Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Diseño de una red neuronal para la automatización del proceso de taladrado mediante la estimación de diámetros de avellanado

Autor: Raúl Fernández Valcárcel
Tutor: Alejandro J. Del Real Torres

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

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Diseño de una red neuronal para la automatización del proceso de taladrado mediante la estimación de diámetros de avellanado

Autor:
Raúl Fernández Valcárcel

Tutor:
Alejandro J. Del Real Torres
Profesor Contratado Doctor

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

Sevilla, 2023

Trabajo Fin de Grado: Diseño de una red neuronal para la automatización del proceso de taladrado mediante la estimación de diámetros de avellanado

Autor: Raúl Fernández Valcárcel

Tutor: Alejandro J. Del Real Torres

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

En estos tiempos que corren dar las gracias se ha vuelto un acto superficial. En lugar de expresar auténtica gratitud hacia el prójimo, en muchas ocasiones este acto se realiza de forma automática impulsado por los protocolos sociales, perdiendo su verdadero significado.

Ser agradecido es un acto de humildad que nos recuerda que en este mundo no todo se puede basar en el individualismo, y a su vez es también una forma de valorar y reconocer lo que otras personas hacen por nosotros.

Por estas razones quiero dar mis más sinceros agradecimientos a todas las personas que han formado parte de mi vida durante una etapa tan importante para mí como lo ha sido la Universidad, tanto a los que me siguen acompañando en mi día a día, como a los que por motivos de la vida ya no están a mi lado.

Quería comenzar dando las gracias a mis padres, mis mayores referentes, por vuestro esfuerzo, por vuestro apoyo incondicional, por vuestro cariño… No hay suficientes páginas como para agradecer todo lo que habéis hecho por mí, así que principalmente os daré las gracias por ayudarme a ser la persona que soy hoy en día. También a mi hermana, una de las personas más importantes de mi vida que, a pesar de ser más pequeña que yo, siempre ha estado ahí para escucharme cuando más lo necesitaba. Os quiero.

Continuar agradeciendo a mis amigos, a los de siempre, y a los que he tenido el placer de conocer estos años tanto dentro como fuera de la Escuela. Gracias por acompañarme en estos últimos cuatro años de mi vida ya sea física o telemáticamente, por celebrar conmigo mis logros y, sobre todo, por no darme la espalda en los momentos difíciles. Sois mi mayor tesoro y espero que podamos compartir muchos más momentos juntos.

Agradecer también a todos los docentes que he tenido durante mi formación académica, desde Infantil hasta la Universidad, quienes no solo me habéis brindado vuestra sabiduría, sino que también me habéis enseñado lecciones de vida que me acompañarán por siempre. Especialmente quiero dar las gracias, primero, a Alejandro del Real, quien ha aceptado tutorear este TFG y es una de las personas que me ha ayudado a completarlo; y, después, al Colegio San Juan Bosco, que apostó por mis estudios en un momento complicado de mi vida.

Finalmente, agradecer a mis compañeros del departamento de Ingeniería y Desarrollo de Alestis por ayudarme a dar mis primeros pasos en el mundo laboral. Especial mención a Ignacio, la otra persona que me ha estado ayudando en este trabajo, quien además me ha enseñado muchísimo acerca de la industria y ha despertado en mí el interés por el mundo de la Inteligencia Artificial.

Todos habéis conseguido dejar una huella imborrable en mí que me acompañará durante el resto de mi vida. Os estaré eternamente agradecido.

Raúl Fernández Valcárcel

Sevilla, 2023

Resumen

Dentro de la industria aeroespacial el respeto a las tolerancias es uno de los elementos más críticos a la hora de garantizar no solo la calidad y el correcto funcionamiento de las aeronaves construidas, sino también la seguridad de las personas que las utilizan. En los últimos años la inteligencia artificial se ha abierto paso como una herramienta clave dentro de esta industria, ayudando en labores de inspección y control de calidad encargadas de asegurar la fiabilidad de los productos.

Este documento plantea un enfoque novedoso para optimizar y automatizar el proceso de taladrado de los bordes de ataque del estabilizador horizontal de los aviones de la familia A220 basado en el uso de redes neuronales. Mediante una red neuronal convolucional se estima el diámetro del avellanado de los taladros y, en base a este, se propone una corrección al avance de la unidad de taladrado.

Con esta corrección se pretende evitar que en el proceso de ensamblaje posterior los remaches y los tornillos se encuentren fuera de las tolerancias que dictamina la norma y que, por tanto, sea necesario desechar la totalidad de la pieza.

Actualmente en el proceso es necesaria la intervención humana a mitad del ciclo para realizar mediciones sobre el borde de ataque y corregir manualmente el avance. Con la red neuronal propuesta se busca reducir al mínimo la participación humana en el proceso de taladrado, mejorando así tanto la velocidad de producción y como la calidad del producto.

En este trabajo se ha realizado un estudio exhaustivo del proceso de taladrado y de ensamble del borde de ataque; y de las redes neuronales convolucionales, proponiéndose un modelo creado desde cero que será evaluado con muestras reales obtenidas manualmente.

Abstract

Maintaining precise tolerances is a crucial factor within the aerospace industry, because they ensure the quality and optimal functionality of aircrafts, and also safeguard the well-being of their users. Over the past few years artificial intelligence has emerged as a pivotal asset in this industry, assisting in inspection and quality control tasks that are critical in product reliability.

This document introduces a novel approach to enhance and automate the drilling process of the leading edges of the horizontal tail plane of the A220 family aircraft using neural networks. Through the implementation of a convolutional neural network, an estimation of the diameter of the countersunk holes is obtained. Based on this estimation, a correction to the electric drilling unit's advance is proposed.

This correction aims to prevent the rivets and screws from deviating beyond the tolerances dictated during the subsequent assembly process, thereby avoiding the need to discard the entire part. Currently, human intervention is required midway through the cycle to measure the drills and manually fine-tune the drilling process. The proposed neural network seeks to minimize human involvement in the drilling process, thus enhancing both production speed and product quality.

This work encompasses a comprehensive study of the drilling and assembly process of the leading edge, as well as convolutional neural networks. A model is proposed, built from scratch, which will be evaluated using manually obtained real world samples.

Índice abreviado

Índice

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

1.1. Alestis Aerospace

Alestis Aerospace es una empresa española fundada en 2009. Es proveedora de primer nivel de aeroestructuras para grandes fabricantes del sector aeronáutico internacional como Airbus, Boeing o Embraer; y líder en el ámbito de ingeniería y fabricación de materiales compuestos. Alestis Aerospace cuenta también con la capacidad de diseñar, desarrollar, certificar y entregar en Línea de Montaje Final de manera competitiva y a largo plazo; procesos respaldados por más de 550 certificaciones de todo tipo. [1]

Dentro de la diversidad de partes que pasan por diferentes procesos en las plantas de Alestis Aerospace, podemos destacar el HTP del Airbus A320, los flaps y el alerón del Boeing B777, o la wing tip del Embraer E190 y el Embraer E195, entre otras. Dichos procesos se llevan a cabo en varias plantas a nivel nacional, como la de San Pablo (Sevilla) o la de Vitoria; e internacional, en São Paulo (Brasil).

1.1.1 Ingeniería, desarrollo e innovación

Entre los distintos departamentos de Alestis Aerospace, se encuentra el departamento de Ingeniería y Desarrollo, el cual se encarga de la automatizar procesos manuales y de proponer soluciones tecnológicas a los distintos problemas a los que se enfrenta la empresa. El rango de trabajo del departamento abarca áreas bastante amplias, desde la programación de PLCs hasta el montaje eléctrico de un robot manipulador, pasando por la optimización de procesos automáticos y el desarrollo software.

Algunos de los proyectos en los que se encuentra trabajando actualmente el departamento son la inspección automática de las tapas del A320 o el taladrado automático del borde de ataque del HTP del A220, en el cual nos centraremos en este documento. En ambos proyectos se han propuesto soluciones innovadoras fundamentadas en la programación de robots y en el procesado de datos mediante inteligencia artificial con el objetivo de maximizar la calidad de la producción de la empresa.

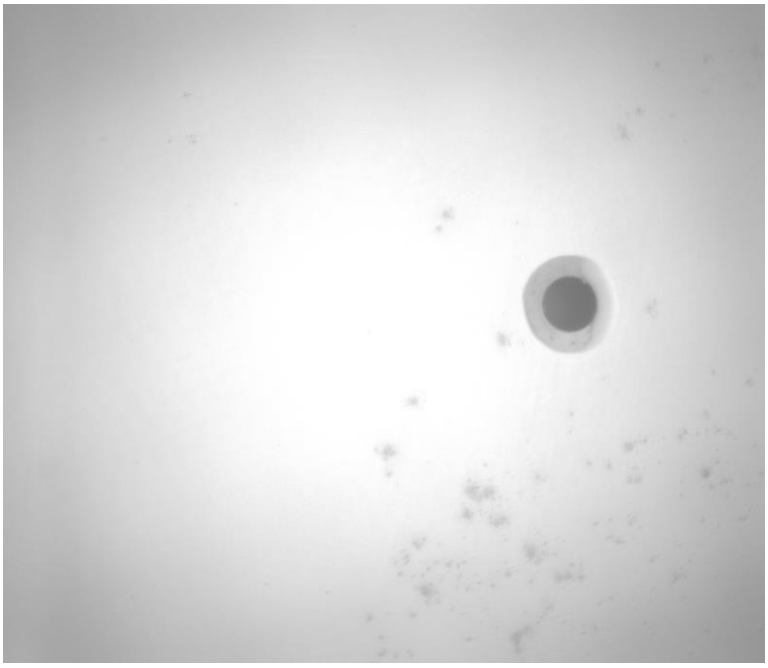

Figura 1-1. Fotografía de defectos de fabricación de una tapa de A320 obtenida por el robot de inspección

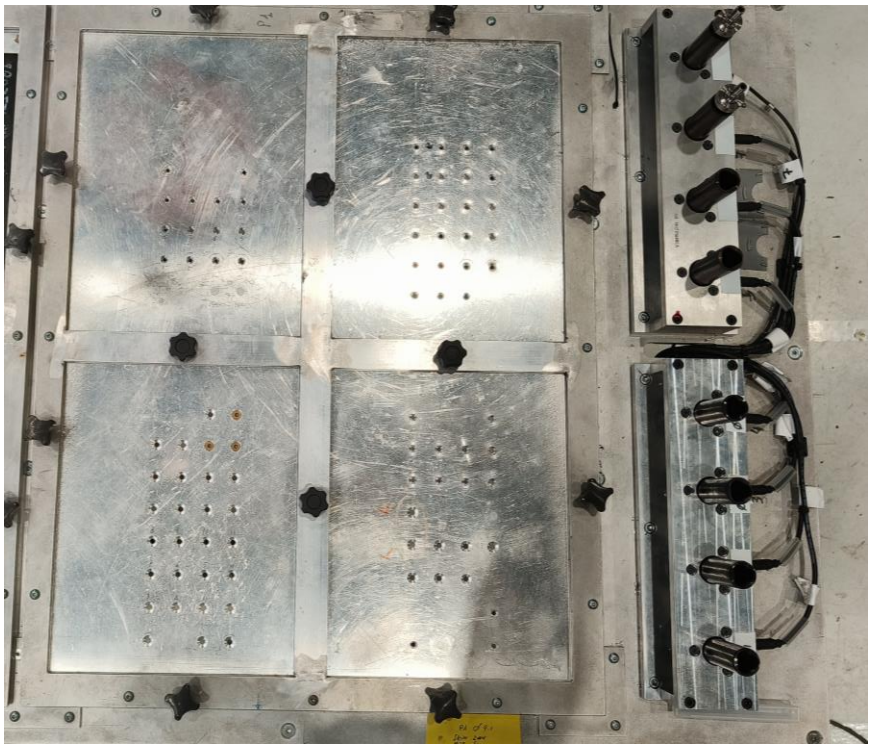

Figura 1-2. Probetas de taladrado realizadas en la célula de taladrado

1.2. Familia Airbus A220

El Airbus A220, anteriormente conocido como Bombardier C Series, es una familia de aviones a reacción bimotor de fuselaje estrecho y pasillo único, diseñados para vuelos comerciales de entre 100 y 160 pasajeros [2]. Algunas de las aerolíneas más famosas que operan con estos aviones son: Delta Air Lines, con 62 aviones; airBaltic con 42; y Air Canada con 33 [3]. Actualmente Alestis Aerospace tiene un contrato con Airbus para la producción de algunas piezas de esta familia de aeronaves.

El programa C Series fue lanzado julio de 2008, siendo a su vez el último programa impulsado por la empresa canadiense Bombardier. El primer avión de este programa que realizó pruebas de vuelo fue el más pequeño de los dos modelos existentes, el CS100 (predecesor del actual A220-100), y lo hizo en 2013. Su hermano mayor, el CS300, realizó su primer vuelo dos años después en 2015, aunque no sería hasta 2016 que ambos comenzarían a operar con normalidad [4].

Dentro de las variantes para vuelos comerciales se encuentran los antes mencionados A220-100 y A220-300, siendo el segundo modelo de mayor tamaño y capacidad. La autonomía de estos es de unas 3200 millas náuticas aproximadamente (alrededor de los 6000 km) [5], lo cual hace ideal su uso para vuelos de media y corta distancia a nivel nacional e internacional.

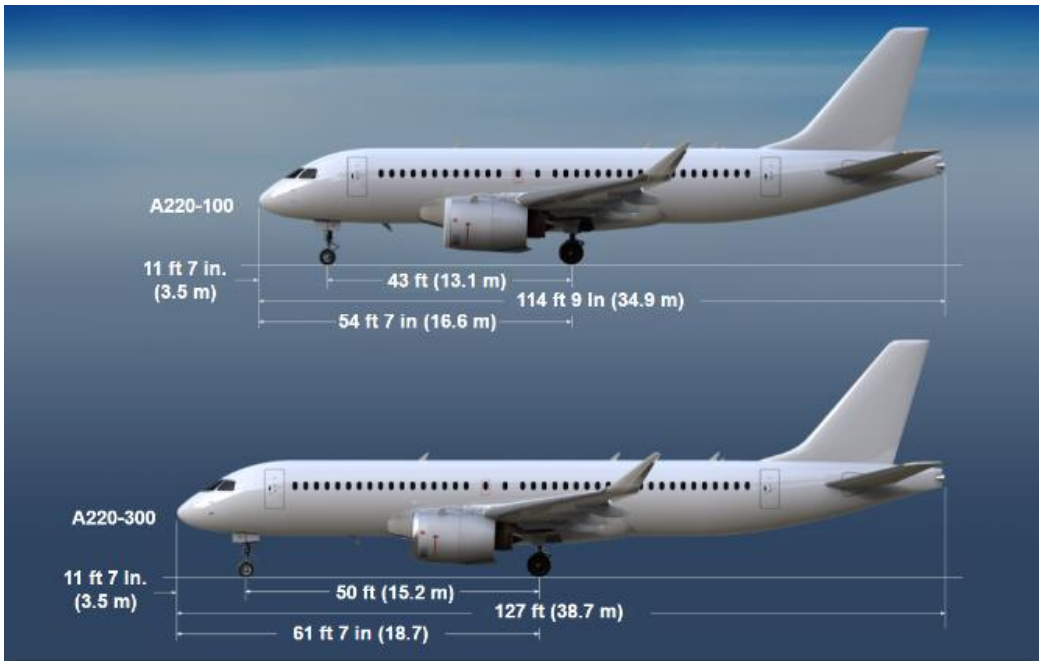

Figura 1-3. Comparación de las dimensiones del A220-100 y del A220-300 [6]

Recientemente Airbus ha planteado la idea de fabricar un nuevo modelo de aeronave para la familia A220, el A220-500, una versión aún más grande y con mayor capacidad que el A220-300 que pretende sustituir al producto insignia de la compañía, el A320, ofreciendo notables mejoras en sus prestaciones. A su vez, este nuevo modelo le serviría a Airbus para competir con el Boeing 737 Max por el dominio del mercado de aviones comerciales de pasillo único [7].

Tal como se ha indicado previamente, la propulsión de estos aviones es llevada a cabo por un par de motores, los dos de tipo turbofán. Ambos motores son del mismo modelo, Pratt & Whitney PurePower PW1500G, y están especialmente diseñados para esta familia de aviones. Estos motores son una evolución de los utilizados en modelos anteriores; puesto que con su uso se ha conseguido una mejora sustancial del consumo de combustible y una reducción tanto de las emisiones como del ruido de operación, haciendo los vuelos más eficientes y respetuosos con el medio ambiente.

Otra de las ventajas a destacar de la familia A220 es que los dos modelos comparten casi la totalidad del diseño, permitiendo a las aerolíneas tener menor cantidad de repuestos en inventario y disminuyendo por ello tanto la inversión inicial que implica adquirir varios de estos aviones, como el coste de mantenimiento del almacén.

1.3. Ensamblaje de una aeronave

El ensamblaje de una aeronave es el proceso mediante el cual se unen las diferentes partes y piezas de un avión (alas, cola, fuselaje…) para completar su estructura. Es una de las partes más importantes en la fabricación de una aeronave, puesto que un correcto ensamblaje asegura la integridad estructural del avión, haciéndolo un vehículo funcional y seguro para el transporte de pasajeros y mercancías.

Durante el vuelo, un avión se ve sometido a una gran cantidad de vibraciones debido a las fuerzas generadas por el aire. Estas vibraciones son un problema muy serio en el diseño aeronáutico, ya que pueden tener consecuencias muy negativas sobre la aeronave como por ejemplo el aleteo o flutter, una serie de vibraciones que a veces aparecen en las alas o los estabilizadores del avión y que pueden llegar a provocar fracturas, poniendo en riesgo la integridad de la estructura [8].

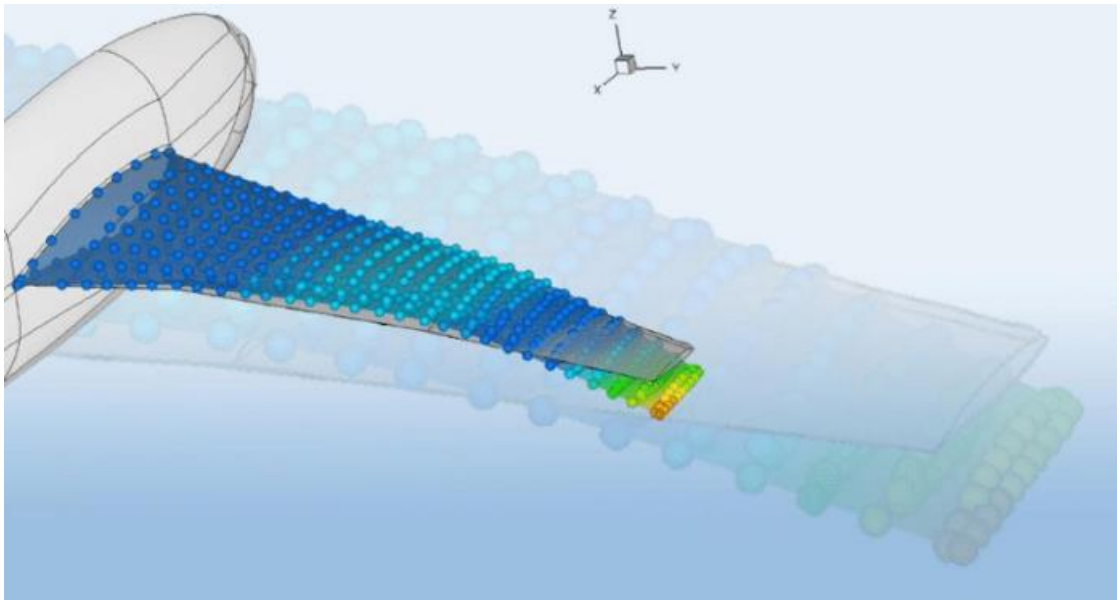

Figura 1-4. Simulación de la deformación provocada en el ala de un avión debido al flutter [9]

Dichas vibraciones son uno de los motivos que descartan la soldadura como método de unión entre las partes estructurales de la aeronave, puesto que la propia soldadura es un punto poco flexible y débil por el que, a determinadas frecuencias en altas velocidades de vuelo, la unión podría fracturarse.

