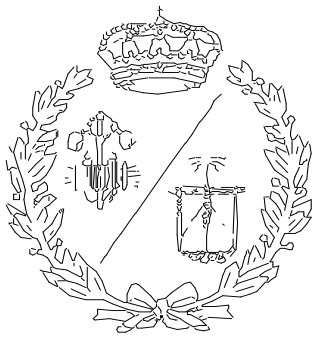
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

MONITORIZADO DEL PROCESO DE TREFILADO MEDIANTE ADQUISICIÓN DE IMÁGENES Y USO DE REDES NEURONALES

Para acceder al Título de

GRADUADO EN INGENIERÍA MECÁNICA

Autor: Raúl Rivero Mier

Febrero
2025

RESUMEN

Este Trabajo de Fin de Grado (TFG) se centra en la monitorización del proceso de trefilado mediante la adquisición de imágenes y su posterior procesamiento utilizando una red neuronal convolucional, específicamente ResNet50, implementada en MATLAB. El objetivo principal es identificar y clasificar imágenes del proceso de trefilado, diferenciando aquellas con defectos de las que no presentan anomalías, así como identificar el tipo de defecto que aparecen en las imágenes.

Para llevar a cabo este estudio, se capturaron imágenes de muestras de cables obtenidos mediante el proceso de trefilado. Las imágenes fueron etiquetadas manualmente para entrenar y evaluar la red neuronal. Se utilizó la red neuronal ResNet50, con el objetivo de encontrar la solución óptima. El modelo fue entrenado y ajustado en MATLAB, empleando un conjunto de datos balanceado para evitar sesgos en la clasificación.

Los resultados muestran una precisión equilibrada del 98.1% en la detección de defectos y un 98.11% en la clasificación de los mismos, destacando la efectividad de ResNet50 para este tipo de aplicaciones industriales. Además, se discuten las limitaciones del estudio y se proponen mejoras futuras, como la utilización de técnicas de aumento de datos y el empleo de cámaras de mayor resolución.

En conclusión, la implementación de un sistema de monitorización basado en la adquisición de imágenes y el procesamiento mediante redes neuronales se presenta como una solución viable y efectiva para la identificación y clasificación de defectos en el proceso de trefilado.

ABSTRACT

This Final Degree Project (TFG) focuses on monitoring the wire drawing process through image acquisition and subsequent processing using a convolutional neural network, specifically ResNet50, implemented in MATLAB. The main objective is to identify and classify images from the wire drawing process, distinguishing those with defects from those without anomalies, as well as identifying the type of defect present in the images.

For this study, images of wire samples obtained through the wire drawing process were captured. The images were manually labeled to train and evaluate the neural network. The ResNet50 neural network was used with the goal of finding the optimal solution. The model was trained and fine-tuned in MATLAB, using a balanced dataset to avoid biases in classification.

The results show a balanced accuracy of 98.1% in defect detection and 98.11% in defect classification, highlighting the effectiveness of ResNet50 for this type of industrial application. Additionally, the study's limitations are discussed, and future improvements are proposed, such as the use of data augmentation techniques and higher-resolution cameras.

In conclusion, the implementation of a monitoring system based on image acquisition and processing using neural networks is presented as a viable and effective solution for defect identification and classification in the wire drawing process.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

1.1 CONTEXTO Y JUSTIFICACIÓN

El trefilado es un proceso fundamental en la industria metalúrgica, utilizado para la fabricación de alambres y otros productos metálicos de alta precisión mediante el paso del cable por hileras reduciendo su sección. La detección de defectos en el proceso de trefilado y su clasificación es crucial para garantizar el buen funcionamiento del proceso, la calidad del producto final y evitar pérdidas económicas significativas. Las técnicas tradicionales de monitorización pueden ser costosas y difíciles de implementar en tiempo real. En este contexto, la utilización de métodos basados en la visión por computadora y el trabajo con redes neuronales ofrece una alternativa prometedora en la detección y clasificación de defectos originados en el proceso de trefilado.

