(año_entrenamiento=año_entrenamiento, id_red=id_red_omision, clases=constantes.clases):


    for clase in clases:
        (x_train, y_train), (x_validation, y_validation)=_cargar_datos_entrenamiento(clase, año_entrenamiento)
        modelo=_cargar_modelo(clase,id_red)
        _entrenar_modelo(modelo, clase, id_red, año_entrenamiento, x_train, y_train, x_validation, y_validation)
```

Figura 38: Entrenamiento de la red - Código -

9.1.1 Carga de los datos

Se ha codificado una función que distribuye y carga los dos conjuntos, el de entrenamiento y el de validación(auto-testeo). Disponemos de una constante que indica que tanto por ciento de los datos se dedicará a cada conjunto. También disponemos de otra constante que dependiendo del año que queramos entrenar indica la ruta de los datos, el año se lo indicamos como parámetro.

La clase que queremos entrenar es el otro parámetro de la función. Recordemos que hemos "binarizado" el conjunto de datos de entrada y disponemos de 30 columnas con datos binarios, una para cada clase, debemos indicar que clase vamos a entrenar para así seleccionar una u otra columna.

```
def _distribuir_datos_entrenamiento(clase, año_entrenamiento):

    file=constantes.input_data_entrenamiento_red_keras % (año_entrenamiento, tipo, metodo, tamaño, tamaño)
    data = pandas.read_csv(file)


    #el 90% de los datos los usamos como training
    bound = int(data.shape[0]*indice_entrenamiento)

    num_columna=_calcula_columna(clase);

    x_train, y_train = data.iloc[:bound,30:].values, data.iloc[:bound,num_columna].values
    x_test, y_test = data.iloc[bound:,30:].values, data.iloc[bound:,num_columna].values

    return (x_train, y_train), (x_test, y_test)
```

Figura 39: Distribuir datos de entrenamiento - Código -

También debemos amoldar esos datos a Keras, ya que serán la entrada de la red, adaptando los valores de las características a valores entre 0 y 1 y categorizando los resultados.

Con esto ya tenemos los datos cargados, una vez carguemos el modelo o creemos uno nuevo, podemos entrenarlo.

```
def _cargar_datos_entrenamiento(clase, año):

    # the data, shuffled and split between train and validation sets
    (x_train, y_train), (x_validation, y_validation) = _distribuir_datos_entrenamiento(clase, año)


    x_train = x_train.reshape(x_train.shape[0], img_cols, img_rows, 1)
    x_validation = x_validation.reshape(x_validation.shape[0], img_cols, img_rows, 1)



    print('epochs:', epocas)
    x_train = x_train.astype('float32')
    x_validation = x_validation.astype('float32')
    x_train /= 255
    x_validation /= 255

    print('x_train shape:', x_train.shape)
    print(x_train.shape[0], 'train samples')
    print(x_validation.shape[0], 'validation samples')


    # convert class vectors to binary class matrices
    y_train = keras.utils.to_categorical(y_train, num_clases)
    y_validation = keras.utils.to_categorical(y_validation, num_clases)

    return (x_train, y_train), (x_validation, y_validation)
```

Figura 40: Carga de datos de entrenamiento - Código -

9.1.2 Carga del modelo

Está función contempla dos escenarios:

- El modelo ya existe y ha sido previamente entrenado.
- El modelo no existe, en este caso se crea usando la arquitectura indicada.

