Trabajo Fin de Máster

# Clasificación de documentos digitalizados

Yolanda Cobas Cornide

**Máster en Técnicas Estadísticas**

**Curso 2022-2023**

# Propuesta de Trabajo Fin de Máster

| |
|---|
| **Título en galego:** Clasificación de documentos dixitalizados |
| **Título en español:** Clasificación de documentos digitalizados |
| **English title:** Classification of digitized documents |
| **Modalidad:** Modalidad B |
| **Autor/a:** Yolanda Cobas Cornide, Universidade de A Coruña |
| **Director/a:** Rubén Fernández Casal, Universidade de A Coruña; Ignacio García Jurado, Universidade de A Coruña |
| **Tutor/a:** José Jorge García Romarís, ABANCA |
| **Breve resumen del trabajo:**<br>El trabajo a realizar consistirá en desarrollar modelos (utilizando preferente Deep Learning) para la clasificación automática de documentos digitales. |
| **Recomendaciones:**<br>Conocimientos avanzados en SQL y programación en Python. Técnicas de Deep Learning. |
| **Otras observaciones:**<br>Bolsa de estudios de 800 € mensuales |

Escriba el texto aquí

Don Rubén Fernández Casal, de la Universidade de A Coruña, don Ignacio García Jurado, de la Universidade de A Coruña y don José Jorge García Romarís, de ABANCA, informan que el Trabajo Fin de Máster titulado

**Clasificación de documentos digitalizados**

fue realizado bajo su dirección por doña Yolanda Cobas Cornide para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En [A Coruña], a 14 de [Julio] de 2023.

Firmado digitalmente por FERNANDEZ CASAL RUBEN - 33317724Q
Fecha: 2023.07.13 19:06:59 +02'00'

El director:
Don Rubén Fernández Casal

GARCIA JURADO IGNACIO MIGUEL - 32451391R
Firmado digitalmente por GARCIA JURADO IGNACIO MIGUEL - 32451391R
Fecha: 2023.07.13 18:01:58 +02'00'

El director:
Don Ignacio García Jurado

El tutor:
Don José Jorge García Romarís

La autora:
Doña Yolanda Cobas Cornide

---

**Declaración responsable.** Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,...sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

# Índice general

# Resumen


## Resumen en español

En las entidades bancarias, la cantidad de documentos que se manejan es muy elevada, es por ello que conseguir una clasificación de los mismos rápida y eficiente puede ayudar a ahorrar costes. Este trabajo consistirá en una revisión tanto teórica como práctica sobre *Computer Vision*, una técnica empleada para la clasificación de imágenes. El ejemplo práctico se realizará sobre una base de datos construida a partir de documentos bancarios de la entidad ABANCA.

Para la construcción del modelo de *Computer Vision* se necesitará realizar un preprocesado previo de los datos, donde se realizará una limpieza de los mismos que permitirá entrenar el modelo de forma eficiente. Posteriormente, se construirá un modelo basado en redes neuronales utilizando la técnica de *transfer learning* o aprendizaje por transferencia, utilizando como base un modelo ya construido y eficiente llamado *MobileNet V2*. Tras el ajuste, este modelo será evaluado con diversas métricas, como la precisión global, el valor predictivo positivo, la sensibilidad y el *F1 score*. Para concluir, se sacarán algunas conclusiones sobre el funcionamiento de este método y su puesta en práctica y se hablará de otras alternativas.

## English abstract

In banking institutions, the amount of documents handled is very high, which is why a fast and efficient classification of documents can help to save costs. This work will consist of a theoretical and practical review of Computer Vision, a technique used for image classification. The practical example will be performed on a database built from ABANCA's bank documents.

For the construction of the Computer Vision model, it will be necessary to perform a previous preprocessing of the data, where a cleaning of the data will be performed in order to train the model in an efficient way. Subsequently, a model based on neural networks will be built using the transfer learning technique, using as a basis an already built and efficient model called MobileNet V2. After tuning, this model will be evaluated with various metrics, such as *accuracy*, *recall* and *F1 score*. To conclude, some conclusions will be drawn about the performance of this method and its implementation and alternatives will be discussed.

# Introducción

La clasificación automática de documentos ha experimentado un auge en los últimos años, debido a la mayor disponibilidad de documentos en formato digital, a la consiguiente necesidad de organizarlos y al avance de las tecnologías de aprendizaje automático. El objetivo de estos sistemas que clasifican documentos es analizarlos y asignarles una categoría o clase específica basada en su contenido.

En la actualidad, muchas empresas cuentan con grandes cantidades de documentos que se encuentran en distintos formatos y sistemas. La mayoría de las empresas no disponen de bases de datos que contengan datos estructurados, es decir, los datos no son archivos con formato estandarizado, con estructura bien definida, de fácil acceso para los programas y las personas y fácilmente procesados por herramientas de minería de datos. Sólo el 20 % de los datos con los que cuenta una empresa serían datos estructurados. El 80 % de la información relevante para una empresa se origina en forma no estructurada, estos datos no estructurados no cuentan con una estructura interna identificable. Es un conglomerado masivo y desorganizado de varios objetos que no tienen valor hasta que se identifican y almacenan de manera organizada. La clasificación automática de documentos se ha convertido en una valiosa herramienta para organizar y estructurar estos datos de manera eficiente, lo que conduce a una mejor gestión de la información y una toma de decisiones más rápida y precisa (Sebastiani, 2002).

La organización automática de objetos en general, basada en las semejanzas entre ellos, puede ayudar a manejar grandes volúmenes de tales objetos, agrupándolos en clases, categorías o clústers de objetos parecidos entre sí. Básicamente podemos distinguir dos grandes formas de abordar la organización automática: la clasificación no supervisada (o clustering) y la categorización o clasificación supervisada. En cualquiera de estas formas, los objetos necesitan ser descritos a través de una lista de características que, en función de las herramientas y las técnicas que se apliquen, pueden tener valores de diversos tipos, no necesariamente numéricos. Las descripciones de los objetos deben poder construirse de manera automática, y deben ser homogéneas entre sí, de forma que sea posible calcular de forma automática la semejanza entre dichas descripciones.

En el caso de la categorización o clasificación supervisada, que es la que se utilizará durante el desarrollo de este trabajo, se parte de una serie de clases o categorías, elaboradas de antemano, manualmente, y cada objeto después de ser procesado, se intenta asignar a la clase que le corresponda. Para ello, los programas de ordenador elaboran modelos o patrones de cada una de las clases o categorías contempladas. Esta fase se conoce como entrenamiento y para ser llevada a cabo es preciso contar con una colección de objetos categorizados previamente. Durante el entrenamiento, se extraen las características que definen cada una de las categorías y, dependiendo de las técnicas utilizadas, también las que determinan lo contrario, es decir, la distancia o no pertenencia a cada clase o categoría

(Figuerola, 2013).

Nos centraremos en la clasificación de documentos en el ámbito bancario, siendo de especial interés para una entidad bancaria el poder clasificar sus documentos de forma inmediata y lo más satisfactoriamente posible. Para realizar la clasificación de los 26 tipos de documentos que se van a considerar en este trabajo se va a utilizar un modelo de *Computer Vision*.

*Computer Vision*, visión por computador o visión artificial es un área de la Inteligencia Artificial que incluye formas de adquirir, procesar, analizar y comprender imágenes y vídeos del mundo real para imitar la visión humana. Su objetivo principal es la deducción automática de la estructura y propiedades de un mundo tridimensional a partir de una o varias imágenes bidimensionales. Un sistema de visión por ordenador puede aceptar diferentes formas de datos como una entrada que incluye, entre otras, imágenes, secuencias de imágenes o vídeos que pueden ser transmitidos desde múltiples fuentes para procesar y extraer información útil para la toma de decisiones (Valenzuela, 2021). En comparación con la visión humana, ofrece una mejor evaluación de las magnitudes físicas y un mayor desempeño en las tareas rutinarias. Algunas de sus múltiples aplicaciones están relacionadas con la vigilancia y la seguridad, el reconocimiento facial, la programación de vehículos autónomos ...

En el primer capítulo de este trabajo se realiza una contextualización del problema que se va a tratar, empezando por una breve historia de la técnica de *Computer Vision* y a continuación explicando el modelo estadístico que hay detrás de este problema y los pasos a seguir para resolverlo.

En el segundo capítulo, se realiza un análisis de la base de datos que se va a utilizar para la realización del ejemplo práctico, incluyendo una descripción de los diferentes tipos de documentos y de sus formatos, así como una explicación de la primera limpieza que se hace a los mismos para conseguir el *dataset* de imágenes con el que va a trabajar el modelo.

El tercer capítulo es una revisión de las bases teóricas en las que se encaja el desarrollo de este problema de clasificación. La técnica de *Computer Vision* se basa, principalmente, en redes neuronales convolucionales, por tanto, se explicarán las mismas comenzando desde las redes neuronales más sencillas. Además, se explicará el funcionamiento teórico de las medidas que se usarán para evaluar el modelo.

En el siguiente capítulo, se trata la metodología con la que se va a construir el modelo. El modelo que se usará en el ejemplo práctico se construye utilizando la técnica de *transfer learning*, que se explicará en este capítulo y se basará en un modelo ya construido llamado *MobileNet V2*, para el cual también se explicará su estructura. Posteriormente, se cuentan los cambios que se realizan en el modelo de base para la construcción del modelo definitivo.

El quinto capítulo versa sobre cómo se ha desarrollado el ejemplo práctico de este trabajo. Se explica el preprocesado que se realiza para cada tipo de documento, la construcción del modelo de clasificación y, posteriormente, la evaluación de los resultados obtenidos. Por último, se explica cómo se clasificarían los documentos completos, y no las imágenes como se había hecho hasta el momento.

El último de los capítulos recoge las conclusiones que se han extraído durante la elaboración de este trabajo, así como posibles trabajos futuros.

# Capítulo 1

# Contextualización del problema de clasificación de documentos

Este primer capítulo se basa en poner en contexto el problema que se trata a lo largo de este trabajo. Se trata en primer lugar la historia de *Computer Vision*, la técnica que se utilizará para el desarrollo del ejemplo de clasificación de documentos. Posteriormente, se pasa a describir el problema a tratar, se estudia tanto la descripción del problema desde el punto de vista bancario como la explicación del modelo estadístico. Por último, se hace una breve introducción sobre cuáles serán los cuatro pasos que se llevarán a cabo para la construcción del modelo de clasificación.

## 1.1. Una breve historia sobre *Computer Vision*

El interés por la clasificación de documentos siempre ha estado ahí, pero a medida que la cantidad documental que se maneja ha ido aumentando, esto ha conllevado una mayor necesidad de clasificar los documentos de forma rápida y óptima. Las técnicas que se utilizan actualmente para la clasificación automática de documentos se recogen dentro de la Inteligencia Artificial. La Inteligencia Artificial, como campo de estudio, engloba el campo del *Machine Learning*, el cual, a su vez, engloba al *Deep Learning*.

A lo largo de este trabajo se va a utilizar un método que se engloba dentro de los campos citados previamente, que es *Computer Vision*. Por ello, se va a introducir una pequeña historia de este, para lo que se seguirá (Borrella, 2022) y (Huang, 1996).

A finales de la década de los 60, la técnica de *Computer Vision* comenzó a desarrollarse con el objetivo de poder conectar una cámara de vídeo a un ordenador en universidades que eran pioneras en Inteligencia Artificial. Se data el inicio de *Computer Vision* en 1959, año en el que un grupo de neurofisiólogos realizó un experimento que consistía en mostrarle a un gato una serie de imágenes para comprender como las entendía y procesaba el animal. Los resultados mostraron que sus primeras respuestas iban dirigidas a los bordes o líneas sólidas, por lo que concluyeron que el procesamiento de las imágenes empezaba con formas simples, como los bordes.

Al mismo tiempo, comenzó a desarrollarse la primera tecnología de escaneo artificial de imágenes,

que permitió a los ordenadores digitalizar y adquirir las imágenes. En 1963 los ordenadores fueron capaces de transformar imágenes bidimensionales en formas tridimensionales. Uno de los trabajos que marcó el inicio de la visión artificial fue el realizado por Larry Roberts en el año 61. Roberts creó un programa llamado "Mundo de Microbloques", en el que un robot podía "ver" una estructura de bloques sobre una mesa, analizar el contenido y reproducirla desde otra perspectiva. Consiguió que la imagen mandada de la cámara al ordenador fuera procesada adecuadamente por este. Además, en la década de los 60 fue cuando la Inteligencia Artificial se convirtió un campo de estudio académico y esta comenzó a usarse para resolver el problema de la visión artificial.

Lo que distinguió *Computer Vision* del campo que predominaba en el procesamiento de imágenes digitales en ese momento fue el deseo de extraer una estructura tridimensional de las imágenes con el objetivo de lograr la comprensión completa de la escena. Los estudios en la década de 1970 fueron la base de muchos de los algoritmos de visión por computador que existen hoy en día, como la extracción de bordes de imágenes, el etiquetado de líneas, el modelado no poliédrico y poliédrico, la representación de objetos como interconexiones de estructuras más pequeñas, flujo óptico y estimación del movimiento.

En la década posterior se realizaron estudios basados en análisis matemáticos más rigurosos y aspectos cuantitativos de la visión por computadora. Gracias a eso los investigadores se dieron cuenta de que muchos de los conceptos matemáticos que hay detrás de *Computer Vision* se podían tratar dentro del marco de optimización y los campos aleatorios de Markov. La década de 1990, supuso el inicio del uso de técnicas de aprendizaje estadístico para reconocer caras en imágenes. Hacia finales de la década de 1990, se produjo un cambio significativo con la mayor interacción entre los campos de los gráficos por computador y la visión por computador. Incluyéndose en esa interacción la representación basada en imágenes, la transformación de imágenes, la interpolación de vistas, la costura panorámica de imágenes y la representación temprana de campos de luz.

En el año 2000, el estudio se centraba en el reconocimiento de objetos, y para el año 2001 se empezaron a crear aplicaciones de reconocimiento facial. Actualmente la visión artificial comprende tanto la obtención como la caracterización e interpretación de los objetos contenidos en una imagen y es uno de los elementos sensoriales más importantes para incrementar la autonomía en robótica. Adicionalmente, se puede destacar que los métodos utilizados de visión artificial son bastante baratos en comparación con otros métodos que ofrecen resultados similares.

## 1.2. Descripción del proceso

La cantidad de documentos que maneja cualquier entidad bancaria es inmensa. Los bancos cuentan con gran cantidad tanto de documentos digitales como de documentos en papel que son luego escaneados. En muchos casos, son los trabajadores del banco los que se encargan de clasificar estos documentos en función de la clase a la que pertenecen y esto conlleva gran cantidad de errores. Es por ello que nace la necesidad de automatizar este proceso. La correcta clasificación de los documentos permite evitar la fuga de información relevante. Si se consiguiera realizar una clasificación de documentos y una extracción de información realmente buena, muchos de los trámites que lleva a cabo una entidad bancaria podrían automatizarse por completo, evitando así errores humanos y disminuyendo el tiempo que tardan en realizarse.

Como se ha avanzado previamente, la idea de este trabajo es a partir de una base de datos documental proporcionada por la entidad bancaria ABANCA, construir un modelo de *Computer Vision* que se encargue de intentar clasificar estos documentos de forma correcta, en otras palabras, que intente predecir la clase a la que pertenecen los mismos.

La construcción del modelo tendrá lugar en cuatro etapas.

1. **Creación del *dataset***: a partir de los documentos proporcionados por ABANCA se construirá la base de datos formada por las imágenes de cada una de las páginas de los documentos, que se utilizará a lo largo del trabajo mediante la limpieza y transformación de los mismos, como se explicará más adelante.

2. **Depuración de la base de datos**: una vez construido el *dataset*, se separa en conjuntos de entrenamiento y test para aplicar un modelo que permita la detección de atípicos y su posterior eliminación.

3. **Modelado**: mediante técnicas de aprendizaje por transferencia se creará la red neuronal que será el cuerpo del modelo. Se entrenará el modelo para que trate de predecir la clase a la que pertenece cada uno de los datos.

4. **Evaluación**: una vez ha sido construido el modelo y se han obtenido las predicciones, se evaluará el mismo usando distintas métricas de evaluación que se explican posteriormente.

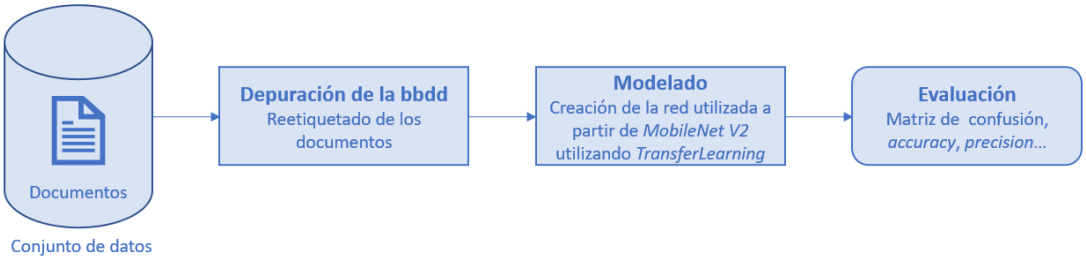


Figura 1.1: Etapas en las que se dividirá el proyecto.