1.2 OBJETIVOS

El objetivo principal de este Trabajo Fin de Grado (TFG) es desarrollar un sistema de monitorización del proceso de trefilado mediante la adquisición de imágenes y su procesamiento utilizando una red neuronal implementada en MATLAB, de modo que el sistema de monitorización sea capaz de detectar la existencia de defectos, así como poder clasificar entre varios tipos de defectos. Específicamente, se pretende:

- Capturar imágenes del proceso de trefilado.
- Procesar las imágenes obtenidas para un mejor funcionamiento de la red neuronal.
- Entrenar y evaluar el modelo de red neuronal ResNet50 para clasificar imágenes con y sin defectos.
- Entrenar y evaluar el modelo de red neuronal ResNet50 para la identificación del tipo de defecto encontrado.
- Analizar la precisión y efectividad del modelo en la detección de defectos.
- Proponer mejoras y aplicaciones futuras del sistema desarrollado.

1.3 ESTRUCTURA DEL DOCUMENTO

Este documento se estructura en seis capítulos principales:

- **Capítulo 2.** Se presenta el marco teórico, incluyendo una descripción del proceso de trefilado y los fundamentos de la monitorización de procesos industriales, el procesamiento de imágenes y las redes neuronales convolucionales, con un enfoque particular en ResNet50.

- **Capítulo 3.** Detalla la metodología empleada, abarcando la adquisición de imágenes, la preparación de datos y la configuración y entrenamiento de la red neuronal.

- **Capítulo 4.** Se presentan los resultados obtenidos del análisis y evaluación del modelo.

- **Capítulo 5.** Discute los hallazgos, las limitaciones del estudio y propone posibles mejoras.

- **Capítulo 6.** Concluye con un resumen de los hallazgos y recomendaciones para futuros trabajos.

2 MARCO TEÓRICO

2.1 PROCESO DE TREFILADO

El trefilado es una técnica de deformación plástica en frío, donde el material es empujado a través de una hilera (troquel o matriz) que tiene una abertura más pequeña que la sección original del material. Este proceso reduce el diámetro del material mientras aumenta su longitud, sin pérdida significativa de masa. El trefilado se utiliza con metales como el acero, aluminio, cobre y sus aleaciones, y es esencial en la producción de cables, alambres de alta resistencia, tubos y varillas. [15][26].

2.1.1 Tipos de trefilado

Existen varios tipos de trefilado, dependiendo del material a procesar, el uso final del producto y la maquinaria empleada [1]. Los tipos principales son:

- **Trefilado en seco**: El material pasa por la matriz sin la presencia de un lubricante líquido. Se utiliza principalmente en procesos donde el acabado superficial no es crítico, como el trefilado de barras y perfiles gruesos.

- **Trefilado en húmedo**: En este tipo de trefilado, la matriz y el alambre están inmersos en un lubricante líquido, lo que reduce la fricción y mejora el acabado superficial. Es común en el trefilado de alambres finos y de alta precisión.

- **Trefilado flotante**: En este proceso, el alambre no tiene contacto directo con la matriz en todo momento, lo que reduce el desgaste de la matriz y es útil para alambres de grandes longitudes.

- **Trefilado múltiple**: Aquí, el material pasa por varias matrices de manera sucesiva en una misma línea de producción. Es utilizado para obtener alambres muy delgados o largos.

En la aplicación el trefilado más utilizado en la industria es el trefilado múltiple con lubricación. El cable pasa por varias matrices y se enrrolla en tambores hasta obtener el producto final.

En la Figura 1 y en la Figura 2 se muestra el funcionamiento de un banco de trefilado común.

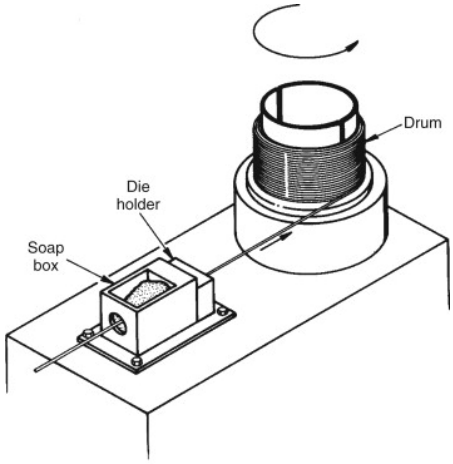


FIGURA 1. Módulo de banco de trefilado.