```
def _cargar_modelo(clase, id_red):

    # cargar json y crear el modelo
    if(os.path.exists(constantes.output_red % (tipo, clase, id_red))):
        json_file = open(constantes.output_red % (tipo, clase, id_red), 'r')
        modelo_json = json_file.read()
        json_file.close()
        modelo = model_from_json(modelo_json)
        # cargar pesos al nuevo modelo
        modelo.load_weights(constantes.output_pesos_red % (tipo, clase, id_red))


        # Compilar modelo cargado y listo para usar.
        modelo.compile(loss=keras.losses.categorical_crossentropy,
                                optimizer=keras.optimizers.Adadelta(),
                                metrics=['accuracy'])

        #print("Cargado modelo desde disco.")
    else:

        os.makedirs(constantes.dir_red_keras % (tipo, clase, id_red))
        input_shape = (img_rows, img_cols, 1)
        modelo=modelos.estructura_A(input_shape, num_clases)
        #print("Creado nuevo modelo")

    return modelo
```

Figura 41: Carga del modelo - Código -

9.1.3 Entrenamiento del modelo

Entrenamos el modelo y lo evaluamos con el conjunto de validación. Guardamos la red y los pesos en archivos ".json" y ".h5" respectivamente para poder reentrenar el modelo en otro momento posterior.

```
def _guardar_modelo(modelo, tipo, clase, id_red):


    # serialize model to JSON
    modelo_json = modelo.to_json()
    with open(constantes.output_red % (tipo, clase, id_red), 'w') as json_file:
        json_file.write(modelo_json)
    # serialize weights to HDF5
    modelo.save_weights(constantes.output_pesos_red % (tipo, clase, id_red))
    print("Modelo guardado en disco")
```

Figura 42: Guardar modelo - Código -

Mostramos por pantalla los resultados y las gráficas obtenidos.
Y por último guardamos los resultados en un archivo CSV para poder contrastar los resultados de los entrenamientos con condiciones diferentes.

```
def _entrenar_modelo(modelo, clase, id_red, año_entrenamiento, x_train, y_train, x_validation, y_validation):

    init_date=time.strftime("%x")
    init_time=time.strftime("%X")
    history=modelo.fit(x_train, y_train,
                       batch_size=batch_size,
                       epochs=epocas,
                       verbose=1,
                       validation_data=(x_validation, y_validation))

    score = modelo.evaluate(x_validation, y_validation, verbose=0)

    print('RED_%d Validation loss:' % clase, score[0])
    print('RED_%d Validation accuracy:' % clase, score[1])

    end_date=time.strftime("%x")
    end_time=time.strftime("%X")

    #una vez entrenado lo salvamos para almacenar ese entreanmiento
    _guardar_modelo(modelo, tipo, clase, id_red)

    #guardamos los logs
    fout= open(constantes.output_entrenamientos_red % (tipo,clase),'a')
    linea=('%d,%s,%d,%d,%d,%s,%d,%d,%d,%f,%f,%s,%s,%s\n' % (id_red, metodo,año_entrenamiento,tamaño,clase, estructura,epocas,
                                                            x_train.shape[0],x_validation.shape[0],score[0],score[1],
                                                            init_date+' '+init_time,end_date+' '+end_time,tipo))
    fout.write(linea)
    fout.close()

    _imprimir_curvas_entrenamiento(history, clase, id_red)
```

Figura 43: Entrenamiento del modelo – Código -

9.2. Predicción de nuevos elementos figurativos

Se predice para una red determinada la totalidad del conjunto de testeo, guardando los resultados de cada clase en un archivo para su posterior estudio.

```
#predice probabilidad de las imagenes para la red y clases que le digamos
def predecir_red_completa(id_red=id_red_omision, clases=constantes.clases):

        #con la combinacion 0,0 realizara la prediccion de todas las imagenes de 2017
        posicion_inicial, numero_ejemplos=0, 0
        x_test, y_test, posicion=_cargar_datos_prediccion(posicion_inicial, numero_ejemplos)

        fout= open(constantes.output_testeo_multired % ('multiclase', id_red),'w')

        for clase in clases:
            print ("Evaluando todos los logos del año 2017 para la clase %d con la red %d \n" % clase)
            modelo=_cargar_modelo(clase, id_red)
            y_predict=modelo.predict(x_test, batch_size, verbose=0)
            linea=''

            num_imagenes=y_test.shape[0]
            for i in range(0,num_imagenes-1):
                linea=linea+str(round(float(y_predict[i,1])*100,2))+','

