Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Clasificación automática de señales de tráfico basada en Deep Learning.

Autor: Lara Sánchez Vidal

Tutor: José Ramón Cerquides Bueno

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Clasificación automática de señales de tráfico basada en Deep Learning.

Autor:
Lara Sánchez Vidal

Tutor:
José Ramón Cerquides Bueno
Profesor titular

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo de Fin de Grado: Clasificación automática de señales de tráfico basada en Deep Learning.

Autor: Lara Sánchez Vidal

Tutor: José Ramón Cerquides Bueno

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis amigos

A mis maestros

Agradecimientos

A mis padres y a mi hermana, por apoyarme en todas mis decisiones, estar en los momentos más difíciles y por confiar siempre en mí.

A mis compañeros y amigos que me llevo para siempre, por hacer este camino más fácil, divertido y especial, sin vosotros no hubiera sido lo mismo.

A mis profesores, por todo lo que me han enseñado a lo largo de estos años, en especial, a mi tutor, Ramón Cerquides, por toda la dedicación y ayuda en el proyecto desde el primer momento.

A Isaac, por ayudarme siempre que lo necesito y su apoyo incondicional.

Gracias a todos, sin vosotros todo esto no hubiera sido posible.

Lara Sánchez Vidal

Sevilla, 2021

Resumen

Las tecnologías que conforman la Inteligencia Artificial, como el Deep Learning han tomado mucha relevancia en diversos sectores a lo largo de los últimos años. Uno de los sectores más influenciados por estas tecnologías ha sido el de la conducción autónoma.

Este Trabajo de Fin de Grado se encuentra en el ámbito del Deep Learning y la conducción autónoma, y tratará de resolver uno de los aspectos más importantes cuando se habla de un coche autónomo, es decir, la clasificación de señales de tráfico. El objetivo del proyecto será la creación de diferentes modelos basados en el Aprendizaje Profundo, en concreto, en redes neuronales convolucionales. Para llevar a cabo el modelo se utilizará el lenguaje de programación Python y la base de datos GTSRB, que es una base de datos de 43 tipos diferentes de señales de tráfico alemanas. Después de crear las estructuras de los modelos, se entrenarán y compararán los resultados obtenidos.

Una vez analizados y comprados los resultados, se consigue una precisión superior al 90% en los diferentes experimentos realizados para la clasificación de las señales de tráfico.

Abstract

The technologies that make up Artificial Intelligence, such as Deep Learning, have become very important in a lot of sectors over the last few years. One of the most influenced by these technologies has been autonomous driving.

This Final Degree Work is in the field of Deep Learning and autonomous driving and will try to solve one of the most important things when talking about an autonomous car, that is, the traffic sign classification. The aim of this project will be the creation of different models based on Deep Learning, specifically on convolutional neural networks. The Python programming language and the GTSRB database which is a database of 43 different types of German traffic signs, will be used to create the model. When the model structures have been created, the results achieved will be trained and compared.

Once the results have been analysed and compared, an accuracy of more than 90% is achieved in the different experiments for the traffic sign classification.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS

1 INTRODUCCIÓN

La inteligencia es la capacidad de adaptarse
a los cambios

- Stephen Hawking -

1.1. Motivación

La Inteligencia Artificial está cambiando nuestras vidas. Fue en el año 1956 cuando el informático John McCarthy acuñó el término de Inteligencia artificial, también conocida por sus siglas, IA. Sin embargo, no ha sido hasta los últimos años cuando el campo de Inteligencia Artificial ha sufrido un despliegue vertiginoso, tanto que muchos la consideran la cuarta revolución industrial. Este avance se ha producido principalmente gracias a la mejora de la potencia computacional, así como al aumento y accesibilidad de datos.

El área de la Inteligencia Artificial es un campo muy extenso que resulta difícil definir, ya que depende en cierto modo de la definición de la inteligencia humana, la cual sigue teniendo diversas interpretaciones a día de hoy. Una definición según Jordi Torres, sería: "El esfuerzo de automatizar tareas intelectuales normalmente realizadas por humanos" [1] .

Se pueden distinguir dos tipos de Inteligencia Artificial, la débil y la fuerte. La IA débil hace referencia a sistemas que sólo pueden llevar a cabo un número acotado de tareas, de forma que, si a ese sistema se le asignara la realización de otra tarea diferente, seguramente no se obtendrían los resultados esperados. Por otro lado, la inteligencia artificial fuerte se refiere a sistemas capaces de realizar muchas tareas diferentes. Los sistemas existentes hasta día de hoy pertenecen al primer tipo de IA, la débil, mientras que la segunda por ahora solo existe en la ficción. [2]

La Inteligencia Artificial está presente en diversos sectores. Por poner algunos ejemplos, en el sector financiero, los bancos usan esta tecnología para poder detectar fraudes o pronosticar modelos del mercado. En el sector comercial ayuda a los comercios a pronosticar las ventas, así como a recomendar el producto apropiado a un determinado cliente. Por su parte, el sector sanitario ha creado chatbots capaces de realizar un diagnóstico a partir de preguntas sobre los síntomas, mientras que en el sector automovilístico se han generado sistemas capaces de evitar colisiones y optimizar el tráfico.[3]

Dentro del mundo de la Inteligencia Artificial se pueden encontrar diferentes campos de estudio como son el procesamiento del lenguaje natural (en inglés, Natural Language Process o NLP), la visión por computadora o el aprendizaje automático.

Este trabajo se centrará en una disciplina concreta de la Inteligencia Artificial, el aprendizaje automático, y dentro de éste, en el aprendizaje profundo o Deep Learning.

1.2. Objetivo

El principal objetivo de este proyecto será el desarrollo de un sistema de ayuda a la conducción, en concreto, un sistema de clasificación de señales de tráfico. Es decir, se tratará de un sistema capaz de clasificar, a partir de una imagen de una señal de tráfico, la clase a la que pertenece dicha señal. Esta clasificación se realizará a partir de técnicas de Deep Learning, en concreto, la creación de una red neuronal convolucional.

Para poder desarrollar dicho sistema, será necesario primero adquirir una serie de conceptos básicos de redes neuronales convolucionales y del lenguaje de programación a usar, Python, así como de las librerías necesarias para la creación de la red.

Una vez alcanzados estos conocimientos previos, se puede proceder a la creación del sistema y su futura evaluación con el fin de alcanzar el modelo óptimo.

1.3. Estructura de la memoria

Este documento se encuentra estructurado en diferentes capítulos, los cuales se explican de forma breve a continuación:

- **Capítulo 1: Introducción.** En este primer capítulo se expone la motivación del proyecto, los objetivos que se quieren alcanzar y la estructura del documento.
- **Capítulo 2: Estado del arte.** Este capítulo estudia la situación actual del área de desarrollo del proyecto, la conducción autónoma.
- **Capítulo 3: Deep Learning**. En este capítulo se estudian los conceptos teóricos sobre Deep Learning necesarios para llevar a cabo el desarrollo del proyecto.
- **Capítulo 4: Metodología propuesta.** En este punto, una vez conocida la teoría necesaria sobre Deep Learning y redes neuronales, se lleva a cabo la creación del modelo. Para ello, se explican previamente las herramientas necesarias para el desarrollo del modelo, así como la base de datos con la que se trabajará, en concreto, GTSRB (The German Traffic Sign Recognition Benchmark).
- **Capítulo 5: Experimentos y resultados.** Una vez creado el modelo se realizarán diversos experimentos y se evaluarán y compararán entre sí, mostrando los resultados obtenidos.
- **Capítulo 6: Conclusiones y líneas futuras.** En este último capítulo se exponen las conclusiones que se han alcanzado tras la realización del trabajo expuesto, así como algunas propuestas de mejora para el futuro.

2 ESTADO DEL ARTE

Las 'leyes del pensamiento' no solo dependen de las propiedades de las células cerebrales, sino del modo en que están conectadas.

- Marvin Minsky -

En este capítulo, se abordará la definición y explicación en profundidad de la conducción autónoma, así como de todos los niveles existentes según la autonomía que presente el vehículo. Adicionalmente, se expondrán los diferentes sistemas de ayuda a la conducción que existen en la actualidad. Uno de estos sistemas es la clasificación de señales de tráfico, la cual se estudiará más detenidamente en capítulos posteriores.

2.1. Conducción Autónoma

Un sector muy impactado por la IA es el sector automovilístico, cuyo objetivo es la creación de coches capaces de conducir de forma autónoma, comúnmente conocidos como coches autónomos. La finalidad de este tipo de coches sería que llegasen a reemplazar a los coches actuales. Desde la aparición del primer coche, cuyo propósito era la posibilidad de desplazarse sin necesidad de usar el transporte público (ya fuera un desplazamiento corto hasta largos viajes entre ciudades), hasta el día de hoy, estos no han parado de evolucionar y por ello la industria del automóvil ha seguido creciendo hasta convertirse en una de las industrias más importantes. Sin embargo, una consecuencia directa que este crecimiento conlleva es el aumento de accidentes de tráfico, lo cual constituye un gran problema a nivel mundial.

Según la OMS (Organización Mundial de la Salud), en las carreteras del mundo mueren cada año 1,35 millones de personas, aunque los accidentes de tráfico también pueden dar lugar a traumatismos no mortales, los cuales padecen entre 20 y 50 millones de personas [4]. Un ejemplo de este problema se muestra en la Figura 1 [5], en la que se contabilizan el número de víctimas en accidentes de tráfico en España, dividido en víctimas mortales, heridos hospitalizados y heridos no hospitalizados desde el año 1960 hasta el 2019. Como se puede observar, aunque el número de víctimas mortales es bastante elevado, es mucho mayor la cantidad de personas que sufren accidentes no mortales.

Asimismo, según datos proporcionados por la DGT (Dirección General de Tráfico), el 90% de los accidentes de tráfico se deben al factor humano, como pueden ser las distracciones o el incumplimiento de las señales de tráfico entre otros [6]. Surge por tanto la necesidad del desarrollo de los coches autónomos, los cuales no necesitarían conductor ya que tienen la capacidad de observar y analizar el medio que les rodea y aplicar las técnicas de conducción adecuadas a dicha situación. Entre las técnicas que deben presentar dichos coches se encuentran la detección de las líneas de los carriles o pasos de cebra, el reconocimiento de peatones y señales de tráfico, etc. Debido a todo esto, son muchas las empresas que hoy en día trabajan en la investigación y el desarrollo de diferentes prototipos de vehículos autónomos como pueden ser Tesla, Ford, Toyota, Mercedes o Audi, y aunque no se ha conseguido aún la creación de un coche completamente autónomo, cada vez se está más cerca.

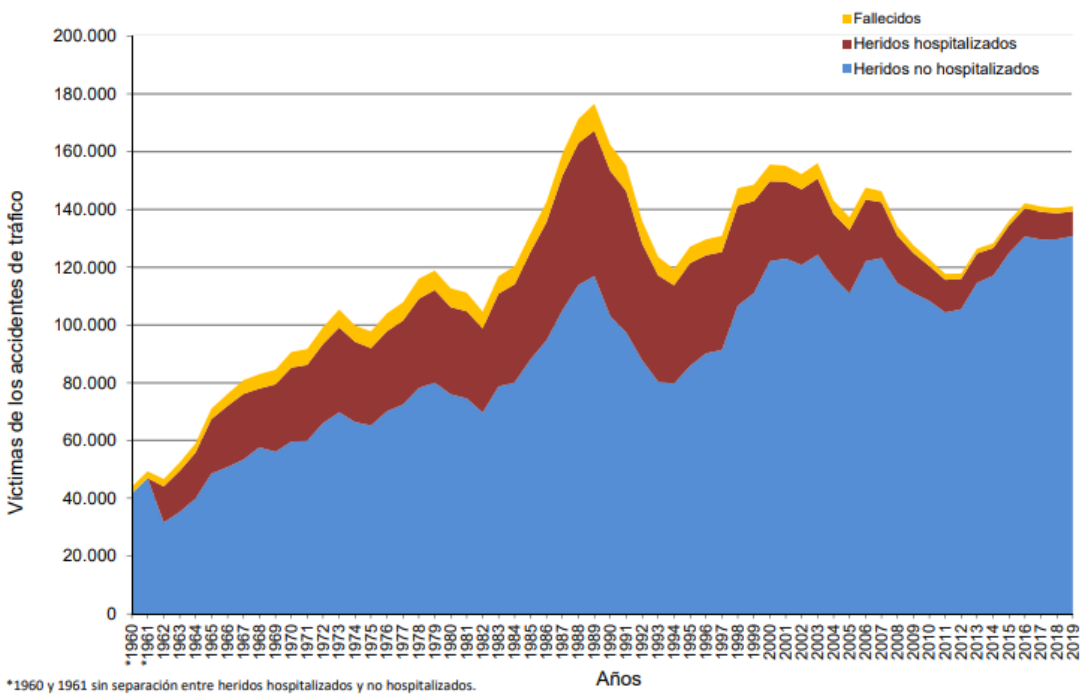


Figura 1. Víctimas accidentes de tráfico desde 1960-2019.

2.2. Niveles de conducción autónoma

Un vehículo autónomo es aquel que cuenta con una serie de sensores, actuadores, procesadores y software que le permiten conducir de manera autónoma. El objetivo final es que en el futuro se consiga una autonomía total del vehículo, sin embargo, su desarrollo se está produciendo de manera gradual. El SAE (Society of Automotive Engineers o Sociedad de Ingenieros de Automoción) realizó una clasificación de los diferentes niveles de automatización de la conducción, la cual consiste en una escala que consta de 6 niveles diferentes que van desde el nivel 0 al nivel 5. Dicha clasificación aparece descrita en el estándar SAEJ3016 ([7]), el cual fue publicado en enero de 2014 y revisado por última vez en abril de 2021, junto a otras consideraciones y definiciones. Sin embargo, aunque dicho estándar SAEJ3016 pretende ser generalista y universal, no es obligatorio su cumplimiento ya que será cada país el que establezca unas leyes determinadas al respecto [8]. Los 6 niveles comentados (Figura 2) se exponen a continuación [9]:

- **Nivel 0:** los coches de nivel de autonomía 0, también conocidos como coches no autónomos son aquellos en los que independientemente del vehículo, será el propio conductor el que llevará a cabo todas las acciones.
- **Nivel 1:** son aquellos en los que, aunque el conductor tiene el control casi total del vehículo, cuenta con ciertos sistemas de ayuda que permiten que la conducción sea más segura. Algunos de estos sistemas pueden ser, por ejemplo, el control de velocidad de crucero o la alerta de cambio involuntario de carril.
- **Nivel 2:** la gran diferencia entre este nivel y el anterior se encuentra en que el nivel 2 incorpora un mayor número de tecnologías y sistemas de ayuda a la conducción que pueden sustituir algunas acciones del conductor.
- **Nivel 3:** este nivel supone un gran avance respecto a los anteriores ya que el vehículo posee una mayor capacidad de autonomía. En este nivel es el propio vehículo el que toma las decisiones y circula libremente, aunque esto no exime al conductor de estar constantemente alerta debido a que el sistema puede fallar y solicitar su intervención.
- **Nivel 4:** este nivel de autonomía roza casi la totalidad, ya que la tecnología implementada será capaz de analizar el entorno, controlar el tráfico y objetos de la vía, determinar la ruta, etc. Esto es posible gracias a cámaras y sensores incorporados en el vehículo, que tienen la capacidad de monitorizar lo

que sucede tanto fuera como dentro del mismo. Hoy en día, los vehículos pertenecientes a este nivel de autonomía no están aún en el mercado, sólo funcionan a modo de prueba.

- **Nivel 5:** este último nivel alcanza la autonomía total. Uno de los grandes cambios de los vehículos incluidos en este nivel se encuentra en la ausencia tanto del volante como de los pedales, ya que en este caso no será necesario el conductor, el cual pasará a ser un pasajero más. Dicha ausencia es posible debido a que, a diferencia de los niveles mencionados anteriormente en los que el conductor tomaba el mando en caso de fallo del sistema, ahora el encargado de solventar dicho fallo será otro sistema de emergencia.

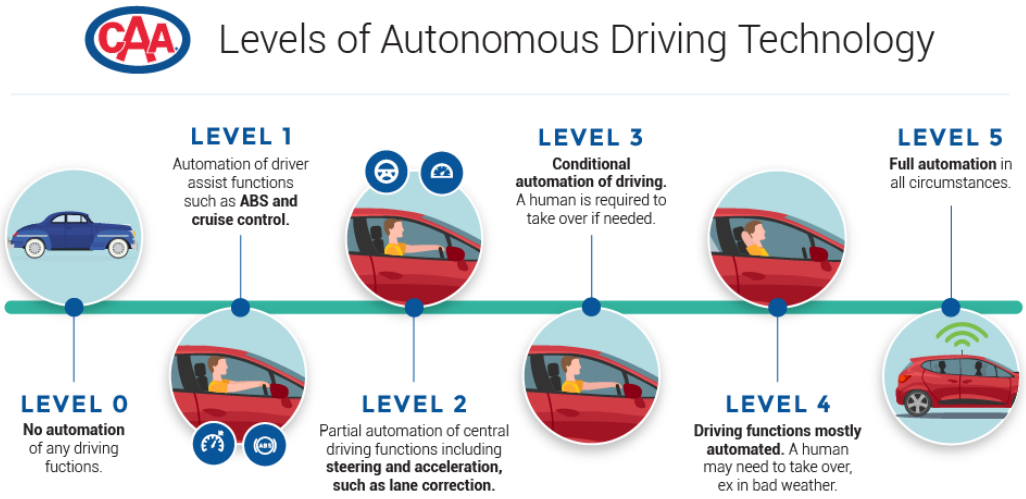


Figura 2. Niveles de conducción autónoma.

2.3. Sistemas de ayuda a la conducción

En relación a los niveles de autonomía vistos anteriormente, es importante tener en cuenta que a día de hoy aún no ha sido posible el desarrollo de un coche completamente autónomo, ya que los pertenecientes a los dos últimos niveles de autonomía (4 y 5), sólo funcionan a modo de prueba. Lo que sí es hoy en día una realidad, es el uso cada vez más habitual de sistemas que hacen la circulación más segura. Esto se debe a que el principal objetivo de la industria del automóvil sigue siendo reducir el número de accidentes de tráfico a partir del desarrollo de vehículos más seguros. Para garantizar esta seguridad, se están llevando a cabo de forma gradual dichos sistemas de ayuda a la conducción, también conocidos por sus siglas en inglés ADAS (Advanced Driver Assistance Systems), los cuales consisten en un conjunto de tecnologías capaces de disminuir la cantidad de riesgos que pueden aparecer mientras se circula, no sólo para los ocupantes del coche sino también para otros usuarios presentes en la vía [10].