Además de por las vibraciones, otra de las razones por la que no se utiliza la soldadura es por los materiales usados en la construcción. La mayoría de los aviones comerciales tienen un cuerpo de aluminio para hacerlos más ligeros y así consumir menos combustible que otros aviones más pesados construidos con otros metales. El problema radica en que el aluminio se ve muy debilitado al exponerse a altas temperaturas, como es en el caso de la soldadura, por lo que las empresas suelen optar por otro tipo de ensamble.

1.3.1 Ensamblaje por remachado y atornillado

En la actualidad, la industria aeronáutica utiliza el remachado como método de unión de las piezas estructurales de sus aeronaves. El remache sólido tradicional es el elemento de unión que más se utiliza en aeronáutica, estando formado este por dos partes: Un extremo denominado cabeza, y el otro llamado vástago, el cual tiene al final lo que se conoce como la cola del remache. La característica principal de estos remaches es que son macizos y de un único material.

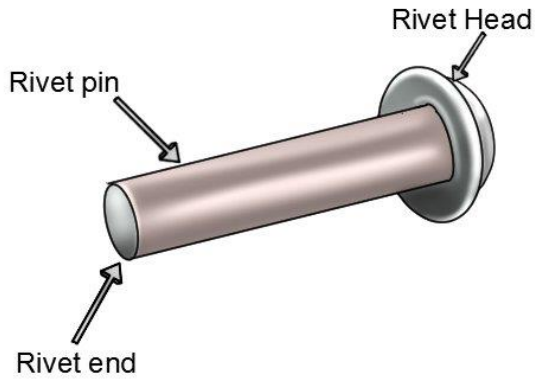

Figura 1-5. Partes de un remache sólido tradicional [10]

La ventaja principal del remachado con respecto a la soldadura es que no necesita la aplicación de calor para la unión de dos piezas, por lo que estas pueden conservar sus propiedades de flexibilidad y resistencia; siendo los únicos daños aplicados sobre el material los asociados al taladrado y a la fuerza aplicada durante el proceso de remachado

Otras de las ventajas de remachar a destacar son el bajo coste del material, la posibilidad de unir dos materiales de distinta naturaleza, o la facilidad que ofrece para comprobar que la unión creada cumple con la norma en labores de inspección de calidad.

Como es de esperar, este método también presenta ciertas desventajas. Uno de los problemas que aparecen debido a la gran cantidad de remaches que hay que utilizar en el ensamblaje, es el peso que estos aportan a la estructura de aluminio del avión, el cual no es despreciable.

Asimismo, es necesario tener en cuenta que los taladros que hay que realizar sobre el material generan tensiones indeseadas. De hecho, la zona de remachado se diseña de tal forma que no exista riesgo de rotura o de agrietamiento por dichas tensiones durante el resto del montaje o durante el vuelo.

El proceso de remachado aeronáutico es sencillo, aunque requiere de cierta habilidad si se hace manualmente: Primero se alinean los taladros de dos piezas, después se introduce el remache hasta que la cabeza se encuentre apretada contra la pieza y, finalmente, se hace fuerza sobre la cola del vástago del remache hasta deformarlo, consiguiendo la unión permanente de las dos partes.

Es importante tener en cuenta que el borde de ataque con el que vamos a trabajar también se somete a un proceso de atornillado que permite a los aviones de la familia A220 su intercambio en caso de rotura, evitándose así la necesidad de desechar el HTP al completo. El atornillado se ayuda de una tuerca para la fijación, y se realiza manualmente con un destornillador y una llave inglesa, la cual tiene un detector de par integrado que ayuda al operario a identificar si el tornillo está lo suficientemente apretado.

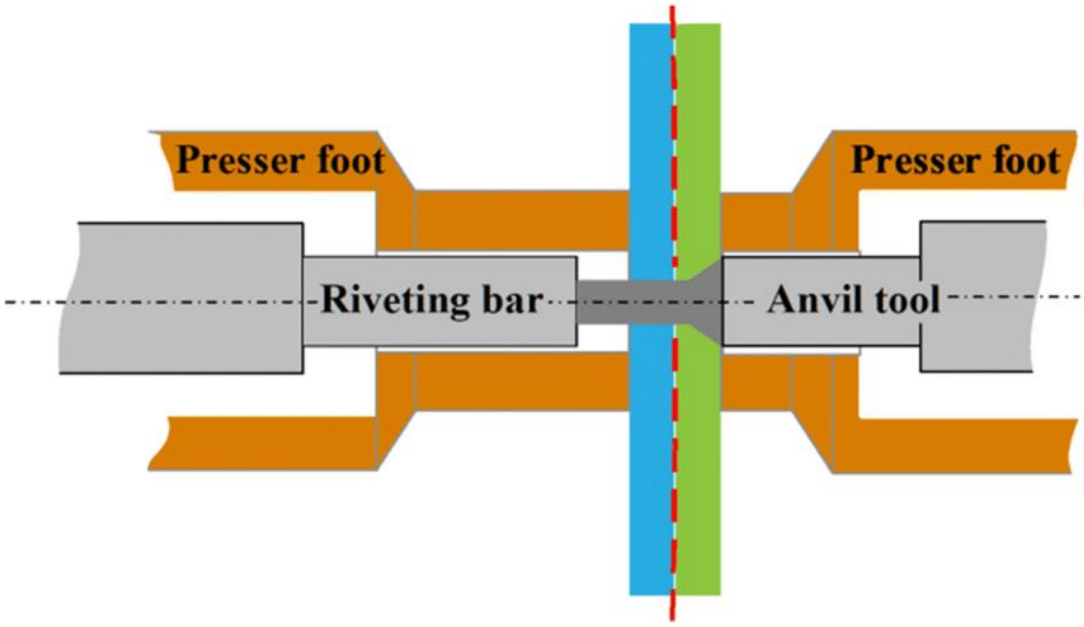

Figura 1-6. Esquema del proceso de remachado de dos piezas planas [11]

1.4. Identificación y delimitación del problema

Como ya mencionamos anteriormente, Alestis Aerospace es la encargada de producción de algunas de las piezas de la familia del A220. La planta de San Pablo en Sevilla tiene asignada la construcción del borde de ataque del HTP de estas aeronaves.

Este documento abarca varios aspectos cruciales del proceso de fabricación: El taladrado avellanado del borde de ataque, y su ensamblaje mediante remaches y tornillos.

1.4.1 Borde de ataque del estabilizador horizontal

El estabilizador horizontal o plano de profundidad de un avión (HTP por sus siglas en inglés) es una especie de ala que se coloca en la cola de los aviones y se encarga de producir el movimiento de cabeceo, es decir, la rotación con respecto al eje transversal de la aeronave. Habitualmente el HTP está formado por una parte fija delantera llamada plano fijo horizontal, y una parte trasera llamada timón de profundidad [12], [13].

En aviones comerciales de gran tamaño suele utilizarse un tipo de estabilizador denominado estabilizador móvil, el cual varía la incidencia de la parte delantera del HTP para compensar el avión de forma automática, permitiendo mantener la altura de vuelo sin una intervención constante del piloto.

El borde de ataque es la parte del ala de una aeronave que primero contacta con la corriente de aire. A veces incluye dispositivos hipersustentadores para aumentar su curvatura, como es el caso de los flaps o los slats. Este juega un papel crucial en el consumo y en la aerodinámica de la aeronave, ayudando a generar la sustentación necesaria para que esta se mantenga en vuelo y reduciendo la resistencia del aire. Por estas razones es muy importante que la pieza final cumpla con las tolerancias impuestas por la norma.

Figura 1-7. Borde de ataque del ala baja del Sling 2 [14]

1.4.2 Procesos de taladrado y de ensamble

El taladrado y el avellanado son llevados a cabo de forma simultánea y automática en una célula de trabajo mediante la taladradora S500 de AEI, la cual está acoplada a un robot de seis ejes KR 120 2700-2 de la marca KUKA. Durante este proceso llamado one-shot drilling, el borde de ataque se fija sobre un raíl y es el robot es el que se desplaza hacia los puntos de acción previamente definidos por el operario.

Figura 1-8. Robot KR 120 2700-2 de KUKA [15]

Figura 1-9. Taladradora S500 de AEI [16]

Debido a la longitud de la pieza con la que se trabaja, el robot cuenta con la ayuda de la KL 4000, una unidad lineal de desplazamiento, también de la marca KUKA, la cual se encarga de desplazar la base del robot manipulador funcionando como un eje adicional.

Figura 1-10. Unidad lineal KL 4000 como eje adicional de un robot KUKA [17]

Los taladros avellanados del borde de ataque del A220 se realizan con dos brocas: Una de 4.1 mm de diámetro para los remaches y otra de 4.9 mm para los tornillos. Ambas tienen un diseño personalizado preparado para el taladrado y avellanado simultáneos.

En la Figura 1-11 podemos observar uno de los dos tipos de broca utilizados, y en la Figura 1-12 un ejemplo de taladro avellanado resultante.

Figura 1-11. Broca de 4.1 mm de diámetro utilizada para el taladrado avellanado del A220

Figura 1-12. Ejemplo de taladro avellanado realizado sobre una probeta

Una vez acabado el proceso de taladrado, se procederá a la colocación manual de los remaches y los tornillos. Para el remachado se utilizan remaches MS20426 (ver Figura 1-13), un tipo de remache sólido avellanado estandarizado en la industria aeronáutica militar desde hace bastante tiempo. Son remaches pequeños, con un ángulo de avellanado de 100º y una composición de aluminio.

En cuanto al tipo de tornillo utilizado, este es administrado por Airbus,y no se han podido extraer sus características.

Figura 1-13. Remache MS20426

1.4.2.1 Tolerancias del proceso de ensamble

Para el ensamble del A220 existe una norma específica que recoge la tolerancia de altura permitida para los tornillos y los remaches con respeto a la superficie de la pieza:

- En el caso de los tornillos, estos no pueden sobresalir ni una micra por encima de la pieza, y pueden estar un máximo de 127 micras por debajo.
- En cuanto a los remaches, estos pueden sobresalir hasta 51 micras por encima de la superficie, pero no pueden estar hundidos.

Esta información sirve para definir qué avellanados son considerados conformes y cuáles no, puesto que un avellanado demasiado grande en un taladro de 4.1 mm haría que el remache se hundiera, y un avellanado demasiado pequeño de un taladro de 4.9 mm haría que el tornillo sobresaliera.

Con las otras dos condiciones se puede estimar el diámetro mínimo y máximo que puede tener el avellanado de las brocas de 4.1 mm y 4.9 mm respectivamente.

1.4.3 Defectos en el proceso de taladrado

Como se puede intuir, lo óptimo es que los remaches y los tornillos queden a la altura de la superficie aerodinámica. Una forma de comprobar que todo se ha hecho correctamente tras el ensamble es pasar la mano por encima y notar la pieza prácticamente lisa. Teniendo en cuenta las tolerancias requeridas y la gran cantidad de taladros necesarios, conseguir un diámetro de avellanado óptimo para el ensamble puede llegar a ser complicado.

En nuestro caso, debido a que las posiciones de taladrado no se encuentran en el mismo plano por la propia forma de la pieza, cada desplazamiento del robot entre posiciones de taladrado va acumulando pequeñas imprecisiones que acaban afectando al diámetro del avellanado.

Como hemos mencionado anteriormente, cuando un avellanado no está hecho con suficiente precisión, el remachado y atornillado posterior puede no cumplir con las tolerancias de altura de la norma.

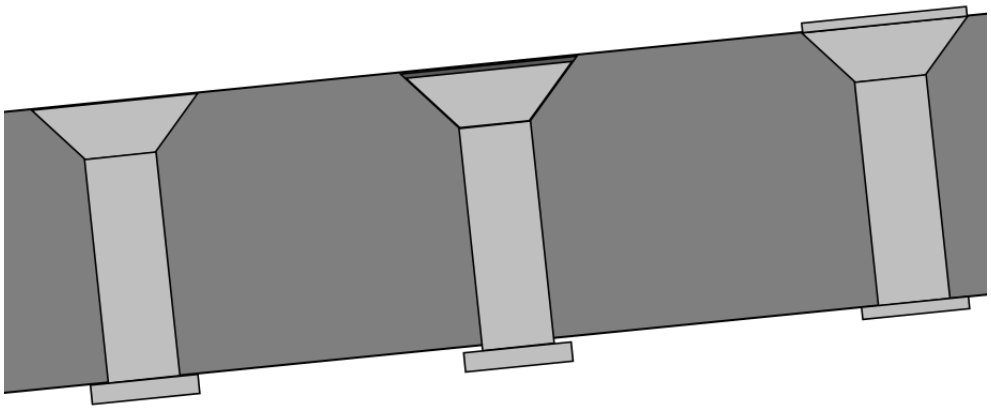

Figura 1-14. Ejemplos de un remache conforme, pasado y saliente, respectivamente

Este fallo en la fabricación da lugar a un mal reparto de tensiones que a su vez puede provocar fracturas o pérdidas de flexibilidad en la pieza, poniendo en riesgo tanto el rendimiento como la seguridad de la aeronave.

Ahora mismo el proceso necesita un operario que cada cierto tiempo mida los avellanados realizados por el robot para comprobar sus tamaños y evitar problemas en el ensamble posterior. Si en algún momento detecta que los avellanados superan ciertos límites, este se encarga de aplicar manualmente una corrección en el proceso.

1.5. Corrección en el avance de la taladradora

Actualmente, las brocas utilizadas en el taladrado avellanado del borde de ataque tiene un largo de cuerpo de 16 mm con 2 décimas de milímetro de incertidumbre (datos del fabricante).

Esta incertidumbre afecta directamente a la dimensión del avellanado: Una broca de 16.2 mm de largo de cuerpo necesita avanzar más distancia que una broca de 15.8 mm de largo para conseguir un avellanado del mismo diámetro.

La EDU (Electric Drilling Unit) tiene parametrizado que el avance al realizar el taladro sea fijo e igual a 16 mm, lo que arrastraría una posible incertidumbre en el largo de cuerpo de la broca a los diámetros del avellanado. La idea es aplicar pequeñas correcciones a ese parámetro durante el ciclo de taladrado. Dicha corrección se haría justo antes de realizar cada taladro y se basaría en el diámetro del avellanado del taladro anterior.

Este trabajo aborda pues la problemática existente con las tolerancias del proceso de ensamble del borde de ataque del HTP del A220 debida a las incertidumbres del proceso de taladrado. Para hacer frente al problema se propone el diseño de una red neuronal convolucional capaz de estimar mediante fotografías el diámetro de avellanado de un taladro que será utilizado para ofrecer correcciones en el avance durante el proceso de taladrado.

2 DEEP LEARNING Y REDES NEURONALES

2.1. Historia de la Inteligencia Artificial

El concepto de inteligencia artificial nace a mitades del siglo pasado gracias a grandes mentes como la del matemático Alan Turing, quien planteó en [18] la idea de crear máquinas inteligentes capaces de actuar como un ser humano a la hora de resolver problemas y tomar decisiones. La gran limitación que impidió a los investigadores de la época poner en práctica sus novedosas ideas fue la escasa capacidad computacional y de almacenamiento de sus ordenadores [19], [20], [21].

Aun así, cinco años después de estos primeros planteamientos, Allen Newell, Cliff Shaw y Herbert Simon consiguieron crear lo que muchos consideran el primer programa de inteligencia artificial de la historia: Logic Theorist, el cual era capaz de probar los complejos teoremas del *Principia Mathematica* aplicando una lógica similar a la que aplicaría un ser humano.

Los años seguían pasando y los ordenadores seguían evolucionando: Aumentaron su capacidad de almacenamiento, redujeron su coste, mejoraron su potencia computacional… Todo esto favoreció a que, en la segunda mitad de siglo, la IA impulsara su desarrollo. Para el cambio de milenio se habían alcanzado muchos logros en este ámbito, como por ejemplo que en 1997 la computadora Deep Blue de IBM fuera capaz de vencer al campeón del mundo Gary Kasparov en una partida de ajedrez.

A principios de los 2000 volvió a aparecer el mismo problema limitante, ya que no existía ni potencia ni almacenamiento suficientes como para poder innovar con el uso de la IA. No es hasta hace unos años que los ordenadores han logrado sobrepasar la barrera computacional y de almacenamiento que estancó el desarrollo de la IA a principios de siglo.

Actualmente los algoritmos de inteligencia artificial se han vuelto una parte fundamental en la industria, pues gracias a ellos las grandes empresas pueden hacerse cargo de la gran cantidad de información que generan y reciben cada día.

En la Figura 2-1 podemos observar como con el tiempo muchas de las funcionalidades que ofrece la inteligencia artificial han ido mejorando hasta llegar incluso a superar las capacidades humanas, rindiendo especialmente bien en tareas como la comprensión lectora o el reconocimiento de patrones en imágenes.

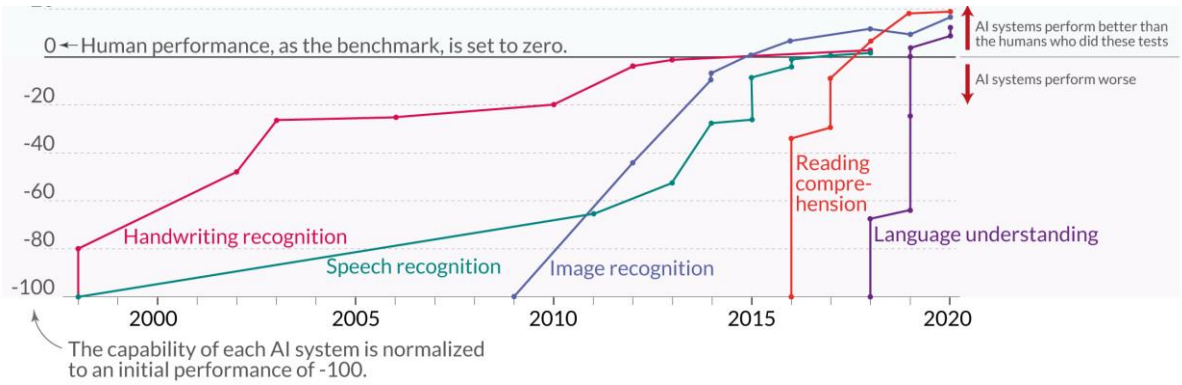

Figura 2-1. Evolución de algunas funcionalidades de la IA respecto a la capacidad humana [22]

2.2. Origen del Machine Learning

El aprendizaje automático o machine learning es una subcategoría de la IA que tiene como objetivo hacer que un ordenador aprenda, permitiéndole así proponer soluciones generales a partir de casos particulares .

Su historia también se remonta a los años cincuenta, con la creación del famoso Test de Turing y la publicación de artículos como [23], donde se expuso por primera vez un modelo matemático para definir el funcionamiento neuronal.

El primer programa de aprendizaje fue creado en 1952 por Arthur Samuel. Este programa permitía introducir a un ordenador jugadas óptimas para una partida de damas entre las que el computador podría elegir para intentar ganar. Más tarde en 1957 Frank Rosenblatt descubrió el perceptrón, el origen de las redes neuronales modernas.

Durante el resto del milenio se alcanzaron otros grandes hitos como la creación del algoritmo de clasificación Nearest Neighbor [24], y se comenzaron a plantear los primeros algoritmos capaces de gestionar grandes cantidades de datos.

En la última década grandes compañías incorporaron el machine learning en productos para el público general. Es el caso de Microsoft, que en 2010 mostró por primera vez al mundo Kinect, una novedosa tecnología basada en aprendizaje automático capaz de interpretar gestos y movimientos del usuario para después realizar acciones en tiempo real sobre un ordenador.

La última gran novedad en machine learning es bastante reciente. Se trata de GPT-3, un algoritmo de deep learning (subcategoría del machine learning) de la compañía OpenAI capaz de procesar un prompt de texto para crear recetas, escribir poesía o incluso programar; todo ello imitando el proceso mental por el que pasaría un ser humano al realizar estas tareas.

2.2.1 Tipos de Aprendizaje Automático

En el ámbito del machine learning, podemos encontrarnos con cuatro grandes tipos de aprendizaje, que son el aprendizaje supervisado, el aprendizaje semisupervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo [25].