# Capítulo 2

# Análisis de los datos

En este capítulo se hace un análisis del conjunto de datos inicial, realizando los pasos necesarios para poder obtener el *dataset* con el que se va a trabajar. El conjunto de datos que se manejará durante el desarrollo del trabajo son documentos de utilidad para la entidad bancaria en formato *.pdf*. El conjunto de datos inicial contaba con 12.694 documentos de 26 tipos diferentes. Para cada uno de los documentos se especifica el tipo con el que se ha clasificado y la fecha en la que se agregó al servidor de objetos de ABANCA, estas fechas abarcan desde octubre de 2005 hasta abril de 2023. Se realizó una primera limpieza en la cual se eliminaron aquellos registros cuya URL al servidor de objetos estaba vacía y los que tenían este campo duplicado, quedando así el conjunto de los datos con 12.297 registros.

## 2.1. Descripción de los documentos

Durante el trabajo se van a clasificar 26 tipos diferentes de documentos. En este apartado se va a explicar cuáles son estos documentos, el código con el que se van a referenciar los documentos a lo largo del trabajo y en qué consisten. Se enumeran a continuación todos ellos.

- Contrato de alquiler ("6002"): el conjunto de datos cuenta con 494 documentos de este tipo. El contrato de alquiler establece una relación entre dos partes, mediante la cual se obligan de manera recíproca y por un tiempo determinado a la cesión de un bien o servicio quedando obligada la parte que aprovecha la posesión a pagar un precio determinado.

- Prestación cese de actividad ("9052"): se dispone de 163 documentos de este tipo. Este tipo de documentos agrupa solicitudes y resoluciones sobre la concesión de prestaciones a los autónomos que cesan su actividad de forma temporal o definitiva.

- Certificado de empadronamiento ("9053"): hay 476 documentos de este tipo. El certificado de empadronamiento es un documento que registra el domicilio de los habitantes de un municipio.

- Declaración de discapacidad ("9054"): existen 408 documentos de este tipo. Son resoluciones al procedimiento administrativo que tiene por objeto la valoración de las situaciones de minusvalía o discapacidad y la calificación de su grado.

- Cuantía mensual de desempleo ("9057"): el conjunto de datos contiene 475 documentos de este tipo. Estos documentos certifican si una persona es o no beneficiaria de algún tipo de prestación por desempleo.

- Certificado de defunción ("9526"): en el conjunto de datos hay 500 documentos de este tipo. Un certificado de defunción es un documento legal emitido por un médico que establece cuándo ha fallecido una persona, o un documento emitido por una oficina de registro civil del gobierno, que declara la fecha, el lugar y la causa de la muerte de una persona, tal como se inscribió en un registro oficial de muertes.

- Extracto ("9533"): existen 499 documentos de este tipo. Un extracto de una cuenta bancaria es un documento que brinda información acerca de las operaciones y movimientos de las cuentas.

- Testamento ("9539"): se dispone de 500 documentos de este tipo. Un testamento es una declaración voluntaria de una persona expresando lo que quiere que se haga con sus bienes después de su fallecimiento.

- Certificado de tasación ("10003"): el conjunto de datos contiene 438 documentos de este tipo. El certificado de tasación es un resumen del informe de tasación donde se reflejan los aspectos básicos y la firma del perito.

- Informe tasadora ("10004"): el conjunto de datos posee 449 documentos de este tipo. El informe de tasadora es un documento oficial por el cual un profesional, perito o tasador, realiza un estudio del inmueble para conocer en cuánto está valorado. Para ello se basa en diferentes factores, como ubicación, orientación, antigüedad ...

- Registro de la propiedad ("10005"): el conjunto de datos parte con 500 documentos de este tipo. En este tipo de documento se agrupan notas simples, consultas de localización de registros y otros documentos relacionados con el registro de bienes inmuebles en el Registro de la Propiedad.

- Escritura ("15005"): se cuenta con 488 documentos de este tipo. La escritura es un instrumento notarial que contiene una o más declaraciones de las personas que intervienen en un acto o contrato, para su incorporación al protocolo del propio notario y, en su caso, para que pueda inscribirse en los registros públicos correspondientes.

- Bienes inmuebles ("15064"): existen 499 documentos de este tipo. En esta clase se engloban documentos relacionados con facturas o contratos de bienes inmuebles.

- DNI ("25000"): el conjunto de datos contiene 496 documentos de este tipo. El DNI es el documento de identidad que se expide en España.

- Nómina ("25002"): se dispone de 469 documentos de este tipo. La nómina es el documento que certifica que una compañía ha cumplido con abonar la remuneración a sus empleados.

- IRPF ("25003"): en el conjunto de datos hay 498 documentos de este tipo. Recoge documentos de ingreso o devolución del Impuesto sobre la Renta de las Personas Físicas.

- IVA ("25005"): el conjunto de datos posee 494 documentos de este tipo. Recoge documentos sobre el Impuesto sobre el Valor Añadido, como declaraciones de resumen anual, autoliquidaciones ...
- Balances ("25009"): existen 497 documentos de este tipo. El balance de situación (o balance general) es el resumen de todas las posesiones, deudas y capital de una organización en un periodo determinado.
- Modelo 347 ("25012"): el conjunto de datos cuenta con 493 documentos de este tipo. El Modelo 347 es la declaración anual de operaciones con terceras personas que permite a Hacienda cruzar datos informados de operaciones económicas entre proveedores y clientes.
- Impuesto de sociedades ("25013"): en el conjunto de datos hay 499 documentos de este tipo. Son documentos de tipo Modelo 200, que es la declaración o liquidación del Impuesto sobre Sociedades (IS), este se aplica sobre los beneficios que obtienen las empresas.
- Vida laboral ("25023"): el conjunto de datos contiene 497 documentos de este tipo. Se trata de un documento oficial en el que queda reflejada cronológicamente la actividad laboral de una persona a lo largo de su vida y su correspondiente cotización en el sistema de la Seguridad Social contabilizada por años, meses y días, independientemente de si ha sido como trabajador asalariado o como autónomo.
- Contrato de trabajo ("25024"): existen 499 documentos de este tipo. Recoge documentos relativos al acuerdo entre empresario y trabajador por el que éste se obliga a prestar determinados servicios por cuenta del empresario y bajo su dirección, a cambio de una retribución.
- Libro de familia ("25053"): en el conjunto de datos hay 466 documentos de este tipo. El libro de familia es un documento expedido por el Registro Civil que certifica los hechos relevantes atinentes a una familia: desde el matrimonio en su caso, incluyendo principalmente el nacimiento o adopción de hijos, así como cualquier hecho que afecte a la patria potestad.
- CIRBE ("25114"): el conjunto de datos cuenta con 500 documentos de este tipo. Recoge las solicitudes de extracción de datos de la plataforma CIRBE de una persona.
- Recibo de otra entidad ("35020"): se dispone de 500 documentos de este tipo. Recoge documentos bancarios que han sido emitidos por entidades distintas a ABANCA.
- DRF ("25037"): el conjunto de datos contiene 500 documentos de este tipo. La Declaración de Residencia Fiscal es un documento que se emite por las administraciones tributarias de cada país, con el fin de acreditar la residencia fiscal de un contribuyente y saber cómo debe tributar.

Por tanto, atendiendo al número de documentos con el que cuenta cada uno de los tipos, el conjunto de datos quedaría distribuido de la forma en la que aparece en la Figura 2.1.

## 2.2. Formato de los documentos

Todos los documentos anteriores están en formato *.pdf*, ya sea porque han sido elaborados directamente en un ordenador de forma digital y se han guardado en ese formato o porque estaban en papel

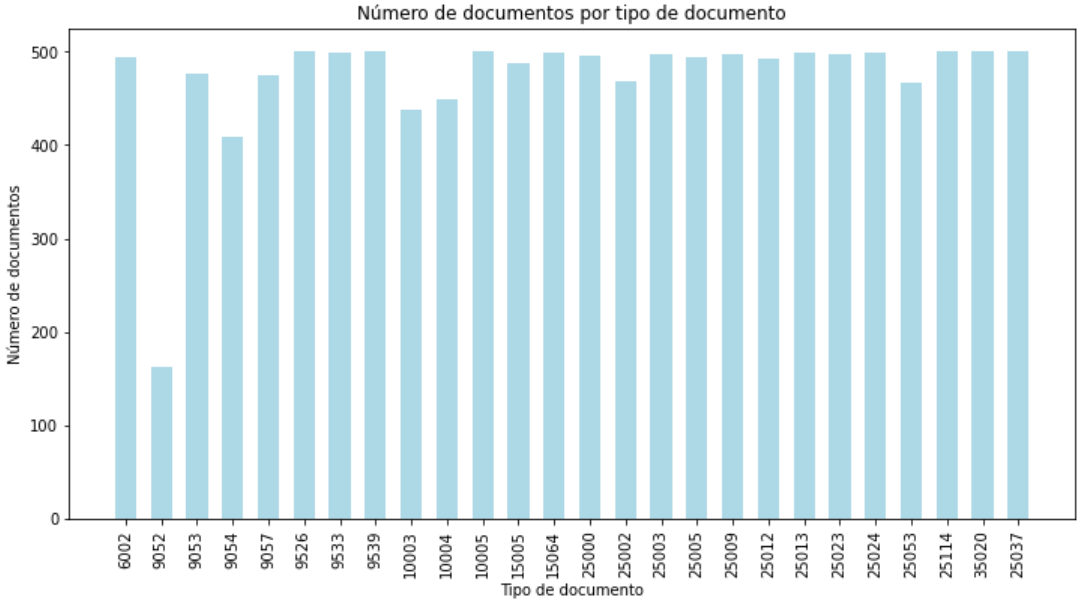


Figura 2.1: Distribución de los documentos en función de la clase.

y han sido escaneados y pasados a ese formato para su guardado.

El modelo que se utilizará a lo largo del trabajo se encarga de la clasificación de imágenes, por lo tanto, para crear la base datos se convertirán los documentos en imágenes separando cada página del documento en una imagen distinta. El modelo se entrena para que intente predecir a qué clase pertenece cada una de las páginas de un documento por separado (no será necesario contar con el documento completo). Por tanto, cada página de cada documento será independiente del resto a lo largo del trabajo, y el *dataset* lo formarán las imágenes de cada una de las páginas.

El primer paso de este trabajo ha sido coger estos documentos y exportarlos a *.png*, de forma que cada página de un documento se corresponda a una imagen distinta. Esta transformación se realiza en *Python*, gracias al comando *.get_pixmap* de la librería *PyPDF2*. Este comando crea un mapa de píxeles a partir de diversos formatos, en este caso, una página de un documento. Los mapas de píxeles son estructuras o ficheros de datos que representa una rejilla rectangular de píxeles o puntos de color, denominada matriz, que se puede visualizar en un monitor. Cada píxel se describe mediante un número de *bytes* que definen su color.

Así, la base de datos que hay construida se conforma de las 87.375 imágenes correspondientes a las páginas de los 12.297 documentos *.pdf*.

## 2.3. Limpieza del conjunto de datos

A partir del conjunto de datos anterior se hace una limpieza de las imágenes. Se van a eliminar tanto las imágenes de las páginas en blanco como aquellas imágenes muy claras u oscuras, que podrían ser por ejemplo las páginas mal escaneadas. Para ello se realizará un análisis de claridad, que consiste en:

1. Calcular la intensidad de cada píxel de la imagen, siendo este un valor entre 0 y 255 de forma que el 0 representa una imagen de oscuridad máxima y el 255 una de luz máxima.
2. Normalizar esta intensidad, es decir, dividir cada valor anterior entre 255.
3. Calcular la claridad de la imagen, que se consigue sumando las intensidades normalizadas de todos los píxeles de la imagen y dividiendo entre el número de píxeles totales de la imagen. Este número total de píxeles de una imagen es $dim_1 * dim_2$, siendo $dim_1$ la altura de la imagen y $dim_2$ la anchura de esta.

$$\text{Claridad} = \frac{\sum \text{Intensidad normalizada}}{dim_1 * dim_2} \tag{2.1}$$

4. Como la idea es seleccionar aquellos documentos que no tengan una claridad ni muy baja (cercana a 0, que se corresponden con documentos muy oscuros), ni muy alta (cercana a 1, que se corresponden con documentos muy claros), vamos a descartar imágenes con claridades próximas a cero y a uno. Los valores concretos de claridad que se descartan para cada documento se seleccionan en función de cómo son esos documentos después de realizar varias representaciones gráficas.

En la Tabla 2.3 se muestran las claridades escogidas para cada tipo de documento, así como el número de imágenes antes y después de la limpieza. Por tanto, después de esta limpieza en función de la claridad, el conjunto de datos está compuesto por 84.111 imágenes.

Atendiendo ahora al número de imágenes tanto inicial como final con el que cuenta cada uno de los tipos, el conjunto de datos quedaría distribuido de la forma en la que aparece en la Figura 2.2.

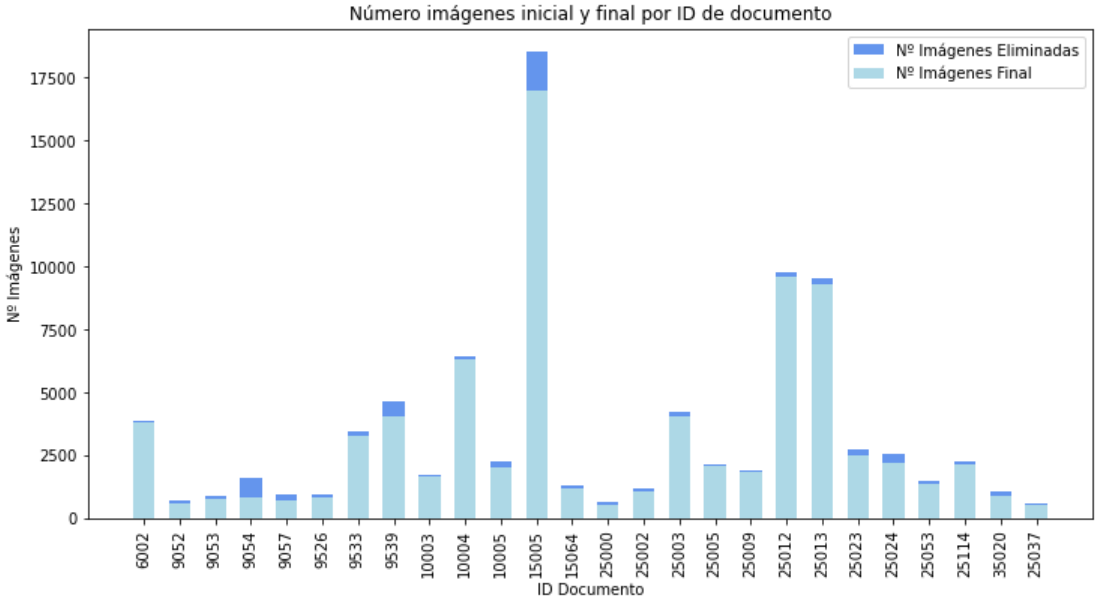


Figura 2.2: Distribución del número de imágenes por clase.

Se observa claramente que, aunque el número de documentos *.pdf* que había en cada tipo de documento era bastante similar, todos rondaban los 500 documentos (a excepción del tipo 9052, que se

| ID documento | Nº inicial de imágenes | Claridades seleccionadas | Nº final de imágenes |
|---|---|---|---|
| 6002 | 3890 | (0.613546, 0.996456) | 3786 |
| 9052 | 709 | (0.8700942, 0.989707) | 585 |
| 9053 | 884 | (0.587021, 0.990237) | 748 |
| 9054 | 1613 | (0.773967, 0.965366) | 839 |
| 9057 | 927 | (0.860074, 0.985037) | 723 |
| 9526 | 929 | (0.857270, 0.987162) | 803 |
| 9533 | 3440 | (0.549965, 0.995219) | 3249 |
| 9539 | 4657 | (0.855135, 0.993117) | 4071 |
| 10003 | 1703 | (0.764721, 0.99214) | 1685 |
| 10004 | 6407 | (0.663895, 0.991125) | 6304 |
| 10005 | 2258 | (0.806979, 0.981742) | 2013 |
| 15005 | 18525 | (0.899769, 0.985937) | 16999 |
| 15064 | 1298 | (0.874963, 0.987242) | 1159 |
| 25000 | 680 | (0.814834, 0.992127) | 556 |
| 25002 | 1179 | (0.876149, 0.982848) | 1050 |
| 25003 | 4202 | (0.829773, 0.99105) | 4028 |
| 25005 | 2148 | (0.811940, 0.984387) | 2064 |
| 25009 | 1883 | (0.450733, 0.99435) | 1839 |
| 25012 | 9788 | (0.884714, 0.983496) | 9624 |
| 25013 | 9533 | (0.692167, 0.994428) | 9320 |
| 25023 | 2730 | (0.600845, 0.988966) | 2524 |
| 25024 | 2549 | (0.767220, 0.990373) | 2210 |
| 25053 | 1503 | (0.602247, 0.98701) | 1393 |
| 25114 | 2283 | (0.715241, 0.991377) | 2132 |
| 35020 | 1055 | (0.893596, 0.993249) | 882 |
| 25037 | 602 | (0.832679, 0.979781) | 525 |

corresponde con “prestación del cese de actividad”, donde contábamos con menos de 200 documentos), al hablar de imágenes hay muchas diferencias. Los documentos del tipo 15005 cuentan con aproximadamente 17.000 imágenes, esto es así porque las “Escrituras” cuentan con entre 30 y 40 páginas por documento. Por otro lado, tenemos los documentos del tipo 25000, que serían los “DNI”, que hay casi el mismo número de documentos como de imágenes, esto es así porque la mayoría solo cuentan con una página, donde se escanea el DNI en ambas caras. El número de documentos y de imágenes no coincide exactamente o bien porque hay DNIs en los que se ha escaneado cada cara en una página distinta o porque hay documentos de otro tipo que están mal clasificados y estos cuentan con varias páginas.