Los sistemas comentados, están formados por diferentes equipos como son: los sensores, las cámaras de video, los radares o los sistemas LIDAR [11]. A continuación se verán algunos ejemplos de estos sistemas de ayuda a la conducción comentados previamente [12]:

- **Avisador de ángulo muerto:** durante una maniobra de cambio de carril, el vehículo avisa mediante una o varias señales de la presencia de otro vehículo en el ángulo muerto. La detección la realiza a partir de sensores de radar situados en la parte trasera del vehículo.
- **Sistema de detección de fatiga:** el objetivo es evitar que el conductor se desconcentre mientras conduce por causas similares al sueño o la fatiga. Esta falta de concentración se detecta a partir de la evaluación de la velocidad angular del volante. En el momento en el que el vehículo detecta un comportamiento durante las maniobras que difieren al típico del conductor, genera una alarma visual, acústica o sensorial.
- **Sensores de aparcamiento:** durante una maniobra de aparcamiento el vehículo avisa al conductor de la proximidad de algún obstáculo o persona, gracias a sensores situados tanto en la parte delantera como trasera del vehículo.

- **Frenado autónomo de emergencia:** permite al vehículo, gracias a cámaras, radares y algunos sensores, detectar otros vehículos parados o que circulan a una velocidad menor, peatones e incluso ciclistas. Una vez detectado el obstáculo y en caso de que exista peligro por colisión, el vehículo avisará al conductor y en caso de que este no haga nada, será el propio vehículo el que frene.
- **Sistema de reconocimiento de señales de tráfico** [13]: detecta a través de cámaras situadas en el espejo retrovisor interior, y en tiempo real, las señales de tráfico tanto permanentes como temporales que el vehículo se encuentre en la vía durante el trayecto, con el fin de informar al conductor y evitar que se despiste. Este sistema tiene dos formas de aplicación:
  - Aplicación pasiva: tiene un objetivo informativo. Avisa a través de pictogramas o sonidos al conductor si infringe alguna señal.
  - Aplicación activa: además de avisar con señales visuales o auditivas, interviene en caso de una posible colisión.
- **Sistema ISA (Intelligence Speed Assistance)** [14]**:** se trata de un asistente de velocidad inteligente impulsado por la DGT y la Comisión Europea que será obligatorio para todos los vehículos nuevos homologados a partir de 2022. El sistema es capaz de leer la máxima velocidad permitida en un determinado tramo y adapta la máxima velocidad del coche a dicha limitación. La detección la realiza de forma similar al sistema de reconocimiento de señales de tráfico ya comentado, a partir una cámara situada detrás del retrovisor, el GPS y otros sensores de los que disponga el vehículo. Hay tres niveles diferentes para el sistema ISA:
  - Informativo o de asesoramiento: en este nivel se avisa al conductor de que está excediendo los límites de velocidad, a partir de sonidos, vibraciones o iconos en el cuadro de mandos como puede observarse en la Figura 3 [15].
  - De apoyo o advertencia: en este nivel, además de avisar al conductor el sistema también ejerce una presión hacia arriba sobre el acelerador, el cual frenará al vehículo.
  - Interviniente u obligatorio: en este nivel, el asistente impedirá el exceso del límite de velocidad actuando sobre la centralita, en caso de querer exceder el límite habrá que pisar fuerte el acelerador o pulsar una tecla con esa función.

Figura 3. Funcionamiento del sistema ISA (Asistente de velocidad inteligente).

Destacar que, en 2017, una tercera parte de los coches vendidos en España, equipaban alguna de estas ayudas, las cuales son cada vez más aceptadas y demandadas por el público, y cuya combinación genera funciones que suponen los niveles ya mencionados, 2 y 3 de conducción autónoma [16]. En los próximos apartados, se diseñará un modelo que permita la clasificación de señales de tráfico.

3 DEEP LEARNING

Las 'leyes del pensamiento' no solo dependen de las propiedades de las células cerebrales, sino del modo en que están conectadas.

- Marvin Minsky -

En este capítulo se estudiarán los conceptos teóricos necesarios para el desarrollo del clasificador de señales de tráfico que se quiere crear. En primer lugar, se analizará el concepto de Machine Learning o aprendizaje máquina, así como la clasificación de sus algoritmos. Por otro lado, se estudiarán tanto las redes neuronales como las redes neuronales convolucionales, su estructura, su funcionamiento y las métricas utilizadas para la evaluación de los modelos.

3.1 Machine Learning

La Inteligencia Artificial mencionada anteriormente, es un campo muy amplio que abarca a su vez muchas ramas diferentes. Entre las más importantes, cabe destacar el Machine Learning también conocido como aprendizaje automático. Con el aprendizaje automático aparece un nuevo paradigma de la programación. En la programación tradicional, se obtenía un resultado a partir de un programa y de unos datos, en cambio, con el aprendizaje automático se obtiene el programa a partir de los datos y los resultados [17].

Puede definirse por tanto el aprendizaje automático como la rama de la inteligencia artificial en la que los ordenadores tienen la capacidad de aprender sin la necesidad de ser programados con anterioridad [1].

3.1.1 Clasificación de algoritmos Machine Learning

La gran cantidad de algoritmos de aprendizaje automático han llevado a clasificarlos en diferentes categorías según cuánta supervisión reciban durante el entrenamiento y su tipo de supervisión. Se pueden distinguir 4 grupos diferentes de Machine Learning [18]:

- **Aprendizaje supervisado:** en este tipo de aprendizaje, los datos que se usan para el entrenamiento contienen etiquetas (también conocidas en inglés como labels) las cuales incluyen la solución deseada. Este tipo de aprendizaje se usa principalmente para dos tipos de tareas: la clasificación y la regresión. Un ejemplo de clasificación puede ser el filtro de spam, como el mostrado en la Figura 4, en el que se entrena al modelo con muchos correos electrónicos pertenecientes a dos clases (spam y no spam) con sus etiquetas correspondientes y éste debe ser capaz de clasificar los correos nuevos. Por otro lado, un ejemplo de regresión puede ser la predicción del precio de una vivienda a partir de unas características determinadas, conocida como predictores. Algunos de los algoritmos más usados en este tipo de aprendizaje son las máquinas de soporte vectorial (SVM), los árboles de decisión y las redes neuronales.

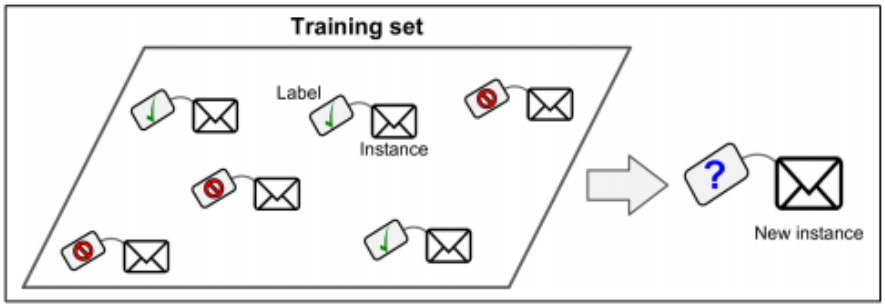


Figura 4. Aprendizaje supervisado para filtro de spam.

- **Aprendizaje no supervisado:** a diferencia del aprendizaje supervisado, ahora los datos que se usan para el entrenamiento no están etiquetados y tendrá que ser el propio algoritmo el que aprenda por sí mismo. Según la tarea realizada se encuentran diferentes algoritmos de aprendizaje no supervisado: clustering, visualización de datos, reducción de la dimensionalidad, detección de anomalías y reglas de asociación. El clustering consiste en detectar grupos con características similares y uno de sus métodos más usados es el k-means. Por otro lado, los algoritmos de visualización de datos se usan para representar los datos de manera visual de forma que se pueda entender su organización e identificar patrones sospechosos, destacando el método del análisis de componentes principales (PCA). La reducción de dimensionalidad es otra tarea relacionada, cuyo objetivo es reducir los datos, tratando de no perder una gran cantidad de información. Otra operación importante es la detección de anomalías, en la que se entrena al sistema con datos normales y cuando ve datos nuevos detecta si se parece a un dato normal o se trata de una anomalía. Por último, las reglas de asociación trabajan con muchos datos e intentan encontrar relaciones interesantes entre los atributos.

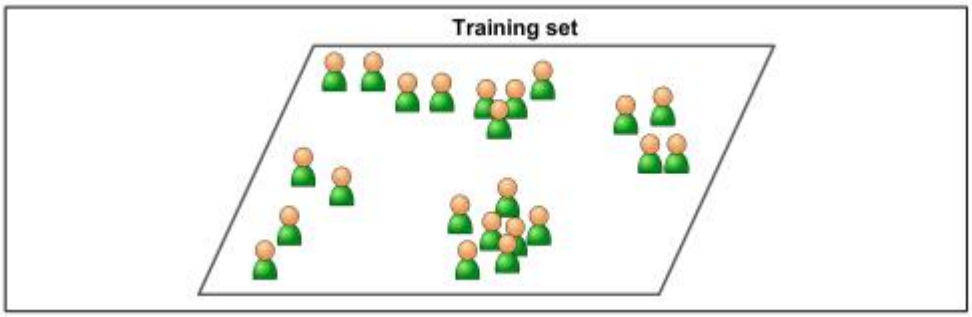


Figura 5. Aprendizaje no supervisado.

- **Aprendizaje semisupervisado:** en este tipo de aprendizaje, los datos usados para el entrenamiento estarán etiquetados de forma parcial, es decir, habrá un porcentaje de estos datos que sí estarán etiquetados y otro porcentaje que no, siendo normalmente el porcentaje de datos no etiquetados mayor que el de datos etiquetados. La mayor parte de los algoritmos de aprendizaje semisupervisado suelen ser una combinación de algoritmos supervisados y no supervisados.

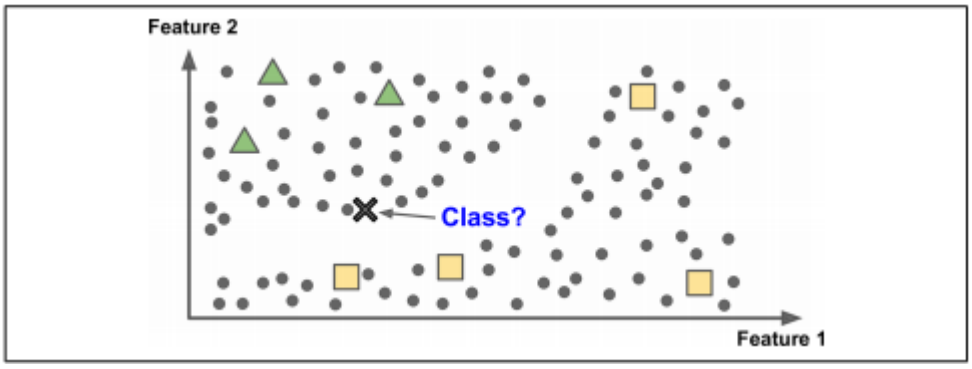


Figura 6. Aprendizaje semisupervisado

- **Aprendizaje por refuerzo:** este tipo de aprendizaje es diferente a los vistos hasta ahora. Un sistema conocido como "agente" tiene que explorar un espacio que resulta desconocido para él y determinar qué acciones llevará a cabo a base de prueba y error, es decir, a partir de una serie de recompensas o penalizaciones que recibirá. El agente deberá aprender de forma autónoma la mejor estrategia (también conocida como política) que le permita obtener la mayor recompensa en el tiempo. Este tipo de aprendizaje está muy moda actualmente, y un ejemplo de ello puede ser la implementación en robots para que aprendan a realizar diversas tareas como puede ser la de andar. De los algoritmos de este tipo de aprendizaje, cuyo funcionamiento puede observarse en la Figura 7 [19], cabe destacar el Q-Learning.

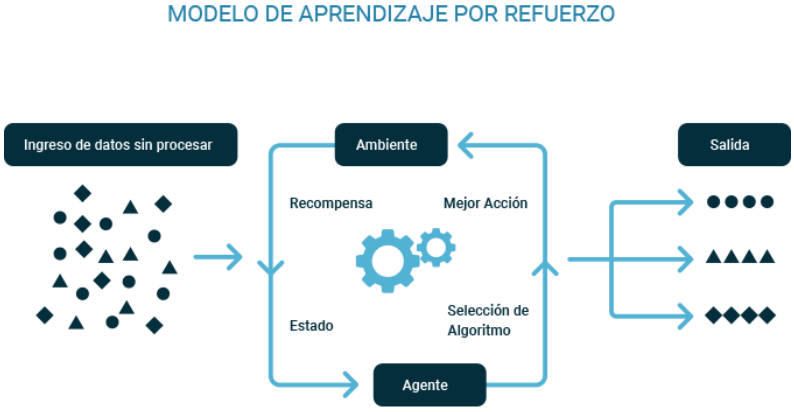


Figura 7. Funcionamiento aprendizaje por refuerzo.

3.2 Redes Neuronales

El Aprendizaje Profundo o Deep Learning es un subcampo del Machine Learning que pertenece al campo de la inteligencia artificial como puede apreciarse en la Figura 8 [20]. El Deep Learning puede considerarse una nueva forma de aprendizaje a partir de datos que se basa en el estudio jerárquico de capas sucesivas de representaciones, desde las menos significativas hasta las más significativas. El término "profundo" en este tipo de aprendizaje representa la idea de sucesión de capas de representaciones, es decir, no se refiere a que el enfoque sea capaz de lograr una compresión más profunda. El número de capas que un modelo de datos tiene que tratar se conoce como profundidad del modelo, siendo variable este número de capas sucesivas, desde una o dos, hasta decenas o cientos de estas capas [17].

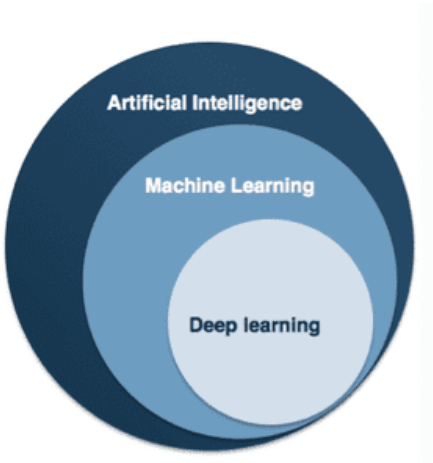


Figura 8. Relación Inteligencia Artificial-Machine Learning-Deep Learning

El aprendizaje profundo se basa en un modelo computacional conocido como redes neuronales artificiales, donde las múltiples capas se apilan unas sobre otras. El término de red neuronal hace referencia a la neurobiología, ya que algunos conceptos se inspiran en el funcionamiento del cerebro humano a partir de la interconexión de sus neuronas. Este nuevo concepto de redes neuronales se comprenderá mejor centrándose primero en la red neuronal más básica existente, el perceptrón, creada por Frank Rosenblatt en el año 1957, la cual consta de una sola capa con una neurona como puede observarse en la Figura 9. El perceptrón tiene varias entradas binarias (x_1, x_2, x_n) y genera una sola salida la cual es binaria también (y_j).Posee además un parámetro (b_j) denominado sesgo o bias que toma siempre el valor 1 multiplicado por un valor. La salida se calcula a partir de unos pesos (w_{1j}, w_{2j}, w_{nj}) asociados a cada entrada. Rosenblatt introdujo el concepto de pesos como un número que define la intensidad con la que una entrada afecta a la neurona. Una vez establecidos estos pesos, se multiplican cada una de las entradas por sus respectivos pesos y se realiza una suma ponderada (z_j). El resultado de dicha suma ponderada pasará por una función de activación y dará una clasificación de los valores de entrada [21].

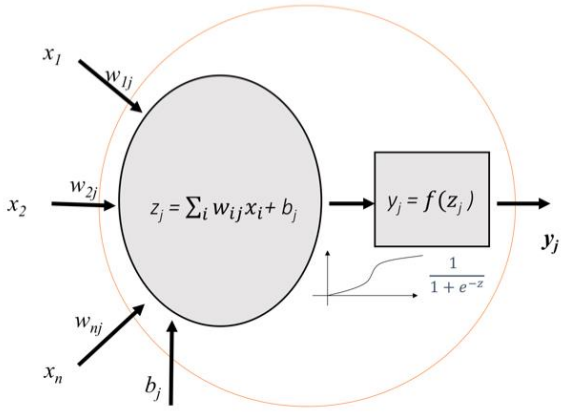


Figura 9. Estructura del perceptrón

El perceptrón resulta útil para la clasificación de datos que se pueden separar de forma lineal, sin embargo, para conjuntos de datos no separables linealmente habrá que añadir más capas y más neuronas, surgiendo así el perceptrón multicapa también conocido por las siglas en inglés MLP (Multilayer Perceptron) o redes neuronales artificiales.

Al igual que el perceptrón simple, el perceptrón multicapa tendrá también capas de entrada (input layer) y de salida (output layer), pero además incluirá una o varias capas ocultas (hidden layer) las cuales estarán completamente conectadas. En la Figura 10 [22] puede observarse la estructura comentada, en concreto, de una red neuronal profunda formada por 5 capas, la capa de entrada, dos capas ocultas totalmente conectadas y la capa de salida.

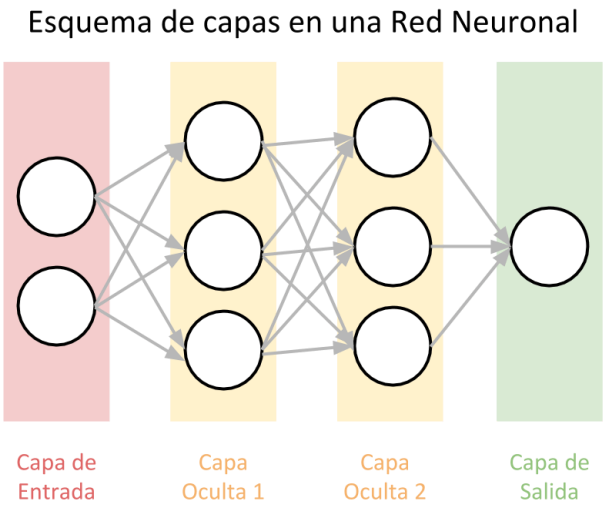


Figura 10. Estructura red neuronal profunda

3.3. Entrenamiento de redes neuronales

3.3.1. Proceso de aprendizaje

Cuando la red neuronal es creada, los pesos y el sesgo se establecen de forma aleatoria, siendo durante la fase de entrenamiento cuando la red aprenda, y estos pesos se ajusten. Primero se realizará la propagación hacia delante o forward propagation, en la que se alimenta a la red con los datos de entrada, los cuales recorrerán toda la red de forma que cada neurona aplique una determinada transformación a la información recibida por la capa anterior y la envíen a la capa próxima. Una vez en la capa final, es decir, recorridas todas las capas y realizados todos los cálculos, se obtendrá una predicción del resultado. Cuando se tiene el valor de salida, se calcula el error, es decir, la diferencia ente el resultado de predicción y el valor real de la salida. Dicha información se propaga hacia atrás, de ahí su nombre retropropagación o backpropagation. Partiendo de la última capa, capa de salida, el error es propagado a las neuronas de la capa oculta que han contribuido de forma directa a la salida, pero dichas neuronas de la capa oculta no reciben el valor de error total sino un porcentaje en función de cómo haya sido su contribución a la salida. Capa por capa se repetirá este proceso hasta que cada neurona reciba su contribución de error. Una vez propagada esta información hacia atrás, ya se pueden ajustar los pesos de nuevo. Para ajustar estos pesos de conexiones entre las neuronas habrá que hacer que el coste sea aproximadamente cero, y para conseguir esto se usa la técnica conocida como descenso de gradiente, la cual es capaz de encontrar el mínimo de una función, en este caso de la función de coste. El descenso de gradiente lo que hace es cambiar los pesos en incrementos pequeños a partir de la derivada del gradiente de la función de coste [1].