            #el ultimo
            linea=linea+str(round(float(y_predict[num_imagenes-1,1])*100,2))
            fout.write(linea+'\n')
        fout.close()
```

Figura 44: Predicción red completa - Código -

BIBLIOGRAFÍA

[1] http://www.wipo.int/classifications/nivilo/vienna/index.htm?lang=EN[en línea][consulta 08/09/2017]

[2] https://euipo.europa.eu/ohimportal/es/open-data [en línea] [consulta 08/09/2017]

[3] https://github.com/conseal/Library/blob/master/Digital_Recognizer_example.R [en línea] [consulta 11/09/2017]

[4] https://www.r-bloggers.com/a-little-h2o-deeplearning-experiment-on-the-mnist-data-set/ [en línea] [consulta 11/09/2017]

[5] Raúl López Briega. Redes Neuronales Convolucionales con TensorFlow, Agosto 2016. [en línea] https://relopezbriega.github.io/blog/2016/08/02/redes-neuronales-convolucionales-con-tensorflow/ [consulta 16/09/2017]

[6] Rubén López. ¿Qué es y cómo funciona Deep Learning? Mayo 2014. [en línea] https://rubenlopezg.wordpress.com/2014/05/07/que-es-y-como-funciona-deep-learning/[consulta 07/01/2018]

[7] Jordi Torres. Redes Neuronales Convolucionales, Junio 2018. [en línea] https://torres.ai/redes-neuronales-convolucionales/ [consulta 15/07/2018]

[8] A. Krizhevsky, I. Sutskever and G.E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems, Lake Tahoe, Nevada. NIPS 2012. [en línea] http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf [consulta 21/07/2018]

[9] Jesús Utrera Burgal. Deep learning básico con Keras, Julio 2018. [en línea] https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-2-convolutional-nets/ [consulta 03/08/2018]

[10] J Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang and G. Wang. Recent Advances in Convolutional Neural Networks. Journal Pattern Recognition Volumen 77, Issue C, May 2018. Elsevier Science [en línea]
preprint versión https://arxiv.org/pdf/1512.07108v5.pdf [consulta 17/8/2018]

[11] https://cs231n.github.io/convolutional-networks/ - case [en línea] [consulta 21/08/2018]

ANEXO

JUGANDO CON EL CLASIFICADOR

Se han realizado infinidad de pruebas que han ido mejorando el clasificador, mostramos aquí algunas de ellas.

A medida que evaluamos el conjunto de testeo vamos almacenando aquellos casos que superan el umbral del 90% para su posterior análisis. Se comparan los resultados de algunos de ellos utilizando la red 1 y la red 4. Ambas responden a la arquitectura de nuestro clasificador, la red 1 ha sido entrenada solamente con el año 2016, mientras la red 4 con el intervalo 2013-2016. Los resultados hablan por sí solos.

```
#estudia una imagen para la clase y redes que le indiquemos
def comparar_redes(clase, redes, posiciones):

    for posicion_inicial in range(0,len(posiciones)):
        x_test, y_test, posicion=_cargar_datos_prediccion(posiciones[posicion_inicial], 1)

        #img=Image.fromarray(x_test,'L')
        #Image._show(img)


        print ("\nEl elemento de la posición %d es de clase %s" % (posicion, y_test[0]))

        for red in range(0,len(redes)):
            modelo=_cargar_modelo(clase, redes[red])
            y_predict=modelo.predict(x_test, batch_size, verbose=0)
                    #snn_predicted = np.argmax(y_predict, axis=1)
            print ("La red %d predice que tiene un %.2f%% de posibilidades de ser clase %d" % (redes[red], float(y_predict[0,1])*100, clase))



red_keras.comparar_redes(24, [1,4], [20470, 35144, 37326])


El elemento de la posición 20470 es de clase 24-26-26-26-26-26-26
La red 1 predice que tiene un 58.16% de posibilidades de ser clase 24
La red 4 predice que tiene un 99.07% de posibilidades de ser clase 24