# Capítulo 3

# Marco teórico

En este capítulo se presentará la parte teórica en la que encaja el método que se utiliza a lo largo del trabajo. El modelo de *Computer Vision* que se va a construir se asienta sobre redes neuronales convolucionales, por lo que se hará una revisión de estas empezando por lo más básico, la inspiración biológica de las mismas, que son las redes neuronales biológicas. Además, se definirán las distintas métricas que se van a utilizar para evaluar el modelo construido.

## 3.1. *Computer Vision*

*Computer Vision*, visión por computador o visión artificial, que se engloba dentro del área de la Inteligencia Artificial, es el conjunto de herramientas y técnicas que permiten obtener, procesar y analizar imágenes reales para que puedan ser tratadas por un ordenador (Ceballo, 2019). Funciona de manera muy similar a la visión humana, excepto que los humanos tienen una ventaja inicial, que es toda una vida de contexto para entrenar cómo distinguir los objetos, a qué distancia están, si se están moviendo y si hay algo mal en una imagen.

La visión por computador entrena a las máquinas para realizar estas funciones, pero tiene que hacerlo en mucho menos tiempo con cámaras, datos y algoritmos en lugar de retinas, nervios ópticos y una corteza visual. Una vez entrenado para realizar una función, permite analizar miles de productos o procesos por minuto, detectando así defectos o problemas imperceptibles, lo que hace que pueda llegar a superar las capacidades humanas.

El campo de la visión por computador se ha revolucionado con las redes neuronales, que permiten a la máquina interpretar y comprender imágenes de forma eficiente. Estas potentes herramientas de aprendizaje automático han impulsado avances significativos en áreas como el reconocimiento de objetos, la detección de rostros, la segmentación semántica y muchas otras aplicaciones relacionadas con la visión artificial.

### 3.1.1. Redes neuronales

El *Deep Learning* o Aprendizaje Profundo, que forma parte del Aprendizaje Automático, se basa en arquitecturas de redes neuronales. Los algoritmos que utiliza se inspiran en la estructura y función

del cerebro, conectando nodos de neuronas hasta formar una red. Se tratará primero la inspiración biológica de la que proceden las redes neuronales y, posteriormente, su definición y sentido al hacer referencia a redes neuronales artificiales.

### Redes neuronales biológicas

El funcionamiento del sistema nervioso humano puede ser pensado en tres etapas:

1. Captación de estímulos por los receptores.
2. Procesado de los estímulos por el cerebro.
3. Desarrollo de la respuesta por los efectores.

La segunda de las etapas es la que cuenta con interés en este trabajo, ya que es donde se encaja la red neuronal, la cual continuamente recibe la información captada por los receptores, la trata y a partir de esta, toma determinadas decisiones que ejecutan los efectores.

Una red neuronal biológica es un conjunto de conexiones sinápticas ordenadas, que se produce como resultado de la unión de unas neuronas a otras en las regiones correspondientes.

Una neurona recoge señales procedentes de otras neuronas a través de un conjunto de delicadas estructuras llamadas dendritas. La neurona emite impulsos de actividad eléctrica a lo largo de una fina y larga fibra denominada axón, que se divide en millones de ramificaciones. El órgano de cómputo es el soma o cuerpo neuronal.

La sinapsis es una aproximación intercelular especializada entre neuronas (puede ser entre neuronas, entre una célula receptora y una neurona o entre una neurona y una célula efectora). Aunque las neuronas son capaces de realizar numerosas tareas por sí mismas, lo verdaderamente interesante surge de la interacción entre las mismas (Artola, 2019).

### Redes neuronales artificiales

Para describir las redes neuronales artificiales se seguirá James et al. (2021).

Una red neuronal artificial toma un vector de entrada de $n$ variables $X = (x_1, x_2, \ldots, x_n)$ y construye una función no lineal $f(X)$ para predecir la respuesta $y$.

En la Figura 3.1 que aparece a continuación se puede ver un ejemplo de una red neuronal artificial sencilla. Este tipo de redes se utilizan en la mayoría de las aplicaciones de aprendizaje automático y reconocimiento de patrones y son redes en las que la información fluye en una sola dirección, desde la entrada hasta la salida, sin bucles ni conexiones hacia atrás. En el ejemplo se utiliza esta red para modelar una respuesta usando $n = 4$ predictores. Las cuatro características $x_1, \ldots, x_4$ constituyen la capa de entrada, que reciben los valores de entrada del sistema. Las flechas indican que cada una de las entradas de la capa de entrada alimenta a cada una de las unidades ocultas (en este ejemplo habría $k = 5$ unidades ocultas). Posteriormente, estas unidades ocultas se conectan con la capa de salida que produce la respuesta $y$.

El modelo de red neuronal es un perceptrón con una sola capa oculta y tiene la siguiente forma:

$$f(X) = \beta_0 + \sum_{i=1}^{K} \beta_i h_i(X) = \beta_0 + \sum_{i=1}^{K} \beta_i g(w_{i0} + \sum_{j=1}^{n} w_{ij} X_j) \tag{3.1}$$

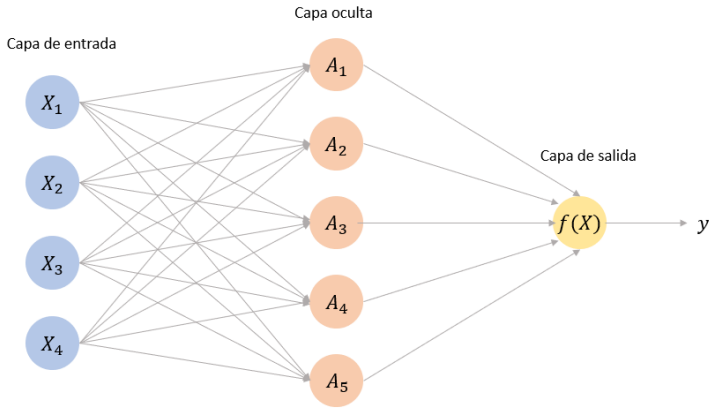


Figura 3.1: Ejemplo de red neuronal.

Este modelo se construye en dos pasos. Primero se construyen las $K$ activaciones $A_k$, $k = 1, \ldots, K$, que se calculan en la capa oculta a partir de las características de entrada $X_1, \ldots, X_n$, de la siguiente forma:

$$A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^{n} w_{kj} X_j) \tag{3.2}$$

donde $g$ es una función de activación no necesariamente lineal (este tipo de funciones se explicarán más adelante). Se puede considerar cada $A_k$ como una transformación diferente de las características originales. Estas $K$ activaciones de la capa oculta se introducen en la capa de salida, dando como resultado la probabilidad estimada de pertenencia a cada clase:

$$P(y = c|X) = \beta_0 + \sum_{i=1}^{K} \beta_i A_i \tag{3.3}$$

en un modelo de clasificación. Todos los parámetros $\beta_0, \ldots, \beta_k$ y $w_{10}, ..., w_{Kn}$ deben estimarse a partir de los datos.

La arquitectura de las redes neuronales depende de gran cantidad de factores, por ejemplo:

- Según el número de capas, pueden ser monocapa o multicapa.
- Según la realimentación, pueden ser no recurrentes, donde la propagación de señales se produce en un único sentido, o recurrentes, las cuales poseen lazos de realimentación que pueden ser entre neuronas de diferentes capas, de la misma capa o incluso de una neurona consigo misma.
- Según el grado de conexión, pueden ser totalmente conectadas, donde todas las neuronas de una capa se encuentran conectadas con las de la siguiente capa (redes no recurrentes) o con las de la anterior (redes recurrentes) o redes parcialmente conectadas, donde no se da la conexión total entre neuronas de diferentes capas, esto ocurre cuando alguna de las sinapsis se pierde (Artola, 2019).

**Función de activación**

Las funciones de activación son componentes clave en las redes neuronales artificiales ya que introducen no linealidades en el proceso de cálculo. Estas funciones se aplican a la salida de cada neurona en la red, transformando la suma ponderada de las entradas en una salida específica.

Al introducir no linealidades, las funciones de activación capacitan a la red para modelar relaciones no lineales en los datos de entrada. Esto es especialmente importante en problemas complejos, como el reconocimiento de objetos en imágenes o el procesamiento del lenguaje natural, donde las relaciones entre los datos no pueden ser capturadas de manera efectiva por funciones lineales. Las funciones de activación también ayudan a regular el flujo de información en la red y a evitar la saturación de las neuronas, permitiendo así un mejor aprendizaje y una mayor capacidad de generalización.

Las funciones de activación más popularmente utilizadas son:

- Sigmoide logístico, que se define formalmente como la ecuación:

$$\sigma(x) = \frac{1}{1 + \exp^{-x}} \tag{3.4}$$

  Esta función toma cualquier valor real como entrada y le asigna un número entre 0 y 1. Su forma característica es una curva en forma de "$S$" como se puede ver en la Figura 3.2, donde los valores muy negativos tienden a 0 y los valores muy positivos tienden a 1. El punto medio de la curva está en $x = 0$, donde la salida es 0.5.

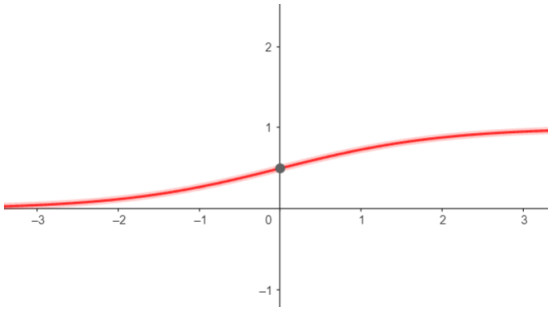


Figura 3.2: Gráfica de la función sigmoide logístico.

El sigmoide logístico es útil en problemas de clasificación binaria, donde se requiere asignar una etiqueta a una muestra entre dos categorías. La salida de la función puede interpretarse como la probabilidad de que una muestra pertenezca a la clase positiva, y la probabilidad complementaria sería la de pertenecer a la clase negativa.

Además, el sigmoide logístico es diferenciable en todo su dominio, lo que es esencial para el entrenamiento de las redes neuronales. La suavidad de la función permite que las actualizaciones de los pesos de la red sean más estables durante el proceso de entrenamiento.

Sin embargo, el sigmoide logístico puede tener problemas en casos de valores extremadamente grandes o pequeños, ya que la función tiende a saturarse en estos rangos, lo que puede dificultar el aprendizaje efectivo. Por esta razón, en muchas situaciones se prefieren otras funciones de activación.

- Tangente hiperbólica, que se define formalmente como la expresión (Sepulveda, 2019):

$$tanh(x) = \frac{2}{1 + \exp^{-2x}} - 1 \tag{3.5}$$

La función *tanh* toma un valor real como entrada y le asigna un número entre $-1$ y 1. Al igual que el sigmoide logístico, tiene una forma en "$S$" como se puede ver en la Figura 3.3, pero está centrada en $x = 0$. Esto significa que el punto medio de la curva es 0, y los valores positivos tienden a valores cercanos a 1, mientras que los valores negativos tienden a valores cercanos a $-1$.

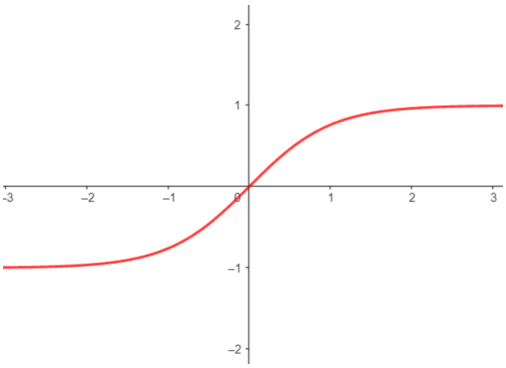


Figura 3.3: Gráfica de la función tangente hiperbólica.

Esta función es útil en redes neuronales debido a su simetría y su rango de valores que incluye tanto valores positivos como negativos. Esta función permite capturar relaciones no lineales más complejas en los datos y es especialmente efectiva en problemas de clasificación binaria donde se busca una salida entre $-1$ y 1.

Al igual que el sigmoide logístico, la función tangente hiperbólica es diferenciable en todo su dominio, lo que es importante para el entrenamiento de redes neuronales.

Aunque esta función tiene ventajas en comparación con el sigmoide logístico, también puede sufrir de saturación en los extremos, lo que dificulta el aprendizaje en casos de valores muy grandes o pequeños. Por esta razón, algunas veces se siguen prefiriendo otras funciones de activación.

- Unidades lineales rectificadas (ReLu), es la más utilizada en los modelos de aprendizaje profundo. Esta función devuelve 0 si recibe una entrada negativa, pero para cualquier valor positivo $x$ devuelve ese valor. Se define como (Artola, 2019):

$$R(x) = \text{máx}(0, x) \tag{3.6}$$

La función ReLU es ampliamente utilizada debido a su eficiencia de cálculo y su capacidad para superar el problema de saturación en las funciones sigmoidales. Al eliminar la región negativa de activación, la función ReLU permite que las neuronas sean más activas y propensas a aprender patrones relevantes en los datos. Además, la función ReLU es lineal para valores positivos, lo que facilita el cálculo de las derivadas durante el entrenamiento de la red neuronal.

Una característica importante de la función ReLU es que no está acotada en la parte superior, lo que significa que las neuronas activadas por ReLU pueden generar salidas arbitrariamente grandes para valores positivos. Esto puede ser beneficioso en algunos casos, pero también puede causar inestabilidad en el aprendizaje si no se controla adecuadamente. Para abordar esto, se han propuesto variantes de la función ReLU que introducen pequeñas pendientes en la región negativa para mejorar el aprendizaje.

En resumen, la función ReLU es una función de activación simple pero potente que ha demostrado ser eficaz en muchas aplicaciones de redes neuronales. Su capacidad para superar la saturación y su eficiencia de cálculo la convierten en una opción popular en la construcción de arquitecturas de redes neuronales para la visión por computadora, el procesamiento del lenguaje natural y otras tareas de aprendizaje automático.

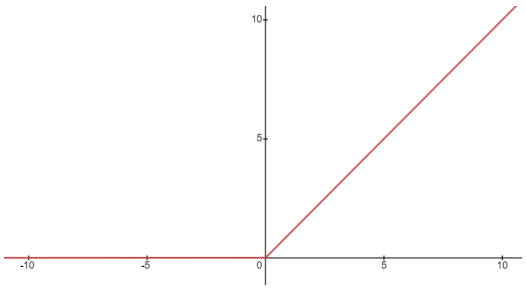


Figura 3.4: Gráfica de la función de unidades lineales rectificadas.

- Softax (función exponencial normalizada), que es una generalización de la función sigmoidea y se define matemáticamente de la siguiente forma (Artola, 2019):

$$\sigma : \mathbb{R}^K \longrightarrow [0,1]^K \; ; \; \sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ para } j = 1, \ldots, K \tag{3.7}$$

Esta función toma como entrada un vector $K-$dimensional de valores reales y produce un vector de la misma dimensión con valores que están en el rango de 0 a 1. Estos valores representan las probabilidades de pertenencia a cada clase. La suma de todas las probabilidades resultantes es igual a 1, lo que garantiza que la salida sea una distribución de probabilidad válida. Es importante destacar que la función softmax, igual que las anteriores, es diferenciable.

La función softmax es especialmente útil en problemas de clasificación multiclase, donde se busca asignar una muestra a una de varias clases posibles. Al utilizar esta función en la capa de salida de la red neuronal, se obtiene una salida que se puede interpretar como la probabilidad de pertenencia a cada clase. La clase con la probabilidad más alta se selecciona como la predicción final.

En resumen, la función softmax es una función de activación utilizada en la capa de salida de redes neuronales para clasificación multiclase. Proporciona una salida que representa una distribución de probabilidad, lo que facilita la interpretación de los resultados y la toma de decisiones basadas en la clasificación de las muestras.

### Fases en el funcionamiento de una red neuronal

En la forma de operar de las redes neuronales se pueden distinguir dos fases claras: la fase de aprendizaje o entrenamiento y la fase de operación o ejecución.