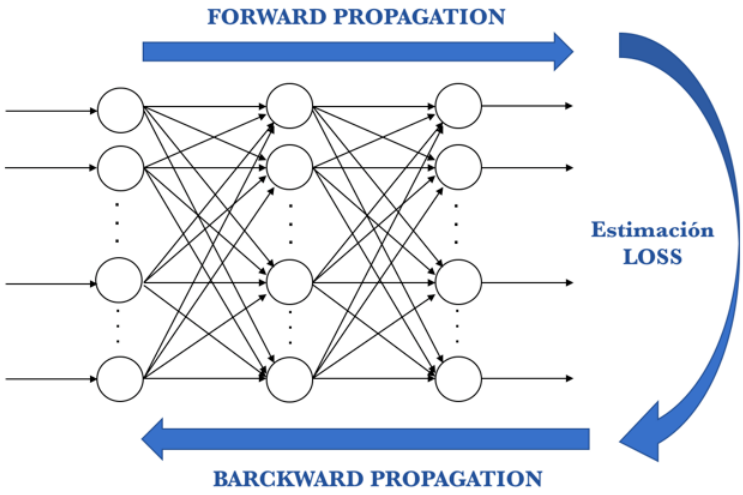


Figura 11. Proceso de aprendizaje de redes neuronales

3.3.1.1. Optimizadores

El descenso de gradiente calcula las primeras derivadas de la función de coste, por eso es un método de optimización considerado de primer orden. Aporta información de la pendiente, pero se necesita información más concreta como es la de la curvatura. Para obtener esta información se podría calcular las segundas derivadas, pero esto supone un gran problema, que es el gran coste computacional. Por ello para el enteramiento se usan optimizadores sobre el descenso del gradiente, entre ellos se encuentran [23]:

- **SGD (Stochastic Gradient Descent):** a diferencia del descenso de gradiente, los pesos se actualizan tras lotes de n muestras, lo que hace que los pesos varíen más rápido y por tanto que converja más rápido hacia el mínimo.
- **AdaGrad (Adapative Gradient Algorithm):** este algoritmo disminuye la tasa de aprendizaje, decae más rápido en pendientes fuertes que en pendientes suaves, lo que se conoce como tasa de aprendizaje

adaptativa. Suele funcionar bien cuando se usa para problemas cuadráticos simples, pero no tanto cuando se trata de entrenar a redes neuronales. Esto se debe a la gran reducción de la tasa de aprendizaje, que hace que el algoritmo se detenga antes de alcanzar el óptimo global.

- **RMSProp (Root Mean Squeare Propagation):** el algoritmo RMSProp soluciona el problema ya mencionado del algoritmo AdaGrad, la no convergencia del optimo global. La solución es acumular solo gradientes de interacciones que se han realizado recientemente, en vez de acumular todos los gradientes del entrenamiento, y para ello realiza un descenso en forma exponencial. Normalmente esta tasa de descenso se suele fijar a 0.9.
- **Adadelta:** es un algoritmo adaptativo que utiliza solo información de primer orden y no requiere que la tasa de aprendizaje se ajuste de forma manual. [24]
- **Adam (Adaptive Moment Estimation):** es uno de los optimizadores más usados hoy en día en el aprendizaje profundo. Combina las ventajas de AdaGrad y RMSProp, lo que permite manejar gradientes dispersos en problemas ruidosos. Se puede profundizar más sobre este optimizador en el artículo [25]

3.3.1.2. Funciones de activación

Estas funciones se usan para realizar la propagación hacia delante de la salida de una neurona, de forma que esta salida será recibida por las neuronas de la siguiente capa y se utilizan para introducir la no linealidad en el modelo creado. Las funciones más usadas se muestran en la Figura 12 [26], junto a sus respectivas ecuaciones y gráficas, y se explican a continuación.

- **Función Lineal:** Se conoce también como función entidad y permite que la entrada mantenga constante su valor, es decir, la siguiente neurona recibirá como entrada los mismos valores que la anterior. Esto se debe a la relación directa y proporcional entre la variable dependiente y la independiente.
- **Función Escalón:** Esta función se conoce también con el nombre de umbral, y asignará valores a la salida con valor 0 o 1. Si la entrada es menor que cero, la salida valdrá cero, en cambio si la entrada es mayor o igual a cero, la salida pasará a valer uno. Se usará esta función cuando se quiera hacer una clasificación o cuando las salidas sean categorías.
- **Función Signo:** esta función es similar al escalón, pero los valores asignados a la salida tendrán valor 1 y -1.
- **Función Sigmoide:** Reduce aquellos valores que son extremos o atípicos en datos válidos, en lugar de eliminarlos. Los valores de entrada de rango casi infinito se convierten en probabilidades entre 0 y 1, de forma que valores altos tenderán de forma asintótica a 1 y los valores bajos tenderán de forma asintótica a 0. Como se puede observar en la Figura 12 [26], esta función no está centrada, lo cual influye tanto en el aprendizaje como en el entrenamiento y esto ha llevado a disminuir su uso y reemplazarla por otro tipo de función.
- **Función Tanh (Tangente Hiperbolica):** Los valores de salida, en este caso, se limitarán entre -1 y 1, lo cual proporciona la ventaja de trabajar de una manera más fácil con números negativos. A diferencia de la sigmoide, si está centrada, pero sigue teniendo el problema de desaparición del gradiente. Se suele usar en redes recurrentes debido a su buen desempeño en estas.
- **Función ReLU (Rectified Lineal Unit):** La función ReLU es muy usada en la práctica gracias a su buen funcionamiento y rápido cómputo. Esta función se activa sólo si los valores son positivos. Cuando la entrada es negativa la pone a cero, en cambio, si es positiva la mantiene. Se caracteriza por ser una función no acotada y que funciona bien con imágenes, además de su buen cometido en las redes convolucionales.

- **Función LeakyReLU:** La función ReLU ya comentada, genera un problema para valores de entrada con valor cero, así como para derivadas iguales a cero. Esta situación da lugar a la muerte no deseada de determinadas neuronas. Para evitar la muerte de dichas neuronas, aparece la función LeakyRelu, creando una pequeña pendiente para valores de entrada menores que cero. Esta pendiente se consigue multiplicando por un coeficiente rectificativo los valores negativos y manteniendo iguales los valores positivos.

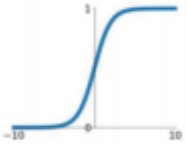
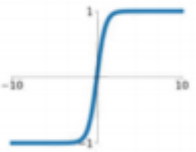
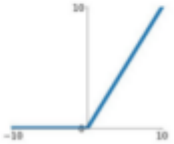
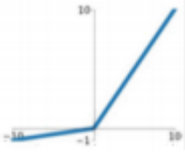
Función	*Ecuación*	*Gráfica*
Escalón	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	
Signo	$f(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases}$	
Lineal	$f(x) = x$	
Logística (sigmoide)	$f(x) = \frac{1}{1+e^{-x}}$	
Tangente hiperbólica	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
ReLU	$f(x) = \max(0, x)$	
Leaky ReLU	$f(x) = \max(0.01x, x)$	

Figura 12. Principales funciones de activación

3.3.2. Hiperparámetros

Los hiperparámetros son variables configurables cuyo valor no puede estimarse a partir de los datos ya que son variables externas al modelo en sí, es decir, es el programador el que las específica para poder así ajustar los algoritmos de aprendizaje. Estos hiperparámetros no se deben confundir con los parámetros del modelo, que son variables cuyo valor si puede estimarse a partir de los datos ya que son internas al modelo. Es importante ajustar bien estos hiperparámetros antes de comenzar el entramiento del modelo para que estos se entrenen mejor y de una forma más rápida. Los hiperparámetros pueden ser de diferentes tipos según su nivel de estructura y la topología de la red como son el número de capas, el número de neuronas, las funciones de activación, etc, los cuales se han visto en apartados anteriores. Por otro lado, se encuentran los hiperparámetros según el nivel de algoritmo de aprendizaje los cuales se introducen a continuación [1].

- **Epochs (épocas):** este hiperparámetro es el encargado de indicar cuantas veces pasan los datos por la red neuronal a lo largo de la fase de entrenamiento.
- **Batch size (tamaño del lote):** los datos de entrenamiento pueden dividirse en lotes, por tanto, el batch size indica cuantas muestras habrá en cada lote para cada iteración durante el entrenamiento.
- **Learning rate (tasa de aprendizaje):** durante cada iteración, se multiplica el gradiente por el learning rate para que se acerque poco a poco al mínimo global y minimizar así el coste de la función. Por tanto, el learning rate indica como de rápido se lleva a cabo el aprendizaje. Es un hiperparámetro muy independiente del problema a tratar, pero generalmente, si su valor es demasiado grande, se está aprendiendo de forma muy rápida y por tanto se puede saltar algún mínimo. De lo contrario, si su valor es muy pequeño, se avanzará poco a poco teniendo más oportunidades de encontrar el mínimo local, pero a su vez hará el proceso de aprendizaje muy lento.

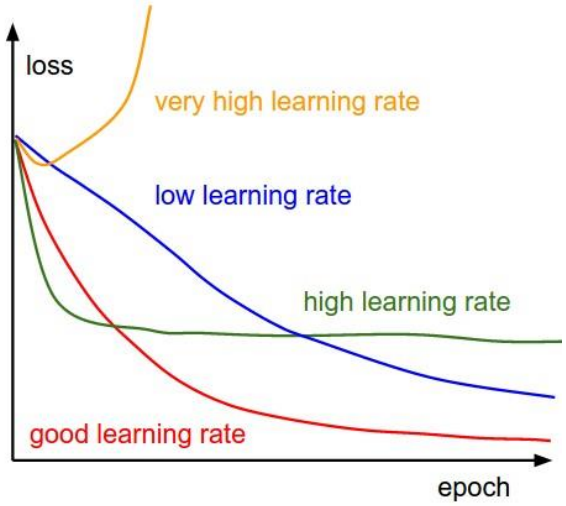


Figura 13. Convergencia según diferentes tasas de aprendizaje.

3.3.3. Resultados

Cuando la red ha sido entrenada se pueden obtener diversos resultados.

- **Buen ajuste:** el resultado idóneo es conseguir un modelo equilibrado, que se encuentre entre el underfitting y el overfitting, aunque es un objetivo difícil de conseguir en la práctica. Se dice que un modelo está bien ajustado cuando es capaz de generalizar, es decir, se ha adaptado bien a los datos de entramiento y cuando se le introduce un dato nuevo es capaz de clasificarlo de manera correcta.
- **Subajuste (Underfitting):** se dice que se ha producido underfitting cuando el modelo no ha sido capaz de ajustarse bien a los datos de entrenamiento. Este problema puede darse por un entramiento durante pocas épocas o tasas de aprendizaje bajas.

- **Sobreajuste (Overfitting):** el sobreajuste, también conocido en inglés como overfitting, se produce cuando el modelo se ajusta muy bien a los datos de entrenamiento, pero no es capaz de generalizar de forma correcta los datos de test, esto se debe a que ha memorizado los datos de entrada.

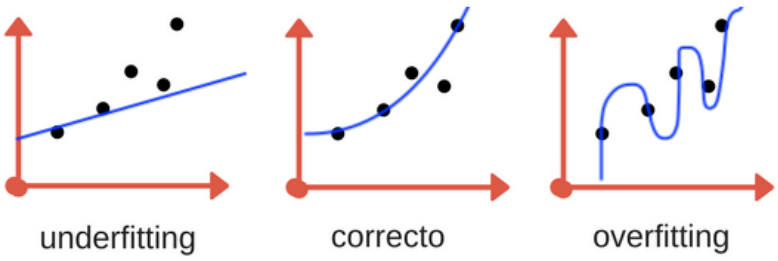


Figura 14. Ejemplos de modelos sobreajustados, bien ajustados y subajustados.

3.3.4. Datos de entrenamiento, validación y test

A la hora de crear, entrenar y evaluar un modelo es necesario un conjunto de datos. Este conjunto de datos se separa en dos conjuntos a su vez: el de entramiento (train) y el de prueba (test). El porcentaje de cada conjunto puede variar según el problema a tratar, aunque los más usados son 80/20 y 70/30. Para evitar el problema de overfitting ya comentado, se suele dividir de nuevo el conjunto de entrenamiento en dos: el conjunto de entrenamiento y el de validación. Esta división del conjunto de datos puede observase en la Figura 15 [27].

El conjunto de entramiento, como indica su nombre es el que se usa para entrenar a la red, es decir, para obtener los parámetros del modelo. Por otro lado, el conjunto de validación no se usará para entrenar a la red, pero servirá para conseguir una solución óptima a partir de la modificación de los hiperparámetros. Por último, el conjunto de test se usará una vez acabo del proceso de entrenamiento y validación y permitirá evaluar el modelo final.

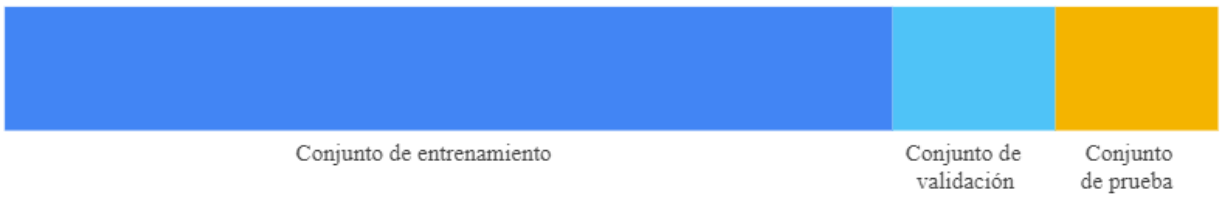


Figura 15. División del conjunto de datos en tres subconjuntos.

3.4. Métricas de evaluación

Una vez creado el modelo hay que evaluarlo, es decir, entender lo bien que clasifica y realizar predicciones en un determinado contexto. Para realizar esta evaluación se utilizan diversas métricas que dependen del problema tratado, ya que a veces solo importa con qué frecuencia acierta el modelo cualquier predicción, y en otras en cambio, lo importante es que el modelo acierte con más frecuencia una determinada predicción [21]. En este apartado se describirán brevemente las métricas más empleadas en la evaluación de modelos y por tanto las que se han usado a la hora de realizar este trabajo.

3.4.1. Matriz de confusión (Confusion matrix)

La matriz de confusión no puede considerarse una métrica en sí, pero si es una herramienta útil para evaluar el rendimiento de un modelo. Es una matriz de m filas y n columnas, donde cada fila contiene el valor real de la clase (etiqueta) y cada columna representa la predicción. La matriz de un clasificador perfecto es aquella que tiene todos los valores iguales a cero excepto los de la diagonal principal.

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

Figura 16. Estructura matriz de confusión para un clasificador binario.

Como puede observarse en la Figura 16 [1], se pueden obtener 4 valores diferentes según la correcta clasificación o no por parte del modelo. Para explicar estos valores, se va a utilizar un ejemplo práctico presente en la actualidad, como es el virus conocido como COVID-19, que ha provocado una pandemia mundial en el año 2020. Para ello se diferenciará entre ser o no positivo en Covid-19 y que la prueba realizada para la detección del virus sea positiva o negativa. Por tanto, suponiendo un clasificador binario con 2 clases, positiva y negativa o Covid-19 (+) y Covid-19 (-) respectivamente, se definen los valores:

- **Verdadero Positivo (VP):** resultado en el que el modelo es capaz de predecir de forma correcta la clase positiva. Es decir, la persona tiene Covid-19 y el modelo lo clasifica como Covid-19 (+).
- **Verdadero Negativo (VN):** resultado en el que el modelo es capaz de predecir correctamente la clase negativa. Es decir, la persona no tiene Covid-19 y el modelo lo predice como Covid-19 (-).
- **Falso Positivo (FP):** resultado en el que el modelo no es capaz de predecir de forma correcta la clase positiva. Es decir, la persona no tiene Covid-19, pero el modelo lo clasifica como Covid-19 (+).
- **Falso Negativo (FN):** resultado en el que el modelo predice de forma incorrecta la clase negativa. Es decir, la persona tiene Covid-19, en cambio el modelo lo clasifica como Covid-19 (-).

Gracias a la matriz de confusión se puede obtener mucha información de forma visual, pero hay casos en los que se necesita una métrica más concisa a la hora de evaluar un modelo. Algunas de estas medidas se verán a continuación [28].

3.4.2. Exactitud (Accuracy)

La exactitud, más conocida por su nombre en inglés, accuracy, es la métrica que se define como la fracción de predicciones correctas realizadas por el modelo.

$$accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

Para conjuntos de datos desequilibrados es aconsejable acompañar esta métrica de otras como pueden ser la precisión y la exhaustividad ya que la exactitud sola no es capaz de mostrar el panorama completo.

3.4.3. Precisión (Precision)

La precisión la métrica que mide la proporción de clasificaciones positivas que han sido correctas.

$$precision = \frac{VP}{VP + FP}$$

3.4.4. Exhaustividad o sensibilidad (Recall)

La exhaustividad o sensibilidad (o tasa de verdaderos positivos), conocida por su nombre en inglés recall, es la métrica que mide el porcentaje de positivos reales clasificados correctamente.

$$recall = \frac{VP}{VP + FN}$$

3.4.5. Puntuación F1 (F1 score)

La precisión y la sensibilidad se pueden combinar en una sola métrica, conocida como F1 score, que es la media armónica de estas dos métricas. A diferencia de la media regular, que trata por igual todos los valores, la media armónica da mayor peso a los valores bajos. Por ello, resulta de gran utilidad para conjuntos de datos desequilibrados.