El elemento de la posición 35144 es de clase 24-26-26-26-26
La red 1 predice que tiene un 65.25% de posibilidades de ser clase 24
La red 4 predice que tiene un 99.60% de posibilidades de ser clase 24

El elemento de la posición 37326 es de clase 01-01-03-05-24-24-24-24-24-24-24
La red 1 predice que tiene un 52.56% de posibilidades de ser clase 24
La red 4 predice que tiene un 99.84% de posibilidades de ser clase 24
```

Elemento 20470

Elemento 35144

Elemento 37326

Como se indica en el estudio, se ha probado a entrenar otras arquitecturas, entre ellas la reconocida red neuronal VGGNet.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 64)	640
conv2d_2 (Conv2D)	(None, 32, 32, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_3 (Conv2D)	(None, 16, 16, 128)	73856
conv2d_4 (Conv2D)	(None, 16, 16, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 128)	0
conv2d_5 (Conv2D)	(None, 8, 8, 256)	295168
conv2d_6 (Conv2D)	(None, 8, 8, 256)	590080
conv2d_7 (Conv2D)	(None, 8, 8, 256)	590080
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 256)	0
conv2d_8 (Conv2D)	(None, 4, 4, 512)	1180160
conv2d_9 (Conv2D)	(None, 4, 4, 512)	2359808
conv2d_10 (Conv2D)	(None, 4, 4, 512)	2359808
max_pooling2d_4 (MaxPooling2	(None, 2, 2, 512)	0
conv2d_11 (Conv2D)	(None, 2, 2, 512)	2359808
conv2d_12 (Conv2D)	(None, 2, 2, 512)	2359808
conv2d_13 (Conv2D)	(None, 2, 2, 512)	2359808
max_pooling2d_5 (MaxPooling2	(None, 1, 1, 512)	0
flatten_1 (Flatten)	(None, 512)	0
dense_1 (Dense)	(None, 4096)	2101248
dense_2 (Dense)	(None, 4096)	16781312
dense_3 (Dense)	(None, 2)	8194

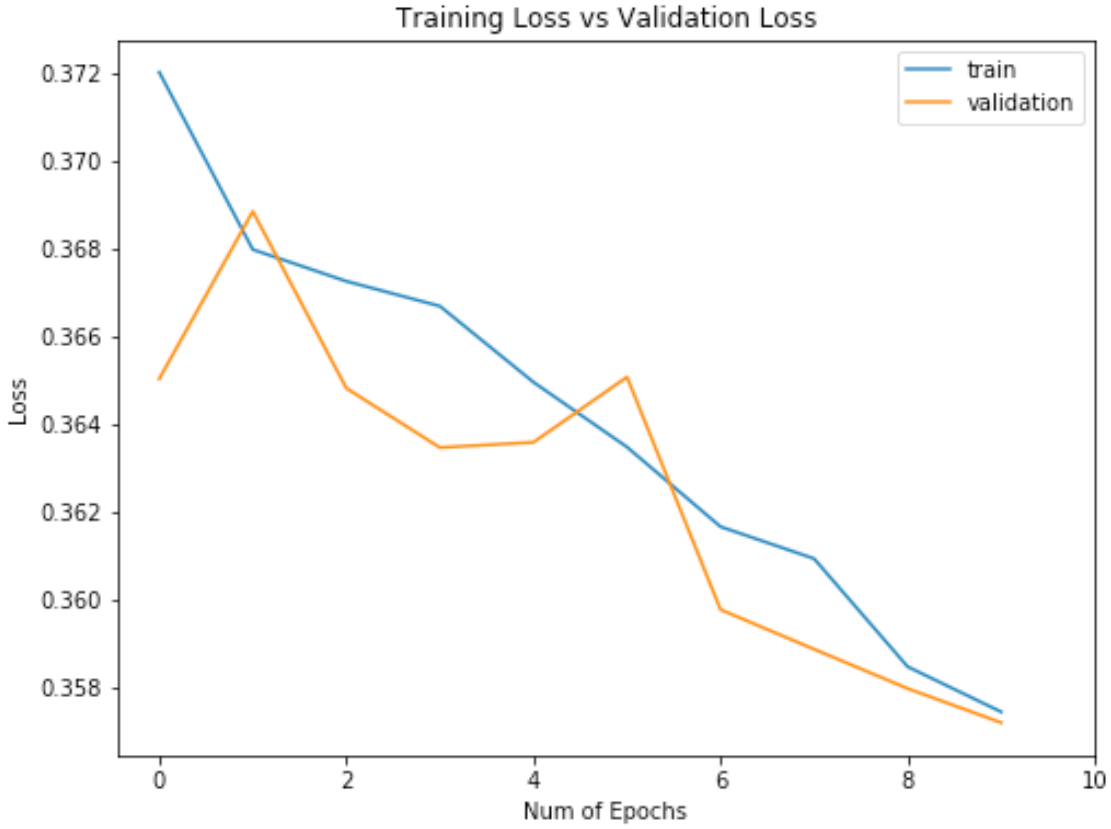
Total params: 33,604,290
Trainable params: 33,604,290
Non-trainable params: 0

Esta red (red 6) tiene un tiempo de entrenamiento de cinco horas por clase (con diez épocas). Entrenando tan solo con el año 2016. Se ha escogido la clase 24 para poder comparar los resultados con nuestro clasificador.