La fase de aprendizaje empieza con un conjunto de pesos aleatorio y busca un conjunto de pesos que permitan a la red desarrollar correctamente una tarea específica. A lo largo de esta fase se va refinando iterativamente la solución hasta alcanzar un nivel de operación suficientemente aceptable. Esta fase se puede desarrollar a dos niveles: a través del modelado de las sinapsis o a través de la creación o destrucción de neuronas. Los algoritmos de aprendizaje se basan normalmente en métodos numéricos iterativos que tratan de reducir al máximo la función de coste. En ocasiones, esto genera problemas en la convergencia del algoritmo. Rigurosamente hablando, la convergencia supone una comprobación de una determinada arquitectura, pues junto con su regla de aprendizaje es capaz de resolver un determinado problema. Los tipos de aprendizaje que pueden distinguirse son: supervisado, no supervisado, híbrido y reforzado.

El aprendizaje supervisado es el tipo más común de aprendizaje en redes neuronales. Se basa en un conjunto de datos etiquetados en el que cada muestra tiene una entrada y una salida deseada asociada. Durante el entrenamiento, el modelo aprende a asociar las entradas a las salidas correspondientes utilizando algoritmos de optimización, como el descenso del gradiente. Una vez entrenado, el modelo puede utilizar este conocimiento para hacer predicciones sobre nuevas entradas no vistas previamente. Este tipo de entrenamiento es el que se utiliza a lo largo del desarrollo práctico de este trabajo.

Por otro lado, el aprendizaje no supervisado implica entrenar un modelo sin la guía explícita de salidas deseadas. En este enfoque, el modelo busca patrones o estructuras ocultas en los datos de entrada sin la necesidad de etiquetas. Algunas técnicas comunes de aprendizaje no supervisado incluyen el agrupamiento (*clustering*) y la reducción de dimensionalidad.

En el aprendizaje reforzado, el modelo interactúa con un entorno y aprende mediante el ensayo y error para maximizar una recompensa acumulada a largo plazo. El modelo aprende a tomar decisiones en función de las señales de recompensa o castigo que recibe a medida que interactúa con el entorno. Este enfoque se utiliza en áreas como los juegos, la robótica y la optimización.

Por último, el aprendizaje híbrido combina múltiples enfoques de aprendizaje para aprovechar sus fortalezas individuales. Por ejemplo, se pueden utilizar técnicas de aprendizaje supervisado y no supervisado juntas para mejorar el rendimiento del modelo. También se puede combinar aprendizaje profundo con otras técnicas de aprendizaje automático tradicionales, como el aprendizaje de características (*feature learning*) o el aprendizaje por transferencia. El objetivo es aprovechar las ventajas de diferentes métodos para abordar problemas complejos o mejorar la eficiencia del entrenamiento.

Una vez que la red ha sido entrenada satisfactoriamente, la fase de aprendizaje termina. Esto implica que los pesos y la estructura de la red quedan fijos, quedando así la red neuronal dispuesta para el procesamiento de los datos. Una de las principales ventajas de las redes neuronales es que la red en sí aprende la relación existente entre los datos, adquiriendo la capacidad de generalizar conceptos.

Dentro de las redes neuronales existen gran cantidad de tipos. En la Tabla 3.1 que se presenta a continuación se explican los tipos más comunes.

| **Redes** | **Aplicación** |
|---|---|
| *Recurrent Neural Networks*(RNN) | Reconocimiento de voz y de escritura manual |
| *Redes Long-Short Term Memory* (LSTM) o *Gated Recurrent Unit* (GRU) | Comprensión de textos de lenguajes naturales, reconocimiento de escritura manual, de voz, de gestos y captura de imágenes |
| *Convolutional Neural Network* (CNN) | Reconocimiento de imágenes, análisis de videos, procesamiento de lenguajes naturales |
| *Deep Belief Network* (DBN) | Reconocimiento y recuperación de imágenes, comprensión de lenguajes naturales, predicción de fallos |

Tabla 3.1: Tipos de redes neuronales más comunes.

Para el problema que se aborda en este trabajo se van a utilizar las redes neuronales con la tercera de las arquitecturas: *Convolutional Neural Network* (CNN) o redes neuronales convolucionales. Estas redes se han convertido en la elección predominante a la hora de ejecutar la técnica de *Computer Vision* debido a su capacidad para aprovechar la estructura espacial de las imágenes, ya que capturan características locales, como bordes, texturas y patrones específicos, a su capacidad de aprendizaje de características invariantes a las traslaciones y a su estructura jerárquica, que permite que estas redes aprendan gradualmente características complejas a partir de las primitivas básicas detectadas en las primeras capas. Estas características les permiten alcanzar un rendimiento sobresaliente en una amplia gama de tareas de visión por computador.

### 3.1.2. Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales representan un modelo de aprendizaje profundo que comenzó a obtener buenos resultados en competiciones de clasificación de imágenes en la década de 2010, primero con Dan Ciresan ganando dos competiciones *ICDAR* 2011 (competición de reconocimiento de caracteres chinos) y el *IJCNN* 2011 (competición de reconocimiento de señales de tráfico alemán) y más tarde, en otoño de 2012, con el grupo de Hinton ganando el reto de reconocimiento visual a gran escala *ImageNet*, de gran repercusión (Chollet, 2022). Este tipo de redes representan el tipo de modelo que ahora se utiliza casi universalmente en aplicaciones de visión por ordenador.

La CNN es un tipo de red neuronal convolucional que procesa sus capas imitando el córtex visual del cerebro humano para identificar distintas características en la entrada. Para ello, contiene varias capas ocultas especializadas y con una jerarquía: esto significa que las primeras capas detectan propiedades o formas básicas y se van especializando hasta llegar a capas más profundas capaces de reconocer formas más complejas como siluetas o rostros (Artola, 2019).

Las redes neuronales convolucionales son simplemente redes neuronales que utilizan la convolución en lugar de la multiplicación general de matrices en al menos una de sus capas. A continuación, se describe lo que es una convolución y la motivación por la que se usa en una red neuronal. Luego se explicará una operación llamada *pooling*, que se emplea en la gran mayoría de redes neuronales convolucionales. Normalmente, la operación utilizada en una red neuronal convolucional no se corresponde exactamente con la definición de convolución utilizada en otros campos como podrían ser las matemáticas puras (Goodfellow, 2016).

**¿Qué es una convolución?**

De forma general, una convolución es una operación sobre dos funciones de un argumento real. En terminología de redes neuronales, una de estas funciones se denomina entrada y la otra núcleo, y la salida se suele denominar mapa de características.

En las aplicaciones de aprendizaje automático, la entrada de esta operación suele ser una matriz multidimensional de datos y el núcleo suele ser una matriz multidimensional de parámetros que son adaptados por el algoritmo de aprendizaje. Como cada elemento de la entrada y del núcleo debe almacenarse por separado, se supone que estas funciones son cero en todas partes excepto en el conjunto finito de puntos para los que se almacenan los valores. Esto significa que, en la práctica se puede implementar la suma infinita como una suma sobre un número finito de elementos de la matriz. Además, a menudo se utilizan convoluciones sobre más de un eje a la vez. Por ejemplo, si se utiliza una imagen bidimensional $t$ como entrada, probablemente también se utilice un núcleo bidimensional $K$:

$$S(i,j) = (t * K)(i,j) = \sum_m \sum_n t(m,n)K(i-m, j-n) \tag{3.8}$$

Como la convolución es una operación conmutativa también se puede escribir:

$$S(i,j) = (K * t)(i,j) = \sum_m \sum_n t(i-m, j-n)K(m,n) \tag{3.9}$$

Se muestra en la Figura 3.5 un ejemplo de una convolución bidimensional. En este ejemplo la salida se va a restringir sólo a las posiciones en las que el núcleo se encuentra completamente dentro de la imagen, lo que se suele denominar "convolución válida".

En las redes neuronales convolucionales se suele escoger un núcleo más pequeño que la entrada. Así, al procesar una imagen que puede tener miles o millones de píxeles si se utiliza un núcleo que ocupe sólo decenas o centenas de píxeles, se reduciría la memoria que se necesita para el modelo y se mejoraría su eficacia.

Para esta sección se ha seguido (Goodfellow, 2016).

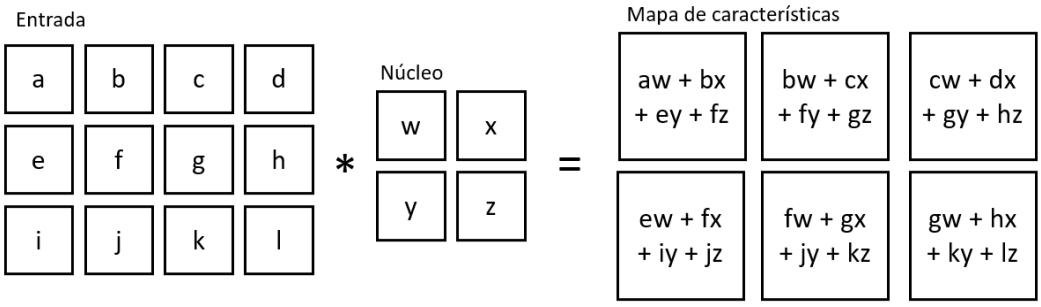


Figura 3.5: Ejemplo de convolución en dos dimensiones.

### Capas completamente conectadas y agrupación o *pooling*

Las redes convolucionales reales rara vez se construyen solo a partir de capas convolucionales. Por lo general, también tienen otros tipos de capas. La más simple es la capa totalmente conectada o *fully connected layer*, un tipo de capa de normalización. Esta es solo una capa de red neuronal normal donde todas las salidas de la capa anterior están conectadas a todos los nodos en la siguiente capa. Normalmente, estas capas aparecen hacia el final de la red.

La función principal de la capa totalmente conectada es tomar las características extraídas por las capas convolucionales y utilizarlas para realizar la clasificación. Esta capa permite que la red aprenda representaciones no lineales complejas a partir de las características extraídas en las etapas anteriores.

Es en la capa totalmente conectada donde se realiza la interpretación global de las características extraídas, ya que el resto de las capas no tienen conocimiento del contexto global de la imagen. Cada unidad o neurona en esta capa está conectada a todas las unidades de la capa anterior. Esto significa que cada salida de la capa anterior contribuye a la activación de cada unidad en la capa totalmente conectada.

La capa totalmente conectada se utiliza para aprender y combinar características de manera no lineal y para mapear las características extraídas a las clases o valores de salida deseados. Cada unidad en esta capa representa un cierto patrón o combinación de características extraídas, y su activación indica la presencia o ausencia de ese patrón en la entrada.

Finalmente, la capa totalmente conectada produce una salida que se puede utilizar para clasificar una imagen en diferentes categorías o para predecir un valor numérico, dependiendo de la tarea específica que se esté abordando.

En resumen, la capa totalmente conectada en una red neuronal convolucional es responsable de aprender características globales y realizar la clasificación o regresión final basada en las características extraídas por las capas convolucionales anteriores.

El otro tipo clave de capa que se ven en las redes neuronales convolucionales es la capa de agrupación o *pooling*. La más común es la agrupación máxima o *max pooling*, en la cual la matriz de entrada se divide en segmentos de igual tamaño y el valor máximo en cada segmento se toma para llenar el elemento correspondiente de la matriz de salida (Ceballo, 2019). Estas capas se dedican a reducir el tamaño de las salidas de las convoluciones. Se ve ahora en la Figura 3.6 un ejemplo del funcionamiento de una agrupación máxima, con una matriz de entrada $4 \times 4$, una agrupación con filtro de $2 \times 2$ y

separación (*stride*) de 2 píxeles.

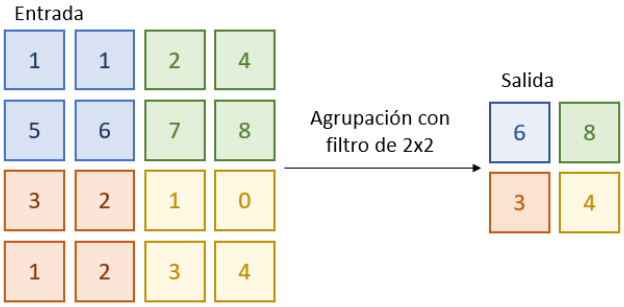


Figura 3.6: Ejemplo de *max pooling*.

Además, la agrupación ayuda a que la representación sea invariante a pequeñas traslaciones de la entrada, es decir, que si en la entrada los valores de la mayoría de las salidas no cambian entonces la mayoría de las salidas agrupadas tampoco cambian. Esta propiedad es útil si importa más la presencia de una característica que la posición en la que esté, por ejemplo, si se está analizando si una imagen contiene una cara, sólo se necesita saber si en la imagen hay ojos, no se necesita saber la ubicación de los ojos con una precisión de píxeles perfecta.

Se muestra ahora en la Figura 3.7 un ejemplo de cómo esta agrupación induce la invarianza. En la fila inferior de muestran las salidas de la capa anterior, es decir, las entradas a la capa de agrupación y en la fila superior las salidas de esta capa. En la primera imagen se ven los resultados de la agrupación máxima, con una separación (*stride*) de un píxel entre las regiones de agrupación y una anchura de la región de agrupación de tres píxeles. Y en la segunda imagen, aparece la misma entrada pero se ha desplazado un píxel a la derecha, así, todos los valores de la fila inferior han cambiado, pero sólo la mitad de los de la fila superior lo han hecho, ya que las agrupaciones máximas sólo son sensibles al valor máximo, no a la ubicación exacta (Goodfellow, 2016).

Una vez que ya se han explicado cuáles son las capas básicas que componen una red neuronal convolucional, en la Figura 3.8 se puede ver la distribución habitual de estos componentes:

### 3.1.3. Redes neuronales convolucionales profundas (DCNN)

Las redes neuronales convolucionales profundas, también conocidas como *Deep Convolutional Neural Networks* (DCNN), son una de las técnicas más populares dentro del campo del *Machine Learning*, y sobre todo, en el reconocimiento de imágenes (Chanampe et al., 2019).

A diferencia de las CNN tradicionales, que suelen tener unas pocas capas convolucionales seguidas de una o dos capas totalmente conectadas, las DCNN están diseñadas con una estructura más profunda, que consta de múltiples capas convolucionales y de agrupación.

El diseño de las redes neuronales convolucionales profundas no es una tarea sencilla, ya que no existe un patrón definido para establecer el número de capas, los tipos de estas, las conexiones entre ellas o un proceso definido que ayude a definirlas (Artola, 2019).

La profundidad adicional en las DCNN permite aprender jerarquías de características más comple-

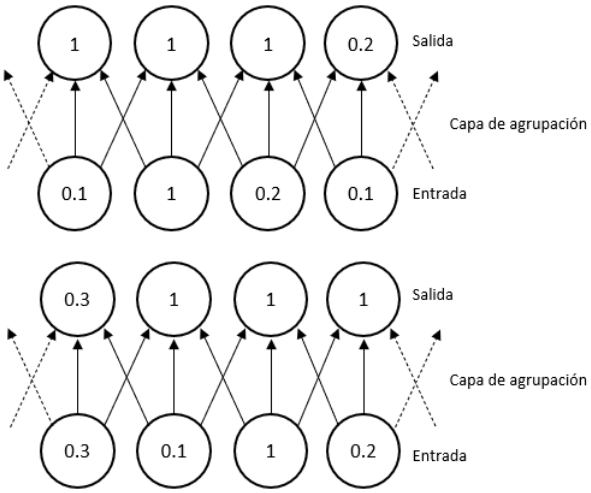


Figura 3.7: Ejemplo de la invarianza en la agrupación.

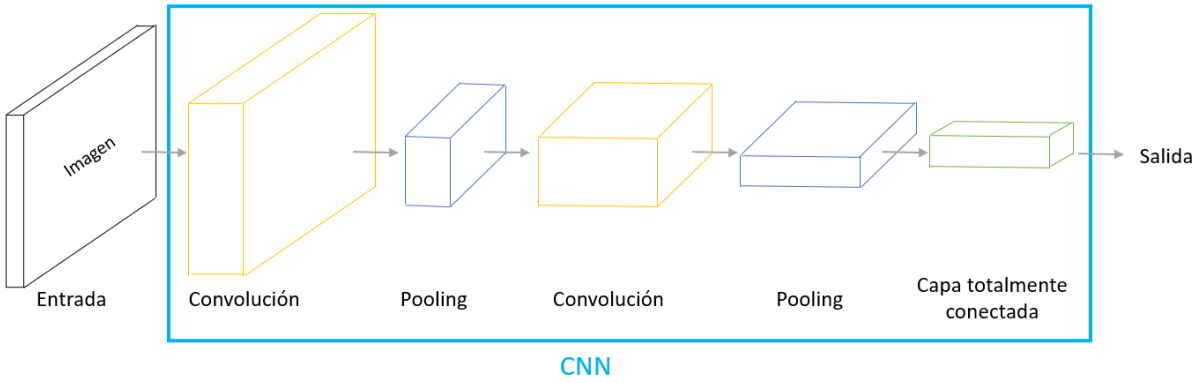


Figura 3.8: Red neuronal convolucional.

jas y abstracciones más profundas. Cada capa convolucional aprende filtros o características locales en diferentes niveles de abstracción, y las capas de agrupación reducen la dimensión espacial de las características, manteniendo la información más relevante. Además de las capas convolucionales y de agrupación, las DCNN también pueden incluir capas de normalización, como la normalización de lotes (*batch normalization*), que aceleran el entrenamiento y mejoran el rendimiento de la red.