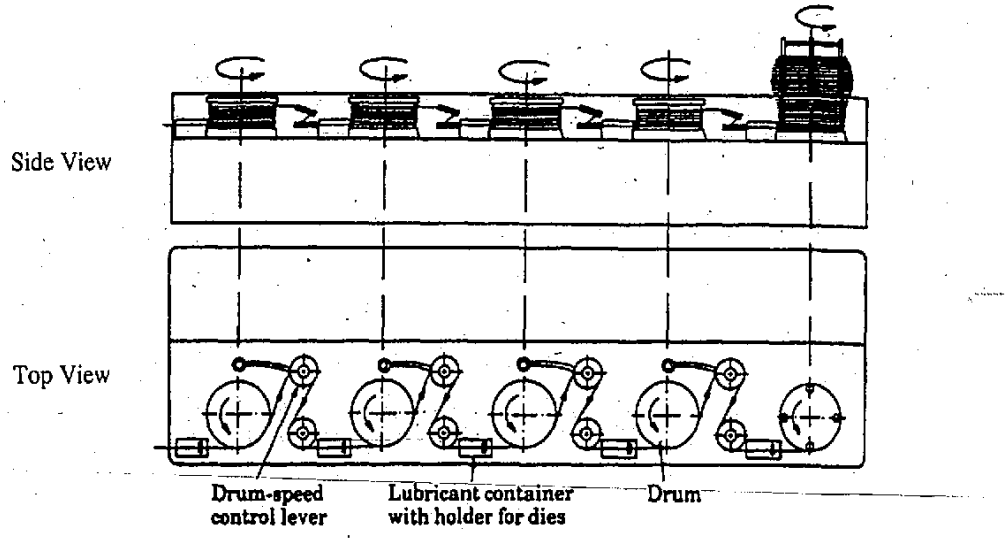


FIGURA 2. Diagrama de funcionamiento de un banco de trefilado múltiple.

2.1.2 Características del proceso de trefilado

Las principales características del proceso de trefilado son [10]:

- **Reducción gradual del diámetro**: Cada pasada a través de una matriz reduce el diámetro del material, lo que permite un control preciso del espesor final.

- **Trabajo en frío**: El trefilado es un proceso en frío, lo que significa que se realiza a temperaturas por debajo de la temperatura de recristalización del material. Esto provoca un endurecimiento del material (aumento de su resistencia) debido a la deformación plástica.

- **Control dimensional**: El proceso permite obtener diámetros muy precisos, lo cual es fundamental en aplicaciones que requieren tolerancias estrechas, como el trefilado de cables eléctricos o alambres para componentes mecánicos.

- **Superficies lisas**: En el trefilado en húmedo, el uso de lubricantes permite obtener acabados superficiales de alta calidad, lo que es importante en aplicaciones donde la rugosidad superficial afecta el rendimiento, como los cables de acero.

El proceso de trefilado presenta grandes ventajas [15][28], entre las que se encuentran:

- **Precisión dimensional**: Permite obtener alambres y perfiles con diámetros y secciones transversales muy precisas.

- **Superficie lisa**: El uso de lubricantes en el trefilado en húmedo proporciona superficies muy suaves y con baja rugosidad.

- **Aumento de la resistencia mecánica**: El trabajo en frío endurece el material, aumentando su resistencia a la tracción y su dureza superficial, lo que es beneficioso en aplicaciones que requieren alta resistencia.

- **Versatilidad**: El proceso se puede aplicar a una amplia variedad de metales y aleaciones, desde cobre y aluminio hasta aceros de alta resistencia.

- **Economía de material**: El trefilado es un proceso de deformación sin pérdida de material significativa, lo que lo hace eficiente desde el punto de vista del desperdicio.

