Proyecto Fin de Carrera
Ingeniería de las Tecnologías Industriales

Identificación de señales de limitación de velocidad usando redes neuronales

Autor: Inés Blanco de la Puerta
Tutor: Antonio Javier Gallego Len
Cotutor: Sara Ruiz Moreno

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Proyecto Fin de Carrera
Ingeniería de las Tecnologías Industriales

Identificación de señales de limitación de velocidad usando redes neuronales

Autor:
Inés Blanco de la Puerta

Tutor:
Antonio Javier Gallego Len
Profesor Sustituto Interino

Cotutor:
Sara Ruiz Moreno

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Proyecto Fin de Carrera: Identificación de señales de limitación de velocidad usando redes neuronales

Autor: Inés Blanco de la Puerta

Tutor: Antonio Javier Gallego Len

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

A mis padres, por apoyarme en todo lo que me propongo y animarme a seguir aun cuando las cosas se ponen difíciles.

A mis compañeros y amigos, por compartir conmigo tantas horas de clases, bibliotecas, descansos y hacer de estos años los mejores de mi vida.

A mis profesores, por todo lo que me han enseñado a lo largo de estos años. En especial, a Antonio Javier Gallego Len y Sara Ruiz Moreno, por enseñarme el apasionante mundo de la Inteligencia Artificial y el Deep Learning.

Inés Blanco de la Puerta

Sevilla, 2022

Resumen

El Trabajo de Fin de Grado que presento a continuación se basa en el estudio y aplicación práctica de técnicas de Aprendizaje Profundo (Deep Learning) con el objetivo de desarrollar un sistema de reconocimiento automático de señales de limitación de velocidad. Este podrá actuar junto con otros sistemas de ayuda a la conducción, también conocidos como ADAS (Advanced Driver Assistance Systems), para garantizar la seguridad en las carreteras.

El objetivo del proyecto será crear un clasificador de imágenes a partir de técnicas de aprendizaje supervisado, en concreto, mediante el entrenamiento de Redes Neuronales Convolucionales. Para llevarlo a cabo haremos uso de diversas aplicaciones ofrecidas por la plataforma Matlab y de una base de datos creada a partir de imágenes tomadas durante la conducción. Se realizará un trabajo previo de detección de señales y, posteriormente, se entrenará el modelo con la técnica de transferencia de aprendizaje. Esta técnica hace uso de una red previamente entrenada con una base de datos general, para especializarse en tareas más específicas utilizando una base de datos considerablemente más pequeña.

Se debe destacar la importancia que tienen el diseño de la arquitectura de la red y la elección de un conjunto de datos apropiado al problema para garantizar el buen funcionamiento del sistema. Al final del proyecto, quedará demostrada la viabilidad del Sistema de Reconocimiento Automático de Señales haciendo uso de una Red Neuronal Convolucional.

Abstract

The Thesis presented in the following lines is based on the study and practical application of Deep Learning techniques with the aim of developing a system capable of automatically recognizing speed limit signs. What is more, this system will be able to work with other driving assistance systems, also known as ADAS (Advanced Driver Assistance Systems), to ensure safety on the roads.

The aim of the project is to create an image classifier based on supervised learning techniques, specifically, by training Convolutional Neuronal Networks (CNN's). In order to accomplish this, we will use different applications offered by Matlab and a database composed by images taken while driving. First, a previous work of sign detection will be carried out. After which, the model will be trained using the transfer learning technique. It uses a previously trained network with a general database, to specialize in more specific tasks using a substantially smaller database.

To ensure the proper functioning of the system, we should highlight the importance of choosing an appropriated design of the networks architecture and dataset for the problem. By the end of the project, the feasibility of the Automatic Sign Recognition System using a Convolutional Neuronal Network will be demonstrated.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

When something is important enough, you do it even if the odds are not in your favour.

- Elon Musk, Tesla CEO -

Los que se mantienen, se quedan atrás, los que innovan, consiguen mantenerse o avanzar. Hoy en día no vale mantenerse en línea, sino que debemos seguir innovando día tras día. Esta es una de las razones que inspiran a la investigación y desarrollo de nuevas técnicas de aprendizaje automático como el Deep Learning.

1.1 Descripción del problema

Uno de los sectores que más fomentan y se benefician de los progresos de la inteligencia artificial es el sector automovilístico. Los usuarios cada vez demandan coches más automatizados y seguros. Para garantizar esta seguridad, cada vez se plantean más sistemas de ayuda a la conducción, también conocidos como ADAS (Advanced Driver Assistance Systems). Se trata de un conjunto de tecnologías capaces de disminuir la cantidad de riesgos que pueden aparecer durante la circulación, no sólo para los ocupantes del coche sino también para los demás usuarios presentes en la vía.

Estos sistemas están formados por sensores, cámaras de video, radares o sistemas LIDAR. Desde julio de 2022 todos los coches nuevos deberán incorporar de serie ocho sistemas de ayuda a la conducción. Entre ellos se encuentran los sensores de aparcamiento, el avisador de ángulo muerto, los sistemas de detección de fatiga, el frenado autónomo de emergencia, el asistente de velocidad inteligente (ISA), la caja negra o EDR y los sistemas de detección de señales. A continuación, trataremos en profundidad alguno de ellos (Baranova, 2022):

- **Sistema de Reconocimiento de Señales de Tráfico:** Este sistema es capaz de reconocer todo tipo de señales de la vía, incluídas las digitales fijas o variables, y las restricciones temporales. Además, este puede actuar junto con el Control de Crucero Adaptativo, controlando de manera autónoma la velocidad máxima de la carretera.
- **Control de Crucero Adaptativo (ACC):** Permite mantener la velocidad programada de manera continuada. Asimismo, es capaz de frenar y acelerar el vehículo para adaptarse al tráfico. Como hemos dicho previamente, este sistema a menudo trabaja junto al sistema de reconocimiento de señales de tráfico.
- **Asistente de Velocidad Inteligente (ISA):** Este sistema adapta la velocidad máxima de un vehículo a las limitaciones vigentes en cada tramo de la carretera de manera autónoma, impidiendo que el conductor acelere más allá de lo establecido.

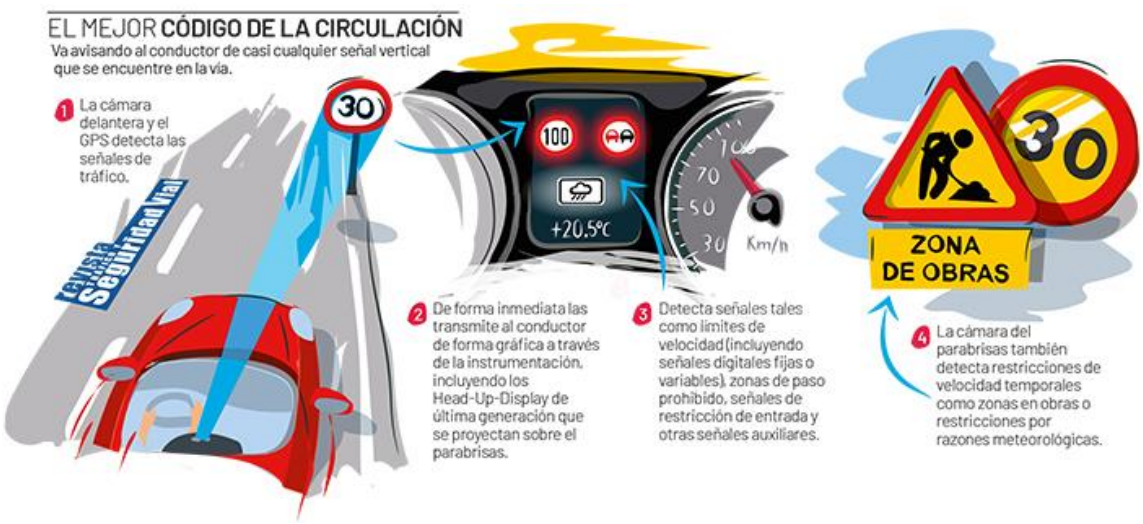


Figura 1. Sistema de Reconocimiento de Señales de Tráfico proporcionado por la DGT.

1.2 Objetivos

1.2.1 Objetivo general

El objetivo de este proyecto será crear un sistema de ayuda a la conducción, en concreto, un sistema de identificación de señales de limitación de velocidad. Este será capaz de clasificar señales en distintas categorías a partir de imágenes de entrada. Para la realización del proyecto, utilizaremos herramientas como Matlab y el Toolbox de Deep Learning, que proporcionan funciones y aplicaciones para crear, entrenar y simular Redes Neuronales profundas. Gracias a ello, realizaremos predicciones a través de redes previamente entrenadas como AlexNet o GoogleNet para así, junto con la variación de distintos parámetros, encontrar la red que mejor se adapte al nuevo set de datos y alcance una mayor precisión en sus predicciones.

Para poder desarrollar dicho sistema, primero debemos entender qué es el Deep Learning y las redes neuronales, por qué son importantes y cómo funcionan.

1.2.2 Objetivos específicos

El objetivo principal de este trabajo es el estudio y aplicación práctica de las Redes Neuronales Convolucionales (CNN) en un problema de visión artificial.

Por otro lado, también será necesario el estudio y aplicación de técnicas de procesamiento de imágenes. Entre las que se encuentra la separación por histograma, métodos morfológicos como el de apertura y cierre, la transformada circular de Hough y la función recorte.

Nuestro algoritmo debe ser capaz de llevar a cabo correctamente los siguientes objetivos:

- Procesar las imágenes de entrada para detectar la posición de la señal de limitación de velocidad.
- Eliminar la información innecesaria mediante el recorte de dichas imágenes.
- Entrenar la red neuronal para que dada una imagen de entrada sea capaz de determinar de qué señal de velocidad se trata.
- Dada una imagen nueva, que no aparezca en la base de datos, ser capaz de realizar los tres objetivos anteriores correctamente.

1.3 Metodología

El concepto de transferencia de aprendizaje permite a modelos, que ya han sido entrenados con bases de datos generales, especializarse en tareas más específicas utilizando una base de datos considerablemente más pequeña y específica para resolver un determinado problema (Janiesch et al., 2021).

Para realizar la transferencia de aprendizaje necesitaremos tres componentes (Mathworks):

- Una red que entrenar que obtendremos a partir de la modificación de una red previamente entrenada. En nuestro caso utilizaremos AlexNet.
- Una base de datos con la que entrenar. Es decir, imágenes para las que ya conocemos la etiqueta correcta.

  Para el conjunto de imágenes de entrada contamos con un set de datos reducido. Este ha sido creado a partir de imágenes reales de la calle tomadas con un iphone 8 y una Nikon D3300. Trabajaremos con 9 clases de señales de limitación de velocidad.
- Por último, tendremos que especificar un conjunto de opciones de entrenamiento o parámetros que afinaremos a medida que se produce el entrenamiento para mejorar la capacidad de la red de identificar correctamente las imágenes del dataset. Entre estos parámetros se encuentra el Batch size, el número máximo de iteraciones, el ritmo de aprendizaje o el grado de agresividad con el que las nuevas iteraciones deben cambiar los parámetros de la red.

Una vez tengamos estas tres variables el reentrenamiento de la red puede comenzar. Si el rendimiento no es el que buscamos ajustaremos las opciones de entrenamiento y volveremos a entrenar la red.

2 ESTADO DEL ARTE

You can focus on things that are barriers or you can focus on scaling the wall or redefining the problem.

- Tim Cook -

En este capítulo hablaremos sobre el auge de las Redes Neuronales. Como su capacidad para aprender ha llevado a campos como el de la medicina o la robótica a la investigación y desarrollo de estas técnicas para el reconocimiento de patrones y la clasificación de imágenes. Ahondaremos en nuevos sistemas que han supuesto grandes avances en dichos campos.

2.1 Sistemas de Visión Artificial y la clasificación de imágenes

La Visión Artificial es una rama de la inteligencia artificial cuyo propósito es proporcionar a los sistemas informáticos la capacidad de percibir y comprender los elementos y características de una escena o imagen del mundo real. Consecuentemente, estará estrechamente ligada a técnicas de clasificación de imágenes y reconocimiento de patrones.

El reconocimiento de imágenes consiste en asignar a una imagen una etiqueta de un conjunto definido de categorías en función de sus características. Este puede parecer un problema trivial para el ojo humano, sin embargo, factores como el de la escala, la iluminación, deformaciones o la visión parcial de objetos en una imagen hacen de dicha clasificación una ardua tarea (Núñez Sánchez-Agustino, 2016).

Antes de la popularización de las Redes Neuronales Convolucionales (CNN's), los sistemas basados en Máquinas de Soporte de Vectores (SVM) ofrecían la mejor solución al problema de reconocimiento y clasificación de imágenes.

2.2 Algoritmo de Máquinas de Soporte de Vectores (SVM)

Las Máquinas de Soporte de Vectores (SVM) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vapnik y Cortes en 1995. Su buen funcionamiento llevó a su uso para solucionar una gran variedad de problemas de clasificación y regresión.

La idea detrás de las SVM consiste en que, a partir de los datos de entrada y sus etiquetas, se entrene al modelo para que sea capaz de predecir la clase de los nuevos datos que se introduzcan. El modelo es representado por un eje de coordenadas donde situaremos el conjunto de datos. El objetivo principal es encontrar un hiperplano que maximice el margen o distancia que separa las clases.

El caso más sencillo es el que presenta un conjunto de datos linealmente separable. Sin embargo, no siempre será posible separar linealmente los datos en su espacio de entrada. En la mayoría de casos, tendremos que transformar o proyectar un conjunto de datos pertenecientes a una dimensión n dada, hacia un espacio de dimensión superior aplicando la función Kernel. En este nuevo espacio euclidiano de mayor dimensión podremos separar los puntos mediante un hiperplano (Sánchez Anzola, 2015).

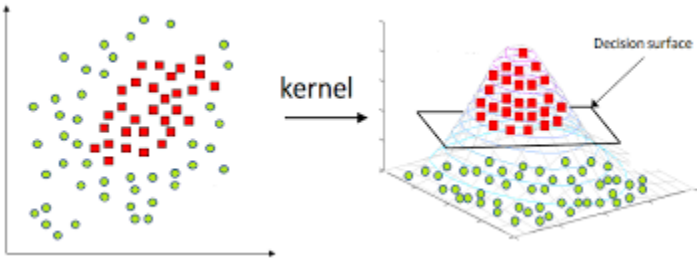


Figura 2. Método Kernel para clasificación (Van Vaerenbergh, 2018)

El desempeño de las SVM no depende del tamaño de la muestra que aportemos a la entrada, por lo que podrá ser utilizado para una cantidad limitada de datos. Esto es un punto a favor y contrasta con otros métodos de clasificación como las Redes Neuronales profundas (DNN's) que presentan un buen desempeño para tamaños de la muestra muy grandes, pero pueden tener un 'comienzo frío' (cold start) si el tamaño de la muestra no es lo suficientemente grande.

El principal inconveniente que presentan los sistemas basados en el algoritmo SVM es que requieren de un proceso previo de extracción de las características más relevantes de las imágenes de entrada. Este tratamiento suele ser diseñado a medida del problema que pretendemos resolver y para ello se emplearán sofisticadas técnicas de detección de objetos y patrones para los que suele ser necesario cierto conocimiento de los datos de entrada. En definitiva, la eficiencia del modelo dependerá de lo buenas que sean estas características extraídas (Núñez Sánchez-Agustino, 2016).

Al no obtener las características por sí mismo, este algoritmo no será capaz de aprender del conjunto de datos. Consecuentemente, este será muy sensible a cambios en la iluminación, escala, perspectiva que pueda presentarse en las imágenes.