Estas redes requieren un gran número de datos de entrenamiento etiquetados y requieren muchos recursos de procesamiento y memoria, lo que podría consumir bastante tiempo. A veces, el entrenamiento de este tipo de arquitecturas se complica por problemas de sobreajuste (*overfitting*) y bajo ajuste (*underfitting*) cuando el conjunto de datos para el entrenamiento o el tiempo de entrenamiento son pequeños (Chanampe et al., 2019).

Para evitar estos problemas, en vez de construir las DCNN desde cero, lo que se hace en el reconocimiento de imágenes es utilizar redes que ya se ha demostrado que son precisas y con tasas de acierto elevadas y hacer cambios sobre algunas de sus capas, mediante una técnica denominada *trans-*

*fer learning* o aprendizaje por transferencia (descrito en el capítulo siguiente). Algunos ejemplos de este tipo de redes convolucionales profundas ya construidas y que obtienen buenos resultados para el reconocimiento de imágenes son: *LeNet, AlexNet, VGG, GoogleNet, ResNet, DenseNet, MobileNet* (Artola, 2019).

## 3.2. Medidas de evaluación

La evaluación de un modelo de clasificación de imágenes es crucial para comprender su rendimiento y su capacidad para generalizar a datos nuevos. Para lograrlo, se utilizan diversas medidas que proporcionan información sobre la precisión y el desempeño del modelo.

En esta sección se hablará de cuatro medidas básicas para evaluar un modelo de clasificación, que son: la precisión global, el valor predictivo positivo, la sensibilidad y el *F1 score*. Para hablar de ellas lo primero que se hará será explicar la matriz de confusión, que es en donde se basan las mismas. Por sencillez primero se explicará cada métrica para un problema de clasificación binaria y posteriormente se generalizarán para un problema con varias categorías, como es el caso del ejemplo práctico que se trata en el trabajo.

Estas medidas, entre otras, proporcionan una evaluación completa del modelo de clasificación de imágenes, permitiéndonos comprender su rendimiento en diferentes aspectos. Al considerar estas medidas en conjunto, podremos tomar decisiones más informadas sobre cómo mejorar y ajustar el modelo para obtener resultados más precisos y confiables en futuras tareas de clasificación de imágenes.

### 3.2.1. Matriz de confusión

Una matriz de confusión o matriz de control (CM) es una tabla de contingencia que sirve como herramienta estadística para el análisis de observaciones emparejadas, permite visualizar el desempeño de un algoritmo que emplea aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa la clase real.

Así, la matriz de confusión es una matriz cuadrada $c \times c$ donde $c = \{c_1, \ldots, c_c\}$ es el número de clases de referencia o la variable respuesta de nuestro problema. Cada uno de los $c^2$ elementos de la matriz se denominan celdas. En cada celda $(i, j)$ se almacena el conteo del número de elementos de la clase $i$ clasificados como clase $j$ por el modelo. Las celdas de la diagonal de la matriz de confusión contienen las cantidades correspondientes a los ítems bien clasificados, es decir, aquellos que pertenecen a la clase $i$ y el modelo los clasifica como clase $i$. Las celdas de fuera de la diagonal se corresponden con los errores de predicción que comete el modelo.

La matriz de confusión ofrece una vista completa de la distribución de los aciertos y errores entre clases, pero es difícil de manejar de manera sencilla (Ariza et al., 2018).

Además, no es una medida de rendimiento como tal, pero casi todas las métricas de rendimiento están basadas en ella y en los números que contiene. Para explicar estos números supongamos que tenemos una matriz de confusión $2 \times 2$ como la de la Figura 3.9, y serían:

- *Verdaderos positivos o True Positive (TP)*: son los casos en los que los elementos de la clase $i$ se han clasificado como clase $i$.

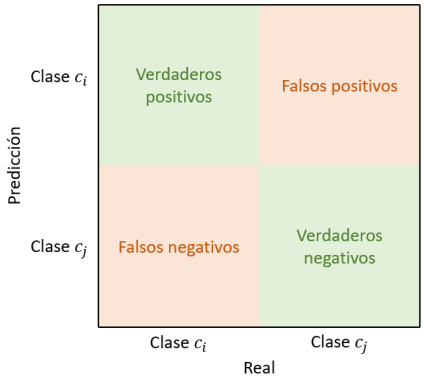


Figura 3.9: Matriz de confusión.

- *Verdaderos negativos o True Negative (TN)*: son los casos en los que los elementos que son de la clase $j$ se han clasificado como clase $j$.
- *Falsos positivos o False Positive (FP)*: son los casos en los que los elementos que son de la clase $i$ se han clasificado como clase $j$.
- *Falsos negativos o False Negative (FN)*: son los casos en los que los elementos que son de la clase $j$ se han clasificado como clase $i$.

El escenario ideal se daría cuando no existieran falsos positivos ni falsos negativos.

Si se habla ahora de un problema de clasificación con $c$ categorías, la matriz de confusión sería similar, pero hay algunas diferencias. Los verdaderos positivos ($TP_i$) asociados a una categoría particular serían el número de observaciones que pertenecen a la categoría y que se han clasificado en ella correctamente. Por otra parte, los falsos negativos ($FN_i$) para una categoría será el número de observaciones pertenecientes a la categoría que se han clasificado incorrectamente. Los verdaderos negativos ($TN_i$) asociados a una categoría serán los elementos que no pertenecen a esa categoría y para los que además no se predice que pertenezcan a ella. Por último los falsos positivos ($FP_i$) para una categoría serán el número de observaciones para las que se predice que pertenecen a la categoría cuando no es así en realidad. En este caso habría que considerar un subíndice $i = 1, \ldots, c$ para indicar a qué categoría hacen referencia los valores anteriores.

### 3.2.2. Precisión global o tasa de aciertos (*accuracy*)

La precisión global representa el número de predicciones correctas que hace el modelo sobre el número total de predicciones realizadas, es decir:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.10}$$

Esta medida por separado puede ser engañosa si las clases están desbalanceadas, por eso es necesario tener en cuenta otras medidas. (Artola, 2019)

Se debe tener cuidado con esta medida cuando las clases no están balanceadas (como es el caso del ejemplo práctico), en esos casos podría emplearse otra medida que sería la precisión balanceada o *balanced accuracy*, que sigue la siguiente expresión (Casal, 2021):

$$Balanced\ accuracy = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{3.11}$$

En el caso de un problema de clasificación multiclase, la precisión global no es más que el cociente entre la suma de verdaderos positivos asociados a cada una de las categorías entre el total de observaciones (supongase $n$), es decir, la proporción de observaciones correctamente clasificadas por el modelo (García, 2022).

$$Accuracy = \frac{\sum_{i=1}^{c} TP_i}{n} \tag{3.12}$$

### 3.2.3. Valor predictivo positivo (*precision*)

El valor predictivo positivo es una medida que indica la proporción de instancias positivas correctamente clasificadas sobre el total de instancias clasificadas como positivas. En otras palabras, mide la exactitud de las predicciones positivas del modelo. La fórmula para calcular el valor predictivo positivo es:

$$Precision = \frac{TP}{TP+FP} \tag{3.13}$$

Esta medida implica que si el modelo solo predice un elemento como clase $i$, y este es clase $i$, el valor del valor predictivo positivo sería del 100 %.

En el caso de un problema de clasificación con $c$ clases, la ecuación del valor predictivo positivo se puede generalizar de manera que cada categoría tenga asociado un valor particular que podrá calcularse para cada $i$, con $i = 1, \ldots, c$, según la siguiente expresión (García, 2022):

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{3.14}$$

### 3.2.4. Sensibilidad (*recall*)

La sensibilidad mide la proporción de instancias positivas correctamente clasificadas con respecto al total de instancias positivas reales. En otras palabras, mide la capacidad del modelo para identificar correctamente los casos positivos. La fórmula para calcular la exhaustividad es:

$$Recall = \frac{TP}{TP+FN} \tag{3.15}$$

Si se supone una sensibilidad de $0,5$, significa que el modelo es capaz de identificar un 50 % de los elementos de clase $i$.

En el caso de un problema de clasificación multiclase, la sensibilidad se puede generalizar de manera que cada categoría tenga asociado un valor particular que podrá calcularse para cada $i$, con $i = 1, \ldots, c$, según la siguiente expresión (García, 2022):

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{3.16}$$

### 3.2.5. F1-score

Representa una puntuación intermedia entre el valor predictivo positivo y la sensibilidad anteriores. Puede hacerse simplemente usando la media aritmética entre ambos:

$$F1 = \frac{precision + recall}{2} \tag{3.17}$$

La media aritmética es la forma más común de calcular un promedio, donde se suman los valores y se dividen por el número total de elementos. Sin embargo, en el caso del *F1 score*, no es apropiado calcular simplemente calcular el promedio aritmético del valor predictivo positivo y la sensibilidad debido a su relación de compensación. Por ellos se utiliza algo más equilibrado, como puede ser la media armónica:

$$F1_{armonica} = \frac{2 * precision * recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \tag{3.18}$$

La media armónica de una cantidad finita de números es igual al recíproco, o inverso, de la media aritmética de los recíprocos de dichos valores.

El valor predictivo positivo y la sensibilidad están inversamente relacionadas entre sí: a menudo, cuando se mejora una de ellas, la otra disminuye y viceversa. En otras palabras, hay un intercambio entre la capacidad de un modelo para hacer predicciones precisas y su capacidad para identificar correctamente todos los casos positivos.

La media armónica es una medida que tiene en cuenta esta compensación y penaliza más fuertemente los valores bajos en comparación con la media aritmética. Al utilizar la media armónica para calcular el *F1 score*, se asegura de que el valor resultante sea más sensible a las bajas puntuaciones tanto en el valor predictivo positivo como en sensibilidad.

El *F1 score* también se puede definir en el caso de hablar de un problema de clasficación multiclase, puede utilizarse la media armónica del valor predictivo positivo y de la sensibilidad o también la siguiente expresión (García, 2022):

$$F1_{armonica,i} = \frac{2TP_i}{2TP_i + FP_i + FN_i} \tag{3.19}$$

### 3.2.6. Pérdida

La función de pérdida o *loss* es una medida que se utiliza para evaluar qué tan bien se está desempeñando un modelo de aprendizaje automático en una tarea en particular. La función de pérdida que se utilice dependerá del tipo de problema y de los objetivos específicos del modelo. Algunos ejemplos comunes incluyen la pérdida de entropía cruzada (cross-entropy loss) para problemas de clasificación, la pérdida cuadrática (mean squared error) para problemas de regresión, entre otras.

En este trabajo se va a utilizar la entropía cruzada, que se explica a continuación. Esta función de pérdida mide la discrepancia entre la distribución de probabilidad predicha por el modelo y la distribución de probabilidad real de las etiquetas de los datos.

En el caso de un problema de clasificación multiclase, se utiliza una versión generalizada de la pérdida de entropía cruzada conocida como *categorical cross-entropy*. La fórmula general para la pérdida de entropía cruzada categórica es:

$$loss = -\sum(c_i * log(p_i)) \tag{3.20}$$

donde $c_i$ es la etiqueta verdadera de la clase $i$, $p_i$ es la probabilidad predicha por el modelo de que un elemento pertenezca a la clase $i$ y el sumatorio representa la suma sobre todas las clases.

El objetivo es minimizar esta función de pérdida durante el proceso de entrenamiento, lo que implica que el modelo ajuste sus parámetros para que las predicciones se acerquen lo más posible a las etiquetas reales.

# Capítulo 4

# Metodología

En el campo del aprendizaje profundo, una de las técnicas más poderosas y ampliamente utilizadas es el *transfer learning*, que permite aprovechar el conocimiento adquirido por una red neuronal entrenada en una tarea para aplicarlo a otra tarea relacionada. Esta técnica ha demostrado ser especialmente efectiva cuando los conjuntos de datos de entrenamiento son pequeños o cuando no se dispone de suficiente potencia de cálculo para entrenar una red desde cero.

Una de las arquitecturas de red neuronal popularmente utilizadas para *transfer learning* es *MobileNet*, una red neuronal convolucional diseñada específicamente para aplicaciones con recursos computacionales limitados, como dispositivos móviles. Su estructura se basa en el concepto de "factorización en profundidad" y utiliza operaciones convolucionales separables en lugar de las convoluciones tradicionales, lo que permite obtener una arquitectura más liviana y eficiente sin comprometer el rendimiento.

Sin embargo, en algunos casos, puede ser necesario construir una red propia adaptada a un problema o conjunto de datos específico, como ocurre en el desarrollo del ejemplo práctico en el que se basa este trabajo. Para ello, se puede utilizar *TensorFlow* y *Keras*, dos bibliotecas de aprendizaje profundo ampliamente utilizadas. *TensorFlow* proporciona una plataforma flexible para construir y entrenar redes neuronales, mientras que *Keras*, que se integra perfectamente con *TensorFlow*, ofrece una interfaz de alto nivel que facilita la construcción y el ajuste de modelos de aprendizaje profundo.

Al combinar las capacidades del *transfer learning,* las arquitecturas de redes neuronales como *MobileNet* y las herramientas como *TensorFlow* y *Keras*, se pueden aprovechar modelos preentrenados, adaptarlos a las necesidades y crear redes neuronales personalizadas para abordar problemas específicos de clasificación de imágenes con mayor eficacia y eficiencia. Esto permite acelerar el proceso de desarrollo de modelos de aprendizaje profundo y obtener resultados prometedores incluso con recursos limitados.

En este capítulo se explicarán estos conceptos con el objetivo de construir de forma eficiente la red que se utilizará en el ejemplo práctico sobre los documentos bancarios.

## 4.1. *Transfer learning*

El *transfer learning*, o aprendizaje por transferencia, es una técnica utilizada en el campo del aprendizaje automático que consiste en aprovechar el conocimiento adquirido por un modelo entrenado previamente en una tarea determinada para mejorar el desempeño en una tarea relacionada pero distinta (Andrade, 2021).

En lugar de entrenar un modelo desde cero para cada tarea específica, este método permite reutilizar los conocimientos y patrones aprendidos por un modelo previamente entrenado en una tarea diferente. Esto es posible debido a que, en muchos casos, los modelos entrenados en grandes conjuntos de datos de una tarea general tienen la capacidad de capturar características generales y representaciones abstractas que son útiles en diferentes dominios.

El proceso de *transfer learning* generalmente se lleva a cabo en dos etapas principales. En primer lugar, se entrena un modelo inicial utilizando un conjunto de datos grande y diverso en una tarea inicial. Este modelo puede ser un modelo de aprendizaje profundo, como una red neuronal convolucional (CNN) o una red neuronal recurrente (RNN), que ha demostrado ser eficiente en algunos campos.

Una vez que el modelo inicial ha sido entrenado en la tarea inicial, se procede a la segunda etapa, que implica ajustar o reentrenar el modelo en la tarea de interés específica. En esta etapa, el modelo se entrena con un conjunto de datos más pequeño y específico para la nueva tarea. A menudo, solo se ajustan los últimos niveles o capas del modelo, mientras que las capas iniciales, que capturan características generales, se mantienen fijas.

El beneficio clave del *transfer learning* es que permite aprovechar el conocimiento previo del modelo para reducir la cantidad de datos necesarios y el tiempo de entrenamiento en la nueva tarea. Esto es especialmente útil en escenarios donde los conjuntos de datos son limitados o cuando entrenar un modelo desde cero no es factible debido a restricciones computacionales.

Además de acelerar el proceso de entrenamiento, el *transfer learning* también puede mejorar el rendimiento del modelo en la nueva tarea, ya que el conocimiento previo y las representaciones aprendidas pueden ayudar al modelo a generalizar mejor y a capturar características relevantes en la nueva tarea.

## 4.2. *MobileNet*

*MobileNet* son una clase de modelos eficientes para aplicaciones de visión móviles. Se basan en arquitecturas simplificadas que utilizan circunvoluciones separables en profundidad para construir redes neuronales profundas y livianas. Estos modelos son eficientes en una amplia gama de aplicaciones y casos de uso, incluida la detección de objetos, la clasificación de grano fino, los atributos faciales y la geolocalización a gran escala.

Estos modelos están entrenados sobre *ImageNet*, un *dataset* de imágenes de $224 \times 224$ píxeles. Es uno de los conjuntos de datos más ampliamente utilizados en el campo del aprendizaje automático y la visión por computadora, que contiene millones de imágenes etiquetadas en miles de categorías diferentes. Este *dataset* se ha utilizado como punto de referencia para evaluar y comparar el rendimiento de diferentes algoritmos de reconocimiento de imágenes y modelos de redes neuronales. Fue creado inicialmente por los investigadores de la Universidad de Stanford y se ha convertido en una referencia estándar en la comunidad científica.

*Imagenet* contiene más de $1,4$ millones de imágenes etiquetadas, y estas imágenes se dividen en aproximadamente 1.000 categorías o clases diferentes. Cada imagen en el conjunto de datos está asociada con una única etiqueta que describe la clase a la que pertenece. El aprendizaje profundo y las redes neuronales convolucionales (CNN) han logrado un gran éxito en la clasificación de imágenes en *Imagenet*, alcanzando niveles de precisión cercanos al rendimiento humano en muchas tareas (Andrade, 2021).