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = 2\ x\ \frac{precision\ x\ recall}{precision + recall} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

3.5. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (Convolutional Neuronal Networks, CNN o Convnet) son unas de las redes más usadas dentro del Deep Learning y las que se usarán a lo largo de este trabajo, por ello es necesario un conocimiento previo de su estructura. Fue Yann LeCun en su documento "Gradient-Based Learning Applied to Document Recognition" [29] el que desarrollo la primera red neuronal convolucional con éxito conocida como LeNet, la cual clasificaba con una probabilidad de acierto alta caracteres escritos a mano a partir del conjunto de datos MNIST. Aunque estas redes se han utilizado desde 1980 en el reconocimiento de imágenes, desde que nacieron al estudiar la corteza visual del cerebro humano, pero ha sido en los últimos años cuando han conseguido un rendimiento excelente en tareas visuales que resultan complejas. Esto se debe a una gran mejora computacional, así como al aumento de los datos disponibles para el entramiento. Por ello, son la base de diversos sistemas como los servicios de búsqueda de imágenes, coches autónomos o sistemas clasificadores de videos. Mas allá de la percepción visual, han conseguido importantes logros en tareas como el reconocimiento de voz o el procesamiento del lenguaje natural [18]. La estructura de una CNN puede observarse en la Figura 17, está formada por sucesivas capas convolucionales y capas de pooling y por último una capa que será la que proporcione la salida del modelo, que es una capa completamente conectada.

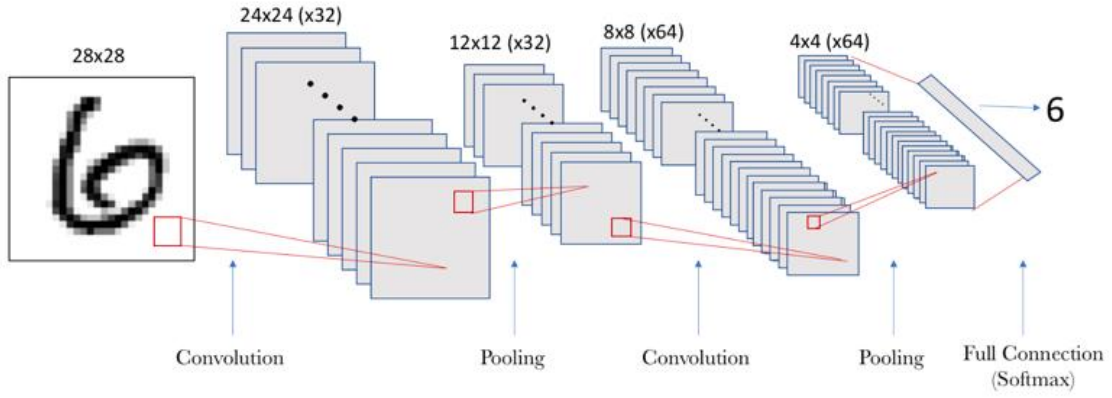


Figura 17. Ejemplo estructura red neuronal convolucional.

Actualmente existen varias arquitecturas populares de redes neuronales convolucionales las cuales tienen nombre propio, como son la ya mencionada LetNet, AlexNet creada por Alex Krizhevsky y ganadora de la competición de ImageNet en el año 2012 [30], GoogLeNet creada por Google [31] o VGG entre otras.

3.5.1. Capas convolucionales

El principal objetivo de estas capas es detectar elementos característicos en imágenes tales como aristas, líneas o colores, y aprender estas características para posteriormente ser capaz de identificarlas, aunque estén localizadas en diferentes puntos de la imagen. Asimismo, estas capas tienen la capacidad de llegar a aprender patrones visuales muy complejos, a partir de patrones más simples aprendidos en capas anteriores, los cuales se basan en elementos básicos aprendidos en la primera capa.

Estas capas reciben su nombre pues cuando les llega una imagen de tamaño mxn pixeles aplican el proceso de convolución, que consiste en agrupar pixeles cercanos de la imagen de entrada a la capa y realizar el producto escalar de estas agrupaciones y una matriz conocida como kernel o filtro. El tamaño del kernel es variable, aunque suele tener un tamaño menor que la imagen como pueden ser 3x3, 5x5,7x7, etc. El kernel recorre toda la imagen de entrada, desplazándose de izquierda a derecha y de arriba abajo, generando una nueva matriz de salida. Realmente a la imagen no se le aplica un solo filtro sino un conjunto de x filtros, que generarán x matrices de salida, cada una de estas matrices de salida se conoce como mapa de características. Esto se debe a que un filtro solo es capaz de detectar una determinada característica, por ello para el reconocimiento de imágenes se usaran varios filtros.

Las capas convolucionales trabajan sobre mapas de características (feature maps) con tres ejes, los cuales son altura, anchura y canales. Este último eje, también conocido como profundidad, puede tomar los valores 1 o 3 en función del número de canales (dimensión) que tenga la imagen. Tomará el valor 1, cuando la imagen tenga un solo canal, es decir, sea una imagen en blanco y negro. En caso contrario, la dimensión del eje será 3 cuando tenga tres canales: rojo, verde, azul, es decir, cuando sea una imagen en color RGB (red, green, blue).

Por ejemplo, si la entrada es una imagen de tamaño 28x28 en escala de grises, y se aplican 32 filtros, se obtendrían 32 matrices de salida de 28x28x1 cada una, lo cual daría lugar a una primera capa oculta de 25.088 neuronas. Este número es muy elevado para una imagen de tamaño 28x28, la cual se considera una imagen pequeña. Para reducir este número tan elevado aparecen las capas pooling [1].

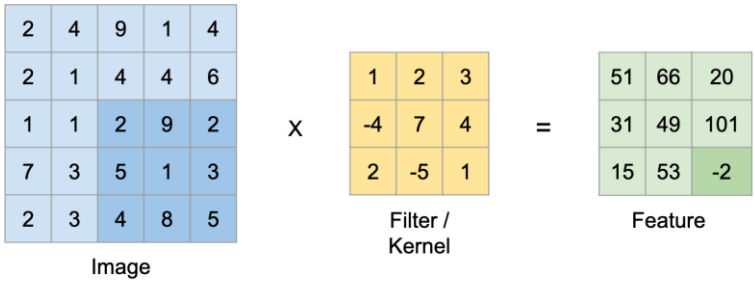


Figura 18. Operación de convolución.

3.5.2. Capas pooling

Estas capas se suelen colocar después de las capas convolucionales, y tienen como objetivo reducir la dimensión de las salidas resultantes a través del submuestreo, pero conservando las características más importantes ya detectadas. Al submuestrear se coge de cada grupo de mxn pixeles uno solo, según el criterio que se use para la elección del píxel se encuentran diferentes tipos de muestreo:

- Max-Pooling: se queda con el píxel de mayor valor.
- Min-Pooling: se queda con el píxel de menor valor.
- Average-Pooling: hace la media de los mxn pixeles.

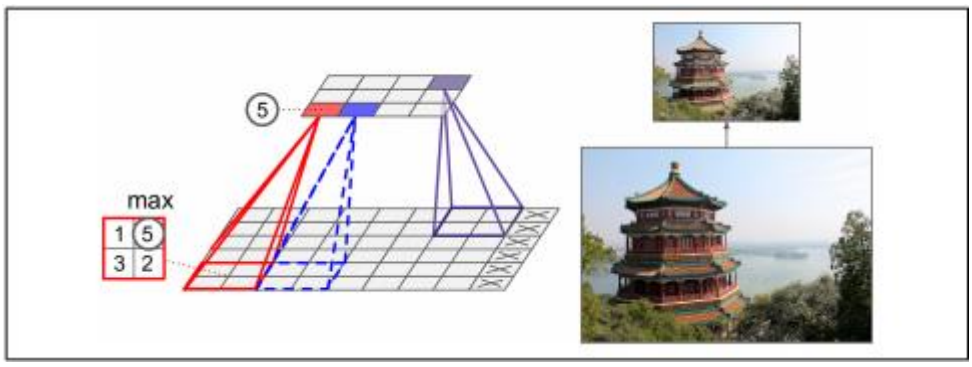


Figura 19. Operación max-pooling.

3.5.3. Dropout y BatchNormalization

Se comentó en apartados anteriores el problema del overfitting, muy presente en la creación de un modelo. Para reducir este problema surgen dos métodos diferentes pero que se apoyan entre sí, conocidos como dropout y batchNormalization. Pueden usarse ambos a la vez, aunque normalmente se usa primero batchNormalization y después si fuese necesario dropout.

La técnica de dropout es una de las técnicas de regularización más populares para redes neuronales profundas. Fue propuesta en el año 2012 por G.E.Hinton y puede encontrarse detallada en el artículo "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" [32]. Esta técnica ha demostrado tener gran éxito ya que redes neuronales complejas han conseguido aumentar su precisión en un 1-2 % añadiendo dropout. El funcionamiento del algoritmo es sencillo, en cada paso del proceso de entrenamiento, cada una de las neuronas tiene una probabilidad aleatoria p de desactivarse de forma temporal, es decir, a lo largo de este paso de entrenamiento se ignorará, aunque puede activarse de nuevo en el siguiente. Esta probabilidad p se conoce como tasa de abandono y suele tomar valores del 25% y el 50%.

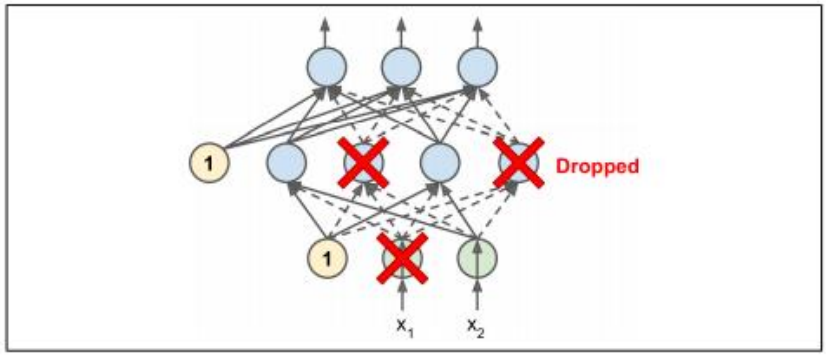


Figura 20. Regularizacion dropout

Por otro lado, la técnica de batchNormalization o normalización por lotes fue propuesta en el año 2015 por Sergey Loffe y Christian Szgedy en el artículo "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift" [33]. La distribución de las entradas de cada capa varia a lo largo del entrenamiento, esto se debe a que los parámetros de las capas anteriores también lo hacen, lo que hace que el proceso sea más lento. Para acelerar el entramiento, esta técnica propone normalizar cada mini lote del entrenamiento, lo cual permite el uso de tasas de aprendizaje más altas y realizar una inicialización menos cuidadosa. Para llevar a cabo esta normalización es necesario estimar tanto la media como la desviación estándar de las entradas y se utiliza detrás de cada capa de activación ReLU.

3.5.4. Capa flatten

Una vez obtenida la última capa submuestreada la cual es tridimensional tenemos que aplanarla para que sea unidimensional y pueda usar redes totalmente conectadas.

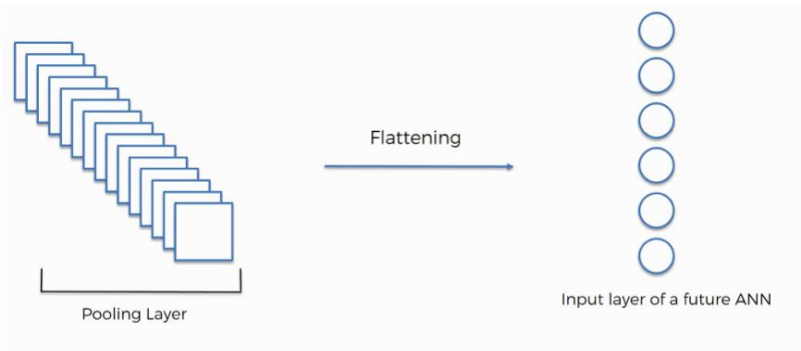


Figura 21. Aplanamiento de la última capa submuestreada.

3.5.5. Capa totalmente conectada

Con el aplanamiento de las ultimas capas de max-pooling, se consiguió un vector de números. Este vector será la capa de entrada de una red neuronal cuyos nodos estarán conectados todos con todos, de ahí su nombre capa totalmente conectada. Es decir, todos los nodos de la red neuronal artificial se conectan entre ellos, pudiendo tener esta red tantas capas ocultas como se necesiten también totalmente conectadas.

Una vez pasadas todas las neuronas de las diferentes capas ocultas se llega a la última capa, la capa de salida que será la encargada de clasificar la imagen de entrada y asignarle una clase.

Cuando se quiere hacer una clasificación múltiple se le aplica a la capa de salida la función Softmax. Esta capa tendrá tantas neuronas como clases de entrada haya. Por ejemplo, si se quieren clasificar imágenes de gatos, perros y caballos serán necesarias 3 neuronas. La función Softmax asigna a cada clase una probabilidad decimal en un caso de múltiples clases. La suma de estas posibilidades debe ser 1.

En este punto, se han adquirido conocimientos teóricos sobre aprendizaje profundo y redes neuronales convolucionales, las cuales se usarán para desarrollar un modelo capaz de resolver el problema de clasificación de señales de tráfico en el próximo capítulo.

4 METODOLOGÍA PROPUESTA

Los grandes conocimientos engendran las grandes dudas.

- Aristóteles-

Una vez estudiado el marco teórico sobre el aprendizaje profundo, en este capítulo se crearan y entrenarán los modelos de redes neuronales convolucionales capaces de resolver el problema planteado al inicio del trabajo. Pero antes, se explicarán las herramientas necesarias para dicho desarrollo, como pueden ser el lenguaje de programación usado, la plataforma en la que se llevará a cabo y la base de datos usada.

4.1. Herramientas

4.1.1. Python

A la hora de realizar un proyecto de Deep learning se pueden utilizar diversos lenguajes de programación, como pueden ser Python, Matlab, R o Julia, siendo Python y R los predominantes. Para llevar a cabo este trabajo se ha elegido la opción de Python ya que se tienen algunos conocimientos básicos de este lenguaje de programación y son muchos los recursos en internet que permiten su aprendizaje de forma autodidacta. Además, es un lenguaje más genérico cuyo aprendizaje servirá en un futuro para la realización de proyectos pertenecientes a diversos campos.

4.1.1.1. Librerías

Python posee una gran variedad de librerías, a continuación, se explican brevemente las utilizadas a la hora de realizar este proyecto:

- **Os:** esta librería permite el uso de funcionalidades que son dependientes del sistema operativo, como puede ser la manipulación de rutas [34].
- **Numpy:** principal paquete para la computación científica que permite trabajar con matrices y campos como el álgebra lineal y la estadística entre otros [35].
- **Matplotlib:** es una biblioteca que permite crear gráficos estáticos, dinámicos o interactivos en Python [36].
- **Random:** es una librería que contiene funciones que permiten generar números aleatorios [37].
- **Opencv:** es una biblioteca de visión artificial que se utiliza para el tratamiento digital de imágenes [38].
- **Pandas:** librería que permite manipular y analizar datos de forma rápida, potente y flexible [39].

4.1.1.2. Frameworks

Los marcos de aprendizaje profundo conocidos también por sus nombres en inglés, frameworks, ofrecen bloques de construcción que permiten el diseño, entrenamiento y validación de redes neuronales profundas, a partir de una interfaz de programación. Entre los frameworks más populares se encuentran: Tensorflow, Keras, Theano,Caffe y Pythorch.

Para el desarrollo de este trabajo se ha elegido Keras como framework, que es una biblioteca de código abierto capaz de proporcionar bloques de alto nivel que permiten el desarrollo de modelos de aprendizaje profundo. Para ello, usa otras librerías como backend como pueden ser Theano, CNTK(Microsoft Cognitive Toolkit) o Tensorflow ,siendo esta ultima la más usada para trabajar bajo keras y la que se usará en la realización de este proyecto.

La librería Sickit-learn es también importante dentro del machine Learning, ya que es una biblioteca de código tanto para aprendizaje supervisado como para el no supervisado y proporciona herramientas que permiten ajustar modelos, preprocesar datos y seleccionar y evaluar modelos entre otras muchas más utilidades. [40]

4.1.2. Google Colab

Google Colaboratory también conocido como Google Colab, es una plataforma gratuita creada por Google que permite la programación de un NoteBook de Jupyter así como su ejecución en un servidor en la nube asignado, sin necesidad de instalar nada, siendo solo necesario tener un navegador y una cuenta de Google.

Como se ha mencionado en apartados anteriores, el entrenamiento de redes neuronales implica una gran cantidad de tiempo ya que estas redes pueden llegar a tener miles de parámetros. Surge por tanto la necesidad de usar unidades de procesamiento grafico (GPU) que aceleren el proceso de entrenamiento. Google Colab permite cambiar el entorno de ejecución y establecer como acelerador de hardware la GPU durante 12 horas, siendo este uno de los principales motivos por los que se ha decidido hacer uso de esta plataforma [41].

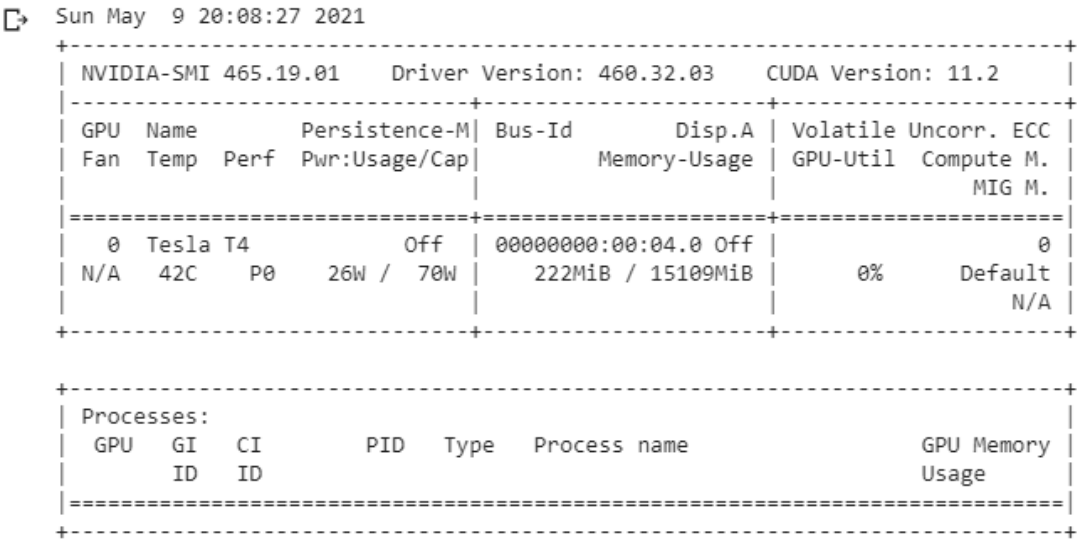


Figura 22. Especificaciones de la GPU asignada por Google Colab.

4.2. Dataset

Para la creación de algoritmos supervisados de Deep Learning es necesario tener una gran cantidad de datos etiquetados, los cuales se conocen con el nombre de dataset. Este conjunto de datos se utiliza para entrenar, validar y testear el modelo creado. La creación de los datasets puede resultar una tarea difícil ya que no sólo se necesita obtener una gran cantidad de datos, sino que también hay que etiquetarlos. Por ello, para la elaboración del trabajo se ha elegido un dataset ya existente, en concreto el GTSRB (German Traffic Sign Recognition Benchmark) de la plataforma Kaggle disponible en la referencia [42].