2.3 Auge del Deep Learning y de las Redes Neuronales

El Deep Learning es un conjunto de técnicas de Machine Learning que utiliza Redes Neuronales profundas (DNN's), es decir, redes con muchas capas, para hacer predicciones. Estas redes son capaces de conseguir una altísima precisión en aplicaciones de reconocimiento de imágenes y patrones, en algunos casos, incluso más que los seres humanos.

Si queremos reconocer un cierto objeto, primero debemos encontrar patrones o características de las imágenes que se puedan utilizar para distinguirlos del resto. Se calculan esas características para el conjunto de imágenes, data set, y posteriormente se utilizan para entrenar el clasificador. El objetivo de Deep Learning es realizar un aprendizaje de principio a fin. Es decir, las imágenes son las entradas, y tanto las características como el clasificador se obtienen a partir del aprendizaje de las imágenes.

Las tareas de reconocimiento de imágenes cada vez desempeñan un papel más importante en la vida moderna. Desde los programas de asistencia al conductor, que detectan si un coche tiene enfrente otro vehículo, un peatón o un árbol a partir de una cámara situada en la parte delantera; hasta sistemas de imágenes médicas capaces de ofrecer un diagnóstico preliminar a partir de una imagen, casi como lo haría un especialista. O un sistema de control de calidad que puede examinar imágenes de los objetos que salen de una línea de producción detectando aquellos que puedan tener un defecto (Mathworks).

2.4 Reconocimiento de patrones en Redes Neuronales

El principal objetivo del Reconocimiento de Patrones es la clasificación ya sea supervisada o no supervisada. Este puede ser usado en campos como el del procesamiento, segmentación y análisis de imágenes, reconocimiento de voz, identificación de huellas dactilares, diagnósticos médicos, análisis del mercado de valores, etc.

Un sistema de reconocimiento de patrones es un proceso de búsqueda de regularidades y similitudes en los datos que nos dan como entrada. La información obtenida nos facilitará el procesamiento posterior. Por ejemplo, en reconocimiento de imágenes las características extraídas contendrán información sobre el tono de gris, la textura, la forma o el contexto de la imagen (Bhattacharya, 2020).

Las Redes Neuronales, a diferencia de la programación convencional, permiten obtener soluciones para aquellos problemas en los que no se conoce algoritmo a utilizar. En concreto, las Redes Neuronales Convolucionales (CNN's) son capaces de analizar grandes conjuntos de imágenes muy diversas y, por tanto, serán capaces de aprender un gran conjunto de patrones.

En las primeras capas aprenderán patrones muy genéricos como bordes, esquinas y texturas. A medida que avanzamos, estos patrones se combinarán para aprender otros más complejos y avanzados. Desde partes de dispositivos tecnológicos hasta cabezas de un tipo concreto de animal. En las últimas capas, los filtros mostrarán trozos de objetos perfectamente reconocibles capaces de resolver nuestro problema de clasificación (Dot CSV, 2021).

Las Redes Neuronales y el Reconocimiento de Patrones son herramientas muy potentes capaces de resolver problemas de clasificación con gran facilidad. Esto es algo que jamás alcanzaríamos mediante métodos clásicos de visión.

2.5 Tipos de Aplicación

A continuación, describiremos algunos de los avances más importantes en reconocimiento y clasificación de imágenes que se han realizado hasta la fecha.

2.5.1 Clasificación de tumores en cáncer de mama

El cáncer de mama es una de las causas más frecuentes de mortalidad entre las mujeres. La llegada de la inteligencia artificial ha supuesto una oportunidad para mejorar el diagnóstico médico mediante el uso de Redes Neuronales Convolucionales. La idea es ser capaz de detectar tumores en mamografías y clasificarlos en dos tipos dependiendo del tipo de tejido: benignos o malignos.

Hoy en día, la mamografía es el método más eficaz para detectar dicha enfermedad. Sin embargo, el diagnóstico de un experto no siempre es acertado al encontrarnos con factores como el cansancio, el descuido o la calidad de la imagen que pueden afectar a su valoración. Un diagnóstico computarizado (CAD) puede proporcionar una segunda opinión capaz de reducir los diagnósticos de falsos negativos.

Estudios como el llevado a cabo por Lourdes Duran-López y sus compañeros del Departamento de Robótica y Tecnología de Computadores de la ETSII, Universidad de Sevilla, nos muestran como mediante CNN's somos capaces de crear un sistema que clasifique tumores con una precisión del 80% (Durán-López et al, 2019).

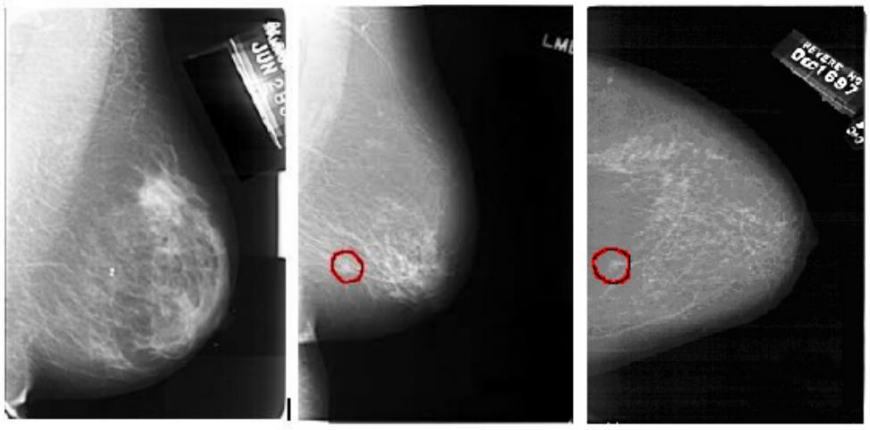


Figura 3. Mamografías extraídas de la DDSM. De izquierda a derecha: mama sin evidencia de anomalía, mama con presencia de tumor maligno y mama con tumor benigno. En rojo: localizador del tumor en la imagen (Durán-López et al, 2019).

2.5.2 TESLA Autopilot

Tesla es a día de hoy una de las mayores empresas de robótica del mundo. Esta ha querido diferenciarse en la industria de los automóviles buscando resolver la ardua tarea que es la conducción autónoma de la mano de nuevos avances en el campo de la Inteligencia Artificial. La idea es que el vehículo sea capaz de observar el entorno que le rodea, lo analice y en fracción de segundos planifique y decida las acciones que debe tomar sobre la mecánica del vehículo para poder desenvolverse en dicho entorno.

Dividiremos el problema que se nos presenta en dos secciones, la visión y la planificación. A continuación, presentaremos un resumen de la propuesta que realiza Tesla en el campo de visión:

Tesla, a diferencia de sus competidores, apuesta por solucionar el problema de la conducción autónoma sin hacer uso de la tecnología LIDAR, un costoso dispositivo que permite trazar un mapa tridimensional del entorno por el que se mueve el vehículo. La estrategia de Tesla trata de aproximar el problema a la realidad en la que los humanos conducimos únicamente haciendo uso de un sistema de 8 cámaras y varios sensores de movimiento.

El nuevo Sistema de Visión, Autopilot, de Tesla se encuentra en la fase Beta. A diferencia del sistema anterior donde tanto el input como el output era información en 2D, se busca que las predicciones que haga la Red Neuronal se proyecten en el espacio vectorial tridimensional. Para ello, cuentan con módulos de visión basados en Redes Neuronales Convolucionales (CNN's) que analizarán y extraerán los patrones observados en cada imagen para cada una de las cámaras. Una vez analizadas, se pone en común toda la información obtenida haciendo uso de Transformers. Además, han incorporado una memoria al sistema debido a la importancia de mantener una coherencia temporal en la información que se está analizando. Esta falta de memoria producía un fuerte parpadeo en los elementos detectados por las CNN's en antiguos modelos del Autopilot, ya que tras cada fotograma el sistema olvidaba lo que había detectado previamente creando cierta inconsistencia en las predicciones. Por ello, el nuevo sistema almacenará los patrones y características obtenidos de los módulos de visión multicámaras en diferentes momentos del tiempo, para que un módulo de video basado en Redes Neuronales Recurrentes (RNN's) se encargue de analizar la temporalidad de dichos patrones guardando una memoria de que es lo que ve y ha visto el coche y situando cada elemento predicho en un espacio tridimensional (Dot CSV, 2021).

Figura 4. Prueba del Sistema Autopilot proporcionada por TESLA

El Sistema Autopilot con conducción completa aún se encuentra en fase de prueba, por lo que los pocos usuarios que han podido acceder a este sistema han comprobado que aún hay errores que pulir antes de su salida al público.

3 DEEP LEARNING

Making everything more efficient should make everybody happier.

- Geoffrey Hinton -

En este capítulo introduciremos los conceptos de Inteligencia Artificial, Machine Learning y Deep Learning. Aprenderemos por qué son importantes las Redes Neuronales, su estructura y en qué consiste su proceso de aprendizaje. Presentaremos técnicas para que el modelo no sufra un sobreajuste y su rendimiento sea óptimo. Por último, hablaremos sobre las Redes Neuronales Convolucionales (CNN's) capaces de resolver el problema de clasificación de imágenes.

3.1 Inteligencia Artificial y Machine Learning

La **Inteligencia Artificial** es una subdisciplina del campo de la informática que busca la creación de máquinas que puedan imitar comportamientos inteligentes. Estos comportamientos son muy diversos, desde conducir hasta analizar patrones y reconocer voces.

Hoy en día, la inteligencia artificial solo es capaz de cumplir con un conjunto muy limitado de tareas, aunque el objetivo final de una IA es poder ser aplicada a una gran variedad de problemas y dominios diferentes.

Cuando hablamos de imitar no significa que dicho comportamiento deba ser realizado de manera cognitiva, ya que al programar de manera clásica los comportamientos de un brazo robótico para que realice una serie de movimientos llevará a cabo un comportamiento aparentemente inteligente. Desde este punto de vista, nos encontraremos con campos como el de la robótica, el Natural Language Processing (NLP) o la visión que estudiarán comportamientos inteligentes.

Si hay una capacidad del ser humano que de verdad nos defina como seres inteligentes es la capacidad de aprender. El campo del Aprendizaje Automático o **Machine Learning** busca dotar a las máquinas de dicha capacidad de aprendizaje mediante algoritmos y modelos estadísticos que nos permitirán realizar tareas sin la necesidad de ser programadas. Esta disciplina es un componente nuclear dentro de la inteligencia artificial, ya que se conecta y comunica con el resto de categorías.

La principal diferencia entre la Inteligencia Artificial y el Machine Learning está en que no es lo mismo programar algo que aprender automáticamente a hacerlo (Dot CSV, 2017).

El Machine Learning utiliza diferentes algoritmos para solucionar los problemas de datos. No existe un 'one-size-fits-all algorithm' que sea capaz de resolver con exactitud un tipo de problema. Por lo que el tipo de algoritmo que utilicemos dependerá del tipo de problema que queramos resolver, el número de variables, el tipo de modelo que se adapta mejor, etc. Entre estos algoritmos se encuentran (Mahesh, 2019):

- Los árboles de decisión (Decision Trees)
- Modelos de regresión como el de las Máquinas de Soporte de Vectores, SVM
- Modelos de clasificación como Naive Bayes
- Técnicas de Clusterización como K-means
- Redes Neuronales

3.1.1 Tipos de aprendizaje

A continuación, explicaremos algunos de los principales tipos de aprendizaje en machine learning y los algoritmos más comúnmente usados en cada caso.

3.1.1.1 Aprendizaje Supervisado

El Aprendizaje Supervisado es la tarea de aprendizaje de Machine Learning que busca encontrar la relación existente entre unas variables de entrada y otras de salida, a partir del conocimiento previo del resultado deseado a la salida. Por tanto, los algoritmos de aprendizaje supervisado necesitarán asistencia externa.

El aprendizaje surge de enseñarle a los algoritmos el resultado que queremos obtener para un determinado valor. Por ello, dividiremos la base de datos de entrada en tres bases de datos de entrenamiento, validación y prueba. Los pares de variables de entrada y salida en la base de entrenamiento se usarán para calibrar los parámetros abiertos del modelo de Machine Learning. Los algoritmos aprenderán patrones gracias a los datos de entrenamiento que luego aplicarán en los datos de prueba para realizar las predicciones y clasificaciones pertinentes (Mahesh, 2019). El flujo de trabajo de los algoritmos de aprendizaje supervisado se muestra en la siguiente figura.

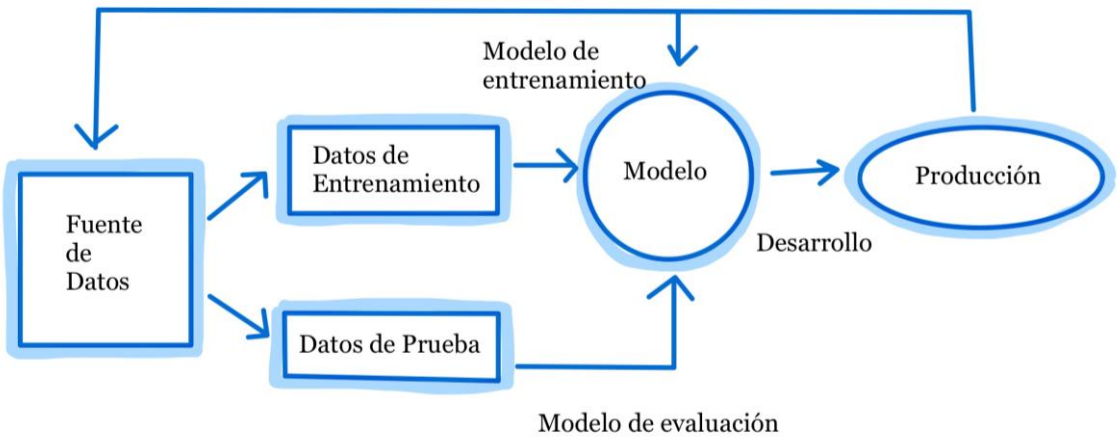


Figura 5. Flujo de trabajo de un algoritmo de aprendizaje supervisado.

Lo denominamos supervisado, ya que al mostrarle los resultados que queremos al algoritmo estamos participando en la supervisión de su aprendizaje. Este tipo de aprendizaje es el paradigma que más aplicación práctica ha tenido en las últimas décadas y se lleva a cabo cuando queremos hacer uso de algoritmos como modelos de regresión, de clasificación y árboles de decisión entre otros.

3.1.1.2 Aprendizaje no Supervisado

Al contrario que en el Aprendizaje Supervisado, ahora los datos no están etiquetados y, por tanto, tendrá que ser el propio algoritmo el que aprenda de manera autodidacta. Aunque no hay conocimiento a priori, le estamos explicando de forma indirecta cuál es el objetivo final, por ejemplo, al decirle que minimice una función de coste o que agrupe los datos cercanos entre sí. La dificultad de estos algoritmos es que no hay una respuesta correcta. Se utiliza principalmente en técnicas de clusterización (agrupamiento) y análisis de componentes (Mahesh, 2019).

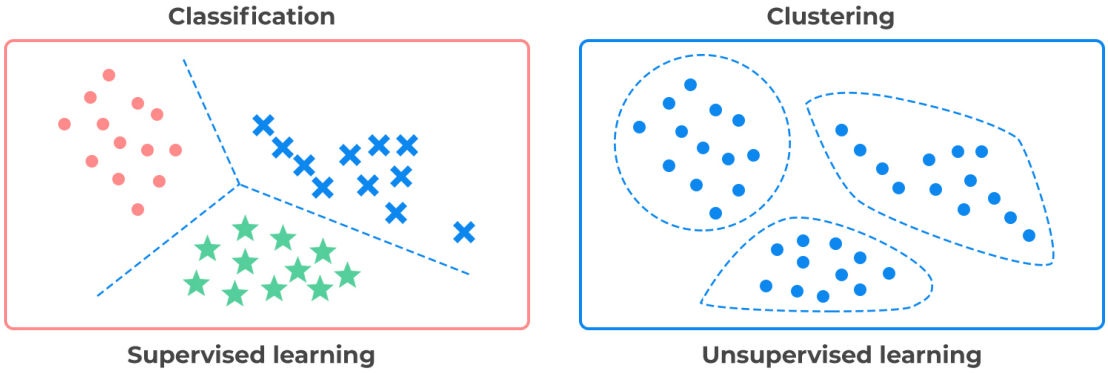


Figura 6. Ejemplos de técnicas de aprendizaje supervisado y no supervisado (Saini, 2021).