### 4.2.1. *MobileNet V2*

El modelo *MobileNet V2* es una red neuronal convolucional formada por 9 capas convolucionales, que son: una capa convolucional normal, una capa de convolución expansión y 7 módulos *bottleneck*. Además cuenta con una capa de aplanamiento, una capa totalmente conectada y una función clasificadora *SoftMax*. En la Tabla 4.1 que aparece a continuación se puede ver en detalle cómo es la estructura de este modelo en función de las capas con las que cuenta.

| **Dimensiones entrada** | **Dimensiones salida** | **Capa** | **Número de repeticiones** | **Paso** |
|---|---|---|---|---|
| $224 \times 224 \times 3$ | $112 \times 112 \times 32$ | Convolución clásica | 1 | 2 |
| $112 \times 112 \times 32$ | $112 \times 112 \times 16$ | Módulo *bottleneck* | 1 | 1 |
| $112 \times 112 \times 16$ | $56 \times 56 \times 24$ | Módulo *bottleneck* | 2 | 2 |
| $56 \times 56 \times 24$ | $28 \times 28 \times 32$ | Módulo *bottleneck* | 3 | 2 |
| $28 \times 28 \times 32$ | $14 \times 14 \times 64$ | Módulo *bottleneck* | 4 | 2 |
| $14 \times 14 \times 64$ | $14 \times 14 \times 96$ | Módulo *bottleneck* | 3 | 1 |
| $14 \times 14 \times 96$ | $7 \times 7 \times 160$ | Módulo *bottleneck* | 3 | 2 |
| $7 \times 7 \times 160$ | $7 \times 7 \times 320$ | Módulo *bottleneck* | 1 | 1 |
| $7 \times 7 \times 320$ | $7 \times 7 \times 1280$ | Convolución expansión | 1 | 1 |
| $7 \times 7 \times 1280$ | $1 \times 1 \times 1280$ | Capa de aplanamiento | 1 | - |
| $1 \times 1 \times 1280$ | $1 \times 1 \times 1000$ | Capa totalmente conectada | 1 | - |
| $1 \times 1 \times 1000$ | $1 \times 1 \times 1000$ | *Softmax* | 1 | - |

Tabla 4.1: Estructura de *MobileNet V2*.

Para comprender correctamente la estructura de este modelo se explicará paso a paso cada una de las capas que utiliza. Algunas de estas capas ya han sido explicadas en una sección anterior, como las convoluciones clásicas, la capa de aplanamiento, la totalmente conectada o la función *softmax*, pero el resto se explica a continuación:

### Módulo *bottleneck*

En *MobileNet V2* se combinan las etapas de convolución, normalización y activación en módulos llamados *bottleneck* formando una capa de la red. Estos módulos buscan aumentar la cantidad de canales para así usar menos pesos y operaciones en la extracción de características mediante las convoluciones *depthwise*. En la Figura 4.1 se muestra la estructura de estos módulos:

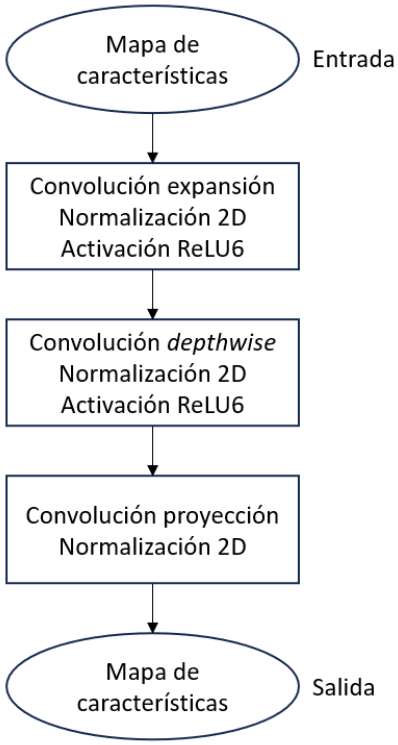


Figura 4.1: Estructura de los módulos *bottleneck*.

Los módulos *bottleneck* reciben como entrada el mapa de características obtenido por la capa anterior, y le aplican una convolución expansión que aumenta el número de canales, una normalización 2D y una activación ReLU6 obteniendo un nuevo mapa. Una vez aumentada la cantidad de canales del mapa de entrada, los módulos utilizan convoluciones *depthwise* para extraer las características, seguidos de la etapa de normalización 2D y activación ReLU6, obteniendo de nuevo como salida otro mapa de características. Finalmente, el módulo reduce la cantidad de canales mediante convoluciones de proyección y normalización 2D, obteniendo el mapa de características de salida.

### Convoluciones *depthwise*

Las convoluciones *depthwise* consisten en realizar una sola convolución de tamaño $3 \times 3$ a cada canal de entrada para obtener un canal de salida, permitiendo disminuir la computación en comparación con las convoluciones clásicas, en las cuales es necesario realizar tantas convoluciones como canales de entrada para obtener un canal de salida. En este tipo de convoluciones el número de canales de entrada es igual al número de canales de salida. En el modelo *MobileNet V2*, estas capas se utilizan para extraer las características de cada mapa.

La ventaja principal de las convoluciones *depthwise* radica en su capacidad para reducir drásticamente el número de operaciones y parámetros necesarios en comparación con las convoluciones convencionales. Esto se debe a que se aplican filtros separados a cada canal de entrada en lugar de tener filtros que se extienden a través de todos los canales. Esto no solo reduce la complejidad computacional, sino que también permite un aprendizaje más eficiente en términos de memoria y capacidad de generalización.

### Normalización

La normalización es una técnica utilizada para mejorar la velocidad de entrenamiento y la estabilidad de las redes neuronales. En el caso de *MobileNet V2* esta técnica se utiliza sobre cada mapa de características para evitar que los valores de estos se disparen. Para aplicar la nomalización 2D se utiliza la siguiente fórmula:

$$\text{MC salida} = \frac{\text{MC entrada} - E(\text{MC entrada})}{Var(\text{MC entrada}) + \epsilon} * \gamma + \beta \tag{4.1}$$

En esta fórmula, MC hace referencia a mapa de características. El mapa de características de entrada en la normalización es la salida de la etapa de convolución, $E(\text{MC entrada})$ es la media del mapa de características de entrada, $Var(\text{MC entrada})$ es la varianza de ese mapa de características, $\gamma$ y $\beta$ son parámetros multiplicativos y adictivos respectivamente y $\epsilon$ es un parámetro de estabilidad para evitar que la división no esté definida. Durante el proceso de entrenamiento se calculan $\gamma$ y $\beta$, y $\epsilon$ suele ser tomado como 0.00001.

Esta normalización 2D o normalización por lotes consiste en normalizar las activaciones en dos dimensiones y se realiza para cada canal de activación por separado. Además, tiene varios beneficios a la hora de utilizarse en el entrenamiento de redes neuronales:

- Estabilización del entrenamiento: ayuda a estabilizar el proceso de entrenamiento, ya que evita que los valores se vuelvan demasiado grandes o pequeños, lo que podría dificultar la convergencia del modelo.

- Aceleración del entrenamiento: al normalizar las activaciones, se reduce la dependencia de la inicialización de los pesos de la red. Esto permite que los pesos se actualicen y ajusten más rápidamente durante el entrenamiento, acelerando así la convergencia.

- Regularización: reduce el riesgo de sobreajuste al proporcionar una regularización adicional durante el entrenamiento.

- Mayor capacidad de generalización: al normalizar las activaciones en cada lote de entrenamiento, la red puede adaptarse mejor a las diferentes estadísticas de los datos en cada lote.

La función de activación utilizada por *MobileNet V2* es ReLU6, una función de activación como las unidades lineales rectificadas pero que evita que las activaciones sean demasiado grandes. La fórmula matemática de ReLU6 es (Hollemans, 2018):

$$y = min(max(0, x), 6) \tag{4.2}$$

Y su gráfica es la de la Figura 4.2:

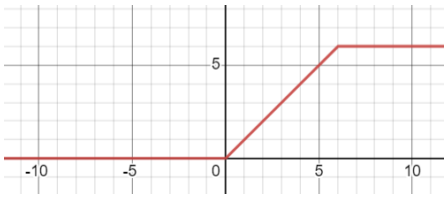


Figura 4.2: Gráfica de la función ReLU6.

### Capa de convolución expansión y proyección

Las convoluciones de expansión y proyección son muy similares a las convoluciones clásicas, pero a diferencia de estas, realizan la convolución siempre por filtros de tamaño $1 \times 1$ en vez de $K \times K$, convirtiendo la operación de convolución en sólo multiplicar la imagen por un escalar y luego sumar los resultados de cada canal de entrada para obtener un canal de salida. La diferencia entre estas convoluciones es que en las de expansión la cantidad de canales de salida aumenta con respecto a la cantidad de canales de entrada y en las de proyección pasa lo contrario.

En las Figuras 4.3, 4.4 y 4.5 ilustramos a continuación cómo se comportaría una convolución clásica, frente a cómo lo harían las convoluciones de expansión y proyección y a cómo lo harían las convoluciones *depthwise* explicadas anteriormente.

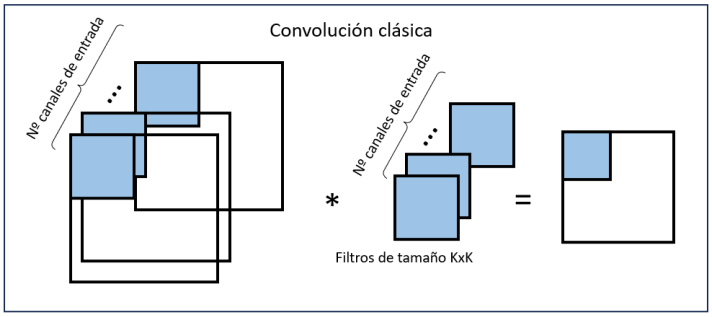


Figura 4.3: Comportamiento de una convolución clásica.

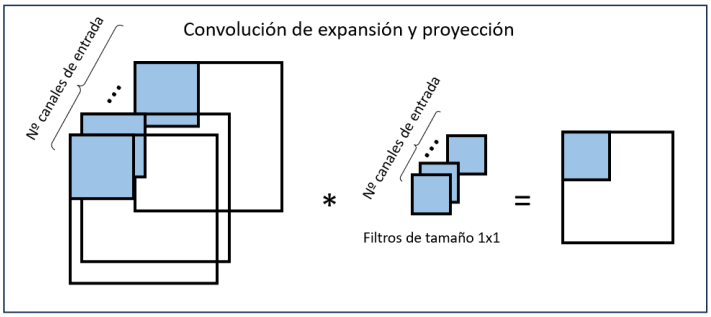


Figura 4.4: Comportamiento de una convolución de expansión y proyección.

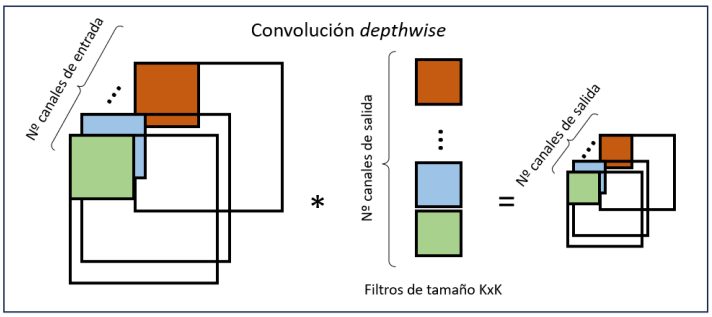


Figura 4.5: Comportamiento de una convolución *depthwise*.

Para esta sección se han seguido Howard et al. (2017) y Pérez Cerdeira (2021)

## 4.3. Construcción de la red utilizada

Para la construcción de la red que se va a utilizar se aplicará la técnica de *transfer learning* usando como base la red ya preentrenada *MobileNet V2*. En esta sección se explica cuáles de las capas de esta red han sido congeladas y qué capas se han añadido para obtener la red a utilizar.

Antes de eso, se introducen las herramientas *TensorFlow* y *Keras*, que son de gran interés, ya que serán las bibliotecas que se utilizarán en *Python* para cargar la red preentrenada.

### 4.3.1. TensorFlow

En primer lugar es necesario introducir *TensorFlow*, una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, que fue desarrollada por *Google* para satisfacer las necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos. Actualmente es utilizada tanto en la investigación como en los productos de *Google*. *TensorFlow* fue originalmente desarrollada por el equipo de *Google Brain* para uso interno en *Google* y posteriormente fue publicada bajo la licencia de código abierto *Apache* 2.0 en 2015.

Posee interesantes integraciones con otras bibliotecas del ecosistema como *Keras*, de la que se hablará más adelante. En este trabajo se ha utilizado para construir y entrenar la red neuronal utilizada. *TensorFlow* es una librería desarrollada a partir de la combinación de *C++* y *CUDA*. El hecho de emplear *Python* supone un ahorro en cuanto a la declaración del tipo de variables que se obtienen como resultado de las ejecuciones, simplicidad a la hora de utilizar dichas variables, el manejo de vectores y matrices, etc. La principal estructura de datos que se maneja en esta librería son los tensores. Con un tensor nos referimos a un conjunto de valores primitivos, por ejemplo, números enteros o flotantes, organizados por un *array* de 1 o $N$ dimensiones, donde el rango del tensor sería el número de dimensiones (Artola, 2019).

### 4.3.2. Keras

*Keras* es una API de redes neuronales a alto nivel, escrita en *Python* y que puede emplearse haciendo uso de *TensorFlow*, *CNTK* o *Theano*. Fue desarrollada con el enfoque de permitir una experimentación rápida, es decir, pasar de la idea al resultado con el menor retraso posible. Entre sus características se puede destacar:

- Permite una creación de prototipos fácil y rápida (a través de la facilidad de uso, la modularidad y la extensibilidad).
- Admite redes convolucionales y redes recurrentes, así como la combinación de ambas.
- Funciona tanto en CPU´s (unidades de procesamiento central, que se encargan de realizar cálculos, y ejecutar instrucciones de programas) como en GPU´s (unidades de procesamiento gráfico, se encargan de procesar gráficos y visualizaciones).

(Artola, 2019)

### 4.3.3. Construcción de la red utilizada

Se explica ahora cuáles han sido los pasos llevado a cabo en *Python* para construir la red neuronal utilizada.

1. En primer lugar, se carga el cuerpo principal de modelo, que sería *MobileNet V2*, mediante el comando *tf.keras.applications.MobileNetV2*, es decir, se carga el modelo preentrenado utilizando la biblioteca *TensorFlow* y *Keras*. En este comando debe especificarse que no se quiere la última capa completamente conectada de la parte superior de la red, las dimensiones de las imágenes de entrada ($224 \times 224$) y que para los pesos se va a utilizar una inicialización aleatoria:

   $$tf.keras.applications.MobileNetV2(input\_shape = (224, 224, 3),\ include\_top = False,\ weights = None)$$

2. A continuación se deben congelar los pesos de todas las capas, para evitar que se vayan actualizando continuamente durante el entrenamiento.

Se añaden ahora las posibles últimas capas, que serían:

- *Preprocesss_input*: que reescala los valores de entrada entre $[0, 1]$, ya que las matrices de entrada tienen 3 canales de color y valores en el rango $[0, 255]$.
  *tf.keras.applications.mobilenet_v2.preprocess_input*
- *Flatten*: una capa aplanadora que convierte los elementos de la matriz de imágenes de entrada en un array plano.
  *tf.keras.layers.Flatten()*
- *GlobalAveragePooling2D*: capa para la agrupación de los datos.
  *tf.keras.layers.GlobalAveragePooling2D()*
- *Dense*: para añadir capas ocultas a la red neuronal. Se añaden dos de estas capas con diferente número de inputs, 1280 y 512, pero con la misma función de activación, ReLU.
  *tf.keras.layers.Dense(*1280*, activation = 'relu')*
  *tf.keras.layers.Dense(*512*, activation = 'relu')*
- *Dropout*: esta capa establece aleatoriamente las unidades de entrada en 0 con una frecuencia, de 0.2 en este caso, en cada paso durante el tiempo de entrenamiento, lo que ayuda a evitar el sobreajuste. Las entradas que no se establecen en 0 se escalan en $\frac{1}{1-0.2}$ de modo que la suma de todas las entradas no cambia.
  *tf.keras.layers.Dropout(*0.2*)*

3. Por último, para realizar las predicciones de los documentos que estén mal clasificados se añade una capa de predicción, que es una cada *Dense* con 26 entradas y función de activación *Softmax*.
   *tf.keras.layers.Dense(*26*, activation = 'softmax')*

Además, se añade también un *Stop Automático* para que, si el modelo deja de mejorar, no se realicen todos los *epochs* o iteracciones, y se evite así el *overfitting*.

# Capítulo 5

# Desarrollo del ejemplo práctico

En este capítulo se muestra en detalle cuáles han sido los pasos que se han llevado a cabo para desarrollar el ejemplo de clasificación de documentos bancarios. Como se mencionaba en la sección 1.2, el ejemplo se va a realizar en 4 etapas: la creación del conjunto de datos, el depurado de los mismos, el modelado (con la red que se ha construido en la sección 4.3.3) y la evaluación de los resultados. Además, se explicará en la sección 5.4 el método que se ha llevado a cabo para clasificar los documentos completos, no las imágenes por separado como se venía haciendo hasta ese momento.