A pesar de presentar grandes ventajas, también presenta una serie de desventajas. Las más importantes son:

- **Endurecimiento excesivo**: El trabajo en frío endurece el material, lo que puede hacer que sea frágil si no se realiza un recocido posterior, lo cual es necesario en algunos casos para recuperar la ductilidad.

- **Límites en la reducción de diámetros**: El trefilado tiene un límite en cuanto a la reducción de diámetro que puede lograrse en cada pasada. Para reducciones significativas, el material necesita múltiples pasadas por varias matrices, lo que incrementa el tiempo de producción.

- **Desgaste de las matrices**: Las hileras o matrices utilizadas en el proceso sufren desgaste, especialmente si se trabaja con materiales duros, lo que implica un mantenimiento constante y la reposición de las matrices.

- **Costo de equipos**: Las máquinas de trefilado, especialmente las de trefilado múltiple y en húmedo, pueden ser costosas, lo que aumenta los costos iniciales de inversión.

2.1.3 Defectos originados en el proceso de trefilado

Durante el proceso de trefilado el material se somete a esfuerzos de tracción y compresión, lo que puede provocar la aparición de defectos si el proceso no se lleva a cabo de manera adecuada [28]. En la Figura 3 se muestran las variables de proceso de trefilado de un alambre.

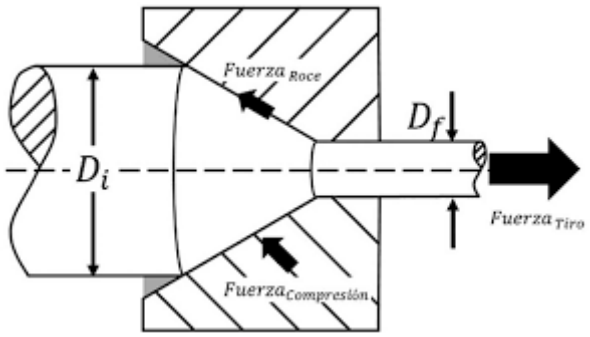


FIGURA 3. Variables del proceso de trefilado.

Los defectos más comunes que se pueden observar en el proceso de trefilado son [12]:

- **Rotura del alambre o barra**: Esto ocurre cuando las tensiones en el material superan su límite de resistencia durante el proceso de trefilado. Puede ser provocado por una reducción excesiva del diámetro en una sola pasada o por una velocidad de trefilado muy alta. (Figura 4)

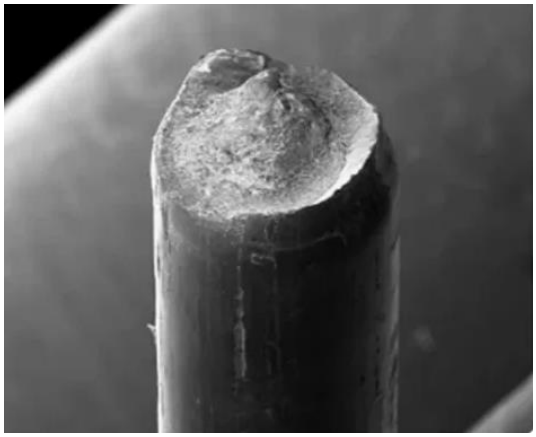


FIGURA 4. Defecto rotura de alambre.

- **Grietas longitudinales**: Estas grietas suelen aparecer a lo largo del alambre o barra y pueden deberse a un material inicial defectuoso, a una reducción excesiva en el trefilado, o a un mal lubricado durante el proceso. (Figura 5)

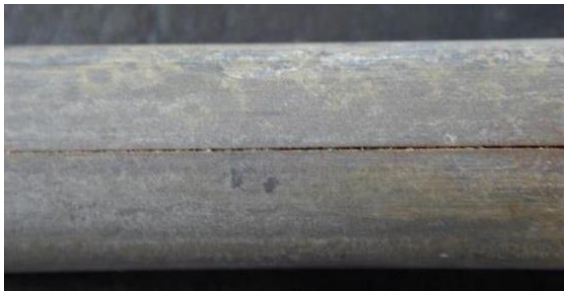


FIGURA 5. Defecto de grieta longitudinal.