3.1.1.3 Aprendizaje Reforzado

En un sistema con Aprendizaje Reforzado, en vez de aportarle pares de variables de entrada y salida, describiremos el estado actual del sistema, especificaremos el objetivo, aportaremos una lista de acciones permitidas y dejaremos que el modelo de Aprendizaje Automático (Machine Learning) trabaje mediante prueba y error para llegar al objetivo final de la manera más eficiente posible. El Aprendizaje Reforzado ha sido aplicado con gran éxito en campos como el de los videojuegos o el de sistemas multi-agentes como son los mercados electrónicos (Janiesch et al., 2021).

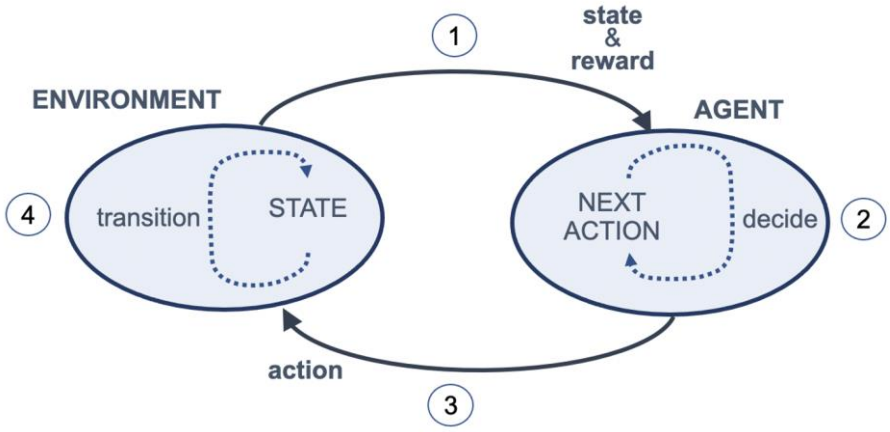


Figura 7. Flujo de trabajo de un algoritmo de aprendizaje reforzado (Torres, 2021).

3.2 Deep Learning y Redes Neuronales

El Deep Learning ha logrado tener un gran éxito recientemente en muchos procesos de reconocimiento de caracteres, de imágenes, de voz, predicción bursátil, generación de texto, traducción de idiomas, conducción autónoma e incluso pronóstico de enfermedades. Sin embargo, las Redes Neuronales profundas o 'deep neural networks (DNNs)' son frecuentemente percibidas como cajas negras, provocando que el procedimiento de toma de decisiones sea difícil de entender y por tanto disminuyendo la posibilidad de que estas sean usadas en posibles aplicaciones. A continuación, trataremos de explicar qué hay detrás de esta caja negra y por qué se ha

convertido en apenas unos años en una de las herramientas más potentes de la inteligencia artificial.

Se trata de un sistema cuya complejidad emerge de la interacción de muchas partes más simples que trabajan conjuntamente. A estas partes se las denomina Neuronas.

3.2.1 La Neurona

La Neurona es la unidad básica de procesamiento que podemos encontrar dentro de una red neuronal. Similar a una neurona biológica, esta tendrá conexiones de entrada a través de las que recibe estímulos externos, también llamados valores de entrada. A partir de estos valores la neurona realizará un cálculo interno para generar un valor de salida. Este cálculo numérico se trata de una suma ponderada de todos los valores de entrada.

Cada conexión de entrada tendrá asociado un peso que determinará la intensidad con la que cada variable afecta a la neurona. En este modelo cada neurona resuelve un problema de regresión lineal, en el que unas variables de entrada definen una recta o hiperplano al que podemos variar la inclinación utilizando los parámetros. Estos parámetros están comprendidos por los pesos y un término independiente cuyo cometido es el movimiento vertical de la recta, por tanto, este nos dará el control sobre el movimiento de la función. A este término se le denomina Sesgo o Bias y se representa como otra conexión de la neurona cuya variable siempre vale 1 (Dot CSV, 2018).

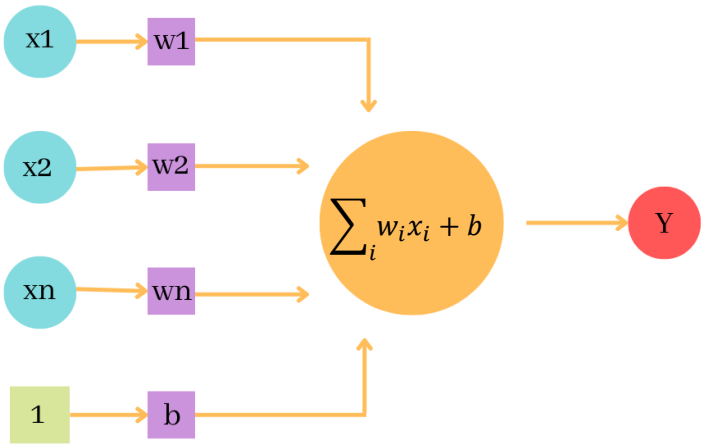


Figura 8. Estructura de una neurona.

Estos pesos junto con el Sesgo son los parámetros de nuestro modelo, con los que la neurona actuaría como un modelo de regresión lineal. No obstante, a continuación, presentaremos un último componente que diferenciará las neuronas del modelo de regresión lineal, las funciones de activación.

3.2.2 La Red

¿Qué ocurriría si comenzásemos a unir neuronas? Dos neuronas que se encuentran en la misma capa reciben la misma información de entrada y pasan los resultados obtenidos al procesar dicha información a la capa siguiente. Gracias a ello, la red presenta un conocimiento jerarquizado.

A la primera capa donde están las variables de entrada se la denomina capa de entrada, y a la última, capa de salida. Las capas intermedias serán denominadas capas ocultas.

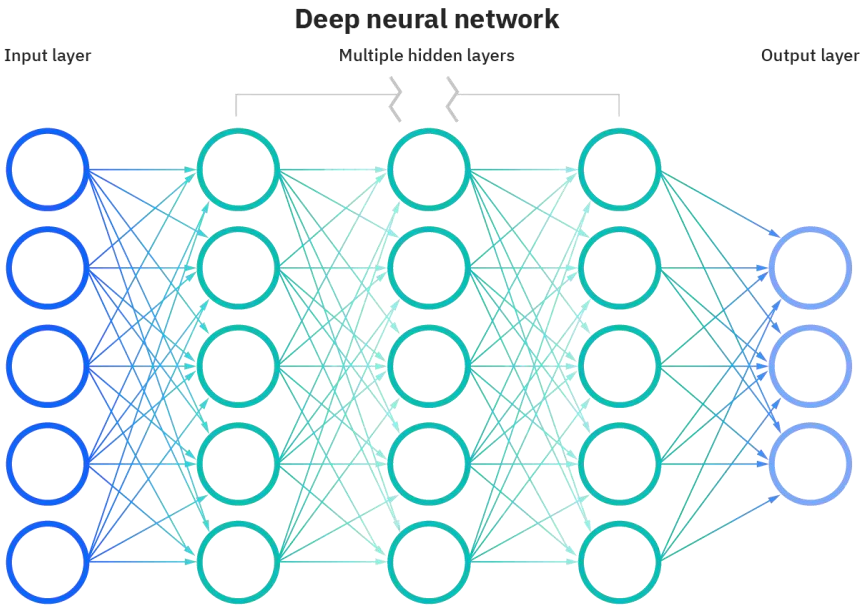


Figura 9. Estructura de una Red Neuronal Profunda (IBM Cloud Education, 2020).

Para alcanzar un aprendizaje profundo (Deep Learning) debemos conectar múltiples neuronas de forma secuencial. Si recordamos del punto anterior, cada una de estas neuronas realiza un problema de regresión lineal, es decir, estamos concatenando diferentes operaciones de regresión lineal. Dado que los resultados de estas operaciones son líneas rectas, la suma de todas ellas será otra línea recta. Visto de otra manera, tal y como está planteada la red y dado el conocimiento que tenemos de las neuronas hasta el momento, la red colapsaría hasta ser equivalente a una única neurona. Para conseguir que nuestra red no colapse, necesitamos que cada una de estas líneas sea distorsionada por una manipulación no lineal provocando que el resultado final sea distinto a una recta. Esta distorsión será llevada a cabo por las funciones de activación (Dot CSV, 2018).

3.2.3 Funciones de activación

Lo que realmente ocurre es que el valor de salida de la suma ponderada calculada por la neurona será pasado como valor de entrada en la función de activación. Esta distorsionará dicho valor añadiéndole deformaciones no lineales para así poder encadenar de forma efectiva la computación de un conjunto de neuronas.

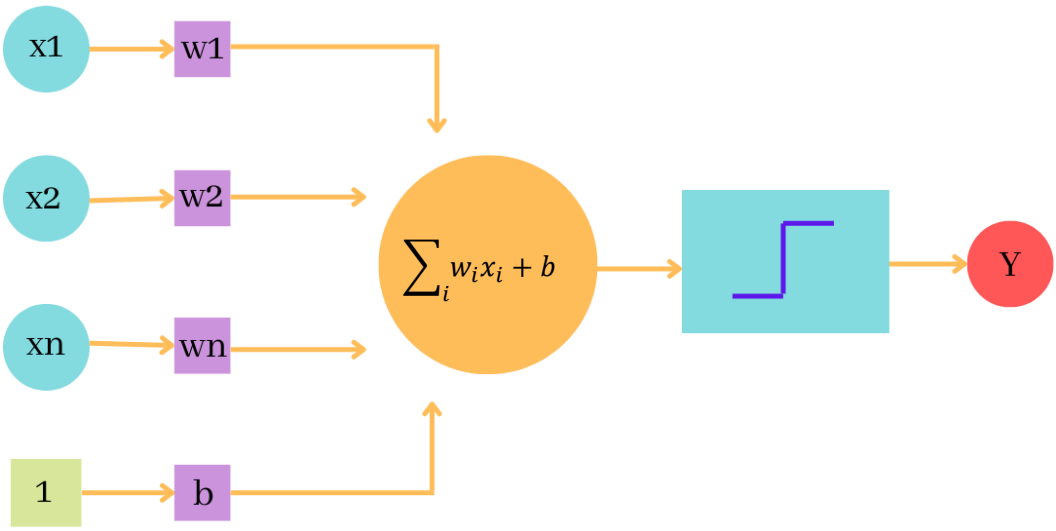


Figura 10. Estructura del perceptrón.

A continuación, presentaremos algunas de las funciones de activación más comunes (Méndez Hernández, 2019):

- **La función escalonada:** Una de las funciones de activación más simples. Para un valor de entrada mayor al umbral, el valor de salida es 1, en caso contrario el valor de salida será igual a 0. Como su propio nombre indica, el cambio de valor se produce instantáneamente produciendo así un escalón. Esta función no favorece al aprendizaje, por lo que no nos interesa.

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

- **La Función Sigmoide:** Para valores de entrada muy grandes, los valores de salida saturarán en 1, mientras que para valores muy pequeños saturarán en 0. El cambio de valor se produce de forma gradual, por lo que favorece al aprendizaje. Además, es útil para representar probabilidades al tener un rango de 0 a 1.

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

- **La Función Tangente Hiperbólica:** Similar a la Sigmoide, diferiendo en que el rango de esta varía entre -1 y 1.

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

- **La Función RELU o Unidad Rectificada Lineal:** Se comporta como una función lineal cuando los valores de entrada son positivos y constante a 0 cuando son negativos.

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

Al colocar un conjunto de neuronas en una misma capa y variar sus parámetros podremos combinar las figuras geométricas resultantes de cada neurona para obtener como resultado una figura más compleja que nos ayude a resolver nuestro problema de clasificación.

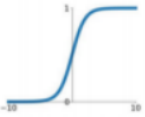
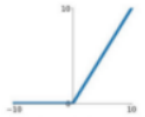
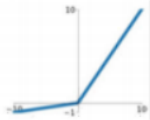
Función	*Ecuación*	*Gráfica*
Escalón	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	
Signo	$f(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases}$	
Lineal	$f(x) = x$	
Logística (sigmoide)	$f(x) = \frac{1}{1+e^{-x}}$	
Tangente hiperbólica	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$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

Figura 11. Tabla con algunas de las principales funciones de activación (Méndez Hernández, 2019).

3.3 Entrenamiento de Redes Neuronales

3.3.1 Proceso de Aprendizaje mediante Backpropagation

El trabajo 'Learning representations by back-propagating errors' publicado por David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams basado en avances en diferenciación automática supuso un punto de inflexión en el avance de las redes neuronales. Este mostró experimentalmente como usando un nuevo algoritmo de aprendizaje se podría conseguir que una red neuronal autoajustara sus parámetros para minimizar la función de error (Rumelhart et al, 1986).

Como hemos explicado anteriormente, cuando pasamos una base de datos por una red neuronal, modelamos la red con pesos y sesgos seleccionados aleatoriamente. Al comenzar la fase de entrenamiento, se realiza la propagación hacia delante o forward propagation, en la que alimentamos a la red con los datos de entrada y dejamos que la información se transforme a medida que avanza por la red de capas hasta recorrer el total de estas. En la capa final, obtendremos una predicción del resultado. La segunda fase, también llamada backpropagation, consiste en comparar los valores obtenidos durante el entrenamiento con los valores de salida esperados para así obtener el error. Transmitiremos hacia atrás el error, desde la capa de salida hacia todas las neuronas del resto de capas que componen la red, recibiendo estas el porcentaje de error aproximado en función de la participación de dicha neurona intermedia en el resultado final a la salida. Este proceso se repetirá capa por capa hasta que todas las neuronas hayan recibido un error que describa su aportación relativa al error total. Basándonos en el valor del error, reajustaremos los pesos y sesgos de cada neurona, de manera que la próxima vez que entrenemos la red, la salida este más cerca de la deseada y, por tanto, disminuya el error (Sánchez Anzola, 2015).

El entrenamiento finaliza cuando se verifica la condición de parada, ya sea porque el error sea inferior a un cierto umbral, o porque se haya superado un cierto número de iteraciones.