Se explicará ahora sobre los datos reales, cómo se han desarrollado estas fases.

## 5.1. Depuración de la base de datos

Una vez que se ha construido el *dataset* con el que se va a trabajar como se ha explicado en el capítulo 2, se procede a realizar una depuración de estos datos. El objetivo de esta depuración es eliminar los datos atípicos que pueda contener el *dataset*, ya que debido a que se cuenta con una gran cantidad de datos (84.111 imágenes) es preferible eliminar observaciones de más que trabajar con el ruido que proporcionan los datos atípicos.

En primer lugar, se trabajará con cada tipo de documentos por separado y se creará para cada uno un modelo de clasificación usando la red ya construida. Para cada tipo de documento se separará el conjunto de datos en conjunto de entrenamiento y conjunto de test.

- Conjunto de entrenamiento: sobre este conjunto de datos es sobre el que se entrena el modelo que se está utilizando.
- Conjunto de test: es el conjunto de datos sobre los que se pretende que el modelo haga las nuevas predicciones y sobre el que se evalúa el mismo.

Para el conjunto de entrenamiento se realizará un reetiquetado inicial de los datos de forma manual, que consiste en etiquetar uno a uno los documentos según pertenezcan a la clase con la que se trabaja o a la clase "Otro" que contiene todos los documentos de cualquiera de las demás clases.

Una vez se ha etiquetado el conjunto de entrenamiento correctamente se procede a realizar un procesado de los dos conjuntos para adecuarlos al formato necesario para el modelo de *transfer learning*.

En este procesado se cargan cada una de las imágenes de cada uno de los conjuntos y se le dan las dimensiones adecuadas (que serán $224 \times 224$ para todos los objetos, ya que son las dimensiones que requiere el modelo para trabajar), además, se va a trabajar con las imágenes con 3 canales de color. Por otro lado, se construirán cuatro listas con las etiquetas de las clases y las imágenes, que será con las que se trabajará posteriormente.

Una vez que se cuenta con las 4 listas (lista de imágenes de entrenamiento, lista de etiquetas de entrenamiento, lista de imágenes de test y lista de etiquetas de test) se procede a entrenar el modelo. Para ello se especifican los siguientes factores:

- *Epoch*: este hiperparámetro define la cantidad de veces que el modelo se ejecuta a través de conjunto de entrenamiento. Así el modelo se ejecuta hasta que el error o el área de mejora haya sido reconocido o suficientemente optimizado. En este caso, se establece el número de epoch en 15.

- *Batch size*: este hiperparámetro define el tamaño de los lotes que se toman de la muestra de entrenamiento para entrenar el modelo. Si este número es pequeño, significa que la red entrena rápido pero tiene poca memoria, entonces es posible que no aprenda las características o detalles más significativos para la predicción. Si este número es grande, es probable que si tenga en cuenta los casos más importantes, pero que el entrenamiento seas más lento. En este caso utilizaremos lo establecido por defecto, que es *batch size* $= 32$.

Una vez que el modelo ha sido entrenado se procede a hacer las predicciones para las nuevas observaciones. Estas predicciones se realizan sobre dos de las listas de test creadas anteriormente y por lotes (de tamaño *batch size*$= 32$), ya que si se realizaran directamente sobre el conjunto de test el coste computacional sería muy alto. Después de este proceso se obtienen las nuevas predicciones para las imágenes del conjunto de test, es decir, para cada una de las imágenes se precide si pertenece a la clase con la que se está trabajando o la clase "Otro" que engloba todas las demás clases .

En conclusión, después de realizar esta depuración en las imágenes, se obtiene un nuevo *dataset* con las imágenes reetiquetadas (tanto las que pertenecen al conjunto de entrenamiento como las que pertenecen al conjunto de test), que se usará para entrenar y evaluar el modelo de clasificación multiclase.

## 5.2. Modelo de clasificación

Una vez que se ha construído el nuevo *dataset* con las imágenes reetiquetadas se procede a entrenar ahora sí el modelo en el que interactúan todas las clases. Como en el *dataset* que se ha creado hay una nueva clase, que sería la clase "Otro", y no nos interesa que nuestro modelo lo tenga en cuenta, ya que son datos atípicos que solo generan ruido, se eliminan todas las imágenes con esa etiqueta. Se dispone así de un conjunto de 69.854 imágenes.

En primer lugar, se debe de dividir el conjunto de imágenes en conjunto de entrenamiento, conjunto de validación y conjunto de test.

- Conjunto de validación: son datos preetiquetados (al igual que el conjunto de entrenamiento) que sirven al modelo para evaluar si la fase de entrenamiento se ha llevado a cabo de manera eficiente

y para ajustar sobre estos datos algunos hiperparámetros. Este ajuste de hiperparámetros podría llevarse a cabo mediante remuestreo o validación cruzada, pero al disponer de una base de datos grande, está justificada la creación de este nuevo conjunto.

Así, durante el entrenamiento, el modelo entrenará solo en el conjunto de entrenamiento y validará evaluando los datos en el conjunto de validación. El modelo aprende las características de los datos en el conjunto de entrenamiento, toma lo que aprendió de estos datos y luego predice en el conjunto de validación. Durante cada época, se irán viendo los resultados de pérdida y precisión global tanto para el conjunto de entrenamiento como para el de validación. Esto ayuda a ver si el modelo se está sobreajustando o no, el sobreajuste ocurre cuando el modelo solo aprende los detalles de los datos de entrenamiento y no puede generalizar bien los datos en los que no se entrenó.

En primer lugar, se redimensionan las imágenes para conseguir que sean de $224 \times 224 \times 3$ y se construyen las listas necesarias, 6 en este caso, una lista de imágenes y otra de etiquetas para cada uno de los conjuntos de entrenamiento, validación y test.

Igual que en la etapa de depuración de datos, se ajusta el tamaño de los lotes de forma que: *batch size*= 32, pero en esta etapa, no se establece el número de etapas (*epochs*) necesarias, si no que se deja que el modelo haga todas las que sean necesarias, en este caso $epochs = 27$. Además, se establece un nuevo parámetro, $Callbacks = EarlyStopping$, gracias a esto se realiza un seguimiento de una métrica de rendimiento, como la pérdida o la precisión (en el caso de este ejemplo práctico la métrica escogida es la pérdida), en un conjunto de datos de validación durante el entrenamiento del modelo, si la métrica de rendimiento no mejora después de un cierto número de *epochs* consecutivas, se detiene el entrenamiento. En la etapa de entrenamiento también se debe tener en cuenta el optimizador o *optimizer*, que determina el algoritmo utilizado para optimizar los pesos del modelo durante esta fase. En este caso se utiliza el optimizador Adam (*Adaptive Moment Estimation*), utilizar este optimizador significa que durante el entrenamiento, los ajustes de los pesos se realizarán utilizando el algoritmo Adam, que adapta la tasa de aprendizaje de manera eficiente y utiliza estimaciones adaptativas de primer y segundo orden del gradiente de la función de pérdida. Esto ayuda al modelo a converger más rápidamente y a obtener un mejor rendimiento.

Durante la etapa de validación, como se ha dicho antes, se ajustan distintos hiperparámetros, que serían:.

- Tasa de aprendizaje o *learning rate*: es un parámetro que determina qué tan rápido se actualizan los pesos del modelo en cada iteración. Aunque no hay un rango sobre el que tenga que oscilar esta tasa, lo habitual es que oscile entre un 0.1 y un 0.0001. Si la tasa de aprendizaje es demasiado alta, es probable que el modelo no converja o que oscile alrededor de un mínimo local sin llegar a una solución óptima. Por otro lado, si la tasa de aprendizaje es demasiado baja, el modelo puede converger lentamente o quedar atrapado en mínimos locales subóptimos. Para llevar a cabo este modelo se ha probado con las siguientes tasas de aprendizaje: 0.1, 0.01, 0.001 y 0.0001. En este modelo, la tasa de aprendizaje es de $1e-04$, este valor representa una tasa de aprendizaje pequeña. En general, esto puede llevar a convergencia más lenta pero a una mejor precisión en la clasificación.

- Función de pérdida o *loss*: especifica la función de pérdida que se utiliza para evaluar el rendimiento del modelo durante el entrenamiento. En este caso, se ha probado con dos tipos de función de pérdida. En primer lugar, la entropía cruzada categórica o *categorical cross entropy*. La entropía cruzada categórica mide la discrepancia entre la distribución de probabilidades predicha por el modelo y la distribución real de las etiquetas de clase en el conjunto de datos. Cuanto mayor sea la discrepancia entre estas distribuciones, mayor será el valor de pérdida. El objetivo del modelo es minimizar esta pérdida durante el entrenamiento para que las probabilidades predichas se acerquen a las distribuciones reales. Esta medida penaliza al modelo por cada vez que predice una probabilidad baja para la clase correcta y recompensa al modelo cuando las probabilidades predichas se acercan a las etiquetas reales. En segundo lugar, la entropía cruzada binaria o *binary cross entropy*, que se utiliza en problemas de clasificación binaria pero puede adaptarse para su uso en problemas de clasificación multiclase mediante la técnica de *one-hot encoding* en la codificación de las etiquetas. Cada etiqueta se representa como un vector binario de longitud igual al número de clases. El vector tiene un valor de 1 en el índice correspondiente a la clase a la que pertenece la imagen y 0 en todos los demás índices. Esta entropía mide la discrepancia entre la distribución de probabilidad predicha por el modelo y la distribución de probabilidad real de las etiquetas, se calcula para cada clase y luego se suma o se promedia para obtener la pérdida total.

Después de probar con estas 8 combinaciones de parámetros, se decide que el mejor modelo con los datos de validación es el que utiliza:

$$\textit{Learning rate} = 0.0001 \text{ y } \textit{Loss} = \textit{categorical cross entropy}$$

Posteriormente, se realiza la evaluación del modelo.

## 5.3. Evaluación de los resultados

Para lleva a cabo la evaluación del modelo primero se presentan dos gráficos que representan la evolución de la función de pérdida y la evolución de la precisión global en las etapas de entrenamiento y validación.

En la Figura 5.1 se tiene en primer lugar la función que representa la evolución de la función de pérdida en las etapas de entrenamiento y validación. Al hablar de la función de pérdida, en ambos casos conforme avanzan las iteracciones las funciones convergen a un mínimo. Si se estuviera produciendo *overfitting* se vería como la función de la etapa de entrenamiento seguiría reduciendo sus pérdidas mientras que en la etapa de validación las pérdidas comenzarían a incrementarse.

A continuación, se representa en la Figura 5.2 la evolución de la precisión global en las etapas de entrenamiento y validación. Se observa que desde que se comienza el entrenamiento hasta que se produce el *earlystopping* no se produce ninguna mejora sustancial. Esto se debe a que al trabajar con modelos preentrenados ya producen buenos resultados desde las primeras iteraciones. Se puede ver como el valor predictivo positivo llega a superar el 80 % en el conjunto de validación. Esto se puede considerar un buen resultado, teniendo en cuenta la dificultad de la tarea que se está a realizar.

En la Figura 5.3 se presenta la matriz de confusión del modelo. Como ya se ha visto, los números de la diagonal de la matriz se corresponden con las imágenes correctamente clasificadas y se puede ver que

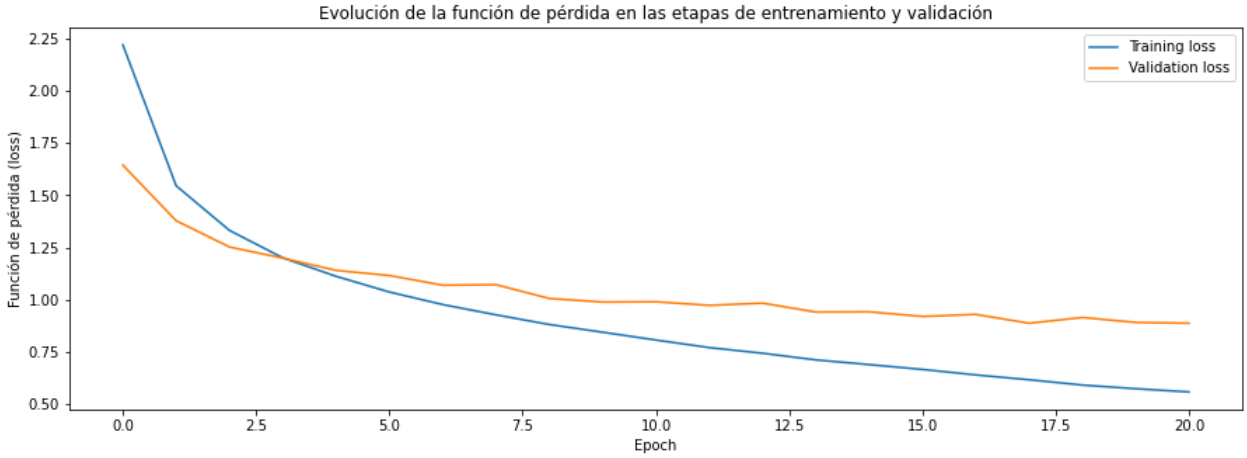


Figura 5.1: Gráfica de la evolución de la función de pérdida.

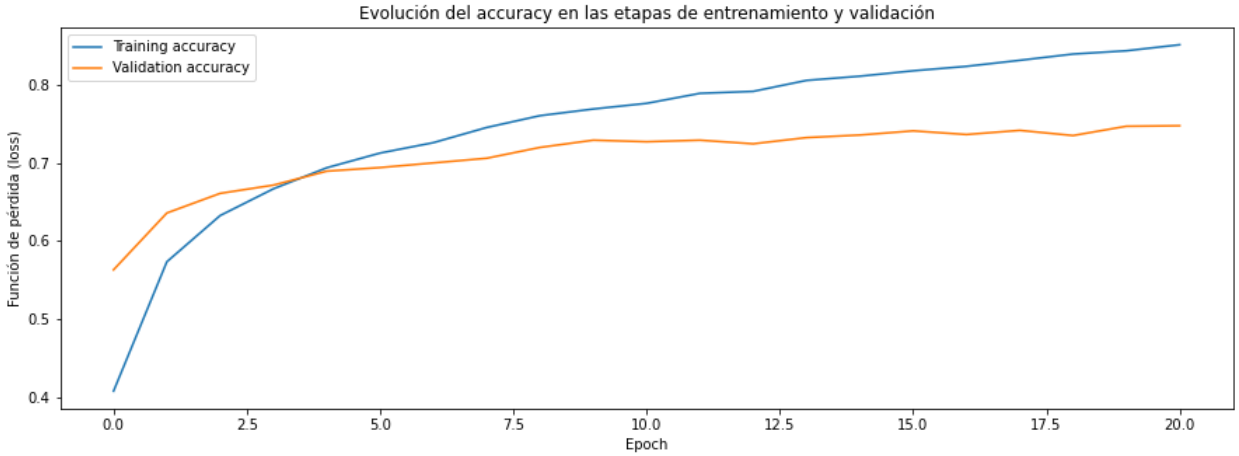


Figura 5.2: Gráfica de la evolución de la precisión global.

para cada una de las clases la mayoría de las imágenes se han clasificado correctamente. Los números que están fuera de la diagonal y que se corresponden con los elementos mal clasificados son muy bajos, salvo en algunos casos. El modelo tiene algunos problemas al intentar clasificar las imágenes de las clases "9539" (testamentos) y "15005" (escrituras), ya que los confunde en múltiples ocasiones, también cuando intenta clasificar las imágenes del tipo "9052" (prestaciones cese de actividad) confunde en gran cantidad de ocasiones los de la clase "9057" (cuantía mensual de desempleo) con esos, en algún otro caso llega a pasar lo mismo pero en menor medida.

Si se analizan ahora el resto de las métricas, se cuenta con una precisión global de 0.74, es decir, el 74 % de las imágenes del conjunto de test han sido clasificadas correctamente en relación con sus etiquetas verdaderas, por lo tanto es una métrica bastante buena, pero como las clases son desbalanceadas esta medida en solitario puede resultar engañosa.