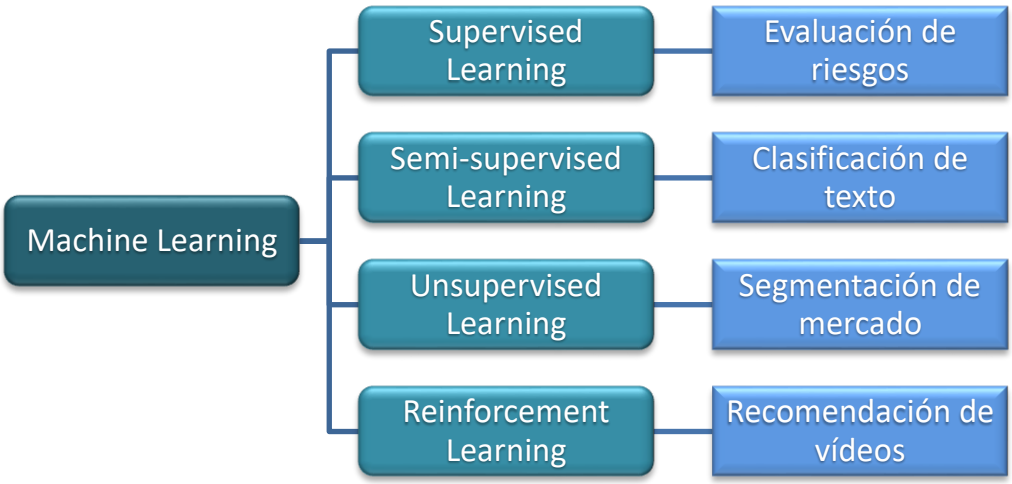

Figura 2-2. Tipos de aprendizaje automático junto a posibles funcionalidades

El objetivo de todos ellos es analizar los datos de entrada, aprender de ellos y estimar una serie de valores de salida dentro de un rango aceptable.

- **Aprendizaje supervisado**: El computador es enseñado a través de ejemplos, es decir, el usuario introduce al algoritmo un conjunto de datos con las entradas y salidas deseadas, y el algoritmo se encarga de encontrar una forma de conectarlas. Este aprendizaje pues, analiza los datos en busca de patrones para ofrecer predicciones que el usuario deberá ir corrigiendo para mejorar la precisión del algoritmo. Entre los algoritmos de aprendizaje supervisado tenemos:
    - **Clasificación**: Dada una serie de clases de salida y un dato de entrada, el algoritmo se encarga de asignar a dicha entrada una clase. Se podría hacer un clasificador que determinara si una imagen de un pastor alemán pertenece a la clase perro o a la clase gato.
    - **Regresión**: Los algoritmos de regresión estudian la relación entre las diferentes variables que componen de los datos de entrada, permitiendo así la estimación de la salida. Este tipo de algoritmos suele utilizarse para predicciones, por ejemplo, para estimar el precio de una noche en un hotel dando como parámetros de entrada las comodidades que ofrece.
- **Aprendizaje semisupervisado**: Se encuentra a medio camino entre el aprendizaje supervisado y el no supervisado, ya que una parte de los datos de entrada tienen etiquetas asignadas con la información clave para el aprendizaje y la otra parte no. Así, se dota al algoritmo con la capacidad de etiquetar datos en base a las etiquetas del entrenamiento.
- **Aprendizaje no supervisado**: En este caso el ordenador analiza los datos en búsqueda de patrones sin ningún tipo de referencia previa. El algoritmo va encontrando relaciones entre numerosas variables y trata de organizar los datos bien por agrupación (clustering) o bien por asociación.
    - **Clustering**: La organización de los datos se realiza en base a similitudes y diferencias. Por ejemplo, una tienda podría agrupar a sus clientes en diferentes clústeres con recomendaciones de productos específicas basadas en los hábitos de compra y la información personal aportada por el consumidor.
    - **Asociación**: En este caso, la organización de los datos es realizada mediante relaciones de interés entre los diferentes datos de entrada. Siguiendo con el ejemplo de la tienda, este algoritmo podría dar información interesante para la toma de decisiones de la empresa, como por ejemplo que dos productos se suelan comprar juntos.
- **Aprendizaje por refuerzo**: El aprendizaje en este caso se basa en dar "premios" y "castigos" al algoritmo en base a si ha tomado una buena o una mala decisión. Puede interpretarse como un aprendizaje por prueba y error donde el algoritmo se encarga de maximizar la recompensa. Este tipo de aprendizaje es el que suele utilizarse para ofrecer anuncios personalizados en Internet.

2.2.2 Deep Learning

El aprendizaje profundo es un campo dentro del machine learning que permite a un sistema estructurar, procesar y extraer de forma automática la información más útil de un gran conjunto desestructurado de datos de entrada de forma similar a como lo haría un cerebro humano, eliminando así el ajuste manual de las entradas que sí necesita el aprendizaje automático [26], [27].

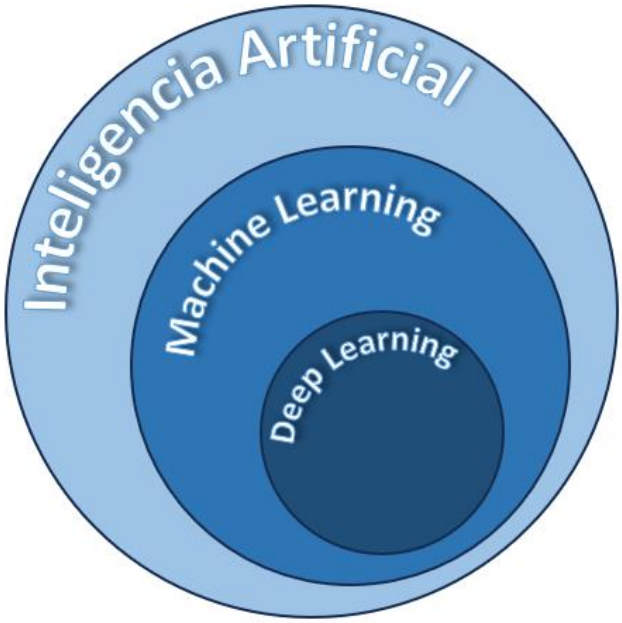

Figura 2-3. Inteligencia artificial, machine learning y deep learning

Este tipo de aprendizaje también requiere de entrenamiento para su optimización, pero en este caso la base de datos de entrada y la capacidad computacional de los equipos han de ser mucho mayores que en el machine learning para obtener los resultados esperados.

Actualmente el deep learning es una de las áreas más prometedoras de la inteligencia artificial y está aportando notables avances en todo tipo de ámbitos, como por ejemplo en el de la medicina. Entre algunas propuestas interesantes encontramos MedWhat [28], un asistente virtual con fines meramente educativos que gracias al deep learning aprende sobre medicina día a día y es capaz de, en base al historial médico del usuario, ofrecer respuestas a las dudas médicas que se le propongan.

2.3. Redes neuronales artificiales

Las redes neuronales artificiales (ANN por sus siglas en inglés), son la base de los algoritmos de deep learning. Estando inspiradas en el funcionamiento del cerebro humano, están formadas por un gran número de nodos interconectados entre sí llamados neuronas que se utilizan para aprender de los datos de entrada con el fin de ofrecer la mejor salida posible.

Una red neuronal bien entrenada permite un procesamiento veloz de grandes cantidades de información, mejorándose las prestaciones de procesos lentos y complejos como la segmentación de datos o el reconocimiento de voz.

2.3.1 Estructura de una red neuronal

Una red neuronal está dividida en capas donde se distribuyen las neuronas. Siempre hay una capa de entrada, una capa de salida, y una o varias capas ocultas intermedias; y todas ellas tienen sus neuronas interconectadas entre sí. Dichas neuronas estás formadas por pesos, sesgos y umbrales: Los pesos se encargan de atribuirle importancia a cada una de las características de la entrada, el sesgo o bias es un offset que se le da a la salida de la neurona para su procesado posterior, y el umbral es un límite puesto a dicho valor de salida que decide si transmitir o no la salida a las capas siguientes [26], [29].

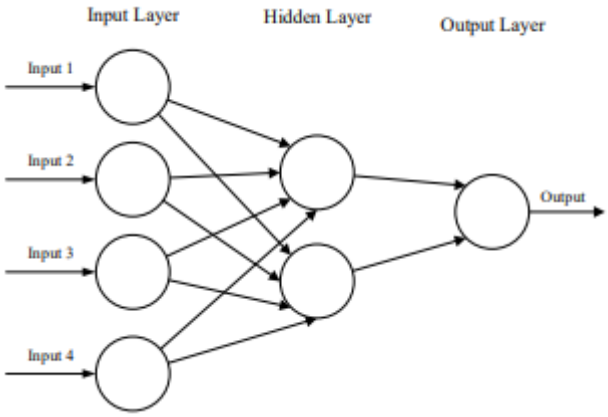

Figura 2-4. Ejemplo de estructura básica de una red neuronal de tipo feedforward [30]

El incremento del número de capas y de neuronas aumenta el coste computacional de la red al añadir más parámetros que optimizar, aunque a veces con ello se logran mejores resultados,

2.3.1.1 Función de activación

Las funciones de activación son una de las partes fundamentales de una red neuronal, ya que son las encargadas de dar forma a la información que se transmite entre las neuronas. Esto permite añadir efectos de no linealidad a la red que sirven para aumentar su complejidad y para que no se adapte en exceso a los datos de entrenamiento. A su vez, nos permiten decidir qué tipo de resultado queremos que devuelva la red [31].

Otra de las cualidades de la función de activación es establecer el rango de salida de las neuronas mediante el umbral que mencionábamos. Si la salida de la neurona no es acotada por la función de activación y toma un valor excesivamente grande, la información que se transmitiría a la siguiente capa causaría problemas en el aprendizaje de la red y provocaría fallos en su funcionamiento.

En la Figura 2-5 se muestran algunas de las funciones de activación más extendidas.

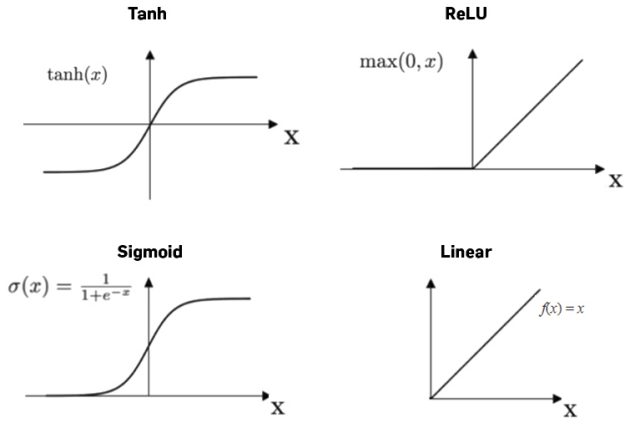

Figura 2-5. Funciones de activación [32]

2.3.2 Entrenamiento de una red neuronal

Al comenzar el entrenamiento de una red neuronal, la distribución de pesos y sesgos se realiza de forma aleatoria, por lo que los resultados que se obtienen al principio no son nada precisos, siendo necesario un método de ajuste de pesos y sesgos para que la red devuelva mejores resultados. Este proceso es lo que se conoce como entrenamiento de una red neuronal.

Para que el entrenamiento funcione, hemos de definir una medida a optimizar. Dicha medida es la función de coste o función de error, la cual devuelve un valor numérico que indica la discrepancia existente entre el resultado esperado y el devuelto por la red. El objetivo del entrenamiento es minimizar dicho valor: Si la función de coste devuelve valores demasiado altos que no se reducen durante el entrenamiento, el aprendizaje no se está realizando correctamente y, por tanto, hay que cambiar el enfoque del entrenamiento. La función de coste más utilizada es el error cuadrático, y será la métrica que minimizaremos en nuestra red neuronal.

2.3.2.1 Backpropagation y descenso de gradiente

Cuando minimizamos la función de coste, el aprendizaje de la red se optimiza, alcanzando sus mejores resultados en el mínimo absoluto de la función de coste. El método más extendido para reajustar los parámetros de una red neuronal es el de backpropagation o propagación inversa, el cual hace que el error obtenido recorra la red al revés para que esta vaya calculando mediante la regla de la cadena y el valor de sus pesos la dirección del gradiente que hace mínimo el error [33], [34].

Las direcciones de gradiente obtenidas son utilizadas por el método de descenso de gradiente para obtener los nuevos valores de los pesos asociados a las neuronas de la red mediante la siguiente fórmula:

$$w_{i+1} = w_i - \alpha \frac{dE}{dw_i}$$

Siendo w_{i+1} los nuevos pesos calculados, w_i el valor actual de los pesos, $\frac{dE}{dw_i}$ el gradiente del error correspondiente a los pesos actuales, y α un hiperparámetro conocido como tasa de aprendizaje o learning rate que define el tamaño del "salto" que se realiza en la optimización.

A mayor sea dicha tasa de aprendizaje, más rápido será el entrenamiento de la red, aunque es importante no elegir un valor excesivamente alto para evitar que la convergencia sea imposible o peor aún, que la optimización diverja. Un learning rate pequeño asegura la convergencia en el mínimo de una función estrictamente convexa, pero en funciones con mínimos locales o puntos de silla puede estancar el entrenamiento.

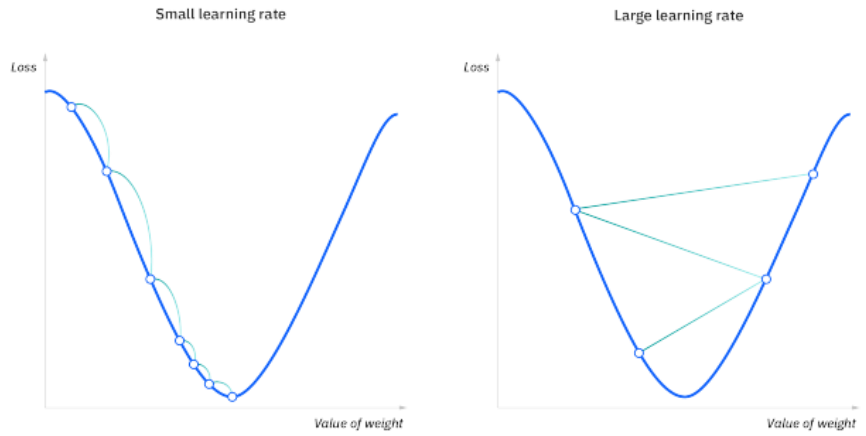

Figura 2-6. Diferencia entre usar un learning rate pequeño o grande en el descenso de gradiente [35]

Existen tres tipos de algoritmos de descenso de gradiente:

- **Gradiente descendiente batch o por lotes:** Este algoritmo suma el error resultante de cada dato del set de entrenamiento y, tras evaluarlo por completo, actualiza los parámetros. Cada ciclo completo de entrenamiento se denomina epoch.
- **Gradiente descendiente estocástico:** En este caso, se va actualizando la red cada vez que se procesa un dato del set. El procedimiento es considerablemente más lento que el método por lotes, que solo actualiza los parámetros una vez, pero a su vez requiere de menos potencia computacional.
- **Gradiente descendiente mini batch:** Es una mezcla entre los dos algoritmos anteriores que divide el set de entrenamiento en varios lotes más pequeños y actualiza la red lote a lote. Suele ser el algoritmo más usado ya que con él puede conseguirse un equilibrio entre la velocidad de entrenamiento del método por lotes, y el reducido coste computacional del método estocástico.

2.3.3 Desempeño de una red neuronal

Una vez finalizado el entrenamiento, existe la posibilidad de que la red no procese correctamente los datos del set de testeo, haciendo que los resultados no cumplan las expectativas. Hay varios problemas que pueden afectar al funcionamiento de la red, como el overfitting y el underfitting, de los que hablaremos a continuación.

2.3.3.1 Problemas de sobreajuste y subajuste en una red neuronal

El sobreajuste y subajuste (overfitting y underfitting, respectivamente) son dos de los problemas más recurrentes en el entrenamiento de una red neuronal. Cuando la red se ajusta demasiado poco o en exceso a un set de entrenamiento, puede presentar problemas al generalizar con nuevas entradas. Para entender la problemática, se propone como ejemplo una red neuronal que clasifique figuras dependiendo de si son o no rectangulares.

Imaginemos que la red es entrenada con una única foto de un rectángulo aleatorio, y que a la hora de probarla utilizamos otro rectángulo de dimensiones distintas y girado. Debido a la falta de ejemplos, la red no será capaz de reconocer la forma, devolviendo como resultado que esta es no rectangular. Esto se conoce como underfitting.

Por otro lado, planteemos una red entrenada con muchas fotos de rectángulos durante mucho tiempo, de tal manera que el error en el entrenamiento sea prácticamente inexistente. Si ahora para probar nuestra red utilizamos un rectángulo con una proporción o una orientación distintas a las entrenadas, la red lo catalogará como forma no rectangular, puesto que no cumple estrictamente con las características de los datos de entrenamiento a las que la red se ha sobreajustado.

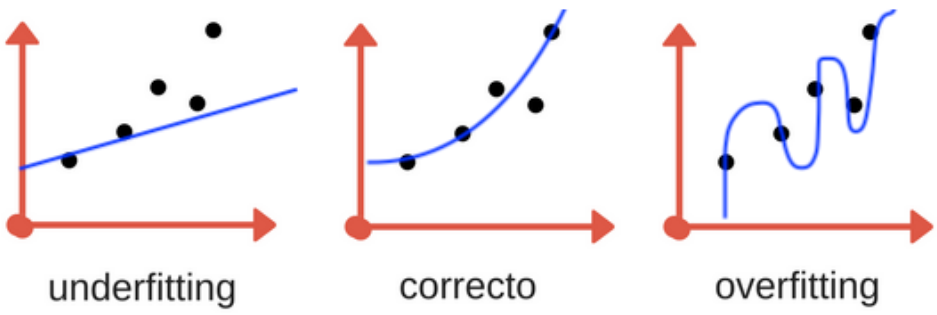

Figura 2-7. Subajuste, ajuste correcto y sobreajuste [36]

Teniendo estos ejemplos en cuenta, podemos analizar cuándo se producen estos problemas.

- **Subajuste:** Ocurre cuando la red neuronal no ha tenido suficientes muestras de entrenamiento y, por tanto, no ha podido extraer suficientes características del set como para poder generalizar. Este problema casi siempre se soluciona añadiendo más muestras al set de entrenamiento.
- **Sobreajuste:** Es el caso contrario. Aparece cuando la red se ha adaptado demasiado a los casos particulares del entrenamiento, volviéndose incapaz de reconocer nuevos datos de entrada si estos no son prácticamente idénticos a los del set de entrenamiento. Para hacer frente al overfitting existen varias técnicas [37]:
  - **Simplificación del modelo:** Hacer una red neuronal más sencilla reduce su número de parámetros, impidiendo así que el ajuste se perfeccione en exceso. Con este método se pierde precisión en el set de entrenamiento porque la red se ajusta peor a sus datos, pero se aumenta en el set de validación, derivando así en un menor overfitting.
  - **Dropout:** Es una de las técnicas más utilizadas, y se basa en desactivar aleatoriamente neuronas de la red para que no se entrenen y no puedan ajustar sus parámetros asociados.
  - **Early stopping:** Se encarga de detener el entrenamiento cuando no se detecte una reducción en el error de la validación tras un cierto número de epochs denominado paciencia. Con esto se evita que el modelo siga entrenando de más y después la red no sea capaz de generalizar.

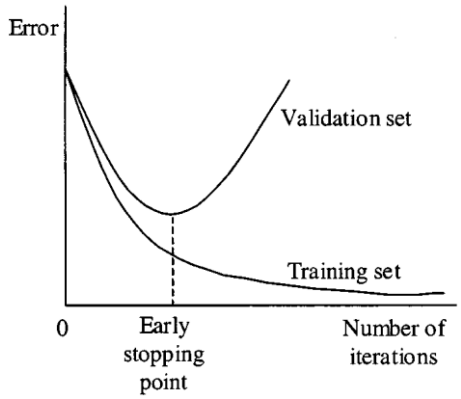

Figura 2-8. Funcionamiento del early stopping [38]

2.3.3.2 Set de validación

La validación forma parte del entrenamiento, y se encarga de evaluar iterativamente nuestra red neuronal con datos distintos a los del set de entrenamiento. La red comprueba su precisión con respecto a los datos de este set, pero no aprende de ellos, sino que informa al usuario de cómo progresa la validación para que, si es necesario, este ajuste algunos parámetros manualmente.

La información devuelta por la validación puede representarse de distintas formas, aunque la más extendida es la métrica de accuracy o precisión de la red. Por ejemplo, en una red neuronal para clasificación se devolvería un porcentaje indicando el éxito que esta ha tenido a la hora de clasificar los datos del set.