- **Grietas transversales**: Se producen por tensiones internas elevadas debido a reducciones rápidas del diámetro o por defectos superficiales previos en el material. También pueden ser causadas por un mal ajuste de las matrices. (Figura 6)

FIGURA 6. Defecto de grieta transversal.

- **Ondulaciones en la superficie**: Este defecto es visible como una serie de ondulaciones en la superficie del alambre y ocurre cuando hay una reducción excesiva en el diámetro durante una pasada, o cuando la velocidad de trefilado es inadecuada.

- **Arrugas o pliegues superficiales**: Las arrugas o pliegues pueden formarse debido a una mala lubricación o a una velocidad de trefilado inapropiada, que genera una acumulación de material en la entrada de la matriz. (Figura 7)

FIGURA 7. Defecto de arrugas superficiales.

- **Desgaste de la matriz**: Ocurre debido al contacto continuo entre la matriz y el material, especialmente si se utiliza con materiales duros como el acero. Este desgaste puede llevar a una reducción en la calidad dimensional del alambre y problemas en el acabado superficial.

- **Variación de sección**: Se da cuando el diámetro del alambre o barra varía a lo largo de su longitud. Esto puede ocurrir por un mal centrado de las matrices o por variaciones en la velocidad de trefilado.

- **Superficie áspera o rayada**: Puede ser producto de un mal lubricado, matrices desgastadas, o la presencia de partículas de material atrapadas entre el alambre y la matriz. (Figura 8)

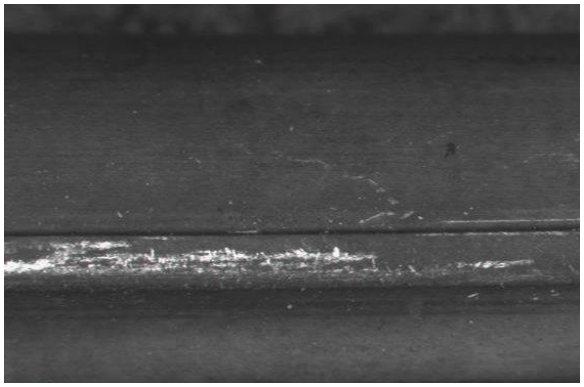


FIGURA 8. Defecto de superficie rayada.

- **Defectos internos (porosidad o inclusiones)**: Son defectos internos en el material que pueden derivarse de impurezas o inclusiones en el metal de partida, o de una deformación interna no homogénea durante el trefilado.

- **Desalineación**: Esto ocurre cuando el alambre o barra no sigue una trayectoria recta al salir de la matriz, generalmente debido a un mal ajuste de la hilera o matrices desalineadas.

- **Otros defectos** en la superficie originados por la selección inadecuada de los parámetros de proceso, lubricación deficiente o un mal estado del dado.

En este trabajo se busca identificar defectos superficiales, como rayaduras, grietas o picaduras. En la Figura 9, Figura 10, Figura 11, Figura 12 y Figura 13 se presentan algunos ejemplos de muestras de cables tomadas con los defectos buscados.

FIGURA 9. Cable con superficie rayada transversalmente.

FIGURA 10. Cable con superficie rayada transversalmente.

FIGURA 11. Cable con defecto de picadura.

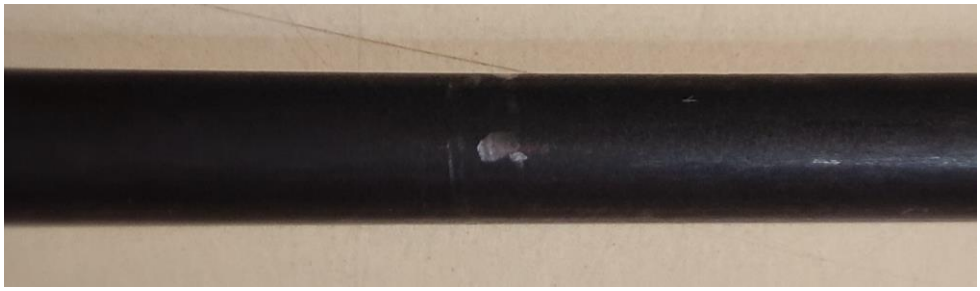


FIGURA 12. Cable con defecto superficial.