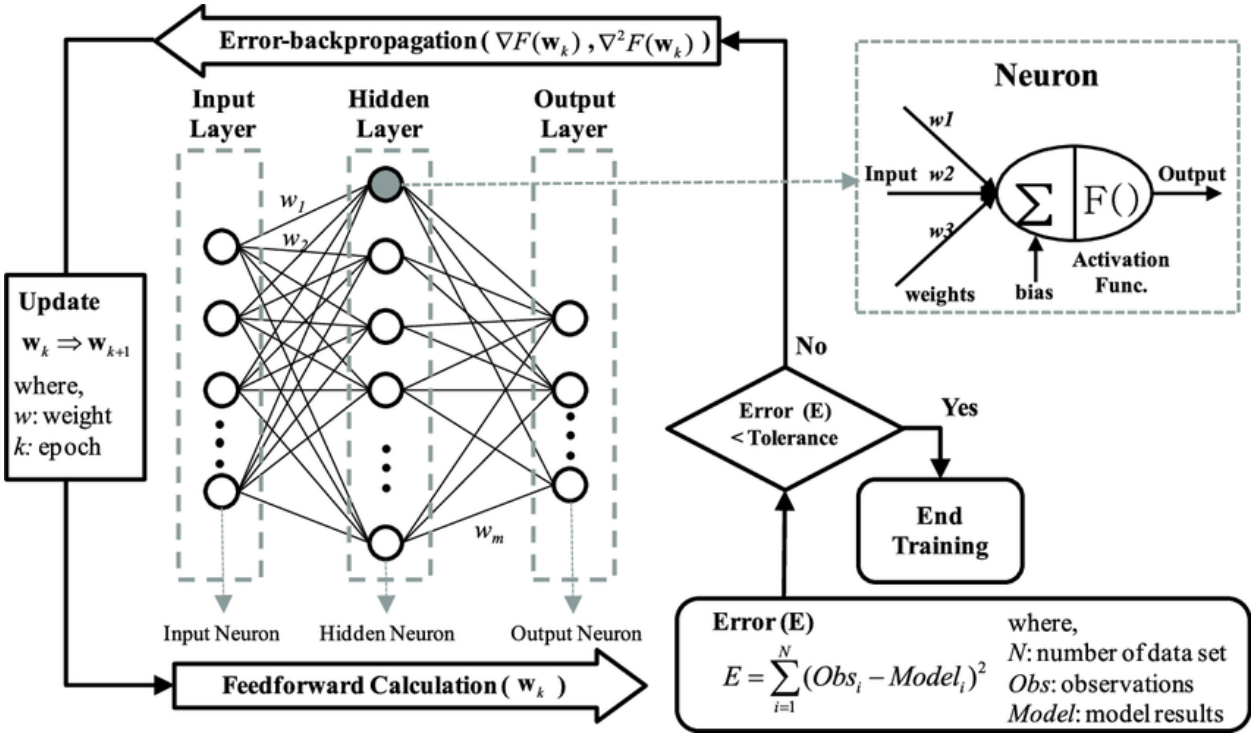


Figura 12. Diagrama del algoritmo de entrenamiento de Backpropagation

3.3.2 Overfitting y Underfitting

En el campo del Machine Learning se busca que un modelo sea capaz de generalizar su conocimiento y no memorizar los datos de entrenamiento. Utilizaremos un modelo de regresión polinomial, ya que el grado del polinomio es determinante a la hora de ajustar nuestro modelo a los datos.

Si este es demasiado bajo, **Underfitting** o Subajuste, no le damos la flexibilidad suficiente al modelo para poder ajustarse a la curva natural de los datos. El modelo tendrá un sesgo o bias muy alto, por lo que realizará una fuerte suposición sobre los datos. Por ende, nuestro modelo no será capaz de hacer buenas predicciones.

El grado de nuestro polinomio va a determinar el número de puntos críticos que va a tener nuestra curva. Mientras mayor sea el grado, más flexible llega a ser nuestro modelo. El problema de **Overfitting** o Sobreajuste aparece al darle un grado de flexibilidad demasiado alto al modelo. Demasiada flexibilidad le dará libertad para ajustarse tan bien a los datos de entrada que acabará por memorizarlos. Este tendrá una varianza muy alta, ya que cambiará significativamente dependiendo de los datos de entrenamiento (Koehrsen, 2018).

Debido al overfitting podemos tener la falsa impresión de que cuando entrenamos nuestro modelo este está aprendiendo correctamente. Sin embargo, cuando aparecen nuevos datos el rendimiento no es el que esperábamos conseguir (Dot CSV, 2019). Afortunadamente, existe una técnica para cerciorarnos de que obtenemos el modelo óptimo, la validación.

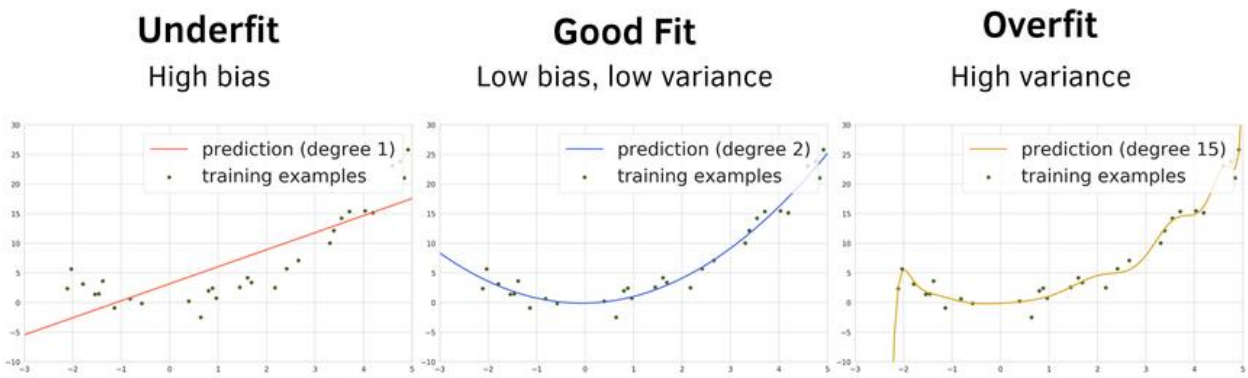


Figura 13. Ejemplos de modelos subajustados, bien ajustados y sobreajustados.

3.3.3 Datos de entrenamiento, validación y prueba

Si queremos saber si nuestro modelo sufre de Overfitting, necesitamos saber cual es el rendimiento para datos diferentes a los de entrenamiento. Por ello, dividiremos nuestro dataset en tres grupos, entrenamiento, validación y prueba. Normalmente, en porcentajes de 75, 15 y 10 respectivamente. Así podremos entrenar nuestro modelo y evaluar su rendimiento con datos que no habrá visto durante el entrenamiento.

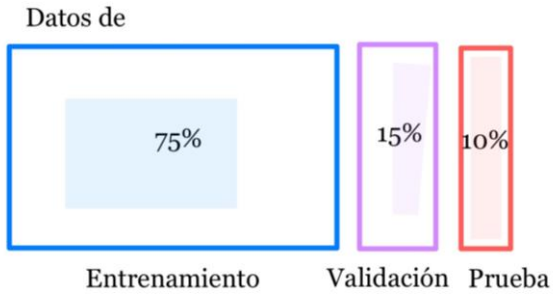


Figura 14. División de la base de datos.

Para que podamos comparar de forma correcta ambas mediciones del error, debemos asegurarnos de que los datos que introducimos en los tres grupos sean similares. Por ello, debemos desordenar de manera aleatoria los datos antes de dividirlos y posteriormente comprobar que estos estén balanceados, es decir que cada clase tenga el mismo número de instancias.

Si evaluamos simultáneamente el error de entrenamiento y validación de la figura 15, veremos que, como es de esperar, el de validación es superior al de entrenamiento. Si seguimos entrenando nuestra red, veremos que ambas gráficas seguirán descendiendo por lo que nuestro modelo estará generalizando. Llegará un punto en el que el error de entrenamiento seguirá descendiendo, mientras que el de validación comenzará a empeorar. Esto se debe a que el modelo sigue aprendiendo a resolver el problema con los datos de entrenamiento, pero su capacidad de generalización comienza a deteriorarse. Este punto es el punto óptimo, ya que previamente el modelo presentaba un subajuste y tras pasar dicho punto entra en una fase de sobreajuste (Dot CSV, 2019).

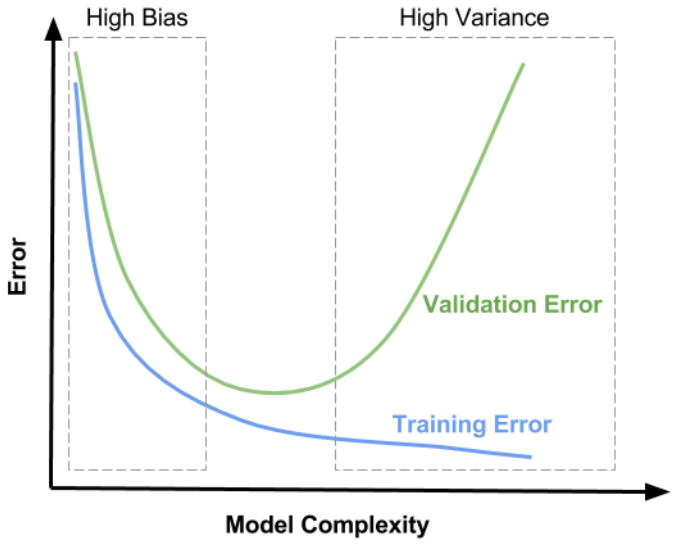


Figura 15. Gráfica de compensación del Sesgo con la varianza (Mallick, 2017).

En definitiva, un modelo subajustado tendrá un error de entrenamiento y validación alto, mientras que un modelo sobreajustado tendrá un error de entrenamiento extremadamente bajo y un error de validación alto (Koehrsen, 2018).

El conjunto de prueba se utilizará una vez acabado el proceso de entrenamiento y validación. Este nos permitirá evaluar el rendimiento del modelo final.

3.3.4 Hiperparámetros

Los hiperparámetros son variables cuyo valor no puede estimarse a partir de los datos. Estas son las encargadas de que la fase de entrenamiento, y por tanto el aprendizaje, se lleve a cabo de la manera más eficiente posible (Mathworks).

- **Mini-batches:** Calcular el error y el gradiente para todos los datos de entrenamiento en cada iteración es muy caro. Por este motivo, dividimos los datos de entrenamiento en mini-batches o pequeños lotes. Cada iteración durante el entrenamiento se realizará con un mini-batch distinto, por lo que el error y el gradiente calculado para cada mini-batch será una aproximación del de los datos de entrenamiento completos.
- **Epochs:** Las épocas o Epochs son el hiperparámetro que nos indica el número de veces que se ha procesado el total de los datos de entrenamiento por la red neuronal durante la fase de entrenamiento.
- **Learning Rate:** Se trata del tamaño del paso que da el algoritmo del descenso del gradiente en la dirección en la que el gradiente disminuye. Podemos pensar en el Learning Rate o la tasa de aprendizaje como la agresividad con la que deseamos modificar los parámetros en cada paso. Si su valor es demasiado grande, el algoritmo aprende tan rápido que puede saltarse algún mínimo. De lo contrario, si su valor es muy pequeño, tendrá más oportunidades de encontrar un mínimo local, pero el proceso puede llegar a ser demasiado lento.

  Podemos tener una tasa de aprendizaje fija para todo el entrenamiento o empezar con un aprendizaje más rápido y disminuirlo periódicamente a lo largo del proceso de entrenamiento.

3.4 Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales o **CNN's** en inglés, son un tipo de red neuronal cuyo diseño ha sido pensado para sacar partido a datos de entrada bidimensionales, como una imagen. Con una red neuronal clásica, al mandar como dato de entrada una imagen, estaríamos introduciendo un vector plano de píxeles, por lo que en ningún momento le estaríamos dando a cada píxel la importancia que tiene su posición dentro de la imagen [14]. Las CNN's ayudan a eliminar el complicado preprocesamiento de imágenes implementando un método gracias al cual podremos añadir directamente las imágenes de entrada. Estas redes están compuestas por capas convolucionales y capas de agrupación, pooling layers (Wu et al., 2017).

Las **Capas Convolucionales** se caracterizan por aplicar una operación matemática llamada convolución. Cada píxel nuevo que vayamos a generar se calculará colocando una matriz de números, llamada filtro o Kernel, sobre la imagen original. Desplazaremos el filtro y haremos uso de la vecindad para multiplicar y sumar cada píxel obteniendo como resultado una nueva imagen. Dependiendo de como configuremos los parámetros de nuestro filtro obtendremos un resultado u otro. Por tanto, esta operación de convolución sobre una imagen puede detectar distintos patrones para distintos valores del filtro. Aprender estos filtros para detectar patrones es el principal trabajo de una Red Neuronal Convolucional (Dot CSV, 2020).

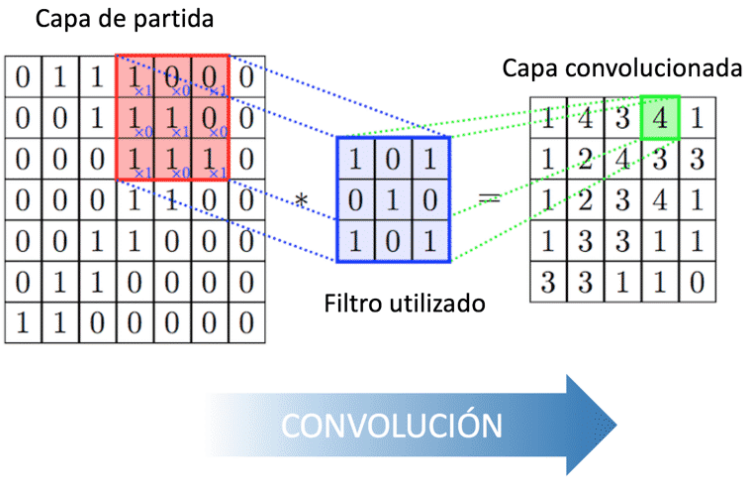


Figura 16. Operación de convolución (Calvo, 2017).

Cada una de las imágenes generadas se conocen como **Mapa de Características**, ya que actúa como un mapa donde se nos indica en que parte de la imagen se ha detectado la característica buscada por el filtro, píxeles en blanco. La operación de convolución por sí sola no es capaz de detectar más que cambios de contraste, bordes y patrones simples. Sin embargo, fijémonos en que donde antes teníamos una región de nueve píxeles, si el filtro fuese de tamaño 3x3, ahora se ha convertido en un único píxel de información. Por tanto, a medida que aplicamos convoluciones sobre estos Mapas de Características estaremos accediendo a cada vez más información espacial de la imagen original, consiguiendo así componer patrones cada vez más complejos. Por este motivo, el diseño de una estructura convolucional normalmente se representa como una especie de embudo donde la imagen inicial se va comprimiendo espacialmente, su resolución disminuye, al mismo tiempo que su grosor va aumentando, es decir, el número de Mapas de Características que vamos detectando va en aumento. Cuando la imagen pase por todo el embudo habremos detectado todos los patrones necesarios y, por tanto, podremos meter los Mapas de Características como inputs de una Red Neuronal multicapa que tratará de resolver el problema de clasificación (Dot CSV, 2020).

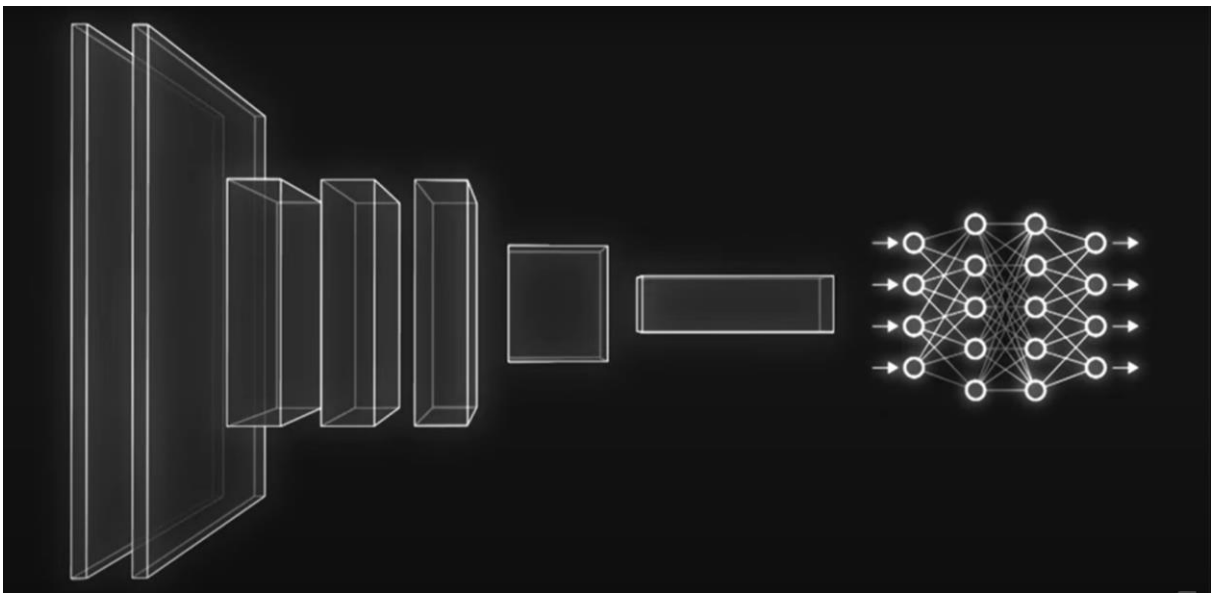


Figura 17. Estructura de una Red Neuronal Convolucional.