Normalmente en deep learning no se revisa únicamente la precisión, ya que no aporta suficiente información. Siguiendo con el ejemplo planteado, puede suceder que la precisión a la hora de clasificar sea muy alta, pero no lo sea tanto la seguridad con la que se realiza dicha clasificación.

Si por ejemplo tuviéramos una red con una precisión del 99% pero con una baja seguridad en sus elecciones, muy probablemente a la hora de evaluar el set de testeo habría problemas con la clasificación. La baja seguridad en la elección nos indica que, a pesar de cumplir durante el entrenamiento, la red no está extrayendo características suficientes de los datos de entrada como para ser capaz de diferenciar entre clases.

Por ello, junto a la precisión del set de validación se comprueban otras métricas como el loss o pérdidas de validación, valor de la función de coste que nos permite conocer la magnitud del error cometido en las predicciones. Esta métrica también es muy útil al analizar su evolución, permitiendo observar el proceso de entrenamiento epoch a epoch para saber si el modelo está aprendiendo progresivamente.

Una red neuronal óptima tiene una precisión alta y unas pérdidas bajas, traduciéndose esto en el ejemplo del clasificador en muchas decisiones acertadas con un alto índice de seguridad.

2.3.3.3 Set de testeo

El set de testeo se utiliza una vez el entrenamiento se ha finalizado y sirve para evaluar el funcionamiento final de la red neuronal. Al igual que el set de validación, son datos de los que la red no aprende y con los que se representa el desempeño real de la red en condiciones normales.

Figura 2-9. Distribución habitual de los sets de entrenamiento, validación y test

2.4. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN por sus siglas en inglés) son un tipo de redes neuronales artificiales basadas en las neuronas de la corteza visual del cerebro animal [26]. Es la red neuronal más utilizada en el procesamiento de imágenes para el reconocimiento y detección de objetos, la estimación de poses y la segmentación. Los dos elementos más característicos de una CCN, que a su vez son los que la diferencian del resto de redes neuronales, son su arquitectura y su forma de procesar los datos de entrada.

2.4.1 Estructura de una red neuronal convolucional

En las tareas de percepción se analizan imágenes, es decir, matrices formadas por uno o varios canales que contienen la intensidad de cada píxel. Teniendo en cuenta que las imágenes suelen estar formadas por tres canales de color, una CNN normalmente tiene capas con distribución tridimensional.

Al contrario que las redes neuronales básicas donde todas las neuronas se conectan entre sí, la mayoría de las capas de una CNN solo conectan sus neuronas con las neuronas de la capa inmediatamente posterior.

Esta forma de distribuir una red neuronal surge de los problemas existentes en las redes básicas de deep learning para hacer frente a tareas de procesado de imágenes como el reconocimiento de formas y texturas o la identificación de patrones visuales. La estructura de una CNN se fundamenta en sus tres capas principales, además de las de entrada y de salida: Las capas convolucionales, las capas de pooling y las capas fully connected.

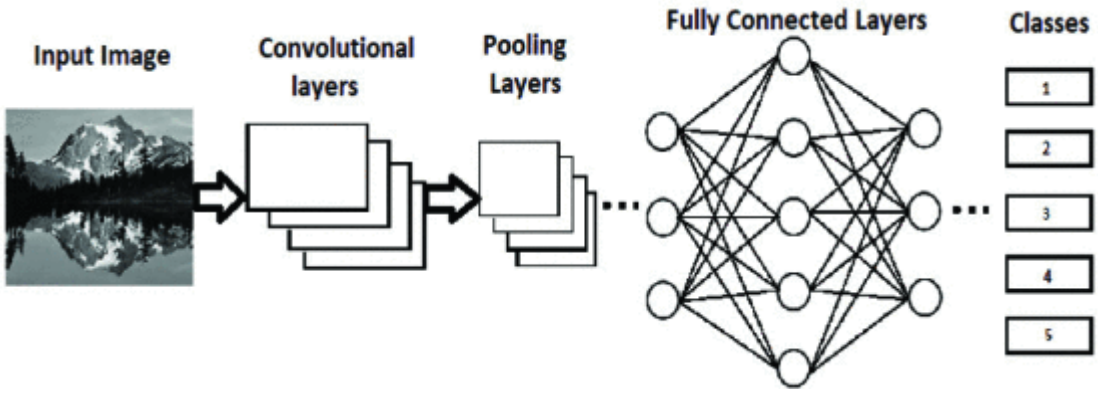

Figura 2-10. Arquitectura básica de una CNN para clasificación de imágenes [26]

Existen otras capas que pueden introducirse en una CNN, como las capas de dropout o las de normalización, pero nos centraremos en las tres mencionadas anteriormente.

2.4.1.1 Capas convolucionales

Las capas convolucionales son la base de las CNN, y en ellas se realiza casi todo el procesado de información. Estás funcionan gracias a una máscara o kernel que va desplazándose por los píxeles de la imagen aplicándole ponderaciones a este y a sus vecinos para comprobar si una característica está presente. Es importante aclarar que la convolución no es un producto de matrices, cada píxel se multiplica únicamente por el valor de la máscara que le corresponde a su posición. El proceso puede observarse en la Figura 2-11.

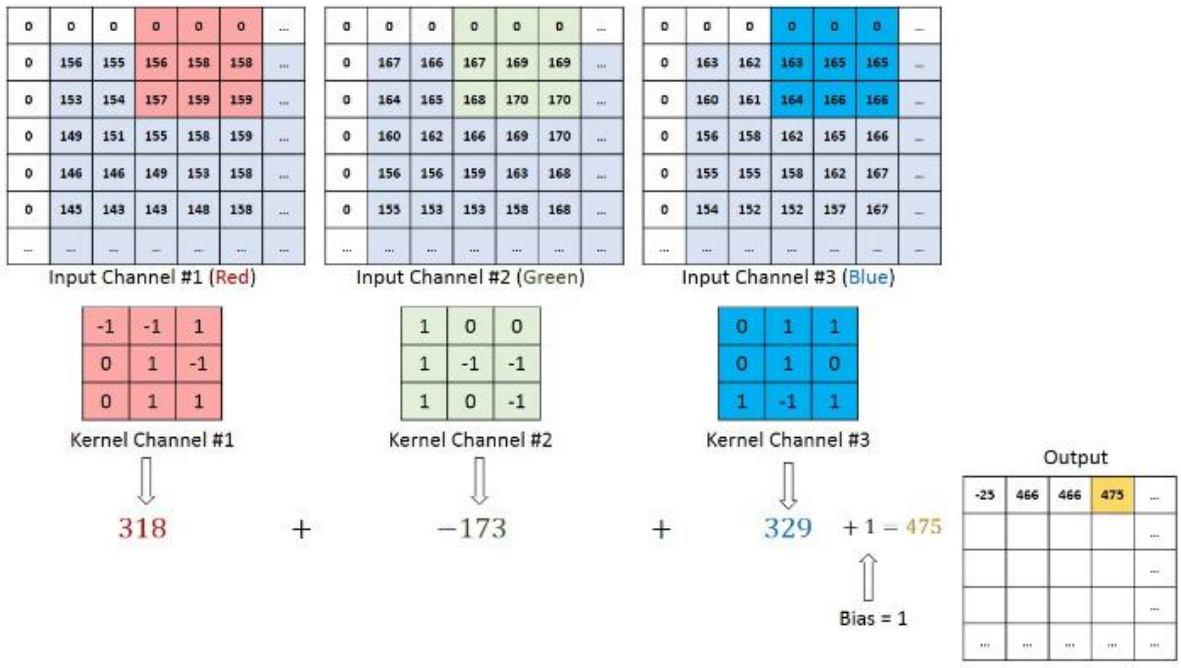

Figura 2-11. Convolución de una imagen RGB con una máscara de dimensión 3x3x3 [39]

En la figura anterior se puede observar como una matriz de tres canales de profundidad necesita como mínimo una máscara de también tres canales de profundidad para realizar la convolución, devolviéndose como resultado de la operación una matriz de un único canal. En algunas ocasiones pueden añadirse más máscaras para sacar más características de la imagen como bordes o texturas, en cuyo caso la matriz resultante aumentaría su profundidad. Es conveniente tener en cuenta la profundidad de las capas convolucionales al construir la red porque de ello depende el número de características que se analizarán y, por tanto, el coste de la red.

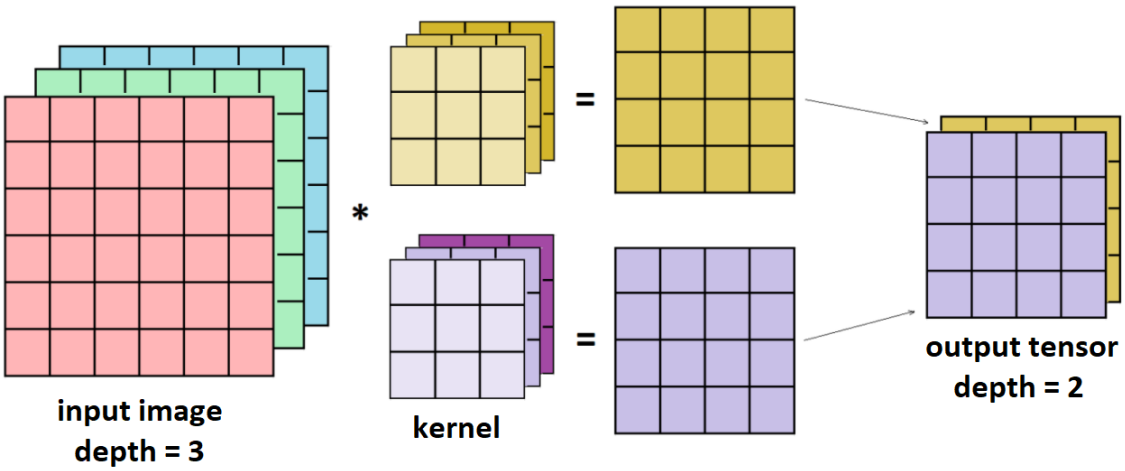

Figura 2-12. Concepto de profundidad en el proceso de convolución [40]

También es necesario añadir que las máscaras no siempre avanzan píxel a píxel. Existe un hiperparámetro llamado stride que define el salto en píxeles que el kernel va realizando iterativamente durante la convolución. Un stride alto puede mejorar la velocidad del proceso al reducir el número de ciclos en la convolución, pero a su vez puede hacer que se pierda información importante.

Así, un stride de 1 aplicaría la máscara píxel a píxel; un stride de 2, se iría saltando un píxel cada vez; y así en adelante. En la Figura 2-13 hay un ejemplo visual del desplazamiento en píxeles del filtro según el valor del stride.

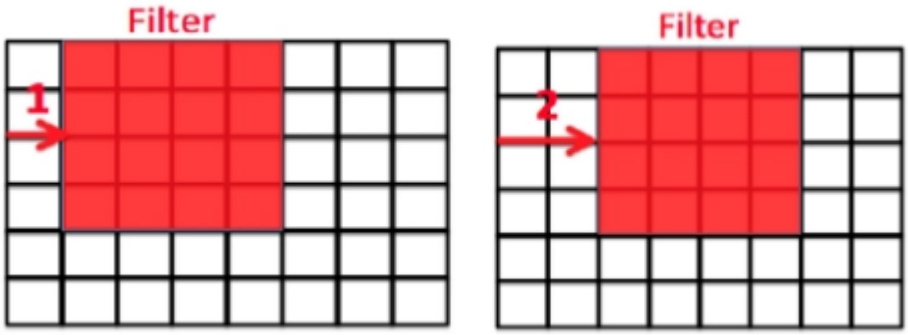

Figura 2-13. Concepto de stride en la convolución [41]

Finalmente, una vez se completa la convolución, se obtiene un mapa de activación que se filtra y se pasa a las siguientes capas de la red neuronal. En la Figura 2-14 pueden observarse los mapas de activación que una CNN crea en su primera capa convolucional para identificar números en una imagen en blanco y negro. Las zonas más iluminadas representan las características que más destacan del número introducido. En cada una de las matrices que crea la capa convolucional, el kernel aplicado varía y, por tanto, las características destacadas también. Dichas características serán procesadas por capas posteriores hasta devolver una predicción.

Figura 2-14. Mapas de activación de la primera capa de una CNN de clasifición de números [42]

En la imagen anterior también se aprecia como el mapa de activación que genera la capa convolucional reduce su tamaño con respecto a la entrada. Esto se conoce como efecto borde, y ocurre debido a que no es posible aplicar la convolución en los píxeles del borde de la imagen. La solución a este problema es el padding, que se divide en tres tipos:

- **Zero padding:** Es el más simple de todos los padding, pues pone a cero la posición correspondiente a los píxeles del borde donde no ha podido aplicar convolución, conservándose el tamaño de la imagen.
- **Same padding**: Agrega ceros alrededor de la imagen para que se pueda aplicar la máscara parcialmente sobre los píxeles del borde, por lo que también se conserva el tamaño de la imagen.
- **Valid padding**: En este caso no se realiza padding de ningún tipo, la convolución simplemente ignora los píxeles del borde, devolviéndose una imagen más pequeña que puede haber perdido información.

2.4.1.2 Capas de pooling

Las capas de pooling se encargan de reducir la dimensión de los mapas de activación, minimizando así el número de parámetros de la red y, por tanto, su coste computacional. Estas capas también utilizan la convolución, solo que en este caso las máscaras no tienen pesos asociados, sino que aplican una función de agregación sobre un área de píxeles en la imagen con la que obtienen el valor de salida.

Otra de las ventajas de utilizar capas de pooling es que, al forzar la pérdida de información, consigue que la red no se ajuste en exceso al set de entrenamiento, reduciendo así la probabilidad de incurrir en overfitting.

Los dos tipos de pooling más utilizados en las CNN son:

- **Max pooling:** La máscara selecciona el píxel de la zona filtrada con el valor máximo para ponerlo en la salida.
- **Average pooling:** Método más costoso que calcula la salida mediante la media de todos los píxeles afectados por el kernel.

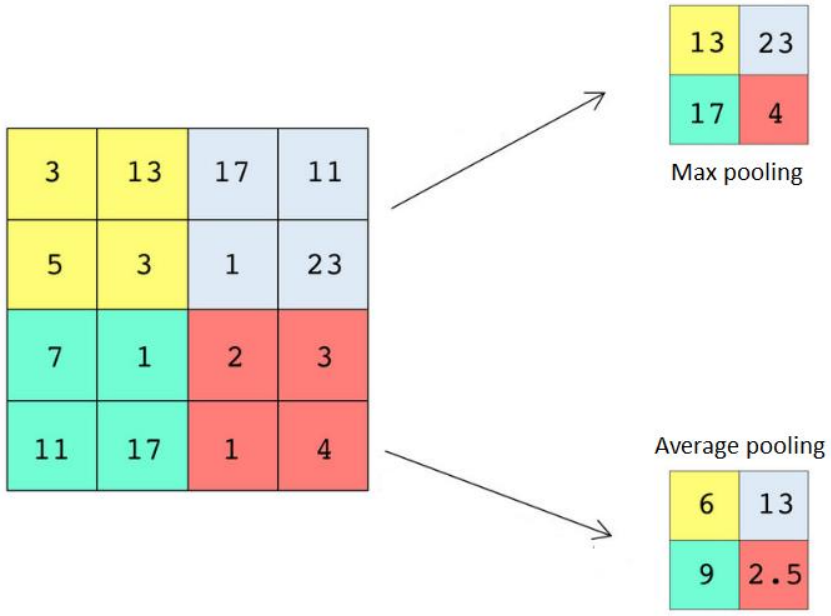

Figura 2-15. Conceptos de max pooliny y average pooling [43]

La capa de pooling también posee hiperparámetros como el stride anteriormente mencionado. Normalmente en esta capa no se modifica la profundidad de la matriz ni se necesita hacer padding.

2.4.1.3 Capas fully connected

Las capas fully connected, también llamadas capas densas, tiene todas sus neuronas conectadas con todas las neuronas de la capa anterior y posterior, como si de una capa de una red neuronal básica se tratara.

Es la encargada del proceso de clasificación de las características extraídas por las capas anteriores. Para ello transforma los mapas de activación de una estructura multidimensional a una de una sola dimensión sobre la que realiza abstracciones mediante razonamiento de alto nivel.

Esta capa también suele ser la última de la red neuronal y, dependiendo de qué tipo de resultado se necesite, se suele elegir entre tres funciones de activación:

- **Softmax:** La función de activación softmax devuelve un valor entre 0 y 1, por lo cual se utiliza en problemas de clasificación para indicar la posibilidad de que una imagen pertenezca a una clase en concreto.
- **Linear o ReLU:** Estas funciones se suelen utilizar para obtener valores continuos en problemas de regresión. La diferencia principal es que la función ReLU aporta no linealidad y la función linear no.

2.5. Red neuronal propuesta

Para crear nuestra red neuronal convolucional se ha utilizado la Keras de Tensorflow, una libfdfíz para Python 3 muy extendida en el ámbito del deep learning. El uso de Keras facilita la creación de redes neuronales con modelos modulares completamente configurables.

Nuestra red neuronal se encargará de procesar la foto de un taladro avellanado para encontrar las características clave que ayuden a estimar con la máxima precisión posible el diámetro exterior del avellanado.

2.5.1 Idea inicial

En un principio se planteó la idea de utilizar herramientas de percepción visual de Python como las de la librería OpenCV para procesar las imágenes e intentar así extraer los diámetros directamente de las fotografías, pero los resultados no fueron satisfactorios.

El problema es que las imágenes tomadas tienen muchas imperfecciones que hacen que las funciones de alto nivel de las librerías de percepción no consigan detectar correctamente la circunferencia del avellanado, haciendo imposible obtener resultados aceptables en el cálculo de los diámetros.

En la Figura 2-16 se compara la relación de los diámetros obtenidos mediante herramientas de alto nivel con la relación real, y se puede observar que no son nada parecidas. Habiendo comprobado esto, se tomó la decisión de abandonar esta propuesta. Las gráficas se han obtenido utilizando la librería de Python Matplotlib.

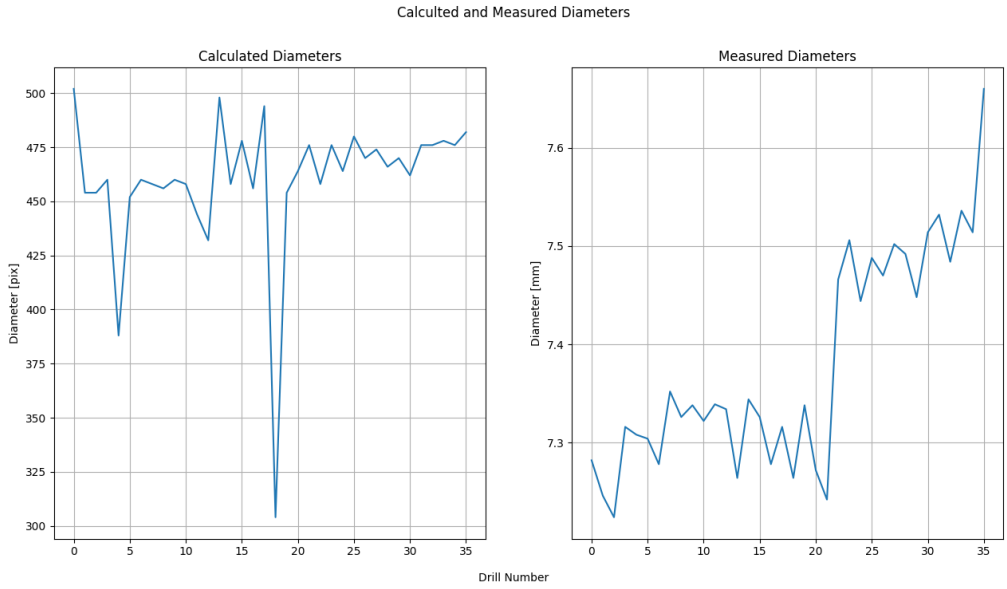

Figura 2-16. Diámetros obtenidos frente a diámetros reales

A pesar de que esta idea no ha llegado a funcionar, los conceptos de percepción aprendidos han sido utilizados en la elaboración del algoritmo de preprocesado de imágenes, del cual se hablará en el próximo capítulo.