En la Tabla 5.1 que aparece a continuación, se pueden ver el resto de métricas para cada una de las

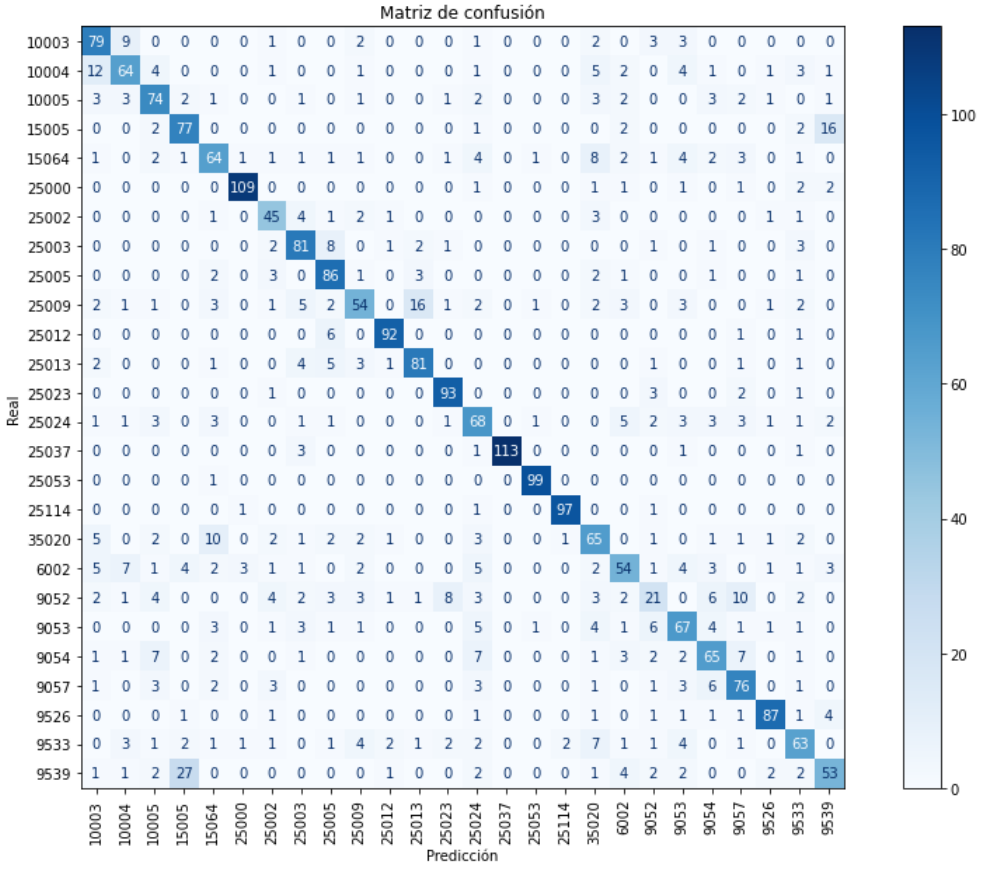


Figura 5.3: Matriz de confusión.

clases, que son relativamente elevadas. Si se habla del valor predictivo positivo, ronda aproximadamente el valor 0.7 en todas las clases, esto significa que, para cada clase $i$ aproximadamente el 70 % de las predicciones clasificadas como clase $i$ son correctas. En algunas clases este valor asciende hasta el 1, en el caso de la clase "25037", y en otras desciende al 0.41, en el caso de la clase "9052".

La sensibilidad toma valores levados en casi todos los casos, ronda también el 0.7 en muchos casos, esto quiere decir, que de todas las imágenes que pertenecen a la clase $i$, el modelo ha conseguido clasificar correctamente el 70 % de ellas. En algunos casos este valor asciende al 0.99, en el caso de la clase "25052" o desciende hasta el 0.26, en el caso de la clase "9052".

El *F1 score* ronda en casi todos los casos el valor 0.73, lo cual es una buena métrica e indica que el modelo tiene un rendimiento razonablemente bueno en términos de equilibrio entre el valor predictivo positivo y sensibilidad para esas clases en particular. En algunos llega incluso a superar el 0.9. En el caso de la clase "9052" (prestaciones cese de actividad), el valor de esta métrica es muy bajo, un 32 %, ya se había visto en la matriz de confusión que el modelo tenía problemas al intentar clasificar esta clase.

## 5.4. Clasificación de los documentos

Una vez que las imágenes han sido clasificadas si se quiere saber cuál es la clase a la que pertenece el documento completo, lo que se hace es escoger la clase que ha sido predicha para el mayor número de imágenes.

Para explicar esta clasificación se pone un ejemplo de clasificación de un documento. Se toma un documento de la clase "6002", un contrato de alquiler, se hace la limpieza necesaria y se le aplica el modelo de clasificación a cada una de las imágenes de las páginas. Una vez hecho esto la salida que proporciona el modelo aparece en la Tabla 5.2 que se muestra a continuación. Se puede ver las predicciones para cada una de las páginas y la probabilidad con la que se calculan las mismas. A partir de estos resultados se calcula cuántas predicciones de cada clase ha hecho el modelo, en este caso:

- 2 imágenes predichas como "9052"
- 15 imágenes predichas como "6002"
- 1 imágenes predichas como "9054"
- 2 imágenes predichas como "35020"
- 1 imágenes predichas como "25009"

En consecuencia, se elegiría como clase correcta la "6002" y se habría predicho correctamente.

| **Clase** | **Valor predictivo positivo** | **Sensibilidad** | ***F1 score*** |
|---|---|---|---|
| 10003 | 0.73 | 0.73 | 0.73 |
| 10004 | 0.61 | 0.66 | 0.63 |
| 10005 | 0.74 | 0.72 | 0.73 |
| 15005 | 0.81 | 0.59 | 0.68 |
| 15064 | 0.64 | 0.63 | 0.64 |
| 25000 | 0.94 | 0.93 | 0.93 |
| 25002 | 0.58 | 0.75 | 0.65 |
| 25003 | 0.75 | 0.86 | 0.8 |
| 25005 | 0.70 | 0.83 | 0.76 |
| 25009 | 0.69 | 0.53 | 0.6 |
| 25012 | 0.96 | 0.9 | 0.93 |
| 25013 | 0.79 | 0.78 | 0.78 |
| 25023 | 0.86 | 0.91 | 0.88 |
| 25024 | 0.62 | 0.64 | 0.63 |
| 25037 | 1 | 0.95 | 0.97 |
| 25053 | 0.91 | 0.99 | 0.95 |
| 25114 | 0.96 | 0.97 | 0.97 |
| 35020 | 0.65 | 0.64 | 0.65 |
| 6002 | 0.52 | 0.6 | 0.56 |
| 9052 | 0.41 | 0.26 | 0.32 |
| 9053 | 0.66 | 0.69 | 0.67 |
| 9054 | 0.69 | 0.6 | 0.65 |
| 9057 | 0.67 | 0.74 | 0.70 |
| 9526 | 0.87 | 0.89 | 0.88 |
| 9533 | 0.76 | 0.58 | 0.66 |
| 9539 | 0.56 | 0.69 | 0.62 |

Tabla 5.1: Tabla con las métricas del modelo.

| **Clase** | **Predicción** | **Probabilidades** |
|---|---|---|
| 6002 | 9052 | 0.429707 |
| 6002 | 6002 | 0.926273 |
| 6002 | 6002 | 0.381075 |
| 6002 | 6002 | 0.483950 |
| 6002 | 6002 | 0.446803 |
| 6002 | 9052 | 0.398037 |
| 6002 | 6002 | 0.504967 |
| 6002 | 6002 | 0.666975 |
| 6002 | 9054 | 0.402017 |
| 6002 | 6002 | 0.937877 |
| 6002 | 6002 | 0.502345 |
| 6002 | 6002 | 0.799422 |
| 6002 | 35020 | 0.336476 |
| 6002 | 6002 | 0.636579 |
| 6002 | 6002 | 0.868084 |
| 6002 | 6002 | 0.806477 |
| 6002 | 35020 | 0.825193 |
| 6002 | 6002 | 0.541988 |
| 6002 | 6002 | 0.810797 |
| 6002 | 6002 | 0.931084 |
| 6002 | 25009 | 0.218087 |

Tabla 5.2: Predicciones para un documento de la clase “6002”.

# Capítulo 6

# Conclusiones

A lo largo de este trabajo, se ha llevado a cabo una pequeña revisión bibliográfica, realizada en el capítulo 3, que ha permitido familiarizarse con los conceptos teóricos sobre los que se basa la técnica de *Computer Vision* y una explicación, en el capítulo 4, sobre cómo se ha construido la red neuronal con la que se construye el modelo utilizado para resolver el problema de clasificación de documentos.

En la sección 5.3 se han detallado cuales han sido los resultados obtenidos al evaluar el modelo sobre el conjunto de test de la base de datos con la que se contaba, y en este último capítulo del trabajo se explicarán las conclusiones que han sido extraídas después de ejecutar el modelo. En la sección 6.1 se hará un análisis del comportamiento del modelo para algunos de los documentos y se explicará por qué dependiendo del formato del documento, el modelo funciona mejor en unos casos que en otros, y en la sección 6.2 se hará una breve introducción a cuáles podrían ser las líneas de trabajo que se adoptarían en un futuro.

## 6.1. Análisis del modelo

Como ya se ha visto anteriormente, los resultados del modelo con respecto a las métricas de evaluación empleadas han sido bastante satisfactorios. En esta sección se analizarán los posibles motivos por los que qué unos documentos se clasifican mejor que otros.

Observando la matriz de confusión se aprecia que hay ciertos documentos que el modelo confunde, algunos de estos ejemplos serían: los testamentos y las escrituras, el informe de tasadora y el certificado de tasación, los documentos de bienes inmuebles y los extractos... Como se ha visto, el modelo de *Computer Vision* se encarga de clasificar las imágenes en función del mapa de píxeles que forman, por ello, algunos de los motivos por los que puede confundir estos tipos de documentos serían:

1. Similitud visual o falta de características distintivas: Si dos documentos tienen una apariencia visual muy similar, es posible que el modelo de *Computer Vision* tenga dificultades para diferenciarlos. Esto puede ocurrir cuando los documentos comparten características visuales comunes, como diseños, estructuras, disposición de texto o elementos gráficos similares.

2. Ambigüedad en la interpretación: Algunos documentos pueden contener información o contenido que puede ser interpretado de diferentes maneras. Esto puede generar ambigüedad en la clasifica-

ción, ya que el modelo puede tomar decisiones basadas en su interpretación de los datos visuales, lo que lleva a confusiones entre documentos con estructuras similares.

3. Limitaciones del conjunto de datos de entrenamiento: Si el modelo de *Computer Vision* no ha sido entrenado con una variedad suficiente de documentos con estructuras similares, es posible que no haya aprendido las diferencias sutiles que existen entre ellos. Como resultado, el modelo puede tener dificultades para clasificar con precisión documentos que comparten estructuras similares.

4. Errores en la extracción de características: En algunos casos, el modelo puede tener dificultades para extraer características relevantes de los documentos durante el proceso de análisis de imágenes. Esto puede deberse a factores como baja resolución de la imagen, mala iluminación, distorsiones o ruido en la imagen, lo que puede afectar negativamente la capacidad del modelo para distinguir documentos con estructuras similares.

El análisis de esta clasificación permite ver que efectivamente el primero de los motivos influye en cómo se han clasificado las imágenes. Los tipos de documentos que, como se ha visto antes, peor se han clasificado comparten características o estructuras comunes. Por ejemplo, en el caso de los testamentos y las escrituras ambos son documentos formados plenamente por texto, por lo tanto, no tienen una estructura en el mapa de píxeles que sea claramente diferenciable la una de la otra.

## 6.2. Líneas de trabajo futuras

El análisis de este trabajo sugiere que mejorar la precisión de clasificación de documentos bancarios puede requerir enfoques y técnicas adicionales. En este sentido, existen diversas recomendaciones y posibles líneas de investigación futura que podrían contribuir significativamente a la mejora de los resultados, algunas de estas recomendaciones son:

1. Ampliación del conjunto de datos: Recopilar y agregar más ejemplos de documentos al conjunto de datos de entrenamiento podría ayudar a mejorar el rendimiento del modelo. Cuanto más variada y representativa sea la muestra de entrenamiento, mejor será la capacidad del modelo para reconocer patrones y características específicas. Además, sería beneficioso considerar diferentes contextos y escenarios para abordar una mayor diversidad de documentos.

2. Introducir información adicional: En este ejemplo solo se han estado considerando las imágenes para llevar a cabo el modelo de clasificación, pero podría añadirse información adicional como la fecha en la que se añadieron los documentos a la base de datos o para qué tipo de transacción se han añadido esos documentos y cómo se hizo.

3. Utilización de técnicas de procesamiento de lenguaje natural (NLP): Dado que muchos de los documentos a menudo contienen terminología y lenguaje especializado, emplear técnicas de procesamiento de lenguaje natural puede resultar especialmente útil para capturar y comprender mejor los detalles y la estructura de dichos documentos. Al incorporar métodos de NLP, se podrían identificar de manera más precisa y eficiente elementos cruciales, como nombres propios, fechas, cláusulas específicas ...

4. Consideración de características multimodales: Los documentos bancarios contienen tanto texto como imágenes, firmas, sellos ... Al combinar información textual con características visuales, como el reconocimiento óptico de caracteres (OCR) y la detección de objetos, se podría lograr un análisis más completo y exhaustivo de los documentos. La integración de enfoques multimodales permitiría aprovechar la información tanto textual como visual, mejorando así la precisión de la clasificación.

5. Enfoque en la interpretabilidad y la transparencia del modelo: A medida que los modelos de clasificación de documentos se vuelven más sofisticados, es fundamental comprender cómo toman sus decisiones. Algunas técnicas comunes son la atención visual, que muestra las regiones específicas de una imagen que el modelo considera más relevantes para su clasificación o la interpretación de características, que consiste en identificar qué partes del texto son más influyentes en la clasificación realizada por el modelo. Puede revelar las palabras clave, frases o elementos particulares que son determinantes en el proceso de toma de decisiones del modelo.

# Bibliografía


Andrade Carrera, H. E. (2021). Desarrollo de un sistema de control de acceso en base a detección de temperatura corporal y al correcto uso de mascarillas en tiempo real [Trabajo de fin de grado, Escuela Politécnica Nacional de Quito]. http://bibdigital.epn.edu.ec/handle/15000/21847

Ariza- López, F. J., Rodríguez-Avi, J. & Alba-Fernández, V. (2018). Control estricto de matrices de confusión por medio de distribuciones multinomiales. Geofocus: Revista Internacional de Ciencia y Tecnología de la Información Geográfica, (21), 215-226. http://dx.doi.org/10.21138/GF.591

Artola Moreno, Á. (2019). Clasificación de imágenes usando redes neuronales convolucionales en Python [Trabajo de Fin de Grado, Universidad de Sevilla]. https://hdl.handle.net/11441/89506

Borrella Petisco, B. (2022). Introducción a la visión artificial: procesos y aplicaciones. [Trabajo de Fin de Grado, Universidad Complutense de Madrid]. https://eprints.ucm.es/id/eprint/74914/1/beatriz_borrella_introduction.pdf

Ceballo Iñigo, A. (2019). Usando técnicas de visión por computador para la validación de firmas manuscritas [Trabajo de Fin de Máster, Universitat Politécnica de Catalunya]. http://hdl.handle.net/2117/168826

Chanampe, H., Aciar, S., Vega, M. D. L., Molinari Sotomayor, J. L., Carrascosa, G., & Lorefice, A. (2019). Modelo de redes neuronales convolucionales profundas para la clasificación de lesiones en ecografías mamarias. XXI Workshop de Investigadores en Ciencias de la Computación (WICC 2019, Universidad Nacional de San Juan).

Chollet, F., Kalinowski, T. & Allaire, J.J. (2022). Deep learning with R. (2ªed). Manning.

Fernández Casal, R., Costa Bouzas, J. & Oviedo de la Fuente, M. (2021). Aprendizaje estadístico. https://rubenfcasal.github.io/aprendizaje_estadistico

García de Figuerola Paniagua, L. C. (2013). Clasificación automática de documentos. Un caso práctico. http://hdl.handle.net/10366/133305

García Noya, J.D. (2022) Desarrollo de un Priorizador para Campañas Comerciales Empleando Modelos Predictivos. [Trabajo de Fin de Máster, Universidad de Santiago de Compostela] http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_1999.pdf

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press. http://imlab.postech.ac.kr/dkim/class/csed514_2019s/DeepLearningBook.pdf

Hollemans, M. (22 de abril de 2018). MobileNet version 2. Machine, Think! https://machinethink.net/blog/mobilenet-v2/

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. https://doi.org/10.48550/arXiv.1704.04861

Huang, T.S. (1996). Computer Vision: Evolution And Promise. 1996 CERN School of Computing, 21-25. http://cds.cern.ch/record/400313

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2021). An Introduction to Statistical Learning with Applications in R. (2ªed). Springer Texts in Statistics. https://doi.org/10.1007/978-1-0716-1418-1_1

Pérez Cerdeira, I. J. (2021). Aceleración hardware para inferencia en redes neuronales convolucionales. [Tesis para optar al grado de Magíster en Ciencias de la Ingeniería con mención en Ingeniería Eléctrica, Universidad de Concepción]. http://repositorio.udec.cl/jspui/handle/11594/6204

Salinas, J., & Izetta, J. (2016). Clasificación automática de textos periodísticos usando Random Forest. XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016). http://sedici.unlp.edu.ar/handle/10915/55733

Sebastiani, F. (2002). Machine learning in automated text categorization. ACM computing surveys (CSUR), 34(1), 1-47. https://doi.org/10.1145/505282.505283

Sepulveda Valdivia, E. (2019). Redes neuronales convolucionales, reconocimiento de imágenes y pronósticos financieros [Trabajo de Fin de Máster, Universitat Politécnica de Catalunya]. http://hdl.handle.net/2117/172426

Valenzuela Cámara, S. H. (2022). Detección y clasificación de enfermedades en el tomate mediante Deep Learning y Computer Vision [Doctoral dissertation, Universidad Nacional de Mar del Plata]. https://doi.org/10.35537/10915/139770