La red previamente entrenada AlexNet es un ejemplo de una Red Neuronal Convolucional, CNN (Mathworks).

4 METODOLOGÍA PROPUESTA

El método de investigación científica no es más que la expresión del modo necesario del funcionamiento de la mente humana.

- Thomas Henry Huxley -

A continuación, presentaremos la metodología seguida a la hora de llevar a cabo el proyecto. Trataremos de descomponer el proceso seguido desde el trabajo previo al entrenamiento hasta la creación de la base de datos y la elección de los diferentes valores para los hiperparámetros.

4.1 Transferencia de Aprendizaje

Crear y entrenar modelos analíticos inteligentes con redes neuronales poco profundas es caro y requiere de una base de datos muy amplia si queremos evitar un arranque frío. Afortunadamente, los modelos no siempre tienen porque estar entrenados desde cero. El concepto de transferencia de aprendizaje permite a modelos, que ya han sido entrenados con bases de datos generales, especializarse en tareas más concretas utilizando una base de datos considerablemente más pequeña específica para resolver un determinado problema (Janiesch et al., 2021).

La transferencia de aprendizaje ofrece una opción factible para que pequeñas y medianas empresas implementen sistemas inteligentes y permite a grandes empresas reutilizar sus propios modelos analíticos generales para aplicaciones específicas (Janiesch et al., 2021).

Para realizar la transferencia de aprendizaje necesitaremos tres componentes (Mathworks):

- En primer lugar, necesitamos una red que entrenar, un arreglo de capas que representen la arquitectura de la red. El objetivo de la transferencia de aprendizaje es que obtengamos esta red a partir de la modificación de una red previamente entrenada. En nuestro caso utilizaremos AlexNet.
- En segundo lugar, necesitaremos datos con los que entrenar. Es decir, imágenes para las que ya conocemos la etiqueta correcta. Si recordamos del capítulo 3, esto quiere decir que realizaremos un aprendizaje supervisado, ya que la red aprenderá de los ejemplos con una respuesta que sabemos es correcta.
- Por último, tendremos que especificar un conjunto de opciones de entrenamiento. El entrenamiento implica aplicar un algoritmo que, de manera iterativa, mejora la capacidad de la red para identificar correctamente las imágenes de entrenamiento. Este algoritmo se puede afinar con muchos parámetros como el Batch size, el número máximo de iteraciones, el ritmo de aprendizaje o el grado de agresividad con el que las nuevas iteraciones deben cambiar los parámetros de la red. Tendremos que especificar las opciones de entrenamiento para la red, aunque a menudo podremos usar los valores predeterminados.

Una vez tengamos estas tres variables el reentrenamiento de la red puede comenzar. Si el rendimiento no es el que buscamos ajustaremos las opciones de entrenamiento y volveremos a entrenar la red.

4.1.1 Flujo de trabajo

Cada aplicación de red neuronal es única, aun así, el desarrollo del proceso de transferencia de aprendizaje de una red suele implicar los siguientes pasos:

1. Preparación previa de las imágenes
2. Creación de la base de datos
3. Separación de los datos de entrenamiento, validación y prueba
4. Modificación de la red neuronal
5. Establecimiento de las opciones del algoritmo de entrenamiento para optimizar el rendimiento
6. Entrenamiento de la red
7. Validación de los resultados de la red mediante imágenes de prueba
8. Evaluación del rendimiento de la prueba

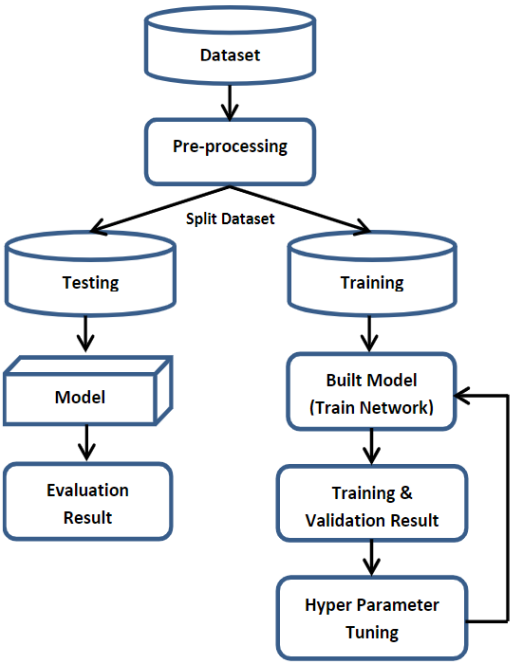


Figura 18. Flujo de trabajo de una Red Neuronal Convolucional

4.2 Herramientas utilizadas

A la hora de realizar un proyecto de Deep Learning, más concreto Redes Neuronales, se pueden utilizar diversos lenguajes de programación como puede ser Python, Matlab, C++, Java o R. Pese a que Python y R son los predominantes, decidimos trabajar con Matlab al haber adquirido conocimientos suficientes como para trabajar con agilidad utilizando esta plataforma. Además, MATLAB ofrece toolboxes especializados para trabajar con Machine Learning, Redes Neuronales, Deep Learning, Visión Artificial y Conducción Autónoma. Estos poseen herramientas y funciones capaces de trabajar con grandes conjuntos de datos. En pocas líneas de código, MATLAB permite desarrollar Redes Neuronales capaces de aprender y solucionar nuestro problema de clasificación.

Al no haber trabajado antes con Redes Neuronales, realicé los cursos de 'Deep Learning Onramp' y 'Deep Learning with Matlab' ofrecidos por MathWorks para tener una base y aprender a trabajar con los Toolbox necesarios para la realización del proyecto.

4.2.1 Toolbox

El Deep Learning Toolbox de Matlab proporciona un marco para diseñar e implementar redes neuronales profundas con algoritmos, modelos previamente entrenados y aplicaciones. Puede utilizar redes neuronales convolucionales (ConvNet y CNN) y redes de memoria de corto-largo plazo (LSTM) para realizar tareas de clasificación y regresión en imágenes, series temporales y datos de texto.

Este Toolbox viene con numerosos ejemplos previamente construidos que podremos utilizar. Incluyendo la clasificación de objetos en movimiento en una escena o la detección de rasgos faciales mediante regresión. También podremos crear arquitecturas de red tales como generativas antagónicas (GAN) y redes siamesas mediante diferenciación automática, bucles de entrenamiento personalizados y pesos compartidos.

La versión de 2018 nos introduce a la app Experiment Manager la cual nos permitirá gestionar varios experimentos de Deep Learning simultáneamente, realizar un seguimiento de los parámetros de entrenamiento, analizar resultados y comparar código de diferentes experimentos. Podremos visualizar la activación de capas y supervisar gráficamente el proceso del entrenamiento. También podremos evaluar modelos entrenados utilizando herramientas de visualización como gráficas y matrices de confusión.

En caso de querer crear un modelo a partir de la Transferencia de Aprendizaje, la app Deep Network Designer nos permite utilizar con facilidad redes neuronales ya entrenadas. Además, se pueden crear redes desde cero, analizarlas y entrenarlas gráficamente (Mathworks).

4.3 Preparación previa: Función Recorte

Las Redes Neuronales Convolucionales utilizan la convolución para aprender a detectar patrones en una imagen. Para facilitar este proceso, hemos creado una función llamada **myimfcn.m** (en Matlab) capaz de detectar una señal en una imagen y recortar su contorno. Eliminando así toda aquella información innecesaria sobre los alrededores de la imagen. A continuación, procederemos a mostrar paso a paso cómo funciona esta función sobre una imagen.

Figura 19. Imagen ejemplo de una señal de velocidad máxima 40 $^{Km}/_{h}$ tomada con una Nikon D3300

La idea es utilizar la Separación por Histograma en una imagen a color para obtener información sobre la localización de objetos. Primero, separaremos los tres canales (RGB) de cada imagen para formar tres imágenes monocromas.

Figura 20. Canales rojo, verde y azul de la imagen

Utilizaremos la Umbralización para separar las regiones que contengan un valor de rojo, verde y azul mayor o menor que cierto umbral y mostrar la silueta correspondiente. El rojo tráfico es (187,40,20), pero este es un color sintético. Por ello, comenzamos probando con puntos que tuviesen un nivel de rojo mayor que 110, verde menor a 70 y azul menor a 50. Tras probar con diferentes combinaciones, un nivel de rojo mayor que 90, verde menor a 70 y azul menor a 80 es la combinación que mejor capta el rojo tráfico distintivo de las señales en estas imágenes.

A continuación, haremos uso de Métodos Morfológicos como el de apertura y cierre. La apertura se emplea en siluetas para eliminar motas y el cierre para rellenar fisuras. Las funciones de Matlab que realizan estas operaciones matemáticas son ‘imopen’ e ‘imclose’.

La función ‘bwareaopen’ elimina los objetos detectados en la imagen de menos de x píxeles.

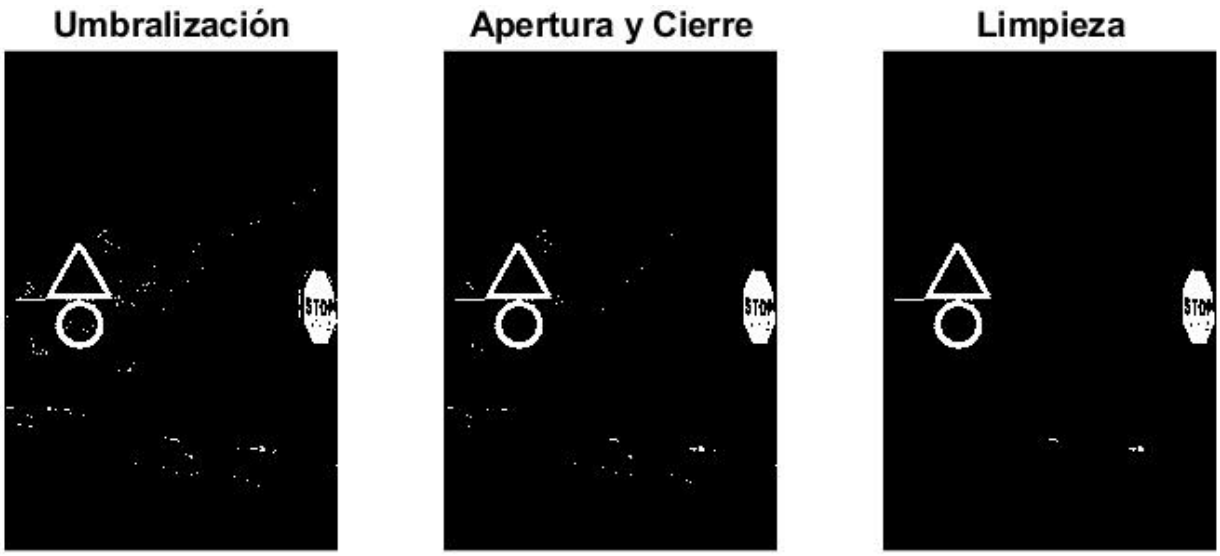


Figura 21. De izquierda a derecha, la imagen tras aplicarle los métodos de umbralización, apertura y cierre, y la función 'bwareaopen'

Acto seguido, procedemos a la detección de la señal mediante la transformada de Hough circular, haciendo uso de la función de Matlab 'imfindcircles'. Esta encontrará los círculos con radio dentro de un cierto rango especificado por nosotros y nos devolverá dos vectores con el radio y centro de estos círculos. Seleccionaremos el círculo de mayor radio.

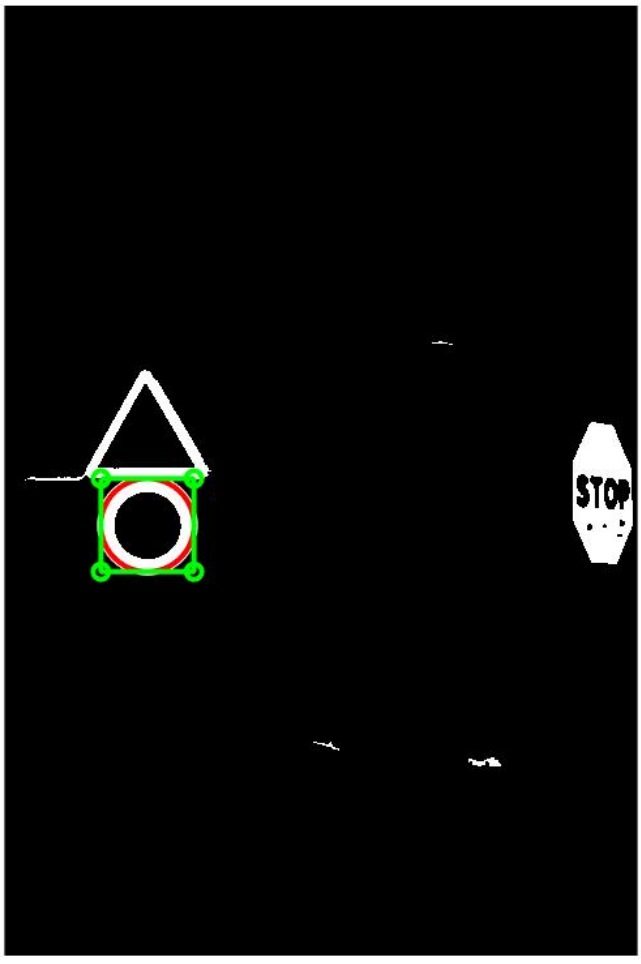


Figura 22. Detección de la señal de velocidad en la imagen

Por último, recortaremos la imagen mediante la función 'imcrop'. Para ello debemos especificar la región que

queremos recortar, en este caso el cuadrado que rodea la señal. Con ayuda de la función 'imresize' devolveremos una imagen de dimensiones [227,227], requisito necesario para las imágenes de entrada en la Red Convolucional AlexNet.

Figura 23. Señal recortada y redimensionada

Aplicaremos la función 'myimfcn' a todas las imágenes recopiladas.

4.4 Base de Datos

La base de datos ha sido creada a partir de imágenes reales de la calle tomadas con un iphone 8 y una Nikon D3300.

Trabajaremos con 9 clases de señales de limitación de velocidad. Comenzaremos con 25 señales de cada tipo, por lo que nuestra base de datos estará formada inicialmente por 225 imágenes divididas en 9 carpetas diferentes, una por cada clase de señal. Estas imágenes son en color y tienen formato .jpg.

El balanceo de datos o resampling, es decir, que el número de imágenes para cada tipo de señal sea el mismo, es muy importante, ya que de no ser así la red aprenderá más de una categoría que de otra pudiendo afectar al rendimiento de esta.

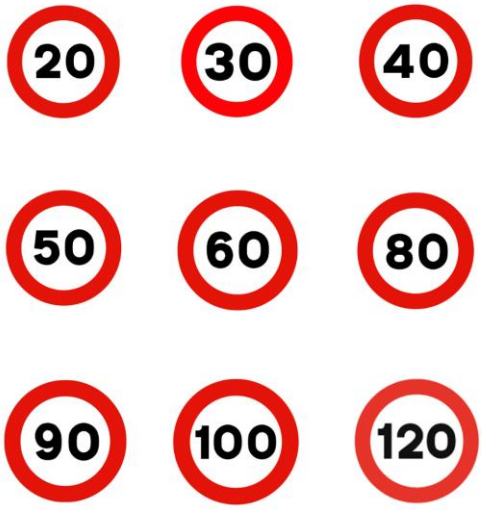


Figura 24. Clases de señales de la Base de Datos

4.5 División de la Base de Datos

Dividimos la Base de Datos para que el 75% de las imágenes sean usadas durante el entrenamiento, el 15% durante la validación y el resto durante la prueba. Esta división se producirá de manera aleatoria para asegurar que los datos introducidos en los tres grupos sean similares.