2.5.2 Estructura de la CNN propuesta

En el Código 2–1 está recogida la arquitectura de nuestra red neuronal programada con Keras. Es un modelo secuencial, por lo que se van añadiendo las capas en el orden que queremos que el dato las recorra. Para llegar a esta estructura se ha partido de los ejemplos básicos propuestos en la página web de Keras [44], realizándose varios experimentos con diferentes combinaciones de capas y parámetros hasta dar con el modelo actual, que es el que nos ha brindado los mejores resultados.

Código 2–1. Modelo secuencial de nuestra CNN

```
model = keras.models.Sequential([
  keras.layers.Input((IMG_SIZE, IMG_SIZE, 4)),
  keras.layers.Conv2D(16, 3, padding='same', activation='relu'),
  keras.layers.MaxPooling2D(),
  keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
  keras.layers.MaxPooling2D(),
  keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
  keras.layers.MaxPooling2D(),
  keras.layers.Conv2D(128, 3, padding='same', activation='relu'),
  keras.layers.MaxPooling2D(),
  keras.layers.Conv2D(256, 3, padding='same', activation='relu'),
  keras.layers.MaxPooling2D(),
  keras.layers.Conv2D(512, 3, padding='same', activation='relu'),
  keras.layers.MaxPooling2D(),
  keras.layers.Flatten(),
  keras.layers.Dense(128, activation='relu'),
  keras.layers.Dense(1, activation='relu')
])
```

Como en toda red neuronal, la primera capa es la capa de entrada, la cual tiene como parámetros el formato de los datos que le suministraremos a la red. En el caso de una imagen, estos serán la altura y el ancho en píxeles junto con la profundidad de canales. Nuestra red recibe imágenes de 800x800x4, ya que además de los tres canales de color introducimos un canal con información adicional del taladro.

Una vez se introduce una imagen en nuestra red, esta pasa por una serie de capas de convolución y de max pooling que van creando los diferentes mapas de activación:

- **Parámetros función Conv2D.** De izquierda a derecha: Número de máscaras de aprendizaje de la capa, tamaño de las máscaras, padding a utilizar y función de activación a la salida.

  El número de máscaras de aprendizaje determina el número de abstracciones o conceptos que puede extraer la red neuronal de una imagen, lo equivalente al número de neuronas de una capa convencional. En la primera convolución, la red se encarga de escoger la información que considere más relevante de los datos de entrada. Una vez extraídas las primeras características útiles, el resto de las convoluciones van aumentando su número de máscaras para poder realizar abstracciones más complejas de forma progresiva. El aumento de kernels se suele hacer en potencias de dos por convención.

  Por otro lado, se utiliza same padding para no perder información que puede ser importante en los bordes de la imagen; y la función de activación ReLU, la cual dictamina la importancia del dato que procesa y añade no linealidad, aumentando la complejidad del sistema.

- **Parámetros función MaxPooling2D.** Se dejan los valores predeterminados que propone Keras: Tamaño de filtro 2x2, stride de 1 y valid padding (los bordes se pierden). Teniendo esto en cuenta, cada vez que se aplica la función de max pooling se reduce el alto y ancho de los mapas de activación a la mitad.

Una vez finalizado el proceso iterativo de convoluciones y max poolings, se introduce una capa de flatten, la cual pasa la información de un array con múltiples dimensiones a un array unidimensional para que la capa fully connected que le sucede pueda procesarla. En la Figura 2-17 se puede como de un array multidimensional 3x2 con profundidad 2 se transforma en un array unidimensional de tamaño 12 (valor obtenido tras multiplicar las tres componentes del array multidimensional).

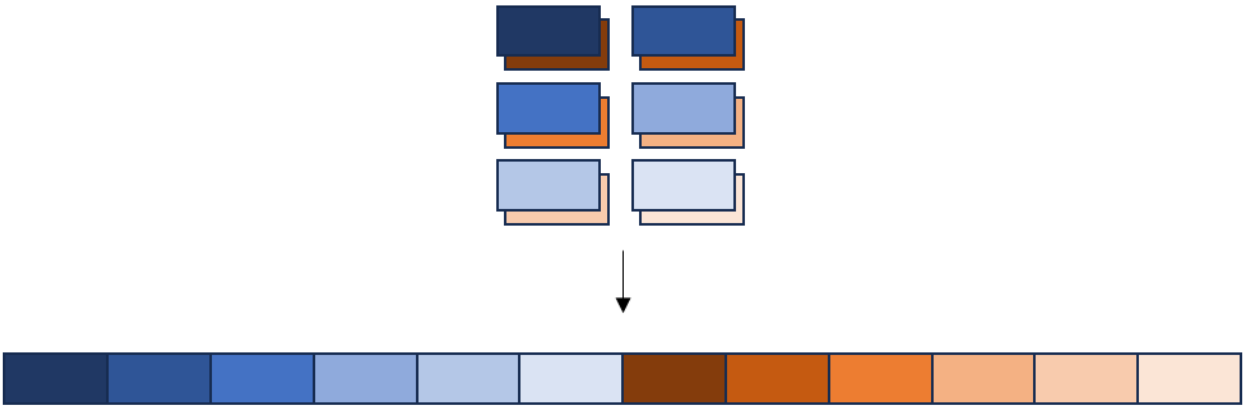

Figura 2-17. Transformación de un array multidimensional 3x2x2 a uno unidimensional 1x12

El flatten recoge toda la información de los mapas de activación de la capa anterior, por lo que aumenta la complejidad de nuestra red. Hay ocasiones donde no es necesario que nuestro modelo tenga tantos parámetros, por lo que en vez de hacerse un flatten se realiza un average pooling de cada campo de activación previo y se reduce su número.

Finalmente, se colocan dos capas dense, también conocidas como capas fully connected. La primera de las dos está formada por 128 neuronas conectadas a cada elemento del array unidimensional, y se encarga de encontrar relaciones entre componentes y clasificar las características.

La última capa es también fully connected y se encuentra conectada a las 128 neuronas anteriores. En nuestro caso, está formada por una única neurona debido a que nuestra CNN funciona como regresor, así que buscamos un resultado continuo (de ahí que utilicemos una función de activación ReLU).

En el caso en el que tuviéramos un clasificador, esta capa estaría compuesta por tantas neuronas como clases existiesen, y la función de activación para todas ellas sería la softmax, como hemos mencionado anteriormente.

Utilizando la función summary de Keras sobre nuestro modelo, la consola nos devuelve las características de nuestra red neuronal. En la Figura 2-18 se puede observar nuestra red capa a capa, junto con la dimensión de los datos que devuelven y el número de parámetros que las componen. Al final se menciona el número total de parámetros de nuestra red, unos once millones, una cantidad bastante estándar en redes neuronales convolucionales pequeñas [45].

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 800, 800, 16)      592

max_pooling2d (MaxPooling2D  (None, 400, 400, 16)      0
)

conv2d_1 (Conv2D)            (None, 400, 400, 32)      4640

max_pooling2d_1 (MaxPooling  (None, 200, 200, 32)      0
2D)

conv2d_2 (Conv2D)            (None, 200, 200, 64)      18496

max_pooling2d_2 (MaxPooling  (None, 100, 100, 64)      0
2D)

conv2d_3 (Conv2D)            (None, 100, 100, 128)     73856

max_pooling2d_3 (MaxPooling  (None, 50, 50, 128)       0
2D)

conv2d_4 (Conv2D)            (None, 50, 50, 256)       295168

max_pooling2d_4 (MaxPooling  (None, 25, 25, 256)       0
2D)

conv2d_5 (Conv2D)            (None, 25, 25, 512)       1180160

max_pooling2d_5 (MaxPooling  (None, 12, 12, 512)       0
2D)

flatten (Flatten)            (None, 73728)             0

dense (Dense)                (None, 128)               9437312

dense_1 (Dense)              (None, 1)                 129

=================================================================
Total params: 11,010,353
```

Figura 2-18. Características de cada capa de la CNN propuesta

2.5.3 Ventajas y desventajas de utilizar una CNN para regresión

El objetivo final de nuestra red neuronal es realizar una predicción del diámetro del avellanado de un taladro dada una fotografía de este. Como hemos mencionado anteriormente, utilizar herramientas de percepción visual de alto nivel no ofrece resultados satisfactorios debido a las características de nuestros datos de entrada, dificultando así el encontrar una relación entre la dimensión del diámetro en píxeles y en milímetros.

Una de las razones por las que se ha decidido utilizar una CNN para hacer frente a este problema, es la capacidad de autoajuste a las no linealidades presentes en las imágenes de entrada. Una CNN es capaz de encontrar relaciones y características clave para la estimación del diámetro que ni los algoritmos de percepción ni el ojo humano pueden percibir.

Los problemas a los que se enfrenta este método son: La duración y el coste computacional del entrenamiento, y la cantidad de datos necesarios. Debido a que la red trata con imágenes de 800x800x4, hace falta una máquina bastante potente para poder entrenarla a buen ritmo. Lo habitual es utilizar mini batches a la hora de procesar la información, lo cual tiene un impacto notable en los tiempos de entrenamiento.

Por otro lado, una CNN necesita de una gran base de datos de entrenamiento para devolver resultados de calidad, algo que ocasiones como esta puede ser difícil de obtener.

3 CREACIÓN DE LA BASE DE DATOS

3.1. Programa de taladrado

Para crear los sets de imágenes con los que alimentamos a la red neuronal, se han realizado probetas sobre cuatro chapas de aluminio con taladros de dos diámetros distintos: 4.1 mm y 4.9 mm. El código utilizado para taladrar las probetas es confidencial, por lo que en este apartado sólo se describirán el proceso de taladrado de forma general (sin entrar en la parametrización y calibración del robot), y el método de obtención de las posiciones de taladrado.

Como se ha comentado, antes de taladrar el borde de ataque es necesario realizar una serie de pruebas sobre una superficie de aluminio para comprobar que la EDU está bien calibrada y que la broca cumple con las especificaciones del fabricante. A continuación, se explicará cómo es el proceso general de taladrado.

El robot cuenta con cuatro palpadores que utiliza para situarse lo más perpendicularmente posible al plano de la pieza, partiendo desde una posición de aproximación situada encima de la posición de taladrado. Una vez el robot se encuentra alineado, un motor de la EDU comienza a desplazar la broca hacia la superficie hasta que detecta un pico de par, instante en el que la taladradora toma su punto de referencia para taladrar. Tras esto, la broca se separa de la superficie para que el otro motor de la EDU pueda hacerla girar. Cuando se encuentre girando, la broca avanza hacia la posición anteriormente referenciada por la EDU y, al llegar, vuelve a avanzar los 16 mm que tiene el cuerpo de la broca, realizando así el taladro.

Este proceso se ha repetido varias veces en cuatro probetas distintas. Las posiciones de taladrado se encuentran definidas en la base de datos del robot, y se puede acceder a ellas si se conoce su situación en la probeta, como si de una matriz se tratara. En la Figura 3-1 se pueden observar las cuatro probetas que se han realizado junto a unas anotaciones que aclaran lo mencionado acerca de la situación de los taladros.

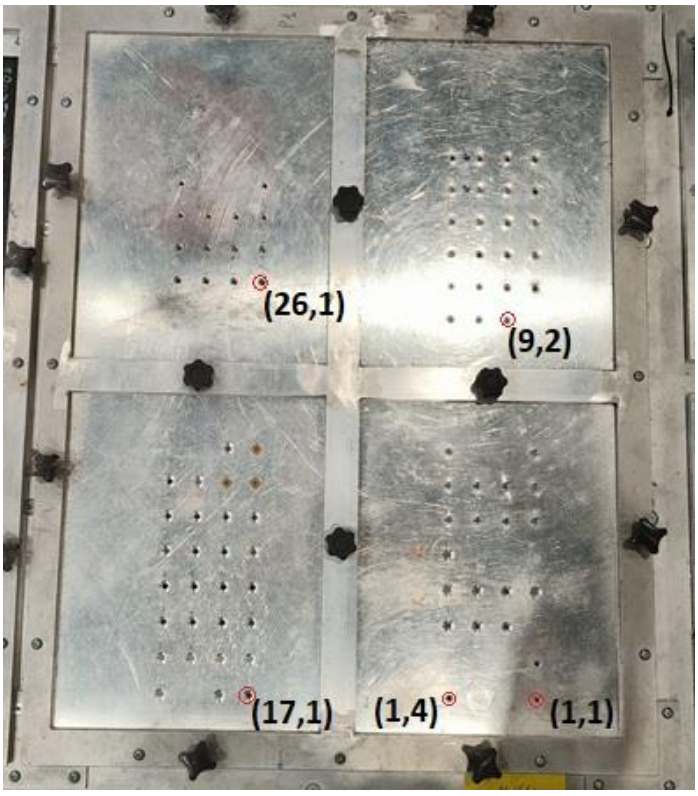

Figura 3-1. Probetas de los taladros realizados sobre chapas de aluminio

Como se puede observar, hay puntos de las probetas que no han sido taladrados por el robot aun estando estos supuestamente definidos en la base de datos. Esto es así debido a que al calcular las posiciones de los taladros en el plano de la chapa no se había planteado aun la necesidad de utilizar aprietes para sujetar la superficie de aluminio. Este factor pues, condiciona el alcance de nuestro robot, ya que, al aproximarse a estos puntos de riesgo, algunas partes del efector como los palpadores chocan con los tornillos de apriete. La solución escogida ha sido obviar vía software estos puntos.

El resto de irregularidades en la posición de los taladros son propias de los experimentos realizados con el programa de taladrado, al cual se le han ido añadiendo limitaciones y se le han ido corrigiendo errores hasta obtener un programa que funcionara como queremos.

La altura a la que se encuentran los puntos de taladrado es utilizada para la aproximación a estos, y ha sido calculada utilizando un sensor de fuerza colocado en el extremo del robot. Para obtener esta información, el robot se desplaza a las coordenadas (X, Y) del punto de taladrado con cierta altura de margen y, una vez ahí, desciende perpendicularmente hacia la chapa a una velocidad muy baja. Cuando el sensor detecta una fuerza ejercida sobre la superficie superior a 20 N, el robot se detiene, observa en qué Z se encuentra, y guarda en la base de datos esa coordenada.

Dicha altura ha de ser calculada para todas las posiciones de taladrado, ya que la coordenada Z en el plano de trabajo no es constante por dos motivos: El grosor no uniforme entre probetas, y la inclinación no nula respecto al suelo de algunas de las superficies.

Las posiciones de los taladros de la base de datos se han extraído a un fichero de texto que se utilizará más adelante en el preprocesado de las imágenes de entrenamiento. Al comprobar los puntos del fichero, se ha observado que ni las coordenadas Z de las posiciones de taladrado asociadas a una misma probeta cumplían con la relación lineal que se podría esperar en el caso de haberse cometido un error colocando la chapa paralelamente al suelo.

Esto se debe a la tasa de refresco de las comunicaciones entre el sensor y el robot. A veces sucede que el instante preciso en el que se alcanzan los 20 N de fuerza no coincide con un pulso del sensor, por lo que el robot sigue haciendo fuerza hasta que el siguiente pulso le hace parar. Este exceso de fuerza hace que el robot avance de más, provocando variaciones en torno a la décima de milímetro en la altura que se guarda, y añadiendo la no linealidad comentada.

3.2. Toma de fotografías

Para el programa de fotografiado hay que realizar un cambio de cabezal y acoplar el efector que tiene la cámara de uso industrial que utilizamos: La GOX-5103C-PGE de la marca JAI. Para el cambio de efector hay dos posiciones definidas con estructuras diseñadas específicamente para poder dejar y recoger con seguridad el cabezal de taladrado y el de fotografiado.

El proceso de fotografiado utiliza las posiciones de la base de datos del robot como puntos de referencia para realizar las fotografías. Al igual que en el taladrado, se utilizan los palpadores para alinear la cámara perpendicularmente con la superficie. Una vez completado el alineamiento, se actualiza la posición del taladro a nivel local, y el robot se desplaza hacia la posición de aproximación actualizada.

Debido a la posición relativa de la lente con respecto al punto calculado, a la posición actualizada se le añade un pequeño offset que hace que en la fotografía aparezca el taladro completo. Este desplazamiento se aprovecha también para que la cámara enfoque correctamente y la imagen no se vea borrosa.

Durante la toma de fotografías, una pequeña linterna acoplada al efector proyecta luz roja sobre el taladro, evitando así la presencia de sombras en el avellanado y de reflejos en la superficie plana, y remarcando las irregularidades del metal. Se utiliza una luz roja por recomendación del fabricante, puesto que los sensores de este modelo de cámara trabajan mejor con este canal de color, como se puede ver en la Figura 3-2.

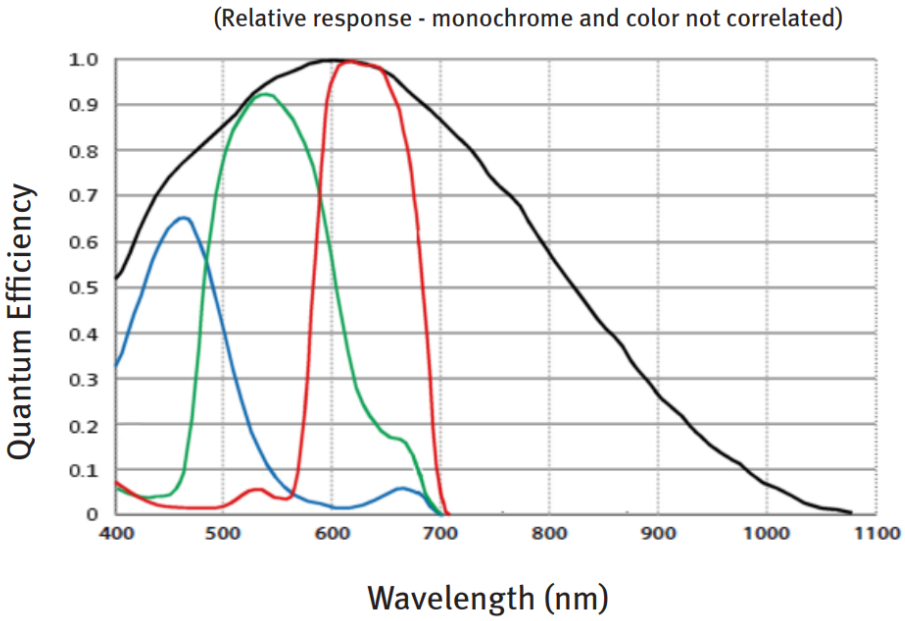

Figura 3-2. Eficiencia de la cámara GOX-5103C-PGE ante los tres canales de color [46]

A continuación, nos basaremos en las anotaciones de la Figura 3-1 para explicar el proceso de fotografiado. Recordar que esta forma de identificar las posiciones de taladrado sirve para extraer las coordenadas de la base de datos a las que están asociadas.

El programa comienza en la posición (1, 1) perteneciente a la probeta de abajo a la derecha. Tras hacer la foto del taladro correspondiente, el robot se desplaza hacia la izquierda hasta llegar a la posición (1, 2), donde se realiza la siguiente foto. Este mismo proceso se repite dos veces más hasta llegar a la posición final de la fila, la posición (1, 4).

En ese instante, el robot salta a la fila superior y vuelve a la primera columna, la de la derecha. Una vez está situado en la posición (2, 1), vuelve a repetir el mismo ciclo de desplazamiento horizontal que la fila anterior.

El bucle se repite hasta ocho veces en cada probeta. En el caso de la probeta que estábamos tratando, finalizaría en la posición correspondiente al taladro de la esquina superior izquierda, la (8, 4).

Después de terminar la primera probeta, se pasaría a la probeta de arriba la derecha, pues es la que contiene la posición que continua con el bucle, la (9, 1).

Hay que tener en cuenta que, si no se hace nada, al cambiar de probeta el robot puede golpearse con los tornillos de apriete. Para evitar problemas, en estos movimientos críticos se añade un punto intermedio de paso para el robot, situado en la posición de reposo, con el que se consigue evitar el choque.

3.2.1 Resultados obtenidos

La imagen capturada por la cámara es de 5.1 Megapíxeles, y tiene unas dimensiones de 2448 píxeles de ancho por 2048 píxeles de alto, además de tres canales de color. En la Figura 3-3 se puede ver una imagen de un taladro de 4.1 mm de diámetro desde el punto de vista de la cámara.

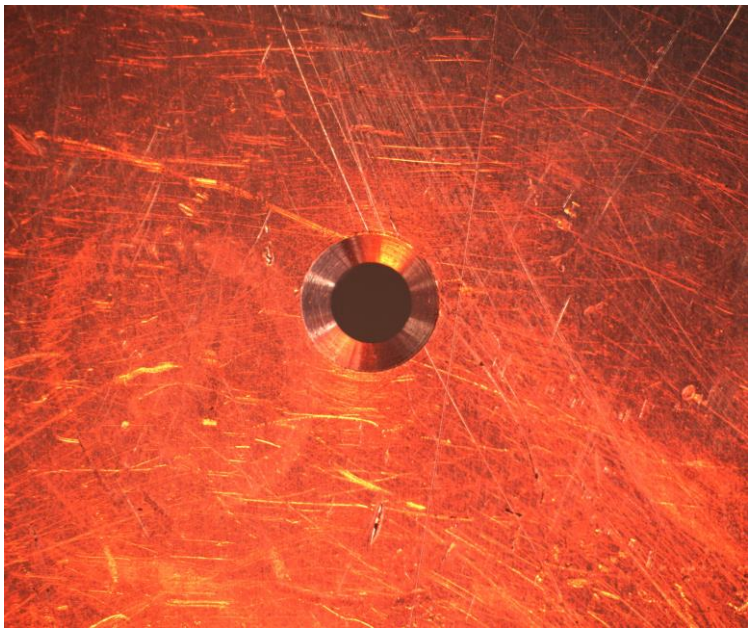

Figura 3-3. Imagen de un taladro de 4.1 mm de diámetro obtenida en el proceso de fotografiado

Para que la fotografía se vea nítida y se perciban correctamente las características del taladro avellanado, se han modificado dos parámetros de la cámara con respecto a los que trae por defecto: La ganancia, que se ha puesto manualmente a 1, y la velocidad de obturación o shutter speed, que se ha puesto a 3000. Estos valores permiten que, bajo las condiciones de luminosidad de la célula de taladrado, las imágenes se vean lo mejor posible con la mínima pérdida de calidad.

3.3. Proceso de medición

Tras haber tomado las fotos, se procede a la medición manual del diámetro del avellanado de cada taladro, proceso necesario para conseguir las etiquetas con las que próximamente entrenaremos nuestra red neuronal. Para la medición se ha utilizado un calibre digital de la marca Trulok, el cual está diseñado específicamente para esta labor.

El primer paso antes de medir es comprobar que el instrumento está calibrado correctamente. Trulok cuenta con un bloque de calibración con un diámetro de avellanado estandarizado sobre el que comprobar el funcionamiento del instrumento. Si este mide correctamente en el bloque de calibración, significa que está listo para ser utilizado.

El proceso de medición es el siguiente:

1. Se coloca la punta del calibre dentro del taladro.
2. Se presiona el instrumento contra la superficie.
3. Manteniendo la presión, se va rotando el calibre lentamente hasta que el valor devuelto no cambie.

Normalmente, se utiliza el valor más alto medido por el calibre, ya que este se obtiene cuando su extremo está lo más introducido posible en el taladro; momento en el que se devuelve la medida del diámetro de avellanado más cercana a la real. En la Figura 3-4 hay un ejemplo visual de cómo se toma la medida.

Figura 3-4. Medida del avellanado de un taladro de 4.9 mm de diámetro con el calibre de Trulok

Podemos observar que el calibre mide hasta la micra, una precisión necesaria al trabajar con tolerancias tan exigentes. Se han tomado un total de 61 medidas efectivas sobre las probetas, descartándose las correspondientes a taladros y avellanados muy lejos de la conformidad. A su vez, cada medida ha sido comprobada al menos tres veces para asegurar que no ha habido ningún error durante la medición.

3.4. Preprocesado de las imágenes

Como hemos mencionado, nuestras imágenes son de 2448x2048 píxeles, dimensiones demasiado grandes como para ser procesadas de forma eficaz por nuestra red neuronal. Es necesario recortar las imágenes para quedarnos sólo con la parte que nos interesa, que es el taladro avellanado.

Teniendo en cuenta el tamaño en píxeles medio del diámetro de avellanado más grande posible (el que realiza la broca de 4.9 mm), se ha comprobado que con un recorte de 800x800 centrado en el taladro se conserva toda la información que nos interesa.

A pesar de que en la toma de fotografías utilizamos la posición sobre la que se ha realizado el taladro como referencia, al realizar siempre un desplazamiento relativo constante para colocar la lente encima del taladro, hay veces que el taladro no se encuentra perfectamente centrado en la imagen.

Esto impide hacer el recorte de 800x800 en el píxel central de la fotografía, ya que las imágenes resultantes no tendrían el taladro totalmente centrado, causando que en algunas ocasiones el recorte no contuviera la totalidad del avellanado.

Es aquí donde se han aplicado los conocimientos obtenidos en la propuesta inicial. A pesar de no haber sido capaz de sacar los diámetros del avellanado correctamente, el algoritmo sí que lograba detectar el centro de los taladros. En base a esto, se ha creado un script que utiliza las funciones de alto nivel para la detección de formas circulares de OpenCV para situar el centro del taladro y luego recortar la imagen alrededor de ese punto. Un recorte obtenido utilizando el algoritmo que acabamos de comentar puede verse en la Figura 3-5.

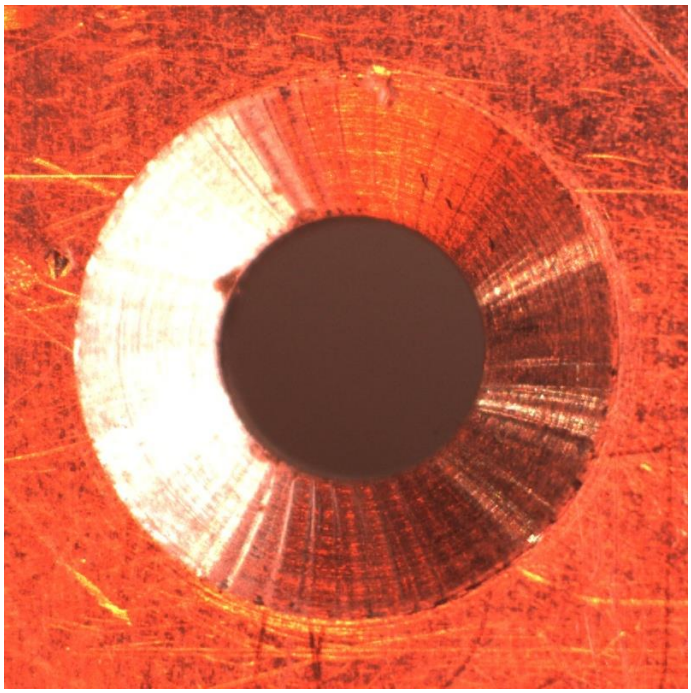

Figura 3-5. Recorte de 800x800 de la fotografía de un taladro de 4.9 mm de diámetro

Los resultados son bastante satisfactorios: El taladro está centrado, se ha logrado reducir considerablemente el tamaño de la imagen y se ha eliminado mucha información irrelevante. Todo esto ayuda al entrenamiento de nuestra red neuronal, optimizando el cálculo de los pesos y aumentando la velocidad de aprendizaje.

Tras finalizar con el proceso de obtención y preprocesado de las imágenes de los taladros avellanados, nuestra base de datos está lista para el entrenamiento de nuestra red neuronal. En total, la base de datos contiene 61 imágenes con sus respectivas etiquetas, de las cuales 45 se utilizan para el set de entrenamiento, 10 para el set de validación, y 6 para el set de testeo.

Tanto la posición de los taladros en la probeta como el diámetro medido por el calibre se han colocado manualmente en el nombre de las imágenes para automatizar procesos posteriores.

4 ENTRENAMIENTO Y RESULTADOS DE LA RED

4.1. Configuración del entrenamiento

Como ya se ha definido en el segundo capítulo, el entrenamiento de una red neuronal es el proceso iterativo mediante el que una red ajusta los pesos de sus neuronas con el objetivo de minimizar las pérdidas definidas por la función de coste.

Antes de comenzar el entrenamiento, será necesario modificar el formato de las imágenes de nuestra base de datos para adaptarlo al formato que espera nuestra red y crear un fichero de etiquetas con las medidas tomadas; tanto para el set de entrenamiento como para el set de validación. Además, Keras nos ofrece la posibilidad de configurar algunos parámetros del entrenamiento, por lo que cambiaremos algunas opciones para adaptar el proceso a nuestras necesidades y limitaciones.

4.1.1 Formato y distribución de los datos de entrada

Para leer las imágenes de la base de datos, utilizamos dos funciones de alto nivel de la librería de preprocesado de imágenes de Keras, las cuales se encargan de transformar cada imagen en un array de NumPy para hacer más fácil su manipulación.

Teniendo en cuenta que las imágenes de nuestra base de datos son interpretadas como matrices de tres canales, antes de introducirlas en nuestra red como datos de entrenamiento hemos decidido añadirle un cuarto canal con información adicional.

Dicha información adicional es la altura del punto de taladrado correspondiente al dato de entrada, la cual se extrae del fichero de texto con coordenadas a través de la posición del taladro sobre la probeta, dato que incluimos anteriormente en el nombre de la imagen. El canal acoplado tiene que ser de 800x800, y contiene en todas sus posiciones el valor de altura asociado.

Esta información extra permite a nuestra red mejorar sus predicciones, puesto que durante el aprendizaje esta logra encontrar una relación entre la altura de la posición de taladrado obtenida y el diámetro del avellanado resultante.

Una vez tenemos nuestro dato de entrada con las dimensiones requeridas por la red (800x800x4), el siguiente paso es normalizar los valores de los cuatro canales, ya que estos no se encuentran en el mismo rango. Esto ayuda al algoritmo de descenso de gradiente durante el entrenamiento puesto que, al utilizar números normalizados, el método calcula un paso de gradiente relativamente estable que facilita la minimización de las pérdidas de la función de coste.

Los tres primeros canales, los correspondientes al color, se normalizan dividiendo por la intensidad máxima posible para un píxel, cuyo valor es 255. Para la normalización del cuarto canal, el que contiene el dato de altura, se ha dividido entre el valor del dato de altura máximo existente en el archivo de coordenadas.

Todo el proceso de cambio de formato se repite tanto para las imágenes del set de entrenamiento, como para las del set de validación. Se devuelven dos arrays de NumPy, uno para cada set, los cuales contienen todas las matrices correspondientes a las imágenes acopladas una detrás de otra.

Este tipo de distribución de los datos del set de entrenamiento y del set de validación es muy importante, puesto que es el único que nuestra capa de entrada a la red neuronal puede procesar.

4.1.2 Recopilación de etiquetas

Como se menciona en el final del tercer capítulo, el nombre de los archivos de imagen contiene las dimensiones del diámetro que hemos medido manualmente. Dichas medidas son las que utilizará nuestra red neuronal como etiquetas en el proceso de entrenamiento.

En el proceso de creación de los sets de entrenamiento y validación, además de dar formato al dato de entrada, se lee la medida del diámetro del avellanado del nombre de la imagen, se pasa a micras para trabajar con números enteros, y se guarda en un dataframe utilizando la librería Pandas.

Así, al acabar de leer todas las imágenes nos quedamos con dos dataframes, uno con las etiquetas del set de entrenamiento y otro con las del set de validación. Ambos dataframes se guardan en dos archivos distintos de formato CSV.

4.1.3 Selección de hiperparámetros

Después de formatear los datos de entrada; distribuirlos correctamente en los sets de entrenamiento y de validación; y crear nuestro fichero de etiquetas; podemos proceder a la configuración de los hiperparámetros de entrenamiento.

Keras permite crear objetos llamados callbacks, los cuales sirven para llevar a cabo acciones durante cualquier parte del entrenamiento. Por otro lado, también permite elegir el optimizador que la red neuronal utilizará durante el reajuste de pesos.

4.1.3.1 Checkpoints

Los checkpoints o puntos de guardado son un tipo de callback utilizado para ir guardando los avances en el entrenamiento de nuestra red después de cada epoch. En el Código 4–1 se muestra cómo se configuran los parámetros de esta función en Python.

Código 4–1. Configuración de los dos checkpoints a guardar