Kaggle es una comunidad en línea para el aprendizaje de ciencia de datos y aprendizaje automático. Dicha plataforma, permite tanto la creación como la búsqueda de conjunto de datos, así como la participación en competiciones que publican.

El dataset con el que se va a trabajar está formado por más de 50.000 imágenes pertenecientes a 43 clases diferentes. Estas imágenes son en color, tienen formato .png y se encuentran distribuidas en tres carpetas diferentes: meta, train, test. La primera carpeta, meta está formada por 43 imágenes, una imagen por cada clase, cada una de las cuales se observa en la Figura 23 junto al número y nombre de la clase perteneciente.

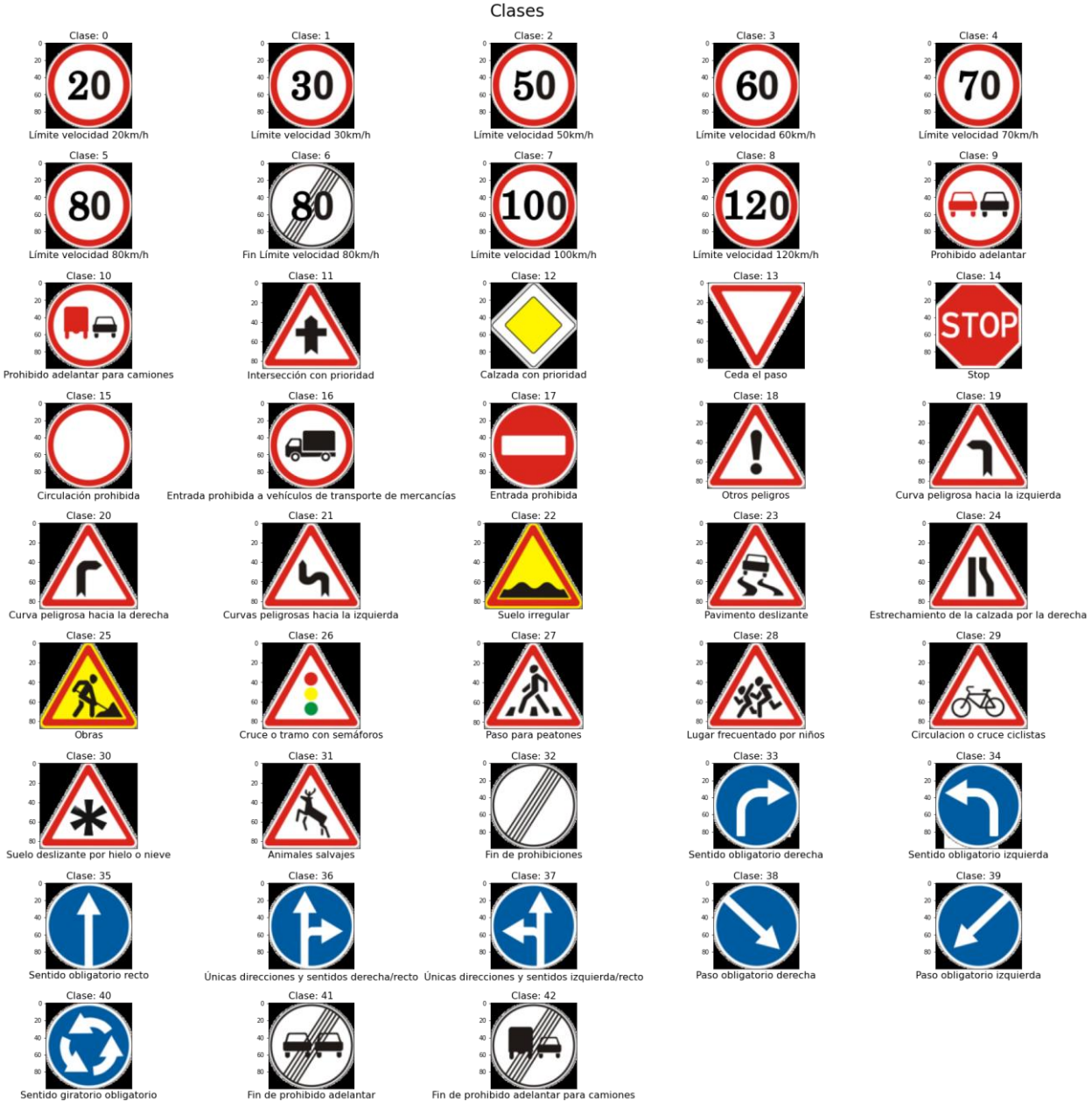


Figura 23. Clases de las señales del Dataset.

Por otro lado, la carpeta test está compuesta por 12630 imágenes de diversas clases y la carpeta contiene a su vez 43 carpetas enumeradas del 0 al 42, de forma que cada carpeta contiene un numero aleatorio de imágenes de una clase, las correspondiente al número de la carpeta.

Adicionalmente se pueden encontrar 3 ficheros de extensión .csv (delimitado por comas) que contienen información de las imágenes: el ancho, alto, las coordenadas del cuadro delimitador de la señal dividida en coordenada x de la esquina superior izquierda, coordenada y de la esquina superior izquierda, coordenada x de la esquina inferior derecha y coordenada y de la esquina inferior derecha, la clase a la que pertenecen y la ruta. Cada uno de estos ficheros contiene información acerca de las imágenes contenidas en las distintas carpetas, meta, train y test. El archivo meta.csv original ha sido modificado ligeramente de forma que sus clases estén enumeradas de forma ascendente. Esta modificación se encuentra en el fichero meta2.sv ya que se ha mantenido el fichero original por si fuera necesario.

4.3. Creación clasificador señales de tráfico

4.3.1. Punto de partida

Como se mencionó en un apartado anterior, se usará la plataforma Google Colab para la creación de la red neuronal convolucional. Para ello, se crea un nuevo cuaderno en el que se escribirá el código necesario, el cual se conecta de forma automática al servidor. Hay que cambiar el entorno de ejecución ya que el establecido es CPU y se quiere usar la GPU. Una vez cambiado, se puede comprobar el uso de la GPU con el Código 1. Si al ejecutarlo da error, quiere decir que no se ha conectado a la GPU, en caso contrario, si se ha conectado de forma correcta e imprimirá por pantalla la GPU encontrada.

```
#Comprobación uso GPU
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU no encontrada')
print('Encontrada GPU: {}'.format(device_name))
```

Código 1. Comprobación uso GPU

Para poder trabajar con el Dataset mencionado en el apartado anterior, es necesario que dicho Dataset esté subido a Google Drive. Una vez en la nube es necesario enlazar el cuaderno creado con la cuenta de Google Drive para poder acceder al Dataset, esto se consigue con el Código 2. Al ejecutarlo se proporciona un enlace para poder obtener un código de autorización, el cual hay que introducir en el espacio destinado a ello.

```
from google.colab import drive
drive.mount('/content/drive')
```

Código 2. Enlazado Googe Colab y Google Drive

Además, resulta también necesario importar las librerías que se van a usar a lo largo del trabajo.

```
  #Importación librerías
  import os
  import cv2
  import keras
  import random
  import numpy as np
  import pandas as pd
  import tensorflow as tf
  import matplotlib.pyplot as plt
  from sklearn.model_selection import train_test_split
  from matplotlib.image import imread
  from keras.models import Sequential
  from keras.utils import to_categorical
  from keras.layers import Dense,Dropout,Flatten,Conv2D, MaxPooling2D,
BatchNormalization
  from keras.utils.vis_utils import plot_model
  from sklearn.metrics import confusion_matrix
  import seaborn as sns
  from sklearn.utils import class_weight
```

Código 3. Importación de librerías

4.3.2. Análisis de datos

Antes de empezar a crear la red, es necesario conocer los datos con los que se trabaja, así como la ruta de trabajo. En primer lugar, se leen los 3 archivos .csv con la función read_csv ( ) de pandas y una vez leídos los archivos se pueden explorar los datos de cada conjunto con la función head ( ), que muestra las 5 primeras filas por defecto. Se analizan también la longitud del conjunto de datos de entrenamiento y la del conjunto de test con la función shape ( ), ya que esta información se usará más tarde.

```
  #Rutas de trabajo
  base_path='/content/drive/MyDrive/TFG/PRUEBAS'

  #Lectura ficheros .csv
  data_meta = pd.read_csv(base_path + '/Meta2.csv')
  data_train = pd.read_csv(base_path + '/Train.csv')
  data_test = pd.read_csv(base_path + '/Test.csv')

  #Exploracion dataset
  print(data_meta.head(5))
  print(data_train.head(5))
  print(data_test.head(5))

  #Longitud
  long_test=str(data_test.shape[0])
  long_train=str(data_train.shape[0])
```

Código 4. Exploración de datos

Por otro lado, también resulta útil la visualización de la distribución de los datos, tanto del conjunto de entrenamiento como del conjunto test. Como puede observarse en la Figura 24, la distribución de los datos viene representada por un diagrama de barras descendente, en el que se ha representado de forma conjunta tanto los datos de entrenamiento (azul) como los de test (verde). Se puede observar que ninguno de los conjuntos está balanceado ya que hay clases que contienen más de 2000 imágenes como puede ser la clase 2 y otras en cambio no llegan a las 200 imágenes como le pasa a la clase 0.

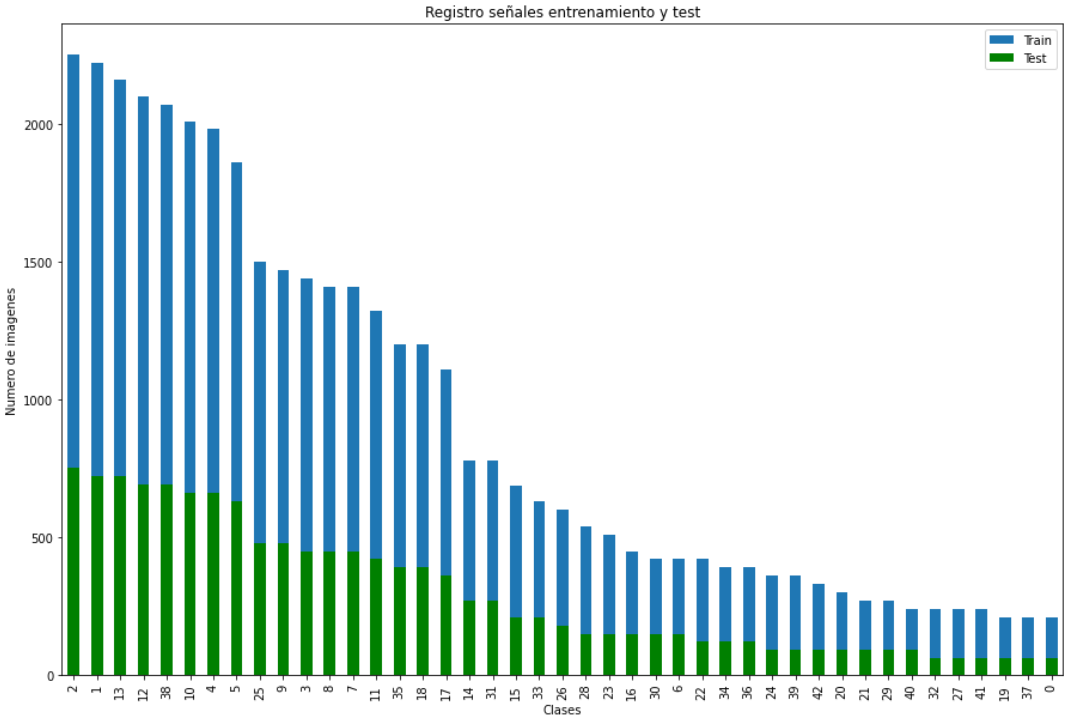


Figura 24. Diagrama de barras de la distribución de datos de entrenamiento y test

4.3.3. Conjunto de datos de entrenamiento y test

Para hacer el proceso más rápido se han guardado tanto las imágenes como las etiquetas de los conjuntos de entrenamiento y test como arrays de numpy en ficheros de tipo .npy. Además, las imágenes de ambos ficheros han sido reescaladas todas a un tamaño de 50x50 pixeles, ya que es el tamaño elegido para la entrada de la red. Para poder hacer uso de estos ficheros hay que importar cada uno de ellos con la función np.load ( ) y guardarlos en una variable.

```
  #Imagenes train
  train_images=np.load('/content/drive/MyDrive/TFG/PRUEBAS/train_img.np
y')
  train_labels=np.load('/content/drive/MyDrive/TFG/PRUEBAS/train_lbl.np
y')

  #Imagenes test
  test_img=np.load('/content/drive/MyDrive/TFG/PRUEBAS/test_img.npy')
  test_lbl=np.load('/content/drive/MyDrive/TFG/PRUEBAS/test_lbl.npy')
```

Código 5. Importación ficheros .npy

Para comprobar que los ficheros se han cargado correctamente se representan 25 imágenes de cada uno de los conjuntos, cada imagen se representa con su clase correspondiente. Para poder visualizar correctamente la imagen en color RGB, es necesario usar la función cv2.cvtColor ( ) que tendrá como parámetros la imagen que se quiere representar y la conversión del espacio de color. En este caso, la conversión del espacio de color será cv2.COLOR_BGR2RGB ya que cuando se lee la imagen con la función imread ( ) de OpenCV, la imagen se guarda en orden BGR en lugar de RGB. El código para representar 25 imágenes del conjunto de entramiento puede observarse en el Código 6 .

```
plt.figure(figsize=(15,15))
plt.suptitle("Algunas imagenes train dataset",y=0.93,fontsize=14)
for i in range(25):
    plt.subplot(5,5, i+1)
    plt.xticks([])
    j=random.randrange(0,int(long_train))
    plt.imshow(cv2.cvtColor(train_images[j], cv2.COLOR_BGR2RGB))
    plt.title("Clase: "+str(train_labels[j]),fontsize=10)
    plt.xlabel(str(classes_names[train_labels[j]]),fontsize=10)
    plt.subplots_adjust(left=0.125,
                        bottom=0.1,
                        right=0.9,
                        top=0.9,
                        wspace=0.2,
                        hspace=0.35)

plt.show()
```

Código 6. Representación de 25 imágenes del conjunto de entrenamiento

4.3.4. Separación imágenes entrenamiento y validación

Hasta ahora se ha trabajado con dos conjuntos de datos diferentes: el de entrenamiento formado por 39209 imágenes y el de test con 12630 imágenes pertenecientes a las 43 clases distintas. Como se comentó anteriormente, para evitar el overfitting se va a dividir el conjunto de entrenamiento en dos subconjuntos: el de entramiento y el de validación, en las proporciones 80%-20% respectivamente, es decir, ahora el conjunto de entrenamiento tendrá 31367 imágenes y el conjunto de validación estará formado por 7842 imágenes. Para llevar a cabo esta división se hace uso de la función train_test_split ( ) de sickit learn, cuyos parámetros son:

- Train_images: imágenes de entrenamiento iniciales
- Train_labels: etiquetas de cada una de las imágenes de entrenamiento.
- Test_size: valor flotante ente 0.0 y 1.0 que representa la proporción de datos que se incluirá en el conjunto de validación tras la división.
- Random_state: número entero que fija la semilla para generar números aleatorios.
- Stratify: esto se aplica pues como se vio en la gráfica de distribución de datos, las clases estaban desbalanceadas. Por ello, interesa que al hacer la división del conjunto de entrenamiento en dos subconjuntos estos mantengan la proporción de datos de cada clase.

```
train_img,val_img,train_lbl,val_lbl=
train_test_split(train_images,train_labels,test_size=0.2,
random_state=42,stratify=train_labels )
```

Código 7. División conjunto entramiento y validación.

4.3.5. Normalización y codificación One-Hot

Cuando se entrena la red hay que evitar que esta encuentre problemas numéricos, por ello es necesario normalizar los valores de los pixeles, es decir, que estén en el rango 0-1 en lugar del rango 0-255. Esto se consigue dividiendo cada uno de los pixeles de la imagen por 255. Como es un problema de clasificación, la categoría de la salida se suele codificar mediante una estrategia denominada "one-hot encoding", que codifica la etiqueta de salida en un vector cuya longitud será la del número de clases que haya, 43 en este caso concreto, dicho vector tiene un 1 en la posición de la clase correspondiente y 0 en el resto de las clases.

```
#Normalizacion 0-1
train_img = train_img / 255
val_img = val_img / 255
test_img=test_img/ 255

# One-hot encoding
train_lbl_one_hot = to_categorical(train_lbl)
val_lbl_one_hot = to_categorical(val_lbl)
test_lbl_one_hot = to_categorical(test_lbl)
```

Código 8. Normalización y codificación One-Hot

4.3.6. Creación del modelo

Para crear el modelo se usa la función Sequential ( ) de Keras, que como indica su nombre permite crear un modelo secuencial, es decir, las capas se apilan una sobre otra, de forma que la salida de una se conecte a la entrada de otra. Con el modelo sequential se pueden definir las distintas capas con el método add().

La entrada de la red serán las imágenes 50x50x3 cargadas con anterioridad, y estará compuesta por 4 capas convolucionales de 2 dimensiones. Cada capa convolucional tendrá un numero de filtros mayor que la anterior ya que las primeras capas detectarán formas más simples y las últimas, formas más complejas. La primera de ellas usará 16 filtros de tamaño 3x3 con función de activación relu, ya que como se vio en el capítulo anterior es una función de activación que funciona bien con imágenes debido a su rapidez computacional y al introducir no linealidad. La segunda capa convolucional será igual que la primera pero seguida de una capa maxpooling 2x2, que reducirá el tamaño a la mitad, 25x25, para evitar así el elevado número de parámetros. La tercera capa estará compuesta por 32 filtros de tamaño 3x3 y función de activación relu, seguida de una capa maxpooling 2x2. La última capa convolucional usará 64 filtros de tamaño 3x3 con la misma configuración que las anteriores. Todas las capas convolucionales tienen un parámetro padding con valor igual a 'same', que lo que hace es añadir filas y columnas de ceros de forma uniforme para que la salida tenga la misma dimensión que la entrada.