4.6 Estructura de la red usada

Las Redes Neuronales Convolucionales varían en su arquitectura y profundidad. Alexnet está formada por 25 capas y GoogleNet por 144. Aun así, las CNN's tienen capas comunes como la capa de imágenes de entrada o las capas convolucionales.

A continuación, mostraremos un esquema con la conexión de capas utilizadas en AlexNet. Como podemos ver, las capas más utilizadas son las ReLU, las convolucionales, las pooling o de agrupación y por supuesto, las fully connected o completamente conectadas.

Figura 25. Estructura de capas de Alexnet. Fuente: Deep Network Designer de Matlab

4.6.1 Tipos de Capas en una CNN

Todas las capas realizan una operación sobre los valores de entrada para devolver un nuevo valor. A continuación, presentaremos algunas de las capas más comunes dentro de la arquitectura de las CNN [2].

- **imageInputLayer:** Esta capa define el tamaño de entrada de la red y normaliza las imágenes de entrada. De forma predeterminada, esta capa lleva a cabo la normalización de los datos por medio de la substraccion de la imagen media del conjunto de entrenamiento. Esto centra las imágenes alrededor de cero.
- **convolution2dLayer:** Esta capa lleva a cabo la operación matemática de convolución de la que hemos hablado en el capítulo 3. Tienen un papel clave en la arquitectura de las CNN.
- **reluLayer:** Las capas de convolución normalmente van seguidas de una capa de activación no lineal como una unidad lineal rectificada (ReLU). Una capa ReLU realiza una operación de umbralización a cada elemento de entrada. Cualquier valor menor que cero se establece en cero.
- **maxPooling2dLayer:** Las capas de agrupación máxima realizan un muestreo descendente mediante la división de los datos de entrada en regiones de agrupación rectangulares y computando el máximo de cada región. Las capas Pooling o de agrupación reducen la complejidad creando una red más general.
- **fullyConnectedLayer:** Las características o patrones que la red va aprendiendo se almacenan en una colección de matrices hasta que se alcanza la Capa Completamente Conectada. Esta 'aplana' los datos que recibe como entrada para que puedan ser asignados a las diferentes clases en la salida. Esta capa representa una red neuronal clásica. El tamaño de salida de esta capa es el número de clases que necesitamos para nuestro problema de clasificación.
- **softmaxLayer:** Esta capa convierte los valores que obtenemos como salida de la Capa Completamente Conectada en puntuaciones normalizadas mediante una función exponencial normalizada. Podemos interpretar cada valor obtenido como la probabilidad de que la imagen de entrada pertenezca a cada clase.
- **classificationLayer:** Esta capa devuelve el nombre de la clase con la probabilidad más alta.

4.6.2 Modificación de la Red Neuronal

La Red Neuronal Convolucional AlexNet ha sido entrenada con alrededor de 1 millón de imágenes para clasificarlas en 1000 categorías. Por ello, si queremos reentrenar dicha red para que esta resuelva nuestro problema de clasificación, primero debemos modificar sus últimas capas.

Las tres últimas capas de la red están configuradas para 1000 categorías, por lo que tendremos que reemplazarlas por nuevas capas: fullyConnectedLayer, softmaxLayer y classificationLayer. Especificaremos las opciones de la nueva Capa Completamente Conectada acorde con nuestra base de datos dándole el nuevo número de categorías que tendrá que clasificar la red.

4.7 Hiperparámetros

Antes de comenzar con el entrenamiento de la red, debemos especificar las opciones de entrenamiento. Mantendremos los pesos de las primeras capas de la red pre-entrenada. Si queremos ralentizar el aprendizaje tendremos que darle un valor pequeño a la tasa de aprendizaje inicial o Initial Learning Rate.

Durante la transferencia de aprendizaje, no tenemos que entrenar la red por un número grande de épocas. Como explicamos en el capítulo 3, las épocas nos indican el número de veces que se ha procesado el total de los datos de entrenamiento durante la fase de entrenamiento. También especificaremos el tamaño de los lotes o mini-batches. Además, tendremos que especificar qué datos utilizaremos para la validación y la frecuencia con la que el sistema validará la red. (Mathworks)

4.8 Entrenamiento

Una vez tengamos la red modificada, la base de datos y las opciones de entrenamiento podremos comenzar con el proceso de transferencia de aprendizaje mediante el reentrenamiento de la red. Proporcionaremos las tres componentes a la función 'trainNetwork' que nos devolverá como salida la red entrenada.

Acabado el entrenamiento, utilizaremos la red obtenida para clasificar imágenes de prueba.

4.9 Investigación del rendimiento de la prueba

Primero almacenaremos las clasificaciones conocidas de las imágenes de prueba en una variable nueva. Haciendo uso de la función 'nnz' y el operador de igualdad '= =' obtenemos el número de clasificaciones correctas y almacenamos el resultado. A continuación, calculamos la fracción de imágenes de prueba correctamente clasificadas, para ello dividiremos el resultado obtenido previamente entre el número de imágenes de prueba.

La función 'confusionchart' calcula y muestra la Matriz de Confusión para las clasificaciones predichas. El elemento (j, k) de la matriz de confusión es un recuento de cuantas imágenes de la clase j predijo la red que estarían en la clase k. Por tanto, los elementos diagonales representan clasificaciones correctas, mientras que los no diagonales representan clasificaciones erróneas.

Si el rendimiento no es el que buscamos ajustaremos las opciones de entrenamiento y volveremos a entrenar la red.

5 PRUEBAS EXPERIMENTALES

Everything is theoretically impossible until it is done.

- Robert A. Heinlein -

En este capítulo, realizaremos una serie de pruebas utilizando la red neuronal convolucional modificada previamente y mostraremos los resultamos obtenidos para cada una de ellas. Con cada prueba variaremos los diferentes hiperparámetros determinantes para evitar el Overfitting del modelo. Posteriormente, evaluaremos los distintos resultados y definiremos el modelo que más se acerque a tener un rendimiento óptimo.

5.1 Pruebas

Llevaremos a cabo el proceso experimental utilizando la aplicación del Toolbox de Deep Learning de Matlab 'Experiment Manager'. Esta nos permite asignar varios valores a determinados parámetros y realiza el entrenamiento para cada posibilidad. La idea es identificar la combinación de parámetros que alcanza el mayor rendimiento.

Crearemos una tabla de hiperparámetros con todas las opciones que hemos definido en la función. Los posibles valores que puede tomar cada hiperparámetro son:

- Optimizadores: sgdm (stochastic gradient descent with momentum), Adam
- Número de épocas: 10, 15, 20, 25, 30
- Tasa de Aprendizaje Inicial: 0.01, 0.001, 0.0001

Se entrenará la red para cada combinación de estos valores.

5.2 Resultados

Entrenando las 30 posibles combinaciones de los distintos parámetros obtenemos la siguiente tabla de resultados. Primero encontraremos los ensayos realizados con el optimizador sgdm y posteriormente los realizados con el optimizador Adam. Para evaluar los resultados nos basaremos en la precisión y el error durante en entrenamiento y validación, cuyos valores podemos encontrar en el lado derecho de la tabla.

Trial	Progress	Elapsed Time	myInitialLearn...	myEpochs	mySolver	Training Accuracy (%)	Training Loss	Validation Accuracy (%)	Validation Loss
1	100.0%	0 hr 1 min 2 sec	0.0100	10.0000	sgdm	11.7188	2.1970	11.1111	2.1973
4	100.0%	0 hr 0 min 39 sec	0.0100	15.0000	sgdm	9.3750	NaN	11.1111	NaN
7	100.0%	0 hr 0 min 51 sec	0.0100	20.0000	sgdm	6.2500	NaN	11.1111	NaN
10	100.0%	0 hr 0 min 54 sec	0.0100	25.0000	sgdm	13.2813	NaN	11.1111	NaN
13	100.0%	0 hr 1 min 3 sec	0.0100	30.0000	sgdm	9.3750	NaN	11.1111	NaN
2	100.0%	0 hr 0 min 52 sec	0.0010	10.0000	sgdm	61.7188	1.1652	80.5556	0.7676
5	100.0%	0 hr 0 min 48 sec	0.0010	15.0000	sgdm	90.6250	0.3175	94.4444	0.2058
8	100.0%	0 hr 0 min 58 sec	0.0010	20.0000	sgdm	94.5313	0.1867	97.2222	0.0675
11	100.0%	0 hr 1 min 9 sec	0.0010	25.0000	sgdm	97.6563	0.0839	100.0000	0.0303
14	100.0%	0 hr 1 min 11 sec	0.0010	30.0000	sgdm	100.0000	0.0223	100.0000	0.0033
3	100.0%	0 hr 0 min 54 sec	0.0001	10.0000	sgdm	21.0938	2.5186	47.2222	1.5583
6	100.0%	0 hr 0 min 52 sec	0.0001	15.0000	sgdm	42.1875	1.6405	69.4444	1.2724
9	100.0%	0 hr 0 min 57 sec	0.0001	20.0000	sgdm	49.2188	1.3696	80.5556	1.0497
12	100.0%	0 hr 1 min 1 sec	0.0001	25.0000	sgdm	64.0625	1.0616	88.8889	0.8702
15	100.0%	0 hr 1 min 15 sec	0.0001	30.0000	sgdm	74.2188	0.8095	88.8889	0.6093
16	100.0%	0 hr 0 min 43 sec	0.0100	10.0000	adam	14.0625	13.3299	11.1111	14.1710
19	100.0%	0 hr 0 min 48 sec	0.0100	15.0000	adam	13.2813	13.5395	11.1111	14.1710
22	100.0%	0 hr 1 min 0 sec	0.0100	20.0000	adam	10.9375	14.1987	11.1111	14.1710
25	100.0%	0 hr 1 min 7 sec	0.0100	25.0000	adam	12.5000	13.8375	11.1111	14.1710
28	100.0%	0 hr 1 min 17 sec	0.0100	30.0000	adam	14.8438	2.3694	11.1111	2.2876
17	100.0%	0 hr 0 min 41 sec	0.0010	10.0000	adam	9.3750	3.1749	11.1111	2.3980
20	100.0%	0 hr 0 min 48 sec	0.0010	15.0000	adam	13.2813	2.2892	16.6667	2.2912
23	100.0%	0 hr 0 min 59 sec	0.0010	20.0000	adam	14.0625	2.4151	13.8889	2.3052
26	100.0%	0 hr 1 min 4 sec	0.0010	25.0000	adam	21.8750	2.1666	13.8889	2.1471
29	100.0%	0 hr 1 min 18 sec	0.0010	30.0000	adam	14.8438	2.5147	22.2222	2.2900
18	100.0%	0 hr 0 min 47 sec	0.0001	10.0000	adam	85.1563	0.4346	91.6667	0.2144
21	100.0%	0 hr 0 min 51 sec	0.0001	15.0000	adam	96.0938	0.0715	100.0000	0.0463
24	100.0%	0 hr 0 min 59 sec	0.0001	20.0000	adam	99.2188	0.0302	100.0000	0.0010
27	100.0%	0 hr 1 min 9 sec	0.0001	25.0000	adam	99.2188	0.0111	100.0000	2.5809e-5
30	100.0%	0 hr 1 min 18 sec	0.0001	30.0000	adam	100.0000	0.0012	100.0000	0.0001

Tabla 1. Resultado de los ensayos obtenidos usando Experiment Manager

5.3 Proceso de evaluación

Para evaluar los resultados, observaremos las diferentes pruebas a la vez que seguimos las siguientes pautas de evaluación:

- Aumentaremos el valor de la tasa de aprendizaje inicial hasta que las pérdidas (el error) lleguen a NaN. A partir de aquí, disminuiremos la tasa de aprendizaje inicial y consecuentemente las pérdidas hasta que el entrenamiento se produzca adecuadamente.
- Observaremos las gráficas de entrenamiento para cerciorarnos de que el error y la precisión de los datos de entrenamiento y validación no se hayan estancado. De ser este el caso, disminuiremos el número de épocas hasta encontrarnos en el inicio del régimen permanente. Puede que la precisión en las predicciones se vea ligeramente afectada, pero es más importante la disminución del tiempo de entrenamiento.
- Vigilaremos que el error final resultado del proceso de entrenamiento y validación sean bajos para asegurarnos de que no nos encontremos con un modelo sub o sobreajustado.

Mostraremos el proceso de evaluación seguido teniendo en cuenta los casos en los que se ha trabajado con el optimizador sgdm y Adam por separado, para posteriormente compararlos. Para ello, mostraremos las gráficas de entrenamiento de distintos ensayos. La siguiente imagen nos indica el significado de cada tipo de línea que podremos encontrar en las gráficas.

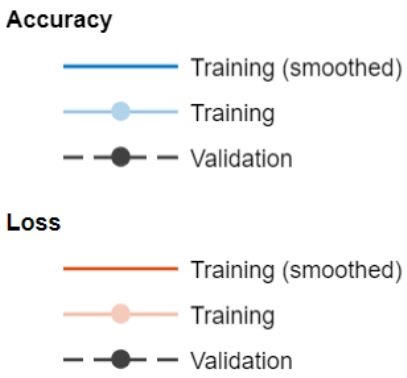


Figura 26. Significado de la gráfica

5.3.1 Proceso de evaluación trabajando con el optimizador sgdm

Comenzamos visualizando el ensayo número 13, con una tasa de aprendizaje inicial alta, 0.01, y 30 épocas. La gráfica nos muestra como alrededor de la novena época el error llega a ser demasiado alto, NaN, estancando la mejora de la precisión durante el entrenamiento de la red, no llegando esta a superar el 12%.

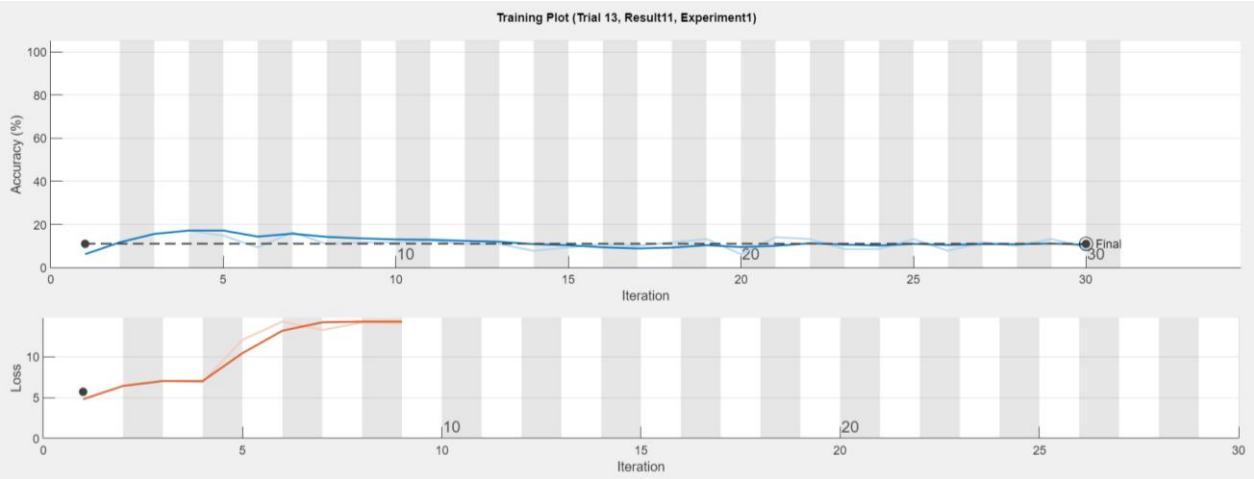


Figura 27. Gráfica de entrenamiento del ensayo número 13

A continuación, presentamos el ensayo 14, esta vez con una tasa de aprendizaje más baja, 0.001, y 30 épocas. Como podemos comprobar en la gráfica de entrenamiento, tanto el error en el dataset de entrenamiento como en el de prueba va disminuyendo a medida que se produce el entrenamiento de la red, así como la precisión aumenta. Esta llega a ser del 100% tanto en el proceso de entrenamiento como en el de validación. Observamos que, la gráfica de la precisión tiene una pendiente muy alta hasta llegar a cierto punto donde la pendiente disminuye considerablemente, llegando a ser prácticamente 0. A partir de este punto el entrenamiento mejora muy lentamente. Si disminuimos el número de épocas, obtendremos una precisión muy alta disminuyendo el tiempo de entrenamiento. En este caso, puede parecer innecesario ya que el tiempo que tarda la red en ser entrenada es tan solo de 1 minuto y 11 segundos. Sin embargo, debemos tener en cuenta que nuestro dataset solo contiene 25 imágenes de cada clase, en total 225 imágenes. En el momento en el que aumentemos la base de datos el tiempo se incrementará, lo cual no nos interesa.