```
ch_loss = keras.callbacks.ModelCheckpoint(
    filepath='loss.h5',
    monitor='loss',
    mode='min',
    save_best_only=True)

ch_val_loss = keras.callbacks.ModelCheckpoint(
    filepath= 'val_loss.h5',
    monitor='val_loss',
    mode='min',
    save_best_only=True)
```

Hay cuatro parámetros que modificamos entre todos los posibles:

- **Filepath** – Es la ruta hasta la carpeta donde queremos guardar el modelo. Aquí también se incluye el nombre del modelo que guardamos junto con la extensión del archivo que se genera.
- **Monitor** – Indica qué parámetro monitoriza nuestro modelo. En nuestro caso, monitorizaremos las pérdidas del entrenamiento y de la validación, generando así dos modelos distintos.

- **Mode** – Este parámetro es el encargado de decir cuándo se genera un checkpoint en base al valor del parámetro monitorizado. Se ha el modo mínimo debido a que la intención final de nuestra red es minimizar las pérdidas.
- **Save_best_only** – Es una variable booleana con la que indicamos que el modelo se guarde únicamente si funciona mejor que el guardado en epochs anteriores.

Realmente no es necesario guardar el modelo de las pérdidas del entrenamiento, ya que el que nos interesa es el de las pérdidas de la validación, pero resulta interesante tenerlo en cuenta a la hora de evaluar los resultados obtenidos.

4.1.3.2 Early Stopping

El early stopping es el segundo y último tipo de función callback que añadimos a nuestro modelo. Como vimos en el segundo capítulo, su función es la de impedir el sobreajuste de la red neuronal deteniendo el entrenamiento al dejar de percibir mejoras en las pérdidas de la función de validación.

En el Código 4–2 se observa como configurar la función: Se monitorizan las pérdidas de validación y se establece la paciencia. El valor de paciencia elegido es de 1000 epochs.

Código 4–2. Configuración del early stopping

```
  earlystop = keras.callbacks.EarlyStopping(monitor='val_loss',
patience=PATIENCE)
```

Una vez configuradas las tres funciones callback que necesitamos, se meten en un array de callbacks que más tarde se pasará a la red neuronal.

4.1.3.3 Compilación del modelo

Justo antes de comenzar con el entrenamiento, es necesario compilar nuestro modelo. En este paso se definen varios parámetros como la función de coste, el optimizador junto a su learning rate, y las métricas que queremos mostrar.

El optimizador elegido en el Código 4–3 es el Adam [47], una mejora del algoritmo de descenso de gradiente más eficiente y menos exigente computacionalmente que modifica iterativamente el learning rate utilizado. A día de hoy es uno de los métodos de optimización más utilizados en problemas con grandes cantidades de datos y parámetros debido a que no necesita un ajuste previo excesivo.

Código 4–3. Compilación del modelo

```
  model.compile(optimizer=keras.optimizers.Adam(learning_rate=LR),
loss='mean_squared_error')
```

Al contrario de lo que puede parecer, es importante elegir una tasa de aprendizaje adecuada, pues en base a ella el optimizador calculará las adaptaciones necesarias en el reajuste de los pesos. Además, esta también se utiliza como referencia para establecer límites a las mismas adaptaciones. Después de probar varios valores, se ha elegido una tasa de aprendizaje de 0.00001 para nuestro modelo, un valor bastante bajo que hace que se alcance el mínimo con mayor exactitud, pero con una cantidad de ciclos considerablemente alta.

La función de coste escogida devuelve el error cuadrático medio de las predicciones con respecto a las etiquetas de nuestras imágenes, valor que nuestra red se encargará de minimizar a través del entrenamiento.

4.2. Fase de aprendizaje

Una vez tengamos los datos de entrada con el formato correcto y sus etiquetas correspondientes; y nuestro modelo esté configurado y compilado; puede dar comienzo la fase de aprendizaje. Para ello, se utiliza la función del Código 4–4, en la cual se introduce el set de entrenamiento junto a su dataframe de medidas correspondiente. Se hace lo mismo con el set de validación.

Código 4–4. Entrenamiento de la red neuronal