```
red_keras.entrenar(2016,6,{24})
```

Este es el resultado de su entrenamiento.

```
Total params: 33,604,290
Trainable params: 33,604,290
Non-trainable params: 0
_________________________________________________________________
None
Train on 43639 samples, validate on 10910 samples
Epoch 1/10
43639/43639 [==============================] - 1716s 39ms/step - loss: 0.3720 - acc: 0.8785 - val_loss: 0.3650 - val_acc: 0.8809
Epoch 2/10
43639/43639 [==============================] - 1703s 39ms/step - loss: 0.3680 - acc: 0.8805 - val_loss: 0.3688 - val_acc: 0.8809
Epoch 3/10
43639/43639 [==============================] - 1702s 39ms/step - loss: 0.3673 - acc: 0.8805 - val_loss: 0.3648 - val_acc: 0.8809
Epoch 4/10
43639/43639 [==============================] - 1705s 39ms/step - loss: 0.3667 - acc: 0.8805 - val_loss: 0.3635 - val_acc: 0.8809
Epoch 5/10
43639/43639 [==============================] - 1746s 40ms/step - loss: 0.3649 - acc: 0.8805 - val_loss: 0.3636 - val_acc: 0.8809
Epoch 6/10
43639/43639 [==============================] - 1722s 39ms/step - loss: 0.3635 - acc: 0.8805 - val_loss: 0.3651 - val_acc: 0.8809
Epoch 7/10
43639/43639 [==============================] - 1738s 40ms/step - loss: 0.3617 - acc: 0.8805 - val_loss: 0.3598 - val_acc: 0.8809
Epoch 8/10
43639/43639 [==============================] - 1795s 41ms/step - loss: 0.3609 - acc: 0.8805 - val_loss: 0.3589 - val_acc: 0.8809
Epoch 9/10
43639/43639 [==============================] - 1765s 40ms/step - loss: 0.3585 - acc: 0.8805 - val_loss: 0.3580 - val_acc: 0.8809
Epoch 10/10
43639/43639 [==============================] - 1763s 40ms/step - loss: 0.3574 - acc: 0.8805 - val_loss: 0.3572 - val_acc: 0.8809
RED_24 Validation loss: 0.35718516078594076
RED_24 Validation accuracy: 0.8809349221335323
```



Se añade esta red a la comparación anterior.