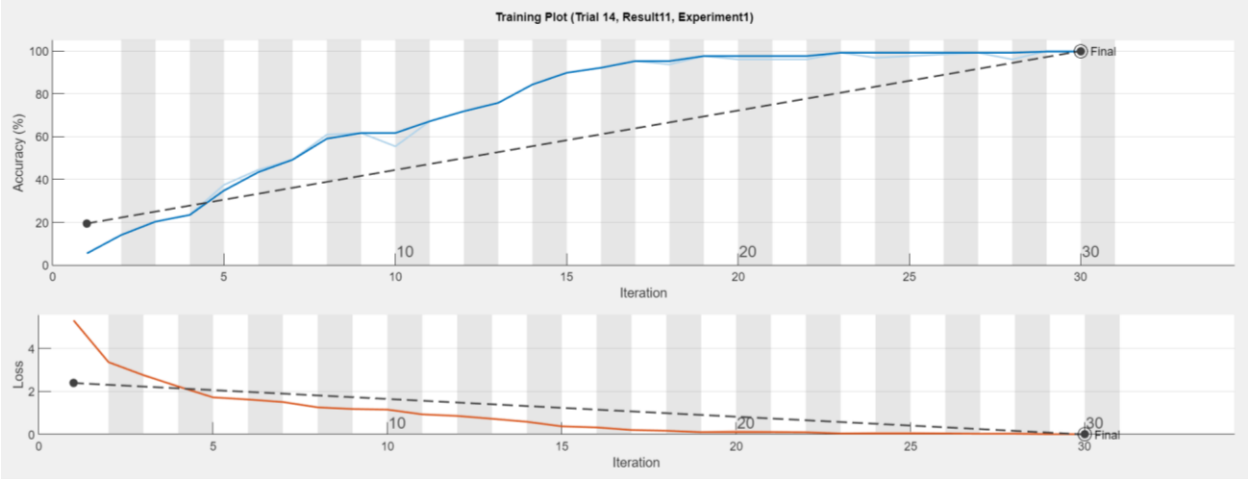


Figura 28. Gráfica de entrenamiento del ensayo número 14

En el ensayo 5, la red ha sido entrenada con una tasa de aprendizaje de 0.001 y 15 épocas. El tiempo de entrenamiento es de 48 segundos, la precisión en sus predicciones es del 90.6250% con los datos de entrenamiento y del 94.4444% con los de validación. Siendo el error en el entrenamiento de 0.3175 y en la validación de 0.2058. Aunque de primeras pueda parecer que el ensayo 14 obtiene mejores resultados, es importante hacer un balance entre todas las variables. Siendo más favorable una red con una precisión menor, si el tiempo de entrenamiento es considerablemente inferior.

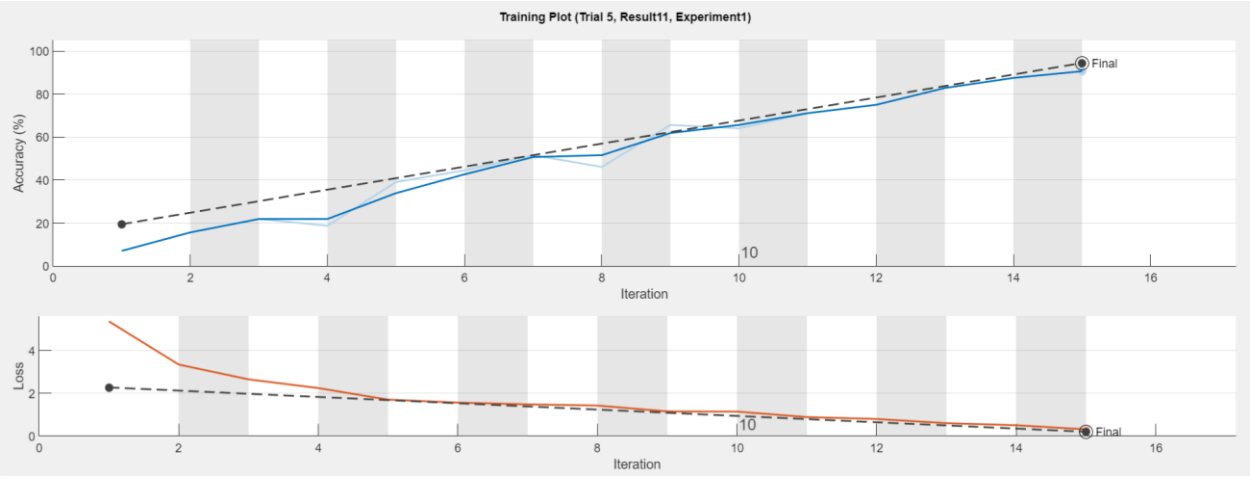


Figura 29. Gráfica de entrenamiento del ensayo número 5

5.3.2 Proceso de evaluación trabajando con el optimizador Adam

Partimos del ensayo número 28, en el que la tasa de aprendizaje es alta, 0.01, y entrenamos la red durante 30 épocas. A pesar de que el error consigue reducirse tras numerosas iteraciones, esto no se debe al aprendizaje de la red. Y aunque no se produce un sobreajuste del modelo, ya que el error de validación no es mucho mayor que el de entrenamiento, la precisión de este no consigue aumentar. La precisión de entrenamiento y validación son de 14.8438% y 11.1111% respectivamente.

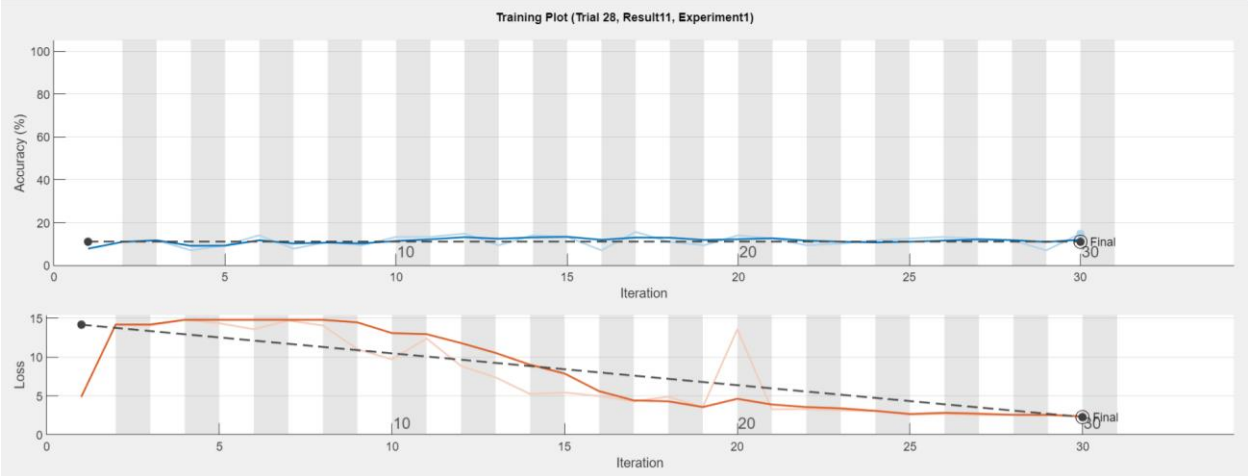


Figura 30. Gráfica de entrenamiento del ensayo número 28

Buscamos un ensayo con una tasa de aprendizaje más baja, como el 29, cuyo valor es de 0.001. Sin embargo, el entrenamiento de la red sigue sin obtener los resultados esperados al no ser capaz de aprender y ajustarse al modelo.

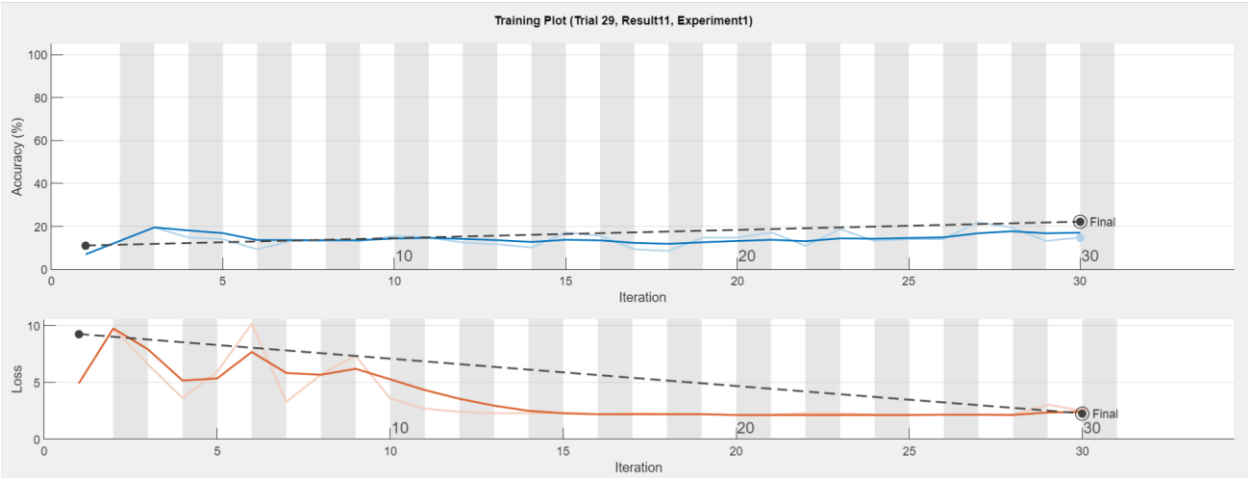


Figura 31. Gráfica de entrenamiento del ensayo número 29

El ensayo número 30 tiene una tasa de aprendizaje menor, en concreto de 0.0001. A medida que se produce el proceso de transferencia de aprendizaje de la red, las pérdidas van disminuyendo y la precisión aumentando hasta llegar a las 30 épocas. Este ensayo llega a alcanzar una precisión del 100% durante el entrenamiento y la validación, con una pérdida de tan solo 0.0012 y 0.0001 respectivamente. Observando la gráfica nos damos cuenta de que llega un momento en el que la precisión ya es lo suficientemente alta y el ritmo de aprendizaje empieza a disminuir. Es por ello que debemos reducir el número de épocas para conseguir un tiempo de procesamiento menor, sin provocar un descenso considerable de la precisión.

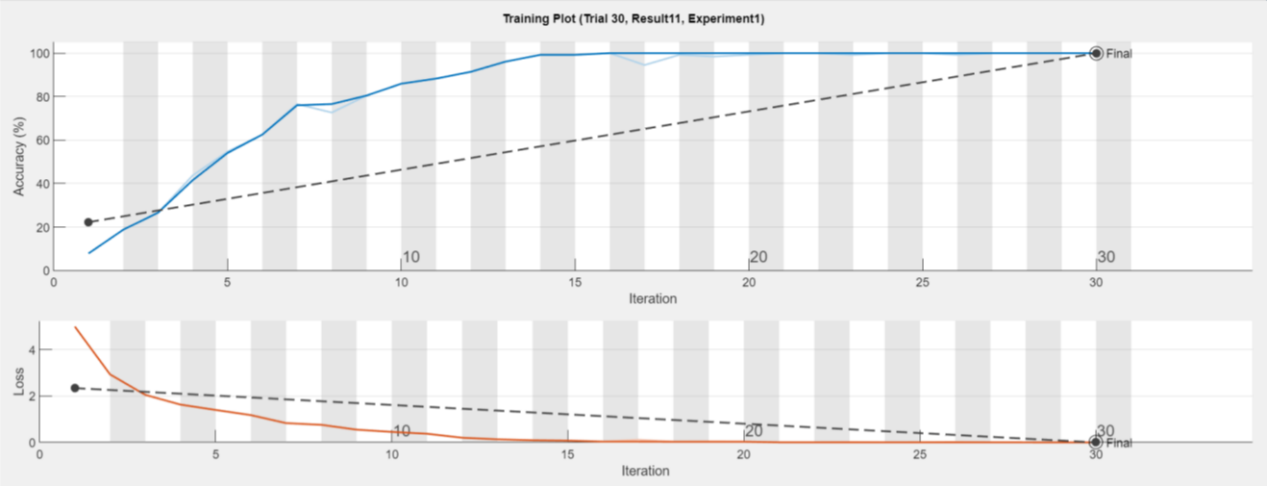


Figura 32. Gráfica de entrenamiento del ensayo número 30

En la gráfica de entrenamiento de la red del ensayo 21 podemos ver como esta es capaz de aprender del problema de clasificación obteniendo una precisión en sus predicciones del 96.0938% con los datos de entrenamiento y del 100% con los de validación. Siendo el error en el entrenamiento de 0.0715 y en la validación de 0.0463. Este ensayo tiene una tasa de aprendizaje inicial de 0.0001 y es entrenado durante 15 épocas. Su tiempo de entrenamiento es de 51 segundos. Como dijimos en el apartado anterior, es importante no guiarnos tan solo por la precisión, sino que debemos hacer un balance entre todas las variables.

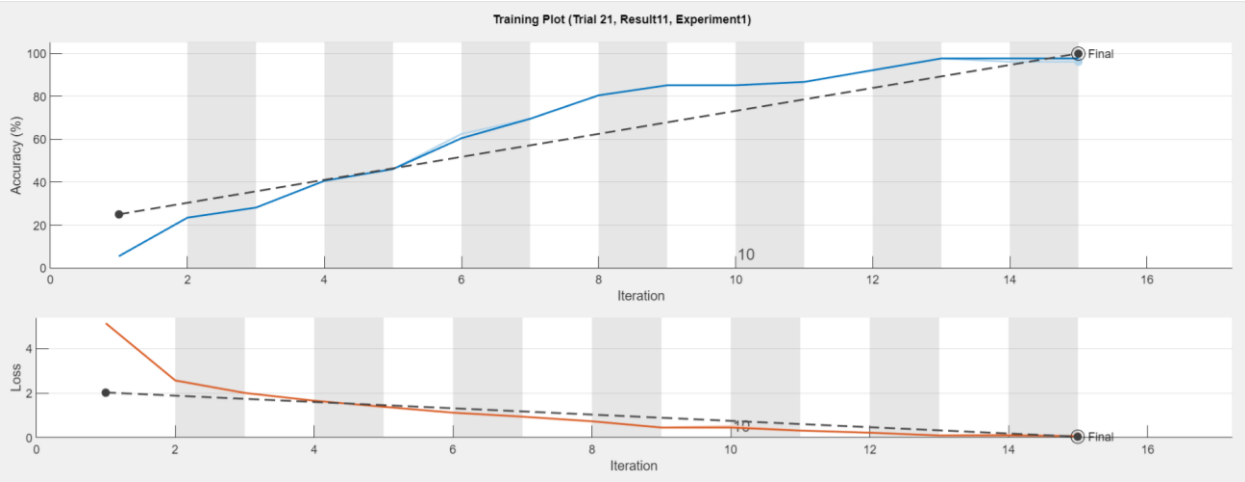


Figura 33. Gráfica de entrenamiento del ensayo número 21

5.4 Comparación de los resultados

Compararemos los ensayos 5 y 21 al ser los que muestran mejores resultados para los optimizadores sgdm y Adam respectivamente según el criterio tomado. Utilizaremos las redes entrenadas para clasificar imágenes de prueba nunca vistas por el modelo. Al haber dividido la base de datos en tres, siendo las imágenes de prueba el 10% del total del conjunto, mostraremos a cada modelo 18 imágenes aleatorias.