Una vez realizadas las convoluciones hay que conectar con una capa flatten, dos capas densamente conectadas de 128 y 64 neuronas respectivamente y por último con una capa de salida con 43 neuronas, una por cada clase que se quiere clasificar, con función de activación softmax.

```
print("----------------------------")
print("Creamos la red")
#Creamos la red
traffic_sign_model = Sequential()

traffic_sign_model.add(Conv2D(filters=16, kernel_size=(3,3), padding=
'same',activation='relu', input_shape=(width,height,channels))),
traffic_sign_model.add(Conv2D(filters=16, kernel_size=(3,3), padding=
'same',activation='relu')),
traffic_sign_model.add(MaxPooling2D(pool_size=(2,2)))
```

```
 traffic_sign_model.add(Conv2D(filters=32, kernel_size=(3,3), padding=
'same',activation='relu')),
 traffic_sign_model.add(MaxPooling2D(pool_size=(2,2)))

 traffic_sign_model.add(Conv2D(filters=64, kernel_size=(3,3), padding=
'same',activation='relu')),
 traffic_sign_model.add(MaxPooling2D(pool_size=(2,2)))

 traffic_sign_model.add(Flatten())
 traffic_sign_model.add(Dense(128,activation="relu"))
 traffic_sign_model.add(Dense(64,activation="relu"))
 traffic_sign_model.add(Dense(43,activation="softmax"))
```

Código 9. Creación de la red neuronal convolucional

Una vez creado el modelo, puede resultar útil obtener un resumen de dicho modelo. En Keras se obtiene mediante la función summary() y se trata de un resumen textual que incluye información sobre las capas y su orden, la forma de salida de cada una de las capas, el número total de parámetros en cada etapa y el número total de parámetros del modelo como puede observarse en la Figura 25.

```
 print("Resumen del modelo")
 traffic_sign_model.summary()
```

Código 10. Resumen del modelo.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 50, 50, 16)        448
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 50, 50, 16)        2320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 25, 25, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 128)               295040
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 43)                2795
=================================================================
Total params: 331,995
Trainable params: 331,995
Non-trainable params: 0
```

Figura 25. Resumen del modelo.

Cuando el modelo es simple el resumen es útil, en cambio, si el modelo tiene múltiples entradas o salidas este resumen puede confundir. Para modelos complejos, Keras proporciona la función plot_model() que proporciona un gráfico de la red como el que se puede observar en la Figura 26.

```
  plot_model(traffic_sign_model,to_file='CNN_plot.png',show_shapes=True
, show_layer_names=True)
```

Código 11. Gráfico de la red

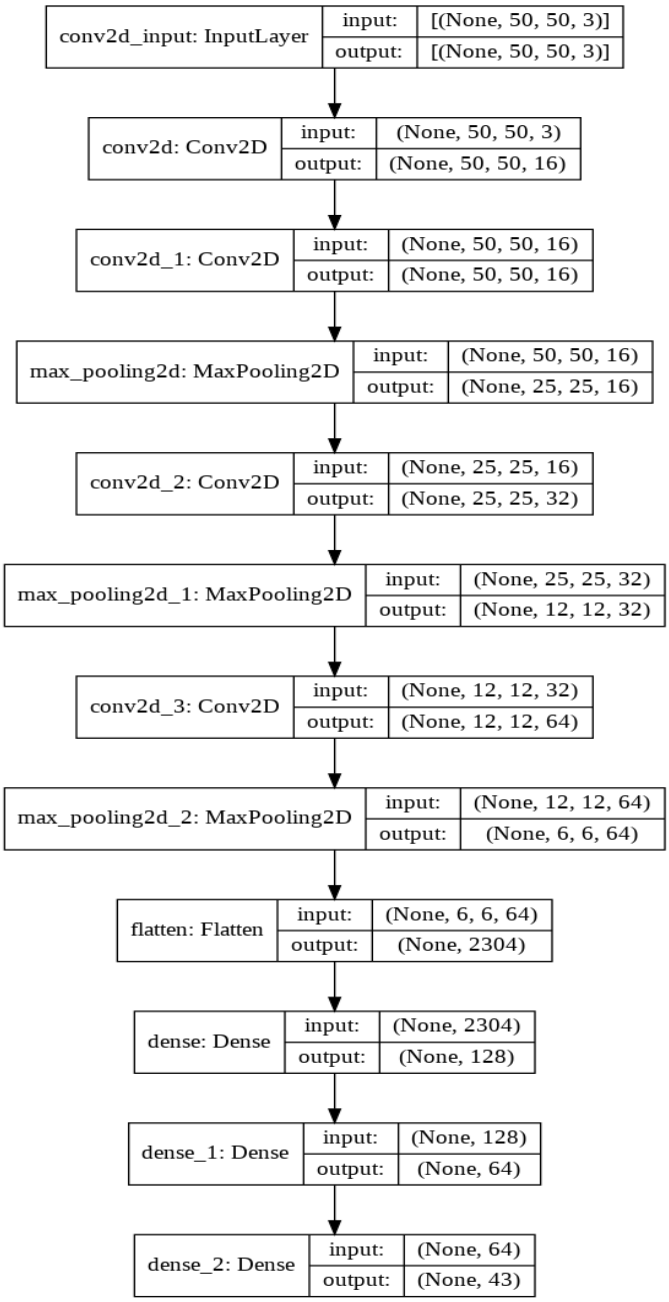


Figura 26. Gráfico de la red.

4.3.7. Compilación del modelo

Una vez creado el modelo es necesario compilarlo, configurando cómo será el proceso de aprendizaje a partir de una serie de argumentos del método compile ( ). El primer argumento es la función de pérdida, loss, que en este caso es la entropía cruzada categórica ya que es un problema de clasificación con más de dos clases, 43 exactamente.

Otro parámetro es el optimizador, que como se vio en la parte teórica, es el encargado de actualizar los pesos y hay varios tipos de optimizadores. De momento no se va a establecer ningún optimizador, ya que será en el próximo capítulo en el que se harán diversos experimentos con los diferentes optimizadores y se evaluará cuál de los optimizadores ofrece un rendimiento mejor. Por último, la métrica elegida para evaluar cuán bueno es el modelo es la accuracy.

```
print("Compilamos la red")
traffic_sign_model.compile(loss='categorical_crossentropy',
optimizer=keras.optimizers.Optimizador(learning_rate=0.001),
metrics=['accuracy'])
```

Código 12. Compilación del modelo

4.3.8. Cálculo de los pesos de clase

En el diagrama de barras representando en el apartado 4.3.2, se vio que los datos no estaban equilibrados, lo cual afecta al algoritmo en el proceso de generalización de la información y perjudicando a las clases minoritarias.Para ello, se van a calcular las ponderaciones de clase con el método compute_class_weight() que se usarán posteriormente para ponderar la función de pérdida durante la fase de entrenamiento.

```
  class_weights = class_weight.compute_class_weight('balanced',
                  np.unique(np.array(data_train['ClassId'])),
                  np.array(data_train['ClassId']))
  print(class_weights)
  class_weights = dict(enumerate(class_weights))
```

Código 13. Cálculo de los pesos

4.3.9. Entrenamiento del modelo

Con los datos preparados y la red creada se puede proceder al entrenamiento de la red con el método fit (), que recibe como parámetros las imágenes de entramiento y sus respectivas etiquetas, el batch size, el número de épocas y los datos de validación, tanto las imágenes como las etiquetas.

```
  print("Entrenamos")
  history = traffic_sign_model.fit(train_img, train_lbl_one_hot,class_w
eight=class_weights, batch_size=32, epochs=50,validation_data=(val_img
,val_lbl_one_hot))
```

Código 14. Entramiento de la red

5 EXPERIMENTOS Y RESULTADOS

Locura es hacer lo mismo una y otra vez esperando obtener resultados diferentes.

- Albert Einstein -

En este capítulo, se realizarán diversos experimentos a partir de la red creada anteriormente y se mostrarán los resultados obtenidos por cada uno de ellos. Dichos experimentos consistirán en el uso de diferentes optimizadores y técnicas para evitar el sobreajuste del modelo. Para conocer la bondad de cada experimento se usarán las métricas de evaluación comentadas en el Capítulo 3 y se compararán para obtener el mejor modelo.

5.1. Experimentos

5.1.1 Experimento 1

El primer experimento consistirá en usar el optimizador Adam para compilar el modelo. Se le pasará como argumento el learning rate con valor 0.001, que es el valor predeterminado.

```
print("Compilamos la red")
traffic_sign_model.compile(loss='categorical_crossentropy',
optimizer=keras.optimizers.Adam(learning_rate=0.001),
metrics=['accuracy'])
```

Código 15. Compilación del modelo con el optimizador Adam.

5.1.2 Experimento 2

En este segundo experimento, se usará el optimizador SGD en lugar de Adam para la compilación del modelo. Como argumento se le pasará el learning rate con un valor predeterminado de 0.01.

```
print("Compilamos la red")
traffic_sign_model.compile(loss='categorical_crossentropy',
optimizer=keras.optimizers.SGD(learning_rate=0.01),
metrics=['accuracy'])
```

Código 16. Compilación del modelo con el optimizador SGD.

5.1.3 Experimento 3

Para este tercer experimento, el optimizador elegido será el RMSprop con un learning rate de 0.001.

```
print("Compilamos la red")
traffic_sign_model.compile(loss='categorical_crossentropy',
optimizer=keras.optimizers.RMSprop(learning_rate=0.001),
metrics=['accuracy'])
```

Código 17. Compilación del modelo con el optimizador RMSprop.

5.1.4 Experimento 4

Con el objetivo de reducir el sobreajuste del modelo inicial, en este experimento se añadirán las capas necesarias para ello ya mencionadas anteriormente, es decir, capas Dropout con probabilidades del 25% y 50% así como capas BatchNormalization cuya distribución se muestra en el Código 18. Para la compilación de este experimento se utilizará el optimizador Adam con un learning rate de 0.001 como en el experimento 1.

```
  print("----------------------------")
  print("Creamos la red")
  #Creamos la red
  traffic_sign_model = Sequential()

  traffic_sign_model.add(Conv2D(filters=16, kernel_size=(3,3), padding=
'same',activation='relu', input_shape=(width,height,channels))),
  traffic_sign_model.add(BatchNormalization())
  traffic_sign_model.add(Conv2D(filters=16, kernel_size=(3,3), padding=
'same',activation='relu')),
  traffic_sign_model.add(BatchNormalization())
  traffic_sign_model.add(MaxPooling2D(pool_size=(2,2)))

  traffic_sign_model.add(Conv2D(filters=32, kernel_size=(3,3), padding=
'same',activation='relu')),
  traffic_sign_model.add(BatchNormalization())
  traffic_sign_model.add(MaxPooling2D(pool_size=(2,2)))
  traffic_sign_model.add(Dropout(rate=0.25))

  traffic_sign_model.add(Conv2D(filters=64, kernel_size=(3,3), padding=
'same',activation='relu')),
  traffic_sign_model.add(BatchNormalization())
  traffic_sign_model.add(MaxPooling2D(pool_size=(2,2)))
  traffic_sign_model.add(Dropout(rate=0.25))

  traffic_sign_model.add(Flatten())
  traffic_sign_model.add(Dense(128,activation="relu"))
  traffic_sign_model.add(BatchNormalization())
  traffic_sign_model.add(Dropout(rate=0.5))
  traffic_sign_model.add(Dense(64,activation="relu"))
  traffic_sign_model.add(BatchNormalization())
  traffic_sign_model.add(Dropout(rate=0.5))
  traffic_sign_model.add(Dense(43,activation="softmax"))
```

Código 18.Modelo del experimento 4

5.2. Resultados

Cuando se entrena la red, se imprime por pantalla la evolución del entrenamiento como se observa en la Figura 27, en concreto, el tiempo que tarda cada época, así como el valor de 4 métricas para cada época: loss, accuracy, val_loss y val_accuracy, donde loss y acuraccy son la perdida y precisión del conjunto de entrenamiento respectivamente y val_loss y val_accuracy son la perdida y precisión del conjunto de validación. Esta evolución del comportamiento del modelo puede visualizarse en la gráfica Acuraccy y la gráfica Loss a partir de los datos del entrenamiento. Por tanto, la gráfica Loss representará por un lado la métrica loss (azul) y por otro, la métrica val_loss (naranja) y la gráfica accuracy representará en azul la métrica accuracy y en naranja la métrica val_accuracy.

```
#GRAFICAS LOSS Y ACCURACY
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()

plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

Código 19. Gráficas Loss y Accuracy.

```
Epoch 1/50
981/981 [==============================] - 50s 7ms/step - loss: 2.3392 - accuracy: 0.3465 - val_loss: 0.2668 - val_accuracy: 0.9178
Epoch 2/50
981/981 [==============================] - 6s 6ms/step - loss: 0.2034 - accuracy: 0.9406 - val_loss: 0.0870 - val_accuracy: 0.9778
Epoch 3/50
981/981 [==============================] - 6s 6ms/step - loss: 0.0699 - accuracy: 0.9796 - val_loss: 0.0821 - val_accuracy: 0.9779
Epoch 4/50
981/981 [==============================] - 6s 6ms/step - loss: 0.0574 - accuracy: 0.9831 - val_loss: 0.0845 - val_accuracy: 0.9777
Epoch 5/50
981/981 [==============================] - 6s 6ms/step - loss: 0.0308 - accuracy: 0.9902 - val_loss: 0.0589 - val_accuracy: 0.9837
```

Figura 27. Evolución del entrenamiento de la red.

Una vez entrenados los modelos de cada uno de los experimentos realizados, se evaluarán dichos modelos a partir de conjunto de test, el cual no ha sido "visto" aún por el modelo.

```
  #EVALUCACION TEST
  test_eval = traffic_sign_model.evaluate(test_img, test_lbl_one_hot,
verbose=1)
  print('Test loss:', test_eval[0])
  print('Test accuracy:', test_eval[1])
```

Código 20. Evaluación del conjunto test.

Como se vio en el capítulo anterior, los datos no están balanceados. Por ello será útil calcular otras métricas ya comentadas en el Capítulo 3 además de la ya calculada (accuracy). Para ello se usaran las métricas de sickit-learn classification_report() que genera un informe de texto con las métricas principales: precision, recall y F1-Score, y confusion_matrix() que genera la matriz de confusión.

```
  print(classification_report(test_lbl, prediction))
```

Código 21. Classification report.

```
  cf = confusion_matrix(test_lbl, prediction)
  df_cm = pd.DataFrame(cf, index =  unique_labels, columns= unique_labe
ls)
  plt.figure(figsize =(100,100))
  plt.title("Confusion Matrix", fontsize=100)
  sns.heatmap(df_cm,cmap=plt.cm.Blues, annot=True)
```

Código 22. Matriz de confusión.

5.2.1. Experimento 1

Para este primer experimento se obtiene un valor de accuracy de 1 en el conjunto de entrenamiento y de 0.99 en el conjunto de validación. En la Figura 28 puede observarse como a partir de la época 20 los valores de la accuracy alcanzan un valor estable tanto en el entrenamiento como en la validación.

En cuanto a la función de pérdidas, se observa como en el entrenamiento disminuyen a medida que aumentan las épocas, en cambio, durante la validación aparecen algunos picos y las perdidas empiezan a aumentar separándose de la perdida de entrenamiento., lo cual indica que hay overfitting.

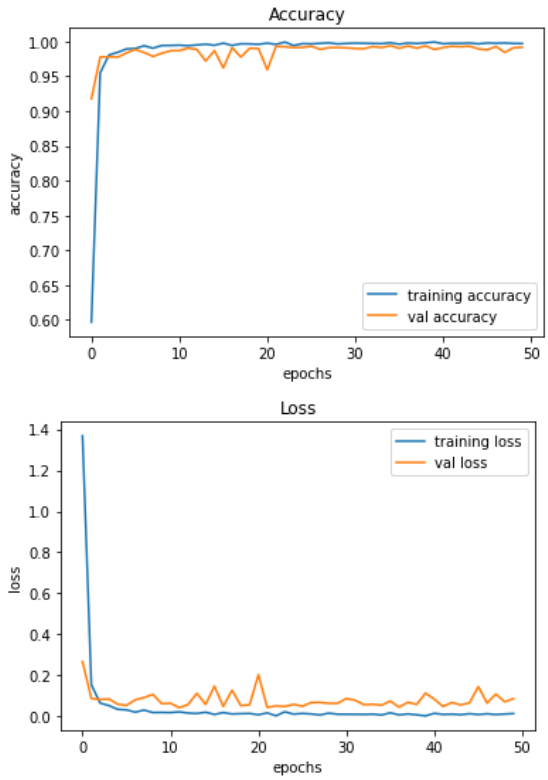


Figura 28. Evolución de la accuracy y de la función de pérdidas del experimento 1

Cuando se le pasa el conjunto de test al modelo se obtiene una accuracy del 0.968, que es menor que la obtenida durante el entrenamiento y la validación. En la Figura 29 se muestra la matriz de confusión de este primer experimento. La mayor parte de los valores se encuentran en la diagonal, lo cual indica que la predicción que se está realizando es acertada. Por otro lado, puede observarse que la mayor parte de los valores que no pertenecen a la diagonal principal son iguales a 0, lo cual indica que el modelo realiza una predicción correcta. Sin embargo, se pueden encontrar algunos valores distintos de cero, que indican que el modelo confunde algunas clases. Como ejemplo de ello, se observa que hay 29 imágenes de la clase 27 que se clasifican como clase 1, o 16 imágenes de la clase 21 que se confunden con la clase 12.

Si se analiza la puntuación F1 obtenida en la Figura 30, se observa que se obtienen valores altos para la mayoría de las clases, a excepción de la clase 27, para la cual se obtiene un valor de 0.55.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
0	57	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	710	4	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	3	740	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	430	0	11	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
4	1	9	5	1	630	4	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	8	4	0	610	0	3	1	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	6	0	8	120	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	3	0	0	0	3
7	0	3	0	0	1	4	0	440	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	2	0	9	0	2	0	1	430	2	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	480	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	1	0	4	0	0	1	1	650	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	1	0	0	0	0	420	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	1	0	0	0	0	0	0	0	0	0	680	3	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1	1	720	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	270	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	210	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	360	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	1	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	360	0	2	0	0	0	2	0	3	17	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	4	0	0	0	0	0	0	0	0	0	1	16	0	0	0	0	0	0	0	0	61	0	1	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	120	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1	0	0	0	0	470	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	8	0	0	10	160	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	1	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	140	0	5	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	2	0	0	2	0	0	0	0	1	2	0	0	0	0	0	0	0	0	2	0	0	14	0	1	0	0	0	10	110	3	1	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	5	0	0	0	0	0	1	0	260	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	210	0	0	1	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	1	0	0	0	0
35	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	390	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0
38	0	0	1	0	1	0	0	0	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	6	0	680	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	4	0	1	0	0	0	83	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	88	0	0
41	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	58	0
42	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	87

Figura 29. Matriz de confusión del experimento 1

	precision	recall	f1-score	support
0	0.97	0.95	0.96	60
1	0.93	0.99	0.96	720
2	0.97	0.99	0.98	750
3	0.95	0.96	0.96	450
4	0.99	0.96	0.98	660
5	0.94	0.97	0.96	630
6	0.99	0.81	0.89	150
7	0.98	0.97	0.97	450
8	0.97	0.96	0.96	450
9	0.99	1.00	0.99	480
10	0.99	0.99	0.99	660
11	0.97	0.99	0.98	420
12	0.97	0.98	0.98	690
13	0.99	1.00	0.99	720
14	1.00	1.00	1.00	270
15	0.96	1.00	0.98	210
16	1.00	0.99	1.00	150
17	0.99	0.99	0.99	360
18	0.99	0.91	0.95	390
19	1.00	1.00	1.00	60
20	0.84	1.00	0.91	90
21	1.00	0.68	0.81	90
22	0.94	0.97	0.96	120
23	0.89	0.99	0.94	150
24	0.98	0.94	0.96	90
25	0.97	0.98	0.98	480
26	0.98	0.87	0.92	180
27	0.60	0.50	0.55	60
28	0.99	0.96	0.97	150
29	0.87	1.00	0.93	90
30	0.94	0.75	0.83	150
31	0.93	0.97	0.95	270
32	0.86	1.00	0.92	60
33	0.98	1.00	0.99	210
34	0.99	0.98	0.99	120
35	1.00	0.99	0.99	390
36	0.93	1.00	0.96	120
37	0.98	1.00	0.99	60
38	0.99	0.98	0.99	690
39	1.00	0.92	0.96	90
40	0.97	0.98	0.97	90
41	1.00	0.97	0.98	60
42	0.96	0.97	0.96	90
accuracy			0.97	12630
macro avg	0.96	0.95	0.95	12630
weighted avg	0.97	0.97	0.97	12630

Figura 30. Métricas conjunto test experimento 1

5.2.2. Experimento 2

Los valores de accuracy obtenidos para el entrenamiento y la validación en este segundo experimento son 0.998 y 0.994 respectivamente. En la Figura 31, se observa que los valores estables de accuracy en ambos conjuntos se logran con pocas épocas.