```
 history = model.fit(train_ds, df['drill_size'],
validation_data=(val_ds, df_val['drill_size']), epochs=EPOCHS,
batch_size=BATCH_SIZE, callbacks=callbacks)
```

El resto de parámetros que aparecen en el código anterior son:

- **Epochs –** Número máximo de ciclos de entrenamiento. En nuestro caso, hemos puesto un límite de 20000 para que el encargado de detener el entrenamiento sea el early stopping.
- **Tamaño del batch –** Debido al coste computacional requerido en el procesamiento simultáneo de todas las imágenes de nuestro set de entrenamiento, se ha optado por dividir el set en lotes de 16 imágenes, tamaño más accesible computacionalmente.
- **Callbacks –** Este parámetro contiene el array de callbacks que creamos en el apartado anterior.

Keras devuelve algunas métricas por consola que nos permiten observar cómo va progresando el entrenamiento de nuestra red, además de otros parámetros de tiempo como la duración de una epoch o el tiempo restante aproximado de esta. En la Figura 4-1 se pueden observar las métricas mencionadas.

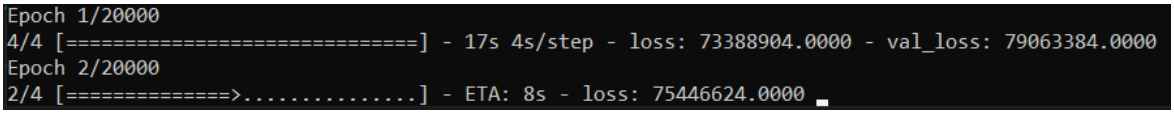

Figura 4-1. Métricas de las dos primeras epoch del entrenamiento

Al finalizar el entrenamiento también se ha decidido guardar el modelo correspondiente a la última epoch, por si en algún momento es necesario reanudar el proceso. También se guarda el historial de las pérdidas de entrenamiento y de validación mediante la librería Pickle.

4.3. Resultados del entrenamiento

El entrenamiento ha durado un total de 7802 epochs. Teniendo en cuenta que colocamos un early stopping monitorizando las pérdidas de validación con una paciencia de 1000 epochs, podemos concluir que en la epoch 6802 se alcanzaron las pérdidas mínimas en la validación

4.3.1 Evolución de las pérdidas

En este apartado se analizará cómo han ido evolucionando las pérdidas en diferentes epochs del proceso de entrenamiento, profundizando en algunos puntos de interés encontrados.

En la Figura 4-2 se ven las fases iniciales del entrenamiento, momento en el que ambas pérdidas son muy elevadas debido a que la red está aún reajustando los pesos aleatorios establecidos en el inicio del proceso. En estas gráficas se puede contemplar como el descendiente de gradiente tiene un paso bastante alto y una pendiente muy pronunciada para reducir rápidamente las pérdidas.

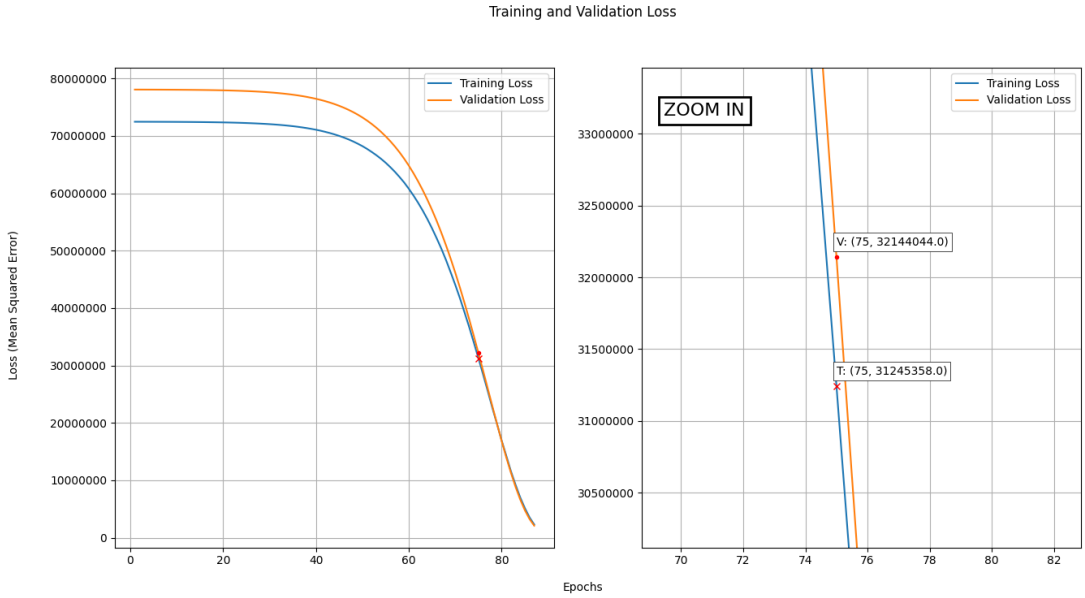

Figura 4-2. Pérdidas tras 75 epochs de entrenamiento

En la Figura 4-3 podemos observar que la red ha logrado adaptarse relativamente rápido a las imágenes de entrada, reduciéndose las pérdidas de forma notoria tras 100 epochs y alcanzando por primera vez lo que parece un mínimo a nivel local.

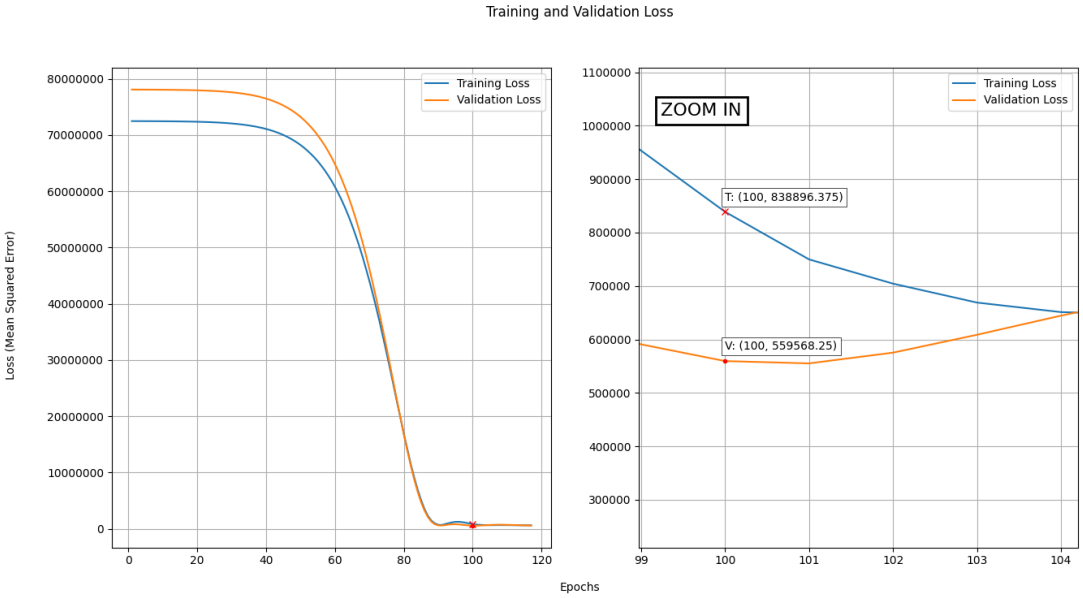

Figura 4-3. Pérdidas tras 100 epochs de entrenamiento

Si seguimos avanzando en el entrenamiento podemos observar que, efectivamente, el mínimo encontrado anteriormente se trataba de un mínimo local; ya que, como se puede observar en la Figura 4-4, la red neuronal ha seguido reduciendo las pérdidas.

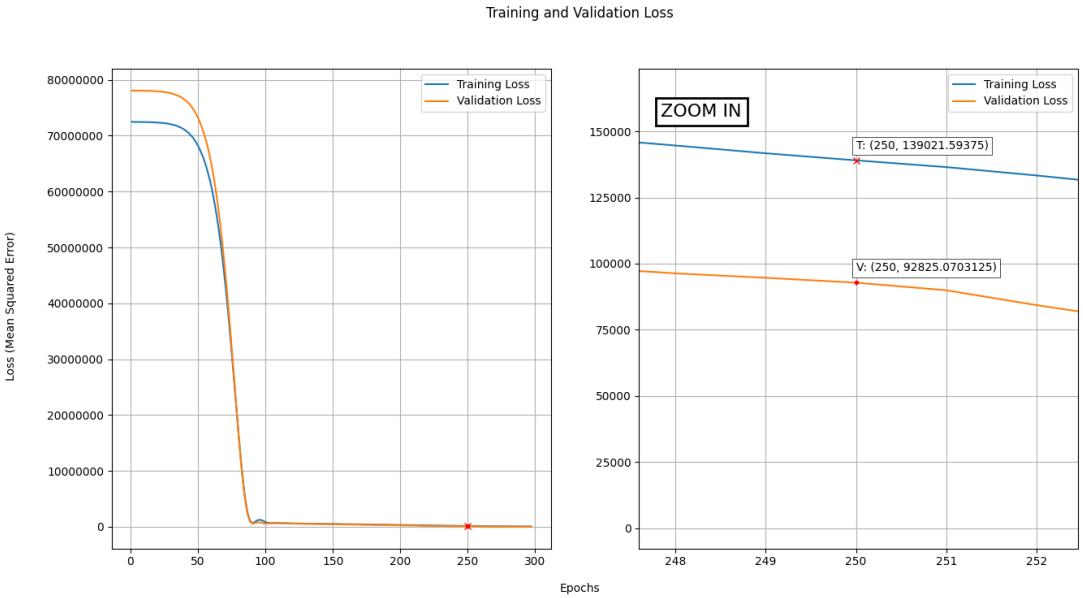

Figura 4-4. Pérdidas tras 250 epochs de entrenamiento

Cabe destacar también que en estas fases del aprendizaje las pérdidas de validación son más bajas que las del entrenamiento, situación que se seguirá repitiendo en las epochs restantes. Esto se debe a las capas de max pooling, las cuales sólo se aplican al set de entrenamiento y aumentan sus pérdidas asociadas. En la validación el único error que se mide es el de la predicción con respecto al valor real, de ahí que las pérdidas sean generalmente menores.

Tras 2500 epochs, en la Figura 4-5 se puede observar que ambas pérdidas se han reducido de manera considerable. Esto significa que la red ha encontrado las características y los patrones de las imágenes que más le sirven para realizar sus estimaciones, teniendo así establecidas las estructuras de los filtros y necesitando sólo realizar pequeñas modificaciones de sus pesos para optimizar los resultados.

A partir de ahora el proceso de minimización avanza mucho más lento en la búsqueda del mínimo absoluto de la función de coste del set de validación.

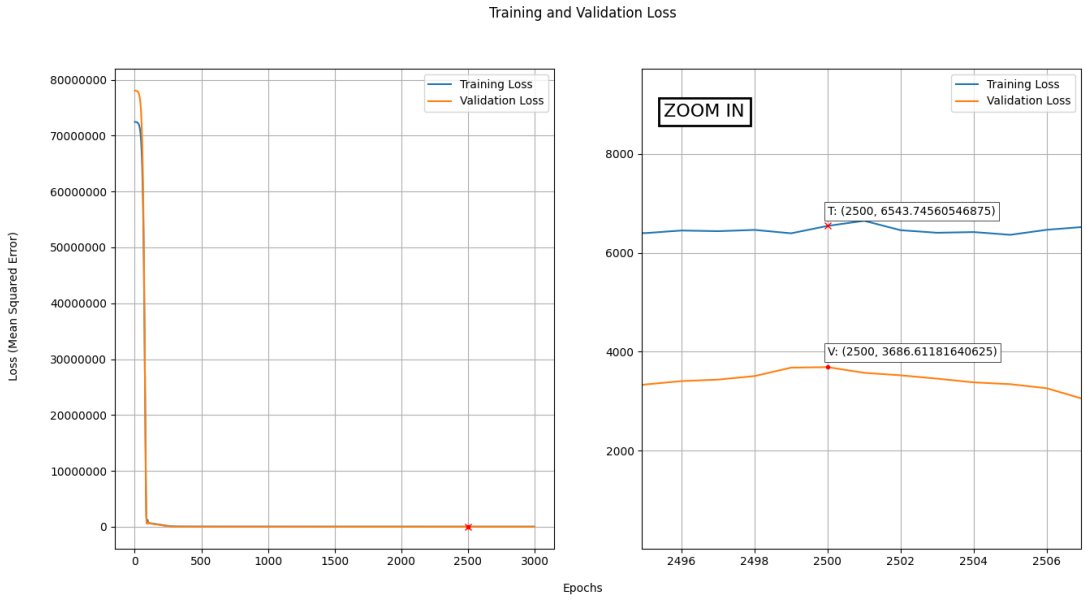

Figura 4-5. Pérdidas tras 2500 epochs de entrenamiento

A las 7802 epochs el entrenamiento finaliza. Como podemos observar en la Figura 4-6, las mínimas pérdidas del set de validación se han dado en la epoch 6802, punto donde comenzó a actuar el early stopping de paciencia 1000 con el que evitamos el overfitting.

El error cuadrático medio asociado a ese punto en la validación tiene un valor aproximado de 1600 micras al cuadrado, por lo que el error medio en la estimación del diámetro de los avellanados sería de aproximadamente 40 micras (0.04 mm). Esta precisión cumple holgadamente con la tolerancia de estimación de 90 micras establecida por el departamento, por lo que el entrenamiento de nuestra red puede considerarse satisfactorio.

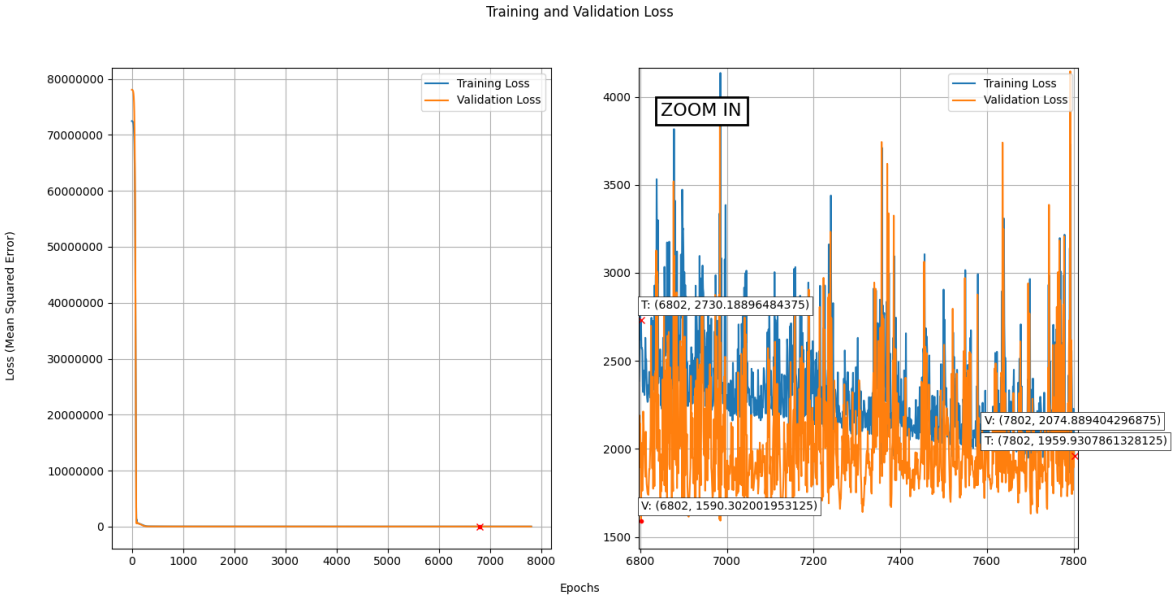

Figura 4-6. Pérdidas tras finalizar el entrenamiento

4.3.2 Mapas de activación de la primera capa de convolución

Si diseccionamos nuestra red neuronal, podemos obtener las ponderaciones de los mapas de activación correspondientes a las distintas capas de nuestra red. Esto nos permite extraer información acerca de cómo nuestra red procesa internamente las imágenes de entrada, con la cual también se puede elaborar un ejemplo visual del funcionamiento de la red, como el que se ve en la Figura 4-7.

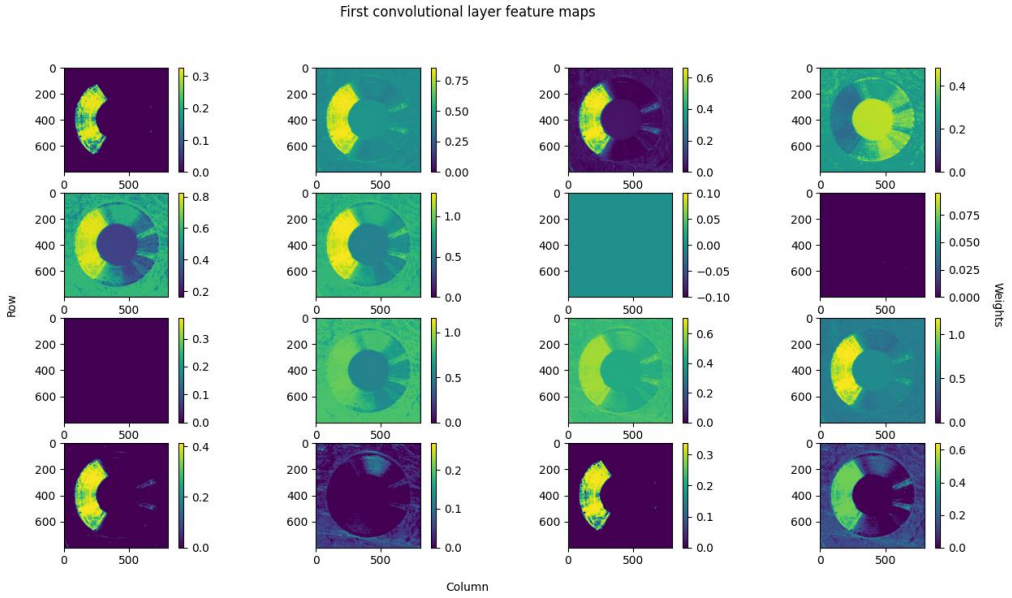

Figura 4-7. Mapas de activación obtenidos en la primera capa de convolución

Resulta interesante analizar cómo los pesos de las máscaras se han ajustado de diferentes formas en cada neurona para extraer distintas características de los taladros avellanados: El diámetro del taladro, el reflejo del avellanado, la superficie plana…

De los mapas de activación de la primera capa convolucional podemos destacar los que presentan un color plano, los cuales aparentemente están desactivados.

Esto es así para los mapas de activación situados en la posición (2, 3) y (2, 4), que no procesan nada para la capa posterior. Esto puede deberse a dos cosas: O sobran parámetros en nuestra red neuronal para la identificación de características (lo que se conoce dying ReLU), o la imagen utilizada no presenta la característica asociada a esta neurona.

Tras analizar una a una todas las imágenes de los sets, podemos concluir que han surgido ambas situaciones. Por un lado, la neurona situada en la posición (2, 4) se encarga de encontrar una característica de las imágenes que no está presente en la fotografía que hemos escogido. En la Figura 4-8 se puede ver como el mapa de activación asociado a esa posición se activa con otra imagen del set de entrenamiento.

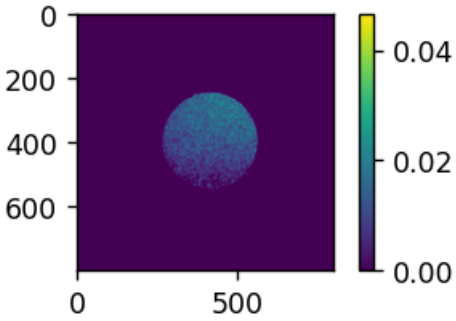

Figura 4-8. Mapa de activación de la posición (2, 4) usando otro taladro avellanado

Por otro lado, la neurona de la posición (2, 3) siempre devuelve cero independientemente de la fotografía seleccionada, por lo que sus parámetros asociados pueden considerarse inútiles. La red pues tiene margen para reducir su tamaño, pero se ha decidido utilizar el modelo obtenido debido a que en su estado actual cumple satisfactoriamente con los requisitos de precisión y velocidad.

Finalmente, si analizamos el mapa de activación situado en la posición (3, 1), este se encuentra siempre desactivado, pero ciertos puntos tienen valores asignados distintos a cero. Esto puede ser indicativo de que nuestra red neuronal utiliza esta neurona para guardar información que utilizará en capas más profundas

.

4.4. Fase de testeo

Esta es la última fase del proceso de entrenamiento. En ella se evalúa nuestra red utilizando las imágenes del set de testeo, las cuales no se han visto en ningún momento de la fase de aprendizaje. Como ya hemos mencionado, el set de testeo está formado por seis imágenes: Tres de taladros de 4.9 mm y tres de taladros de 4.1 mm. En la Figura 4-9 podemos ver cómo, al contrario que en la propuesta inicial, las relaciones entre diámetros medidos y diámetros estimados son mucho más parecidas.

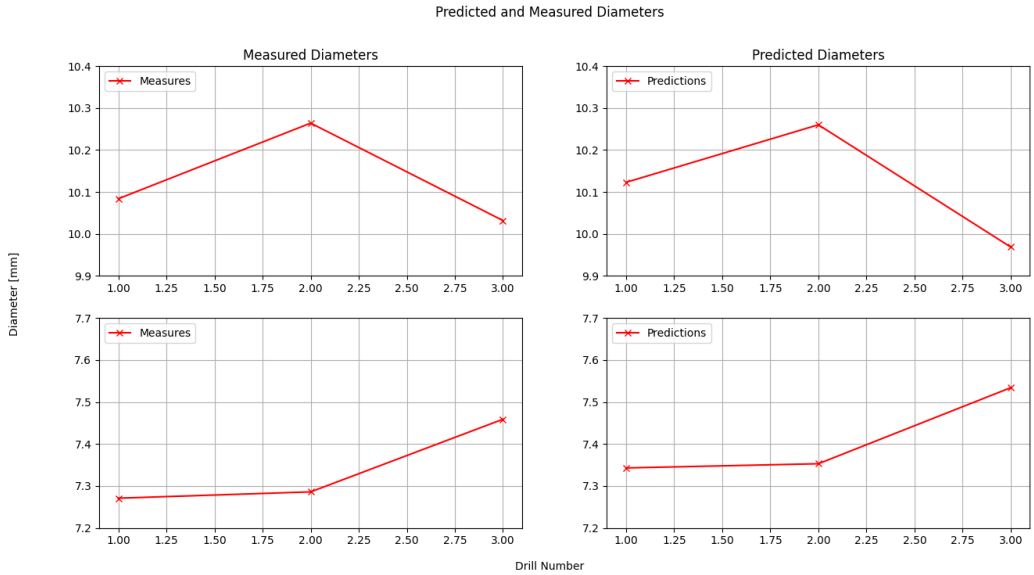

Figura 4-9. Comparación de la relación entre predicciones y entre medidas

Al analizar los errores relativos cometidos en la predicción de los diámetros del avellanado del set de testeo (Figura 4-10 y Figura 4-11) podemos observar que todos se encuentran dentro del margen de tolerancia de 90 micras que habíamos comentado en apartados anteriores, así que parece que nuestra red funciona correctamente y cumple con nuestros requisitos.

Es destacable el hecho de que nuestra red estime con mayor precisión los diámetros de los taladros de 4.9 mm. Muy posiblemente esto se deba a la escasez de imágenes con las que ha contado la red durante el entrenamiento, razón que ha impedido que esta llegue a aprender cómo estimar correctamente el diámetro del avellanado de los taladros de 4.1 mm.

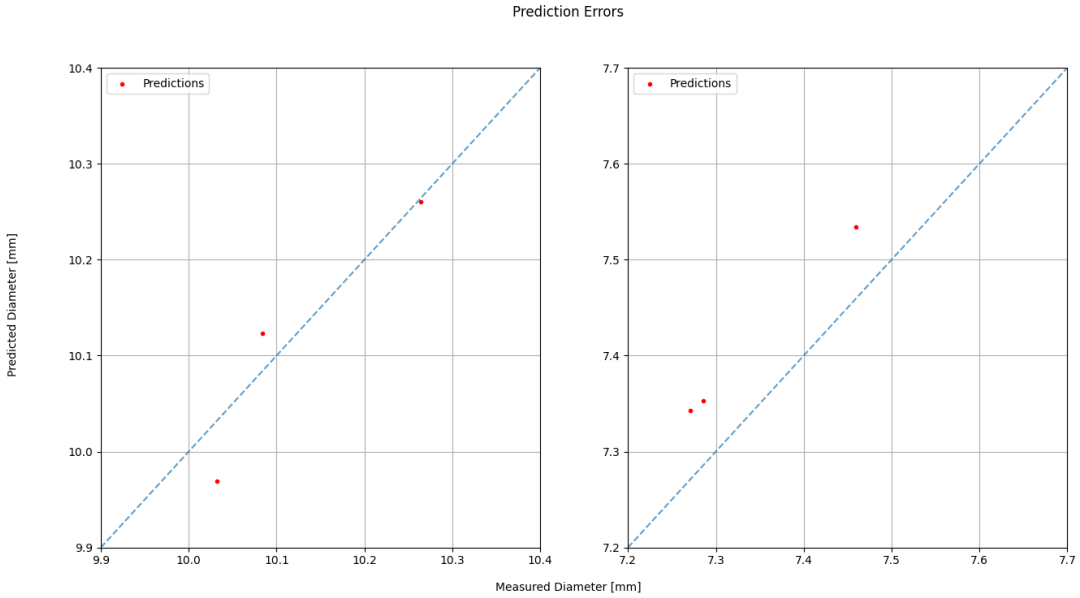

Figura 4-10. Errores en la predicción con respecto al valor medido del diámetro

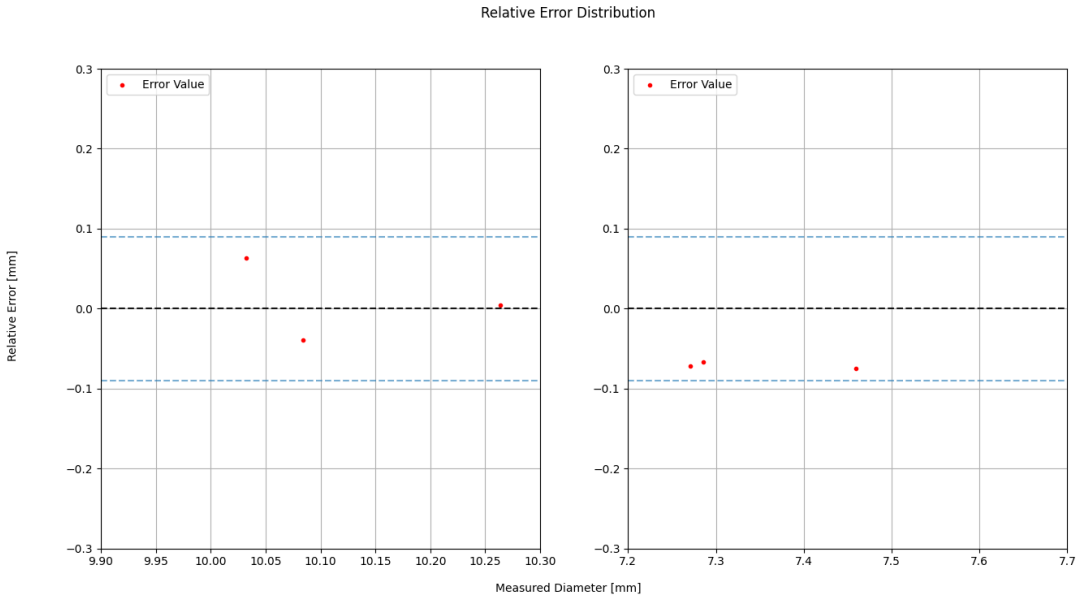

Figura 4-11. Errores cometidos en la predicción con respecto a los límites establecidos

5 CONCLUSIONES Y FUTURAS AMPLIACIONES

5.1. Implementación de la red neuronal en el flujo de trabajo

Una vez comprobado que la red neuronal devuelve predicciones en el rango de tolerancia deseado, esta estaría lista para ser implementada en la célula de taladrado automático.

Como ya hemos mencionado, las predicciones se realizarían justo antes de realizar un taladro en base al diámetro de avellanado resultante en el taladro inmediatamente anterior. Por cada taladrado pues, se pasaría por las siguientes fases:

1. Fotografía del taladro anterior
2. Predicción del diámetro de avellanado de la fotografía
3. Corrección en el avance de la EDU en base al diámetro estimado
4. Realización del taladro avellanado con el avance corregido

El cálculo de la corrección del avance de la EDU con respecto al diámetro de avellanado anterior se realizaría mediante una relación incorporada en un script de Python.

La obtención de dicha relación está delegada a otro departamento encargado de la fabricación, y a la fecha de finalización de este trabajo aún no ha comenzado. Por esta razón no ha sido posible realizar un estudio sobre las correcciones en el avance.

5.1.1 Mejoras esperadas

La implementación de la red neuronal en el flujo de trabajo conseguiría mejorar la eficiencia y la velocidad del proceso de taladrado por dos razones principales: La eliminación parcial del proceso de medición manual y el aumento de precisión en las mediciones.

Debido a que nuestra red neuronal es capaz de realizar predicciones del diámetro del avellanado de manera completamente autónoma, el proceso de medición del diámetro del avellanado utilizando el calibre queda relegado a un segundo plano.

Ya no será necesario medir manualmente con tanta frecuencia sobre los bordes de ataque puesto que será la red neuronal junto con la cámara la que se encargará de comprobar que no se están cometiendo errores sustanciales que acaben afectando al remachado y taladrado posteriores. Con la red también se evitarían los problemas asociados al error humano de la medición.

Gracias al entrenamiento al que hemos sometido a nuestra red neuronal se han conseguido realizar estimaciones de una precisión bastante elevada. De hecho, en algunas pruebas hechas con el set de validación se han devuelto predicciones con errores relativos destacables que no han tenido su origen en un malfuncionamiento de la red, sino en un error a la hora de tomar medidas.

Con la implementación de la red se busca reducir considerablemente el número de bordes de ataque desechados y, además, aumentar la velocidad de producción. La medición manual seguirá siendo necesaria de manera puntual en tareas de calibración o en el caso en el que se quieran añadir más muestras a los sets.

5.2. Posibles ampliaciones

5.2.1 Inclusión de más imágenes en los sets

Debido a que el trabajo se está realizando sobre una máquina operativa en la planta, hay dos grandes problemas a la hora de obtener imágenes para el entrenamiento de nuestra red neuronal.

Por un lado, está la disponibilidad de la célula de taladrado. Actualmente, la programación de la producción es bastante rígida debido al advenimiento de fechas de entrega críticas, por lo que la máquina taladradora no suele estar disponible para realizar todas las probetas que necesitaría nuestra red.

Por otro, la realización de probetas necesita de superficies de aluminio que imiten las chapas utilizadas en el borde de ataque, elementos que han sido difíciles de conseguir debido a la falta de material y a la saturación del departamento encargado de facilitarlas.