```
red_keras.comparar_redes(24, [1,4,6], [20470, 35144, 37326])
```

Conviene recordar:

RED	AÑO_ENTRENAMIENTO	ARQUITECTURA
1	2016	NUESTRO MODELO
4	2013-2016	NUESTRO MODELO
6	2016	VGGNET

```
El elemento de la posición 20470 es de clase 24-26-26-26-26-26-26
La red 1 predice que tiene un 58.16% de posibilidades de ser clase 24
La red 4 predice que tiene un 99.07% de posibilidades de ser clase 24
La red 6 predice que tiene un 34.17% de posibilidades de ser clase 24

El elemento de la posición 35144 es de clase 24-26-26-26-26
La red 1 predice que tiene un 65.25% de posibilidades de ser clase 24
La red 4 predice que tiene un 99.60% de posibilidades de ser clase 24
La red 6 predice que tiene un 34.76% de posibilidades de ser clase 24

El elemento de la posición 37326 es de clase 01-01-03-05-24-24-24-24-24-24-24-24
La red 1 predice que tiene un 52.56% de posibilidades de ser clase 24
La red 4 predice que tiene un 99.84% de posibilidades de ser clase 24
La red 6 predice que tiene un 33.12% de posibilidades de ser clase 24
```

También se entrenó esta otra estructura (red 2), utilizando el conjunto de datos del año 2016.

```
def estructura_B(input_shape, num_classes):

    model = Sequential()

    model.add(Conv2D(32, kernel_size=(3, 3), padding='same',input_shape=input_shape, activation='relu'))

    model.add(MaxPooling2D(pool_size=(2, 2)))

    #again
    model.add(Conv2D(64, (2, 2),padding='same', activation='relu'))

    #choose the best features via pooling
    model.add(MaxPooling2D(pool_size=(2, 2)))


    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes,activation='softmax'))



    model.compile(loss=keras.losses.categorical_crossentropy,
                        optimizer=keras.optimizers.Adam(lr=0.0005),
                        metrics=['accuracy'])

    return model
```

Incluimos a la red 2 y 3 en el juego, ambas con estructura B y entrenadas con 2016, y 2013-2016 (monolabel, solo con elementos "puros") respectivamente.

```
red_keras.comparar_redes(24, [1,2,3,4,6], [20470, 35144, 37326])
```

RED	AÑO_ENTRENAMIENTO	ARQUITECTURA
1	2016	NUESTRO MODELO
2	2016	ESTRUCTURA B
3	2013-2016 (monolabel)	ESTRUCTURA B
4	2013-2016	NUESTRO MODELO
6	2016	VGGNET

```
El elemento de la posición 20470 es de clase 24-26-26-26-26-26-26
La red 1 predice que tiene un 58.16% de posibilidades de ser clase 24
La red 2 predice que tiene un 58.12% de posibilidades de ser clase 24
La red 3 predice que tiene un 13.84% de posibilidades de ser clase 24
La red 4 predice que tiene un 99.07% de posibilidades de ser clase 24
La red 6 predice que tiene un 34.17% de posibilidades de ser clase 24

El elemento de la posición 35144 es de clase 24-26-26-26-26
La red 1 predice que tiene un 65.25% de posibilidades de ser clase 24
La red 2 predice que tiene un 59.76% de posibilidades de ser clase 24
La red 3 predice que tiene un 3.96% de posibilidades de ser clase 24
La red 4 predice que tiene un 99.60% de posibilidades de ser clase 24
La red 6 predice que tiene un 34.76% de posibilidades de ser clase 24

El elemento de la posición 37326 es de clase 01-01-03-05-24-24-24-24-24-24-24-24
La red 1 predice que tiene un 52.56% de posibilidades de ser clase 24
La red 2 predice que tiene un 55.19% de posibilidades de ser clase 24
La red 3 predice que tiene un 17.63% de posibilidades de ser clase 24
La red 4 predice que tiene un 99.84% de posibilidades de ser clase 24
La red 6 predice que tiene un 33.12% de posibilidades de ser clase 24
```

NAHİTA

Probamos a evaluar para todas las clases.