En cuanto a la función de perdidas, igual que en el experimento anterior, durante el entrenamiento disminuye el valor con las épocas, en cambio, durante la validación se producen picos y se separa de los valores de entrenamiento, dando lugar a un overfitting mayor que el del experimento anterior.

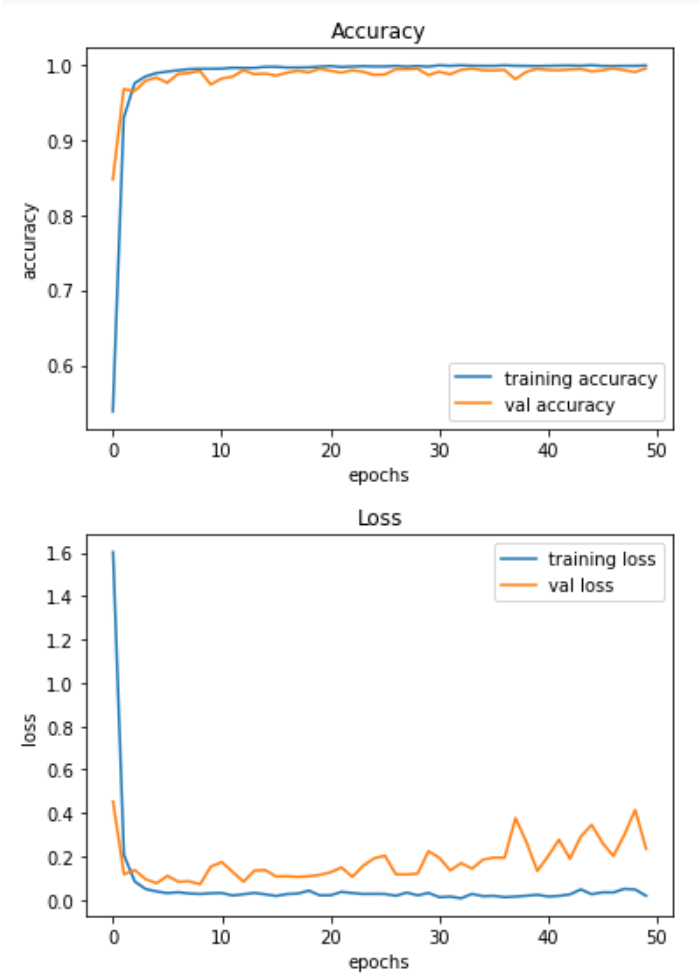


Figura 31. Evolución de la accuracy y de la función de pérdidas del experimento 2

Para el conjunto de test se obtiene una accuracy del 0.957, menor que la obtenida en el entrenamiento y validación y también menor que la obtenida en el primer experimento realizado.

La matriz de confusión de este segundo experimento se puede observar en la Figura 32. Al igual que en el experimento anterior, la mayor parte de los valores se encuentran en la diagonal principal, indicando que la predicción que se está realizando es correcta. Sin embargo, también se encuentran valores diferentes de cero en el resto de la matriz, por ejemplo, se observa que hay 15 y 13 imágenes de la clase 21 que se clasifican como clase 11 y 18 respectivamente, o 45 imágenes de la clase 30 que se confunden con la clase 23.

Si se analiza la puntuación F1 obtenida, se observa que se obtienen valores altos para la mayoría de las clases, siendo las clases 21 y 30 las que obtienen menores valores, 0.74 y 0.71 respectivamente.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
0	59	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	710	2	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	4	740	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	430	0	20	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	4	4	0	640	1	0	2	0	1	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	8	8	0	0	620	0	4	0	1	4	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	1	0	27	120	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	4
7	0	1	0	0	0	6	0	430	2	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	3	0	12	0	2	430	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	1	0	0	0	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	8	0	0	2	0	640	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	0	0	0	1	0	410	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	680	1	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	3	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	720	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
14	0	1	0	6	0	0	0	0	0	0	0	0	0	1	260	0	0	4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	210	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	2	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	360	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	3	0	0	6	0	0	0	0	0	0	1	0	0	0	0	0	0	350	1	6	0	1	1	2	1	3	14	0	0	1	3	0	0	0	0	0	0	1	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	89	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	2	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	13	0	0	57	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	110	0	0	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	6	0	0	2	0	0	68	0	0	5	0	2	1	1	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	1	0	0	0	0	0	0	8	0	0	0	1	0	0	0	0	0	3	0	0	0	440	11	5	0	8	0	7	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	25	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	140	3	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	88	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	1	0	0	0	0	0	0	0	0	0	4	1	0	0	0	0	0	0	0	1	2	0	45	0	0	0	0	1	7	88	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	3	0	0	5	0	0	0	0	0	0	0	260	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	210	0	0	1	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	1	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	390	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	4	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0
38	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	2	680	0	4	0	0
39	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	5	0	73	0	0	0
40	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	82	0	4
41	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	0
42	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	84

Figura 32. Matriz de confusión del experimento 2

	precision	recall	f1-score	support
0	1.00	0.98	0.99	60
1	0.96	0.99	0.97	720
2	0.98	0.99	0.99	750
3	0.97	0.95	0.96	450
4	0.98	0.98	0.98	660
5	0.89	0.96	0.92	630
6	1.00	0.77	0.87	150
7	0.98	0.96	0.97	450
8	0.99	0.96	0.97	450
9	0.96	1.00	0.98	480
10	0.99	0.97	0.98	660
11	0.92	0.98	0.95	420
12	1.00	0.98	0.99	690
13	0.99	0.99	0.99	720
14	1.00	0.95	0.97	270
15	0.95	1.00	0.97	210
16	0.96	0.99	0.97	150
17	0.98	0.99	0.98	360
18	0.93	0.89	0.91	390
19	0.95	0.98	0.97	60
20	0.84	0.99	0.91	90
21	0.89	0.63	0.74	90
22	0.98	0.94	0.96	120
23	0.72	1.00	0.84	150
24	0.97	0.76	0.85	90
25	0.94	0.91	0.92	480
26	0.90	0.84	0.87	180
27	0.69	1.00	0.82	60
28	0.99	0.97	0.98	150
29	0.79	0.98	0.88	90
30	0.90	0.59	0.71	150
31	0.95	0.97	0.96	270
32	0.98	1.00	0.99	60
33	0.95	1.00	0.97	210
34	0.96	0.99	0.98	120
35	1.00	1.00	1.00	390
36	0.99	0.96	0.97	120
37	0.88	1.00	0.94	60
38	0.99	0.99	0.99	690
39	1.00	0.81	0.90	90
40	0.94	0.91	0.93	90
41	0.88	0.77	0.82	60
42	0.91	0.93	0.92	90
accuracy			0.96	12630
macro avg	0.94	0.93	0.93	12630
weighted avg	0.96	0.96	0.96	12630

Figura 33. Métricas conjunto test experimento 2

5.2.3. Experimento 3

Los valores de accuracy obtenidos en este tercer experimento son 1 durante el entrenamiento y 0.986 en la validación. Se observa en la Figura 34 que los valores estables de la accuracy se logran en pocas épocas a excepción de un pico en torno a la época 32 durante el entrenamiento.

En cuanto a la función de perdidas, durante el entramiento decrece con las épocas exceptuando el pico coincidente con el pico de la accuracy en la época 32. En la validación, las pérdidas empiezan disminuyendo durante las primeras épocas, para después empezar a crecer en las últimas épocas, siendo este aumento muy pequeño y manteniéndose cercano al valor de las pérdidas de entrenamiento. Por lo tanto, el overfitting existente es menor que el obtenido en los experimentos anteriores.

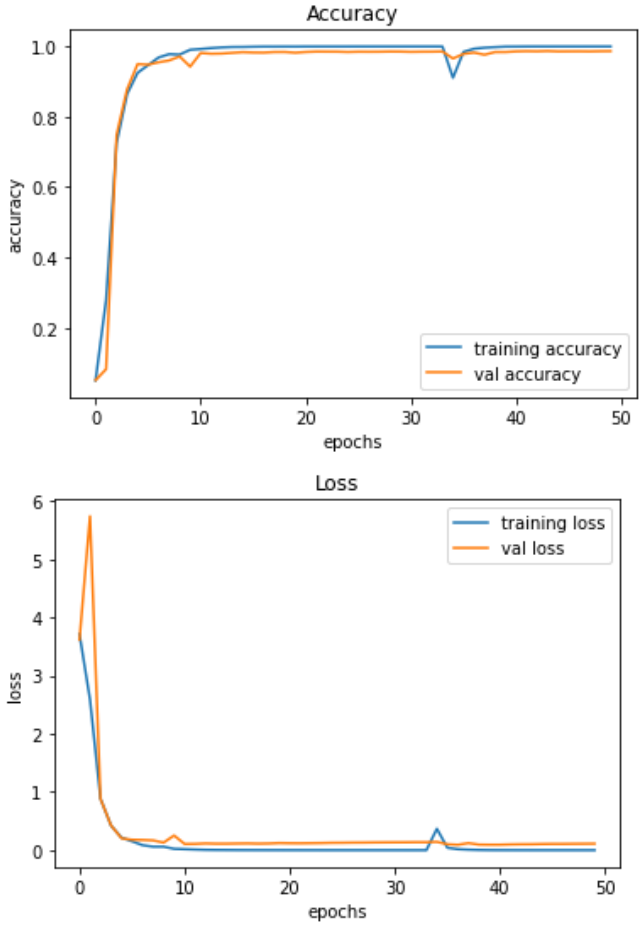


Figura 34. Evolución de la accuracy y de la función de pérdidas del experimento 3

Para el conjunto de test se obtiene un valor de accuracy del 0.906, menor que la obtenida en el entrenamiento y validación y bastante menor que la obtenidas en los dos experimentos ya realizados.

En la Figura 35 se muestra la matriz de confusión del tercer experimento. Al igual que en los experimentos anteriores, la mayor parte de los valores se encuentran en la diagonal principal. En este caso, se observan más valores diferentes de cero en el resto de la matriz que en los casos anteriores, como, por ejemplo, las 28 imágenes de la clase 7 que se clasifican como clase 5 o las 37 imágenes de la clase 25 que se confunden con la clase 11.

Si se analiza la puntuación F1 generada, se observa que los valores obtenidos son menores que los de los experimentos anteriores, destacando los de la clase 0 y 27 que poseen un valor de 0.59 y 0.54 respectivamente.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
0	37	8	0	0	12	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	700	11	0	2	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0
2	0	26	700	6	4	8	0	0	0	3	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4	0	0
3	0	0	0	420	0	13	0	1	1	0	4	0	1	2	0	3	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
4	2	22	9	1	600	11	0	3	4	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0
5	0	20	16	3	1	570	0	5	6	3	2	0	1	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
6	0	1	0	16	0	7	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1
7	0	1	0	13	1	28	0	380	15	0	1	0	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	2	0	0	0	0
8	0	0	2	16	1	8	0	0	420	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	0	0	0	22	0	1	0	0	1	430	22	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	1	2	0	2	10	0	6	4	0	630	0	0	0	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	390	1	0	0	0	0	0	1	0	2	0	0	0	0	0	7	0	1	0	19	0	0	0	0	0	0	0	0	0	0	0	0
12	0	2	0	0	0	0	0	0	0	0	0	1	680	6	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0	0
13	0	1	0	0	0	0	0	0	0	0	0	1	2	710	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0
14	0	0	0	2	3	0	0	0	0	0	0	0	0	0	250	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	1	0	0	0
15	0	1	0	0	2	0	0	0	0	0	0	0	1	0	0	210	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	24	2	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	320	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0
18	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	310	4	3	2	0	6	4	3	6	14	7	0	6	9	0	6	0	0	0	0	1	0	3	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	87	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	20	2	0	0	0	0	0	0	8	0	52	0	0	0	0	0	0	0	0	2	6	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	110	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	16	0	0	0	110	0	0	0	0	8	2	9	0	0	0	0	0	0	0	0	0	0	0	0
24	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	3	0	0	1	0	2	48	2	6	5	3	15	1	0	0	0	0	0	0	0	1	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	6	1	0	0	0	4	2	420	0	2	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0
26	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	3	150	0	1	0	0	0	0	0	0	0	0	4	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	14	1	30	0	0	0	0	0	0	0	0	9	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	3	0	0	3	0	0	0	0	130	8	6	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	0	0	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	1	0	0	2	1	0	0	0	0	3	5	1	0	1	0	0	0	0	6	6	0	0	8	3	1	4	0	2	0	100	2	0	0	0	0	0	0	0	2	0	0	0
31	0	0	0	0	0	2	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1	0	1	0	0	0	1	0	260	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	210	0	0	0	0	0	0	0	0	0
34	0	1	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	1	0	0	0	0	0	0	0
35	0	0	2	7	1	0	0	0	0	10	5	0	0	6	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	350	0	0	0	0	0	0	0
36	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	110	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	58	0	1	0	0	0
38	1	5	1	2	6	1	0	0	0	0	1	0	1	5	4	8	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	1	0	3	0	0	0	630	0	0	0	0
39	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	83	0	0	0
40	0	5	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0	0	0	0	1	0	0	65	0	0
41	0	0	0	18	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	1
42	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	83

Figura 35. Matriz de confusión del experimento 3

	precision	recall	f1-score	support
0	0.57	0.62	0.59	60
1	0.88	0.97	0.92	720
2	0.94	0.93	0.93	750
3	0.79	0.94	0.86	450
4	0.92	0.91	0.92	660
5	0.86	0.90	0.88	630
6	1.00	0.82	0.90	150
7	0.96	0.85	0.90	450
8	0.92	0.94	0.93	450
9	0.96	0.90	0.93	480
10	0.94	0.95	0.95	660
11	0.85	0.93	0.89	420
12	0.96	0.98	0.97	690
13	0.97	0.99	0.98	720
14	0.96	0.93	0.94	270
15	0.92	0.98	0.95	210
16	0.94	0.98	0.96	150
17	0.99	0.90	0.94	360
18	0.92	0.81	0.86	390
19	0.63	1.00	0.77	60
20	0.84	0.97	0.90	90
21	0.93	0.58	0.71	90
22	0.94	0.88	0.91	120
23	0.82	0.76	0.79	150
24	0.80	0.53	0.64	90
25	0.94	0.89	0.91	480
26	0.72	0.84	0.77	180
27	0.59	0.50	0.54	60
28	0.85	0.86	0.86	150
29	0.76	0.93	0.84	90
30	0.69	0.68	0.68	150
31	0.94	0.97	0.95	270
32	0.87	1.00	0.93	60
33	0.95	0.99	0.97	210
34	0.92	0.96	0.94	120
35	0.99	0.91	0.95	390
36	0.99	0.93	0.96	120
37	0.85	0.97	0.91	60
38	0.96	0.91	0.93	690
39	0.95	0.92	0.94	90
40	0.83	0.72	0.77	90
41	0.86	0.63	0.73	60
42	0.98	0.92	0.95	90
accuracy			0.91	12630
macro avg	0.88	0.87	0.87	12630
weighted avg	0.91	0.91	0.91	12630

Figura 36. Métricas conjunto test experimento 3

5.2.4. Experimento 4

Los valores de accuracy obtenidos para el entrenamiento y la validación en este cuarto y último experimento son 0.984 y 0.998 respectivamente. Se observa que los valores estables de accuracy se logran con pocas épocas, en torno a las 14.

En cuanto a la función de pérdidas, tanto durante el entramiento como durante la validación, los valores disminuyen con las épocas, estando muy próximos entre ellos y siendo las pérdidas obtenidas durante la validación menores que las obtenidas en el entrenamiento.

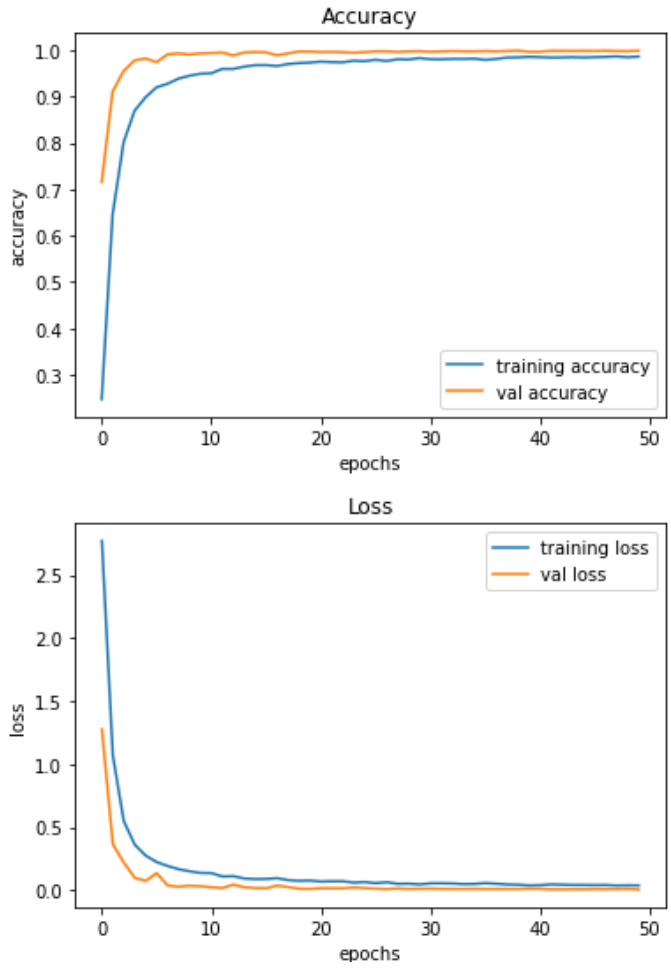


Figura 37. Evolución de la accuracy y de la función de pérdidas del experimento 4

Para el conjunto de test el valor de accuracy obtenido es del 0.987, el cual es mayor que el obtenido en los experimentos ya realizados.