En el caso del ensayo número 5, la fracción de imágenes de prueba correctamente clasificadas es de 0.9444 y la matriz de confusión generada se muestra en la siguiente figura. Como podemos apreciar, la mayoría de elementos se encuentran en la diagonal, lo que quiere decir que su predicción es correcta.

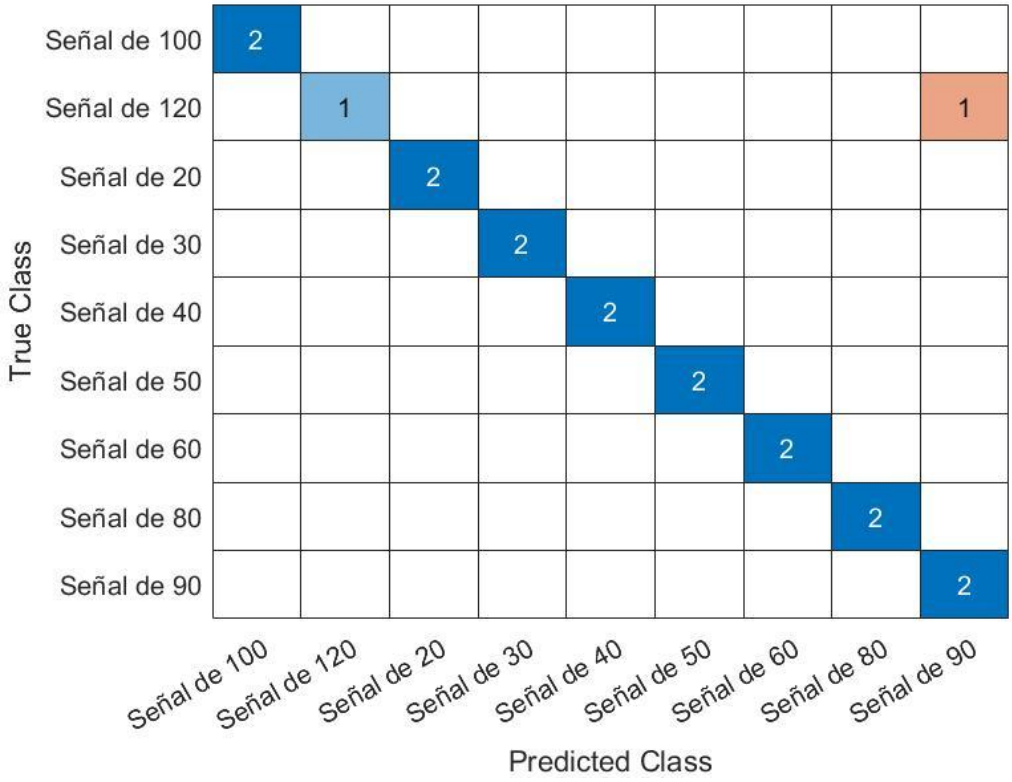


Figura 34. Matriz de confusión del ensayo número 5

En el caso del ensayo número 21, la fracción de imágenes de prueba correctamente clasificadas es de 1, lo que quiere decir que todas han sido clasificadas correctamente. A continuación, mostramos la matriz de confusión obtenida.

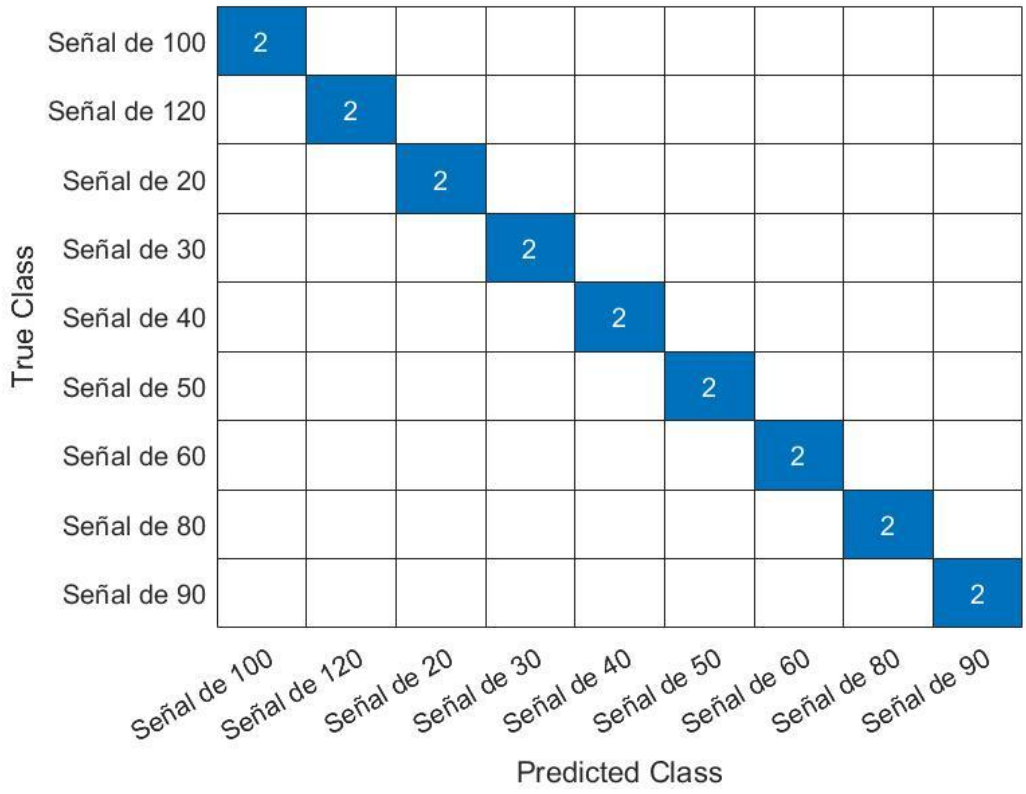


Figura 35. Matriz de confusión del ensayo número 21

Es importante destacar que cada vez que se entrena una red, aun tomando los mismos parámetros, obtenemos una solución distinta. Esto se debe a que los pesos iniciales se eligen de manera aleatoria, por lo que esta tomará diferentes caminos para llegar a la solución final. Y, por ende, no siempre obtendremos los mismos resultados en los valores de las precisiones y errores de entrenamiento y validación, así como con los datos de prueba. Dicho esto, aunque aparentemente el ensayo entrenado con el optimizador Adam obtenga mejores resultados en este caso, no siempre tiene por que ser así. Ambos optimizadores son capaces de resolver el problema de clasificación obteniendo unos resultados óptimos.

6 CONCLUSIONES

El ignorante afirma, el sabio duda y reflexiona.

- Aristóteles -

En este capítulo expondremos las principales conclusiones obtenidas como resultado del estudio de las redes neuronales y del desarrollo del proyecto mediante la realización de experimentos que nos ha llevado a la obtención de nuestro modelo final.

6.1 Conclusiones

Primero, debemos destacar el éxito obtenido en la creación de un sistema de reconocimiento de señales de limitación velocidad usando redes neuronales convolucionales, CNN's. Gracias a estas podemos crear un sistema capaz de aprender las características propias de diferentes señales de velocidad y de clasificar imágenes nunca antes vistas por la red en las distintas clases. Para la resolución de dicho problema solo hemos necesitado una buena base de datos y una arquitectura de la red adecuada.

En la misma línea, se ha podido constatar el buen funcionamiento del proceso de transferencia de aprendizaje. Este permite que modelos ya entrenados con bases de datos generales y de gran tamaño, como AlexNet, se especialicen en tareas más específicas como la de la clasificación de señales de velocidad utilizando una base de datos considerablemente más pequeña. De haber creado una arquitectura exclusiva para este problema, hubiésemos necesitado un conjunto de imágenes mucho más amplio para asegurarnos de no tener un arranque frío.

Por último, me gustaría resaltar que no hay un único modelo de resolución de nuestro problema. Existen numerosos caminos y puede que alguno de ellos obtenga mejores resultados. Otras alternativas podrían haber sido partir de otro tipo de Aprendizaje Supervisado, como el de las Máquinas de Soporte Vectorial (SVM), haber creado una arquitectura de red neuronal convolucional específica para nuestro problema o hacer uso de un tipo de red neuronal más avanzado llamado YOLO (You Only Look Once) capaz de detectar objetos en tiempo real con gran rapidez y precisión.

Para realizar este TFG he requerido de un importante trabajo previo de documentación y estudio de áreas desde Visión Artificial hasta Deep Learning y Redes Neuronales. Estos temas no son apenas tratados a lo largo del Grado de Ingeniería de las Tecnologías Industriales. Sin embargo, el auge de estos sistemas y los nuevos avances en la industria dejan ver como sin lugar a duda en poco tiempo estos sistemas van a ser capaces de resolver grandes problemas y ayudar en campos tan dispares como son la industria y la medicina. Por ello, me gustaría que este trabajo sirva para animar a que los altos cargos de la escuela y los encargados de los planes de estudio faciliten a las futuras generaciones de ingenieros la posibilidad de aprender más sobre este apasionante mundo.

7 BIBLIOGRAFÍA

Baranova, M. (2022). *Así funcionan los 8 sistemas de seguridad que llevarán los coches nuevos desde el 6 de julio.* https://neomotor.sport.es/conduccion/dgt-estos-son-los-8-sistemas-adas-obligatorios-en-los-coches-a-partir-de-julio-de-2022.html

Bhattacharya, S. (2020). An Overview of Neural Approach on Pattern Recognition. *Analytics Vidhya, Data Science Blogathon, 20*. https://www.analyticsvidhya.com/blog/2020/12/an-overview-of-neural-approach-on-pattern-recognition/#:~:text=Pattern%20recognition%20is%20a%20process,world%20or%20in%20abstract%20notions.

Calvo, D. (2017). Red Neuronal Convolucional CNN [Imagen]. https://www.diegocalvo.es/red-neuronal-convolucional/

Dot CSV (2017). ¿Qué es el Machine Learning?¿Y Deep Learning? Un mapa conceptual [Video].

https://www.youtube.com/watch?v=KytW151dpqU&list=PL-Ogd76BhmcC_E2RjgIIJZd1DQdYHcVf0

Dot CSV (2018). ¿Qué es una Red Neuronal? Parte 1: La Neurona [Video].

https://www.youtube.com/watch?v=MRIv2IwFTPg&list=PL-Ogd76BhmcC_E2RjgIIJZd1DQdYHcVf0&index=6

Dot CSV (2018). ¿Qué es una Red Neuronal? Parte 2: La Red [Video].

https://www.youtube.com/watch?v=uwbHOpp9xkc

Dot CSV (2018). ¿Qué es una Red Neuronal? Parte 3: Backpropagation [Video].

https://www.youtube.com/watch?v=eNIqz_noix8&list=PL-Ogd76BhmcB9OjPucsnc2-piEE96jJDQ&index=4

Dot CSV (2019). Como identificar el Overfitting en tu Red Neuronal [Video].

https://www.youtube.com/watch?v=ZmLKqZYlYUI&list=PL-Ogd76BhmcB9OjPucsnc2-piEE96jJDQ&index=8

DotCSV (2020). Redes Neuronales Convolucionales ¿Cómo funcionan? [Video]. https://www.youtube.com/watch?v=V8j1oENVz00&list=PL-Ogd76BhmcB9OjPucsnc2-piEE96jJDQ&index=9

Dot CSV (2021). Extraños patrones dentro de una Red Neuronal [Video].

https://www.youtube.com/watch?v=ysqpl6w6Wzg&list=PL-Ogd76BhmcB9OjPucsnc2-piEE96jJDQ&index=10

Dot CSV (2021). Así funciona realmente el TESLA AUTOPILOT | La IA de Elon Musk [Video].

https://www.youtube.com/watch?v=xKxim55h1UI&t=1195s

Durán-López, L., Domínguez-Morales, J.P., Luna-Perejón, F., Amaya-Rodríguez, I., Civit-Masot, J., Vicente-Díaz, S., Linares-Barranco, A. (2019). *Clasificación de tumores en cáncer de mama basado en redes neuronales de convolución* [Ponencia, ETSII, Universidad de Sevilla]. idUS.

https://idus.us.es/bitstream/handle/11441/88870/linares_ponencia_sevilla_2019_clasificacion.pdf?sequence=1&isAllowed=y

IBM Cloud Education (2020). ¿Qué son las redes neuronales? [Imagen].

https://www.ibm.com/cl-es/cloud/learn/neural-networks

Janiesch, C., Zschech, P. & Heinrich, K. (2021). Machine learning and Deep Learning. *Electronic Markets, 31, 685-695.* https://link.springer.com/content/pdf/10.1007/s12525-021-00475-2.pdf

Koehrsen, W. (2018). *Overfitting vs Underfitting: A Complete Example.* Towards Data Science. http://www.pstu.ac.bd/files/materials/1566949131.pdf

Mahesh, B. (2019). *Machine Learning Algorithms- A Review*. International Journal of Science and Research

https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf?eid=5082902844932096

Mallick, S. (2017). Bias-Variance Tradeoff in Machine Learning [Image]

https://learnopencv.com/bias-variance-tradeoff-in-machine-learning/

Mathworks https://es.mathworks.com/

Méndez Hernandez, R., Acha Piñero, B. & Serrano Gotarredona, M.C. (2019). *Aprendizaje profundo para la segmentación de lesiones pigmentadas de la piel* [Trabajo Fin de Máster]. idUs

Núñez Sánchez-Agustino, F.J. (2016). *Diseño de un sistema de reconocimiento automático de matrículas de vehículos mediante una red neuronal convolucional* [Trabajo de Fin de Máster, Universitat Oberta de Catalunya]. file:///C:/Users/inesb/Downloads/estado%20del%20arte/fnunezsTFM0616memoria.pdf

Rumerhart, D.E., Hinton, G.E. & Williams, R.J. (1986) Learning representations by back-propagating errors. *Nature 323, 533-536.* DOI: https://doi.org/10.1038/323533a0

Sánchez Anzola, N. (2015). Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento USD/COP spot intradiario. *Odeon, 9, 113-172.* DOI: https://doi.org/10.18601/17941113.n9.04

Saini, S. (2021). Supervised vs. Unsupervised Learning [Image].

https://www.linkedin.com/pulse/supervised-vs-unsupervised-learning-whats-difference-smriti-saini

Torres, J. (2021). Introducción al aprendizaje por refuerzo profundo [Imagen].

https://torres.ai/capitulo-1-introduccion-al-aprendizaje-por-refuerzo/

Van Vaerenbergh, I.S. (2018). Métodos kernel para clasificación [Imagen].

https://gtas.unican.es/files/docencia/APS/apuntes/07_svm_kernel.pdf

Wu, Q., Liu, Y., Li, Q., Jin, S., & Li, F. (2017). The Application of Deep Learning in Computer Vision. *Chinese Automation Congress (CAC), IEEE, 6522-6527*.

file:///C:/Users/inesb/Downloads/estado%20del%20arte/The_application_of_deep_learning_in_computer_vision.pdf

GLOSARIO