```
El elemento de la fila 10711 es de clase 27
La red 4 predice que tiene un
         4.33% de posibilidades de ser clase 1
         0.86% de posibilidades de ser clase 2
         1.08% de posibilidades de ser clase 3
         0.11% de posibilidades de ser clase 4
         3.78% de posibilidades de ser clase 5
         0.05% de posibilidades de ser clase 6
         0.56% de posibilidades de ser clase 7
         0.25% de posibilidades de ser clase 8
         0.46% de posibilidades de ser clase 9
         0.10% de posibilidades de ser clase 10
         1.04% de posibilidades de ser clase 11
         0.11% de posibilidades de ser clase 12
         0.27% de posibilidades de ser clase 13
         0.34% de posibilidades de ser clase 14
         0.74% de posibilidades de ser clase 15
         0.47% de posibilidades de ser clase 16
         0.34% de posibilidades de ser clase 17
         0.33% de posibilidades de ser clase 18
         0.04% de posibilidades de ser clase 19
         0.52% de posibilidades de ser clase 20
         0.29% de posibilidades de ser clase 21
         0.02% de posibilidades de ser clase 22
         0.06% de posibilidades de ser clase 23
         9.84% de posibilidades de ser clase 24
         2.19% de posibilidades de ser clase 25
         7.42% de posibilidades de ser clase 26
         93.38% de posibilidades de ser clase 27
         1.43% de posibilidades de ser clase 28
         16.95% de posibilidades de ser clase 29
         0.00% de posibilidades de ser clase 98
```