Bajo estas dos circunstancias, la cantidad de imágenes que se han podido obtener se aleja de la óptima. Nuestra red ha arrojado buenos resultados, pero debido a que el set de testeo no contiene muestras suficientes no podemos sacar una conclusión definitiva acerca de cómo se desenvolvería nuestra red en condiciones de trabajo.

Otro de los problemas que surgen de la falta de imágenes es el que comentábamos al final del capítulo anterior: La red no ha aprendido suficiente sobre los taladros de 4.1 mm, por lo que es posible que, en un futuro, con ciertas imágenes de taladros de 4.1 mm, las estimaciones superen el margen de error establecido.

Uno de los objetivos primordiales es aumentar el tamaño de los sets para mejorar el funcionamiento global de la red, ya sea mediante la realización de más probetas, o utilizando directamente las imágenes que se vayan obteniendo al taladrar directamente sobre el borde de ataque.

5.2.2 Mejora de la cámara

Actualmente la cámara utilizada en la toma de fotografías cumple los requisitos necesarios para que el entrenamiento de nuestra red sea satisfactorio.

Asimismo, es necesario recordar que ha habido que reajustar algunos parámetros de la cámara para que las imágenes se vean lo suficientemente nítidas, generándose por ello un ruido artificial que empeora la calidad de la imagen y afecta al aprendizaje.

Si se utilizara una cámara de mayor calidad y más adecuada al entorno de trabajo en el que funciona, la parametrización podría no ser necesaria y, por tanto, no se generaría el ruido artificial correspondiente. Esto podría llegar a hacer que las funciones de percepción visual de alto nivel funcionaran, no haciendo necesaria la implementación de una red neuronal.

Igualmente, si aun así fuera necesario utilizar una CNN, una mejor cámara produciría imágenes de mayor calidad que ayudarían en el aprendizaje para obtener predicciones más exactas.

5.2.3 Uso de Transfer Learning

Se conoce como transfer learning al proceso mediante el cual se utiliza el aprendizaje adquirido por una red neuronal convolucional para transferirlo y aplicarlo a tareas de un carácter similar, normalmente de clasificación y regresión.

La ventaja principal de este método es que solo necesita entrenar las últimas capas de la red neuronal, haciendo el proceso mucho menos exigente si los sets de entrenamiento son muy grandes. Esto se debe a que las primeras capas de la red ya están entrenadas para la extracción de características de las imágenes y no necesitan seguir reajustándose.

Algunas arquitecturas como la VGG-16 podrían haber servido como punto de partida para nuestra red neuronal, pudiendo incluso llegar a devolver mejores resultados. El problema es que muchas de estas arquitecturas han sido creadas específicamente para la clasificación de imágenes, por lo que los resultados obtenidos en problemas de regresión pueden no ser tan buenos.

Un factor a destacar de este tipo de redes son su tamaño en disco, que suele ser considerablemente superior al de redes de propósito específico puesto que han sido entrenadas con imágenes de todo tipo.

También es necesario tener en cuenta el formato de entrada que aceptan. En el caso de la VGG-16 el pre entrenamiento se ha realizado con imágenes de dimensión 224x224x3, por lo que para que nuestras imágenes de dimensión 800x800x4 pudieran introducirse en la red haría falta o bien reducir nuestras dimensiones, o bien reajustar los parámetros de la red para adecuarse a nuestro formato de entrada.

Ya sea por la pérdida de información resultante del reescalado o por el reajuste de los parámetros de la red, existe la posibilidad de que la precisión se reduzca drásticamente, quedando el modelo inutilizable para nuestra labor.

6 BIBLIOGRAFÍA

[1] Alestis Aerospace S.L., «Acerca de nosotros: Alestis Aerospace,» [En línea]. Available: https://www.alestis.aero/. [Último acceso: 3 Ago 2023].

[2] Airbus, "The A220 Family," [Online]. Available: https://web.archive.org/web/20180828224714/https://www.airbus.com/aircraft/passenger-aircraft/a220-family.html. [Accessed 31 Jul 2023].

[3] Planespotters.net, "Aviation Photos, Airline Fleets & more," [Online]. Available: https://www.planespotters.net/. [Accessed 31 Jul 2023].

[4] O. Memon, "How The Bombardier C-Series Program Transitioned To The Airbus A220 Family," 14 Mar 2023. [Online]. Available: https://simpleflying.com/bombardier-c-series-airbus-a220-transition-story/. [Accessed 31 Jul 2023].

[5] Bombardier, "Bombardier announces new commercial aircraft family name at Farnborough Airshow 2004," 19 Jul 2004. [Online]. Available: https://web.archive.org/web/20160920134840/http://www.bombardier.com/en/media/newsList/details.1304-bombardier-announces-new-commercial-aircraft-family-name-at-farnborough-airshow-2004.bombardiercom.html. [Accessed 31 Jul 2023].

[6] CPaT.com, "Airbus A220-100 To Airbus A220-300 Differences Training Course," [Online]. Available: https://www.cpat.com/courses/airbus-220/a220-100-to-a220-300-training-course/. [Accessed 3 Aug 2023].

[7] P. Díaz, «Se viene el Airbus A220-500: reemplazo del A320, rival del Boeing 737 MAX,» 5 May 2023. [En línea]. Available: https://www.aviacionline.com/2023/05/se-viene-el-airbus-a220-500-reemplazo-del-a320-rival-del-boeing-737-max/. [Último acceso: 3 Ago 2023].

[8] J. M. González, «Los remaches en aeronáutica,» 12 Mar 2018. [En línea]. Available: https://aertecsolutions.com/2018/03/12/los-remaches-en-aeronautica/. [Último acceso: 3 Ago 2023].

[9] M. Abdulnasir and J. R. Banerjee, "Free vibration and flutter analysis of high aspect ratio wings," 1 May 2017. [Online]. Available: https://www.researchgate.net/figure/Aircraft-flutter-representation_fig1_318274988.

[10] Wonkee Donkee Tools, "What are the parts of a solid rivet?," [Online]. Available: https://www.wonkeedonkeetools.co.uk/rivets/what-are-the-parts-of-a-solid-rivet. [Accessed 3 Aug 2023].

[11] A. Zhao, Y. Liu, H. Dong, Y. Bi and J. Liu, "Numerical and experimental investigation on the rivet head flushness in automatic countersunk riveting," 1 Sep 2020. [Online]. Available: https://www.researchgate.net/figure/The-schematic-diagram-of-the-riveting-process-for-the-countersunk-rivet_fig1_343649703.

[12] IVAO Documentation Library, "Leading Edge Devices," 29 Apr 2023. [Online]. Available: https://wiki.ivao.aero/en/home/training/documentation/Leading_Edge_Devices. [Accessed Jul 31 2023].

[13] TMAS Aviación, «Estructura y tipos de estabilizadores,» [En línea]. Available: https://www.tmas.es/blog/mecanica-de-aviones/estructura-y-tipos-de-estabilizadores/#horizontal. [Último acceso: 4 Ago 2023].

[14] "Leading edge skin right wing," 19 Sep 2019. [Online]. Available: https://www.latten.net/sling2/2019/09/19/leading-edge-skin-right-wing/. [Accessed 4 Aug 2023].

[15] KUKA, «KR 120 R2700-2,» 26 May 2022. [En línea]. Available: https://www.kuka.com/-/media/kuka-downloads/imported/8350ff3ca11642998dbdc81dcc2ed44c/0000325899_es.pdf?rev=d7e4a8ad4101434fbb6831cb9e6d3e84&hash=48E2F5A580E507879C36DB9FACCE6F14.

[16] REEL AEI, «Productos,» [En línea]. Available: https://www.reel-aei.com/es/productos/. [Último acceso: 4 Ago 2023].

[17] KUKA, "Industrial robotics: Linear units and positioners," [Online]. Available: https://pdf.directindustry.es/pdf-en/kuka-ag/kuka-linear-units-positioners/17587-752108.html#open2420765.

[18] A. M. Turing, "COMPUTING MACHINERY AND INTELLIGENCE," 1950. [Online]. Available: https://redirect.cs.umbc.edu/courses/471/papers/turing.pdf.

[19] IBM, "What is artificial intelligence (AI)?," [Online]. Available: https://www.ibm.com/topics/artificial-intelligence. [Accessed 5 Aug 2023].

[20] Gobierno de España, «Qué es la Inteligencia Artificial,» 19 Abr 2023. [En línea]. Available: https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr. [Último acceso: 5 Ago 2023].

[21] A. Rockwell, "The History of Artificial Intelligence," 28 Aug 2017. [Online]. Available: https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/.

[22] D. Kiela *et al.*, "Dynabench: Rethinking Benchmarking in NLP," 7 Apr 2021. [Online]. Available: https://arxiv.org/abs/2104.14337.

[23] LightsOnData, "The history of Machine Learning," [Online]. Available: https://www.lightsondata.com/the-history-of-machine-learning/#:~:text=Machine%20learning%20history%20starts%20in,by%20Donald%20Hebb%20is%20published.. [Accessed 1 Aug 2023].

[24] T. Cover and P. Hart, "Nearest neighbor pattern classification," Jan 1967. [Online]. Available: https://ieeexplore.ieee.org/document/1053964.

[25] R. Sharma, "4 Types of Machine Learning and How to Build a Great Career in Each," 10 Mar 2023. [Online]. Available: https://emeritus.org/in/learn/types-of-machine-learning/. [Accessed 1 Aug 2023].

[26] N. K. Chauhan and K. Singh, "A Review on Conventional Machine Learning vs Deep Learning," 28 Sep 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8675097/.

[27] IBM, «¿Qué es Deep Learning?,» [En línea]. Available: https://www.ibm.com/es-es/topics/deep-learning. [Último acceso: 1 Ago 2023].

[28] MedWhat, «Tu asistente médico personal,» [En línea]. Available: https://medwhat.com/es/inicio/. [Último acceso: 5 Ago 2023].

[29] IBM, «What are neural networks?,» [En línea]. Available: https://www.ibm.com/topics/neural-networks. [Último acceso: 5 Aug 2023].

[30] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," 2 Dec 2015. [Online]. Available: https://arxiv.org/pdf/1511.08458.pdf.

[31] V. Jain, "Everything you need to know about "Activation Functions" in Deep learning models," 30 Dec 2019. [Online]. Available: https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253?gi=38fd531ba9b9. [Accessed 5 Aug 2023].

[32] AI Wiki, "Activation Function," 2020. [Online]. Available: https://machine-learning.paperspace.com/wiki/activation-function. [Accessed 5 Aug 2023].

[33] Google Developers, «Algoritmo de propagación inversa,» [En línea]. Available: https://developers-dot-devsite-v2-prod.appspot.com/machine-learning/crash-course/backprop-scroll?hl=es-419. [Último acceso: 5 Ago 2023].

[34] F. Sancho, «Entrenamiento de Redes Neuronales: mejorando el Gradiente Descendente,» 25 Dic 2020. [En línea]. Available: http://www.cs.us.es/~fsancho/?e=165. [Último acceso: 5 Ago 2023].

[35] IBM, "What is gradient descent?," [Online]. Available: https://www.ibm.com/topics/gradient-descent. [Accessed 5 Aug 2023].

[36] Aprende Machine Learning, «Qué es overfitting y underfitting y cómo solucionarlo,» 12 Dic 2017. [En línea]. Available: https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/. [Último acceso: 6 Ago 2023].

[37] M. Sotaquirá, «Tutorial Python: ¿Cómo combatir el Overfitting en el Machine Learning?,» 1 Oct 2019. [En línea]. Available: https://www.codificandobits.com/blog/tutorial-overfitting-machine-learning-python/#tres-t%C3%A9cnicas-para-reducir-el-overfitting. [Último acceso: 6 Ago 2023].

[38] R. Gençay and M. Qi, "Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging," 1 Aug 2001. [Online]. Available: https://www.researchgate.net/figure/Early-stopping-based-on-cross-validation_fig1_3302948.

[39] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," 15 Dec 2018. [Online]. Available: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way. [Accessed 6 Aug 2023].

[40] E. Zvornicanin, "What Is Depth in a Convolutional Neural Network?," 5 May 2023. [Online]. Available: https://www.baeldung.com/cs/cnn-depth. [Accessed 6 Aug 2023].

[41] Tashmit, "Convolution layer, Padding, Stride, and Pooling in CNN," 30 Jun 2023. [Online]. Available: https://www.codingninjas.com/studio/library/convolution-layer-padding-stride-and-pooling-in-cnn. [Accessed 6 Aug 2023].

[42] A. W. Harley, "An Interactive Node-Link Visualization of Convolutional Neural Networks," 2015. [Online].

[43] N. Aljaafari, «Ichthyoplankton Classification Tool using Generative Adversarial Networks and Transfer Learning,» 10 Feb 2018. [En línea]. Available: https://www.researchgate.net/figure/Example-of-max-pooling-and-average-pooling-operations-In-this-example-a-4x4-image-is_fig4_332092821.

[44] TensorFlow, «Keras,» 24 Ene 2022. [En línea]. Available: https://www.tensorflow.org/guide/keras. [Último acceso: 7 Ago 2023].

[45] Ultralytics, "YOLOv8," 5 Aug 2023. [Online]. Available: https://github.com/ultralytics/ultralytics. [Accessed 7 Aug 2023].

[46] JAI, «GOX-5103-PGE 5.01-megapixel CMOS global shutter,» [En línea]. Available: https://multipix.com/wp-content/uploads/2021/01/Datasheet_GOX-5103-PGE.pdf.

[47] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 22 Dec 2014. [Online]. Available: https://arxiv.org/abs/1412.6980.

[48] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," 1943. [Online]. Available: https://link.springer.com/article/10.1007/BF02478259.