En la Figura 38 se muestra la matriz de confusión del presente experimento. Al igual que en los experimentos anteriores, la mayor parte de los valores se encuentran en la diagonal principal. En este caso, se observan menos valores diferentes de cero en el resto de la matriz que en los casos anteriores. Además, dichos valores son menores que en los anteriores experimentos, como se puede observar en la cantidad de imágenes de la clase 5 que se clasifican como clase 2 o de la clase 6 que se confunden con la clase 34, siendo dicha cantidad 4.

Si se analiza la puntuación F1 generada, se observa que los valores obtenidos son mayores que en los experimentos previos, siendo para muchas clases 1 o muy cercano a 1. Por otra parte, el menor valor obtenido es de 0.85 para la clase 41, lo que indica que el modelo tiene un funcionamiento adecuado.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	710	1	0	0	4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	750	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	440	0	5	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	0	660	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	4	1	0	620	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	140	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	2	0	450	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	0	3	0	0	450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	480	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0	0	660	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	420	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	670	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	2	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	720	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	270	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	210	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	360	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	380	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	89	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	140	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	89	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	1	460	0	3	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	170	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	88	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	2	2	0	0	0	140	3	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	270	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	210	0	0	0	0	0	1	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	1	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	390	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0
38	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	680	0	2	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0
40	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	87	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	70

Figura 38. Matriz de confusión del experimento 4

	precision	recall	f1-score	support
0	0.98	1.00	0.99	60
1	1.00	0.99	1.00	720
2	0.99	1.00	0.99	750
3	0.99	0.98	0.99	450
4	1.00	1.00	1.00	660
5	0.97	0.99	0.98	630
6	0.99	0.94	0.97	150
7	0.99	0.99	0.99	450
8	1.00	0.99	0.99	450
9	1.00	0.99	1.00	480
10	1.00	0.99	0.99	660
11	0.99	1.00	0.99	420
12	1.00	0.98	0.99	690
13	0.99	1.00	0.99	720
14	1.00	1.00	1.00	270
15	0.95	1.00	0.97	210
16	0.99	1.00	1.00	150
17	1.00	0.99	1.00	360
18	0.99	0.99	0.99	390
19	0.92	1.00	0.96	60
20	0.97	1.00	0.98	90
21	0.99	0.99	0.99	90
22	0.99	0.98	0.99	120
23	0.97	0.97	0.97	150
24	0.99	0.99	0.99	90
25	0.99	0.96	0.98	480
26	0.98	0.96	0.97	180
27	0.94	1.00	0.97	60
28	1.00	1.00	1.00	150
29	0.99	0.98	0.98	90
30	1.00	0.93	0.97	150
31	0.94	0.99	0.97	270
32	0.87	1.00	0.93	60
33	1.00	1.00	1.00	210
34	0.99	0.99	0.99	120
35	1.00	1.00	1.00	390
36	0.98	1.00	0.99	120
37	1.00	1.00	1.00	60
38	1.00	0.99	1.00	690
39	0.98	1.00	0.99	90
40	0.95	0.97	0.96	90
41	0.74	1.00	0.85	60
42	1.00	0.78	0.88	90
accuracy			0.99	12630
macro avg	0.98	0.98	0.98	12630
weighted avg	0.99	0.99	0.99	12630

Figura 39. Métricas conjunto test experimento 4

5.1.5 Comparación de resultados

Tabla 1. Comparativa de resultados

	Train Accuracy	Validation Accuracy	Test Accuracy	F1- Score
Experimento 1	1	0.99	0.968	0.95
Experimento 2	0.998	0.994	0.957	0.93
Experimento 3	1	0.986	0.906	0.87
Experimento 4	0.984	0.998	0.987	0.98

Una vez realizados los cuatro experimentos y analizados los respectivos resultados se procede a hacer una comparación de ellos. En la Tabla 1 se muestran, por un lado, los valores de la accuracy obtenidos en cada experimento para los diferentes conjuntos de entrenamiento, validación y test. Se observa que, el experimento 4 es el que alcanza una accuracy mayor en el conjunto de test, con un valor del 0.987. Por otro lado, se muestra la puntuación F1 obtenida en los diferentes experimentos, siendo de nuevo la del experimento 4 la mayor de todas, con un valor de 0.98 lo que supone un 11% más que la obtenida por el experimento 3, cuyo valor es de 0.87.

Otra comparación interesante, es la del overfitting que posee cada uno de los modelos, ya que como se vio en capítulos anteriores, hay que evitar la presencia de sobreajuste a la hora de crear el modelo. En base a las gráficas de accuracy y loss de los cuatro experimentos, se puede observar que el experimento 2 es el que más sobreajuste presenta y el experimento 4 el que menos.

Por último, si se comparan las matrices de confusión de los experimentos, se puede comprobar que la matriz que más se parece a la de un modelo ideal, siendo la matriz ideal la que tiene todos los valores en la diagonal central, es la generada por el modelo 4, ya que, aunque no todos los valores se encuentran en la diagonal, los que se encuentran fuera de ella son pocos y además poseen valores pequeños, sin llegar a superar para ninguna clase 4 imágenes mal clasificadas. Por el contrario, la matriz de confusión obtenida por el modelo 3 posee más valores fuera de la diagonal, y confundiendo más imágenes.

En la Figura 40 se muestra un ejemplo del funcionamiento del modelo creado en el experimento 4, con imágenes distintas a las de la base de datos, que han sido extraídas de la web. El ejemplo se ha realizado para 20 imágenes pertenecientes a 20 clases diferentes y se puede observar en la Tabla 2, que el clasificador funciona correctamente para la mayor parte de las imágenes, ya que, de las 20 imágenes a clasificar, clasifica 18 de forma correcta y 2 de forma incorrecta. Las imágenes mal clasificadas pertenecen a las clases 5 y 6 (limite velocidad 80 km/h y fin limite velocidad 80 km/h), en cambio el modelo las clasifica como clase 2 y 23 (límite de velocidad 50 km/h y pavimento deslizante) respectivamente. Si se observa la Figura 38, que contiene la matriz de confusión del experimento 4, se observa que, para el conjunto test, el modelo clasificó 4 imágenes de clase 5 como modelo 2, esto se debe a que ambas señales se parecen ya que ambas son señales de límite de velocidad. El otro caso de fallo, en el que el modelo clasifica una señal de clase 6 como clase 23, siendo amabas clases muy diferentes, se debe a que la imagen que se la ha pasado al clasificador no es la adecuada, ya que no es una imagen de una señal de tráfico real.

Tabla 2.Resultados del experimento 4 para imágenes de la web

	1	2	3	4	5
Clase real	1 Límite velocidad 30 km/h	11 Intersección con prioridad	12 Calzada con prioridad	14 Stop	13 Ceda el paso
Clase Predicha	1 Límite velocidad 30 km/h	11 Intersección con prioridad	12 Calzada con prioridad	14 Stop	13 Ceda el paso
	6	**7**	**8**	**9**	**10**
Clase real	0 Límite velocidad 20 km/h	2 Límite velocidad 50 km/h	3 Límite velocidad 60 km/h	6 Fin límite velocidad 80 km/h	5 Límite velocidad 80 km/h
Clase Predicha	0 Límite velocidad 20 km/h	2 Límite velocidad 50 km/h	3 Límite velocidad 60 km/h	23 Pavimento deslizante	2 Límite velocidad 50 km/h
	11	**12**	**13**	**14**	**15**
Clase real	8 Límite velocidad 120 km/h	7 Límite velocidad 100 km/h	9 Prohibido adelantar	16 Entrada prohibida a vehículos de transporte de mercancías	17 Entrada prohibida
Clase Predicha	8 Límite velocidad 120 km/h	7 Límite velocidad 100 km/h	9 Prohibido adelantar	16 Entrada prohibida a vehículos de transporte de mercancías	17 Entrada prohibida
	16	**17**	**18**	**19**	**20**
Clase real	18 Otros peligros	20 Curva peligrosa hacia la derecha	21 Curvas peligrosas hacia la izquierda	22 Suelo irregular	23 Pavimento deslizante
Clase Predicha	18 Otros peligros	20 Curva peligrosa hacia la derecha	21 Curvas peligrosas hacia la izquierda	22 Suelo irregular	23 Pavimento deslizante

Figura 40. Ejemplo de funcionamiento del modelo del experimento 4

6 CONCLUSIONES Y LÍNEAS FUTURAS

La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia.

- John Ruskin-

En este último capítulo del trabajo, una vez conocida la teoría sobre Deep Learning y redes neuronales, creado el modelo y realizados varios experimentos, se procederá a sacar una serie de conclusiones del trabajo realizado y en base a ello, algunas mejoras que podrían llevarse a cabo en un futuro.

6.1. Conclusiones

El gran avance del aprendizaje profundo en la actualidad ha sido el que ha llevado a enfocar el presente trabajo en esta tecnología. Durante el estudio e investigación del Deep Learning se ha comprobado su complejidad, así como la cantidad de campos en los que está presente. Dentro de los diversos campos, se ha optado por el de la conducción autónoma, ya que son cada vez más los coches autónomos que circulan por las carreteras, aunque sin alcanzar la autonomía completa, de momento.

La idea principal es aumentar la seguridad de los vehículos, mediante sistemas como el cambio de carril, la detección de peatones o la clasificación de señales de tráfico. Estos problemas pueden resolverse con Deep learning, y será el último problema mencionado el que se ha tratado de resolver en este proyecto.

Para ello, se ha elegido la base de datos GTSRB, que es una base de datos con imágenes suficientes y reales, lo cual es importante a la hora de elegir la base de datos ya que esto afectará al buen funcionamiento del modelo. Para el desarrollo, se ha elegido lenguaje de programación Python, del cual se han adquirido muchos conocimientos ya que se empezaba con muy poca base, y como plataforma para el desarrollo Google Colab, la cual ha permitido comparar la gran velocidad de la GPU frente a la CPU a la hora de trabajar con aprendizaje automático.

Se han realizado diferentes experimentos con modelos basados en redes neuronales convolucionales, las cuales son complejas de diseñar ya que es importante la elección del número de capas o el tipo de capas, así como los hiperparámetros y optimizadores usados a la hora de compilar el modelo, los cuales afectarán a los resultados obtenidos.

Se pueden considerar buenos los resultados obtenidos a lo largo del trabajo, pudiéndose realizar algunas mejores que se expondrán en el próximo apartado.

6.2. Líneas futuras

En primer lugar, para resolver el problema propuesto, la clasificación de señales de tráfico, se han usado redes neuronales convolucionales, pero se podría haber usado otro tipo de aprendizaje supervisado como máquinas de soporte vectorial (SVM) o ambos y comparar los resultados obtenidos para cada uno de ellos.

Por otro lado, se podría haber hecho uso del transfer learning, que es un método de machine learning que reutiliza modelos desarrollados para otras tareas como punto de partida para otro modelo, en este caso el

modelo clasificador de señales de tráfico. Para ello, se podría haber elegido un modelo de Deep Learning entrenado previamente para un problema de clasificación complejo, con un número elevado y desafiante de imágenes como es el ImageNet 1000.

Por último, este modelo podría incorporarse junto a otras tecnologías en un coche de forma que este supiera como actuar de manera correcta al encontrarse una determinada señal de tráfico, como por ejemplo pararse en caso de encontrarse un stop. Para ello, antes de hacer la clasificación correcta de una señal de tráfico, debe detectarla y una vez detectada, clasificar la señal en un tipo u otro. Esto podría lograrse haciendo uso de YOLO (You Only Look Once) que es una red neuronal capaz de detectar objetos en tiempo real, con gran rapidez y precisión. Para realizar la detección lo que hace es dividir la imagen en una cuadricula de tamaño nxn, prediciendo celda por celda la posibilidad de encontrar un objeto en ella y creando un cuadro delimitador alrededor de dicho objeto, denominado 'bounding-box'. Por cada cuadro delimitador, se predecirá el objeto que se encuentra en su interior, y se fijará un umbral mínimo de probabilidad para eliminar los cuadros delimitadores que no tengan ese umbral y quedarse solo con uno.

REFERENCIAS

[1] J. Torres Viñals, *Deep learning : introducción práctica con Keras, primera parte* . Barcelona: Kindle Direct Publishing Amazon, 2018.

[2] "¿Qué es el Machine Learning?¿Y Deep Learning? Un mapa conceptual | DotCSV - YouTube." https://www.youtube.com/watch?v=KytW151dpqU (accessed Jul. 09, 2021).

[3] "¿Qué es la Inteligencia Artificial? - Iberdrola." https://www.iberdrola.com/innovacion/que-es-inteligencia-artificial (accessed May 20, 2021).

[4] "OMS | 10 datos sobre la seguridad vial en el mundo." https://www.who.int/features/factfiles/roadsafety/es/ (accessed May 20, 2021).

[5] MINISTERIO DEL INTERIOR, "ANUARIO ESTADÍSTICO DE ACCIDENTES 2019."

[6] J. Soria and M. González, "CAUSA EL 90% DE LOS ACCIDENTES."

[7] "J3016C: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International." https://www.sae.org/standards/content/j3016_202104/ (accessed May 20, 2021).

[8] "De 0 a 5: cuáles son los diferentes niveles de conducción autónoma, a fondo." https://www.xataka.com/automovil/de-0-a-5-cuales-son-los-diferentes-niveles-de-conduccion-autonoma (accessed May 20, 2021).

[9] "Self-Driving Cars & AI — An Intro | by Data-Driven Science | Medium." https://medium.com/@datadrivenscience/self-driving-cars-ai-28a55e9bb62a (accessed May 20, 2021).

[10] "¿Qué son los Sistemas ADAS?" https://www.sistemas-adas.org/que-son-los-sistemas-adas (accessed May 20, 2021).

[11] "Sistemas de seguridad ADAS: cuáles son y cómo utilizarlos antes de que sean obligatorios." https://www.autopista.es/noticias-motor/sistemas-de-seguridad-adas-cuales-son-y-como-utilizarlos-antes-de-que-sean-obligatorios_155205_102.html (accessed May 20, 2021).

[12] "¿Qué Sistemas ADAS puede incorporar mi vehículo?" https://www.sistemas-adas.org/tipos (accessed May 20, 2021).

[13] "Cómo es el Sitema de Reconocimiento de Señales de Tráfico | Carglass." https://www.carglass.es/blog/conduce-seguro/sistema-de-reconocimiento-de-senales-de-trafico/ (accessed May 20, 2021).

[14] "Sistema ISA: así funciona el nuevo asistente obligatorio de la DGT para respetar los límites de velocidad." https://www.xataka.com/vehiculos/sistema-isa-asi-funciona-nuevo-asistente-obligatorio-dgt-para-respetar-limites-velocidad (accessed Jun. 23, 2021).

[15] "Un asistente de velocidad inteligente." https://revista.dgt.es/es/multimedia/infografia/2021/02FEBRERO/0204-Asistente-velocidad-inteligente.shtml (accessed Jun. 28, 2021).

[16] "Estos son todos los sistemas de seguridad ADAS que puede equipar un coche." https://neomotor.sport.es/conduccion/estos-son-todos-los-sistemas-de-seguridad-adas-que-puede-equipar-un-coche.html (accessed Jul. 09, 2021).

[17] F. Chollet, *Deep learning with Python*, vol. 361. Manning New York, 2018.

[18] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems* , 1st ed. Sebastopol, CA: O'Reilly Media, 2017.

[19] "Tipos de aprendizaje automático. La Inteligencia Artificial (IA) está en… | by Javier Luna Gonzalez |

SoldAI | Medium.” https://medium.com/soldai/tipos-de-aprendizaje-automático-6413e3c615e2 (accessed May 20, 2021).

[20] “La diferencia entre Inteligencia Artificial, Machine Learning y Deep Learning | by SPOT | Medium.” https://medium.com/@spot_blog/la-diferencia-entre-inteligencia-artificial-machine-learning-y-deep-learning-cc415f20e63a (accessed May 20, 2021).

[21] J. Patterson and A. Gibson, *Deep learning : a practitioner’s approach* . Sebastopol, California: O’Reilly, 2017.

[22] “Guía Rápida sobre Deep Learning | Aprende Machine Learning.” https://www.aprendemachinelearning.com/aprendizaje-profundo-una-guia-rapida/ (accessed May 20, 2021).

[23] “Todo lo que Necesitas Saber sobre el Descenso del Gradiente Aplicado a Redes Neuronales | by Jaime Durán | MetaDatos | Medium.” https://medium.com/metadatos/todo-lo-que-necesitas-saber-sobre-el-descenso-del-gradiente-aplicado-a-redes-neuronales-19bdbb706a78 (accessed May 20, 2021).

[24] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv Prepr. arXiv1212.5701*, 2012.

[25] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” 2015.

[26] R. Méndez Hernández, B. Acha Piñero, and M. del C. Serrano Gotarredona, “Aprendizaje profundo para la segmentación de lesiones pigmentadas de la piel Trabajo Fin de Máster .” El autor, Sevilla, 2019.

[27] “Conjuntos de entrenamiento y prueba: Separación de datos.” https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data?hl=es-419 (accessed May 21, 2021).

[28] “La matriz de confusión y sus métricas – Inteligencia Artificial –.” https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/ (accessed May 20, 2021).

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 1097–1105, 2012.

[31] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[33] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *32nd International Conference on Machine Learning, ICML 2015*, 2015, vol. 1, pp. 448–456.

[34] “os — Miscellaneous operating system interfaces — Python 3.9.5 documentation.” https://docs.python.org/3/library/os.html (accessed May 20, 2021).

[35] “NumPy.” https://numpy.org/ (accessed May 20, 2021).

[36] “Matplotlib: Python plotting — Matplotlib 3.4.2 documentation.” https://matplotlib.org/ (accessed May 20, 2021).

[37] “random — Generate pseudo-random numbers — Python 3.9.5 documentation.” https://docs.python.org/3/library/random.html (accessed May 20, 2021).

[38] “Home - OpenCV.” https://opencv.org/ (accessed May 20, 2021).

[39] “pandas - Python Data Analysis Library.” https://pandas.pydata.org/ (accessed May 20, 2021).

[40] “scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation.” https://scikit-learn.org/stable/index.html (accessed May 21, 2021).

[41] “Te damos la bienvenida a Colaboratory - Colaboratory.”

https://colab.research.google.com/notebooks/intro.ipynb (accessed May 20, 2021).

[42] "GTSRB - German Traffic Sign Recognition Benchmark | Kaggle." https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign (accessed May 20, 2021).