```
El elemento de la fila 10713 es de clase 26-26-26
La red 4 predice que tiene un
        9.13% de posibilidades de ser clase 1
        1.52% de posibilidades de ser clase 2
        9.82% de posibilidades de ser clase 3
        0.37% de posibilidades de ser clase 4
        3.03% de posibilidades de ser clase 5
        0.06% de posibilidades de ser clase 6
        0.65% de posibilidades de ser clase 7
        0.99% de posibilidades de ser clase 8
        1.84% de posibilidades de ser clase 9
        0.09% de posibilidades de ser clase 10
        0.34% de posibilidades de ser clase 11
        0.19% de posibilidades de ser clase 12
        0.23% de posibilidades de ser clase 13
        1.02% de posibilidades de ser clase 14
        0.39% de posibilidades de ser clase 15
        1.88% de posibilidades de ser clase 16
        0.46% de posibilidades de ser clase 17
        2.16% de posibilidades de ser clase 18
        0.85% de posibilidades de ser clase 19
        0.50% de posibilidades de ser clase 20
        1.74% de posibilidades de ser clase 21
        0.17% de posibilidades de ser clase 22
        0.20% de posibilidades de ser clase 23
        6.67% de posibilidades de ser clase 24
        24.68% de posibilidades de ser clase 25
        89.85% de posibilidades de ser clase 26
        18.73% de posibilidades de ser clase 27
        1.14% de posibilidades de ser clase 28
        13.37% de posibilidades de ser clase 29
        0.00% de posibilidades de ser clase 98
```

Matriz de confusión de nuestro clasificador para el criterio de la probabilidad mayor.

En la matriz de resultados se observan, para cada elemento, las probabilidades de pertenecer a todas las clases. Se considerará una predicción positiva la clase con la mayor probabilidad.

Con ese criterio la matriz de confusión es la siguiente.

Clase	VP	VN	FP	FN	Acierto	Error	Precisión	Especificidad
1	15	34036	21	3825	0,9	0,1	0,42	0,9
2	32	34693	53	3119	0,92	0,08	0,38	0,92
3	101	34785	159	2852	0,92	0,08	0,39	0,92
4	0	37278	0	619	0,98	0,02	0	0,98
5	22	34678	59	3138	0,92	0,08	0,27	0,92
6	0	37338	0	559	0,99	0,01	0	0,99
7	0	36904	0	993	0,97	0,03	0	0,97
8	0	37482	0	415	0,99	0,01	0	0,99
9	0	37450	0	447	0,99	0,01	0	0,99
10	0	37731	0	166	1	0	0	1
11	0	37402	0	495	0,99	0,01	0	0,99
12	0	37821	0	76	1	0	0	1
13	0	37766	0	131	1	0	0	1
14	0	37524	0	373	0,99	0,01	0	0,99
15	0	37339	0	558	0,99	0,01	0	0,99
16	0	37511	0	386	0,99	0,01	0	0,99
17	0	37567	0	330	0,99	0,01	0	0,99
18	0	37131	0	766	0,98	0,02	0	0,98
19	79	37234	46	538	0,98	0,02	0,63	0,99
20	0	37521	0	376	0,99	0,01	0	0,99
21	0	37454	0	443	0,99	0,01	0	0,99
22	0	37781	0	116	1	0	0	1
23	0	37748	0	149	1	0	0	1
24	397	32693	507	4300	0,87	0,13	0,44	0,88
25	114	30558	107	7118	0,81	0,19	0,52	0,81
26	5764	18507	3349	10277	0,64	0,36	0,63	0,64
27	13769	8532	12515	3081	0,59	0,41	0,52	0,73
28	0	36985	0	912	0,98	0,02	0	0,98
29	310	29164	410	8013	0,78	0,22	0,43	0,78
98	0	37705	0	192	0,99	0,01	0	0,99

Matriz de confusión de nuestro clasificador para el criterio de umbral fijo (50%)

En la matriz de resultados se observan, para cada elemento, las probabilidades de pertenecer a todas las clases. Se considerará una predicción positiva para una clase cuando la probabilidad de pertenecer a dicha clase sea mayor del 50%.

Con ese criterio la matriz de confusión es la siguiente.

Clase	VP	VN	FP	FN	Acierto	Error	Precisión	Especificidad
1	274	33307	750	3566	0,89	0,11	0,27	0,9
2	152	34379	367	2999	0,91	0,09	0,29	0,92
3	528	33512	1432	2425	0,9	0,1	0,27	0,93
4	0	37278	0	619	0,98	0,02	0	0,98
5	333	33783	954	2827	0,9	0,1	0,26	0,92
6	0	37337	1	559	0,99	0,01	0	0,99
7	0	36903	1	993	0,97	0,03	0	0,97
8	0	37482	0	415	0,99	0,01	0	0,99
9	0	37450	0	447	0,99	0,01	0	0,99
10	0	37731	0	166	1	0	0	1
11	0	37402	0	495	0,99	0,01	0	0,99
12	0	37821	0	76	1	0	0	1
13	0	37766	0	131	1	0	0	1
14	0	37524	0	373	0,99	0,01	0	0,99
15	0	37339	0	558	0,99	0,01	0	0,99
16	0	37511	0	386	0,99	0,01	0	0,99
17	0	37567	0	330	0,99	0,01	0	0,99
18	0	37131	0	766	0,98	0,02	0	0,98
19	104	37198	82	513	0,98	0,02	0,56	0,99
20	0	37521	0	376	0,99	0,01	0	0,99
21	0	37454	0	443	0,99	0,01	0	0,99
22	0	37781	0	116	1	0	0	1
23	0	37748	0	149	1	0	0	1
24	971	30633	2567	3726	0,83	0,17	0,27	0,89
25	1511	27840	2825	5721	0,77	0,23	0,35	0,83
26	14096	6590	15266	1945	0,55	0,45	0,48	0,77
27	16479	1932	19115	371	0,49	0,51	0,46	0,84
28	10	36982	3	902	0,98	0,02	0,77	0,98
29	6274	15913	13661	2049	0,59	0,41	0,31	0,89
98	0	37705	0	192	0,99	0,01	0	0,99