FIGURA 13. Cable con superficie rayada longitudinalmente.

La detección temprana de estos defectos es crucial para garantizar la calidad del producto final y reducir costos de producción.

2.2 MONITORIZACIÓN DE PROCESOS INDUSTRIALES

La monitorización de procesos industriales es una práctica esencial para garantizar la calidad y eficiencia en la producción. Involucra la recopilación y análisis de datos en tiempo real para detectar y corregir problemas antes de que afecten significativamente el producto final. Las técnicas tradicionales de monitorización incluyen inspecciones visuales, mediciones físicas y el uso de sensores específicos. Algunas de las técnicas de monitorización más utilizadas en la industria son:

- **SCADA (Supervisory Control and Data Acquisition)** [2][3]: Sistema de control y supervisión de procesos a gran escala. Permite la recolección de datos en tiempo real y el control de procesos industriales desde ubicaciones remotas. Utilizado en plantas de energía, redes de distribución de agua, sistemas de transporte y manufactura.

- **DCS (Distributed Control Systems)** [3]: Sistema de control distribuido que divide las tareas de control entre varias unidades dedicadas. Proporciona control local con supervisión centralizada. Común en industrias de procesos como la petroquímica, la química y la farmacéutica.

- **PLC (Programmable Logic Controllers)** [3]: Dispositivos electrónicos utilizados para automatizar procesos industriales. Programables para realizar tareas específicas de control y monitorización. Muy utilizados en manufactura, ensamblaje y control de maquinaria.

- **IoT (Internet of Things)** [2]: Uso de dispositivos conectados a internet para recopilar y compartir datos de procesos. Permite la monitorización en tiempo real y análisis avanzados. Aplicaciones en diversas industrias, incluyendo manufactura, energía y agricultura.

- **Mantenimiento Predictivo** [5]: Técnica que utiliza datos de sensores y análisis predictivo para anticipar fallos en los equipos antes de que ocurran. Utilizado en maquinaria crítica en manufactura, minería y transporte.

- **Análisis de Vibraciones** [6]: Técnica de monitorización que analiza las vibraciones de máquinas y equipos para detectar desbalances, desalineaciones y otros problemas mecánicos. Común en mantenimiento de equipos rotativos como motores, bombas y compresores.

- **Termografía Infrarroja** [6]: Uso de cámaras infrarrojas para detectar variaciones de temperatura en equipos y procesos. Permite identificar puntos calientes que indican problemas potenciales. Utilizado en mantenimiento eléctrico, mecánico y en inspección de edificios.

- **Análisis de Aceite** [6]: Evaluación de la condición del aceite lubricante en maquinaria para detectar desgaste, contaminación y otras condiciones anormales. Común en la monitorización de motores, engranajes y sistemas hidráulicos.
- **Medición de emisiones y calidad del aire** [20]: Monitorización de emisiones de gases y partículas para asegurar el cumplimiento de normativas ambientales y mejorar la eficiencia de los procesos. Importante en industrias químicas, petroquímicas y de generación de energía.

- **Sensores y Actuadores Inteligentes** [6]: Dispositivos que no solo recogen datos, sino que también pueden procesar información y actuar sobre el proceso en base a algoritmos predefinidos. Utilizados en manufactura avanzada y procesos automatizados.

- **Técnicas de Machine Learnig y Análisis de Datos** [13]: Aplicación de algoritmos de aprendizaje automático y análisis de big data para mejorar la monitorización y el control de procesos. Utilizados en procesos complejos donde se requiere análisis predictivo y detección de anomalías.

Con los avances en tecnología, las técnicas de visión por computadora y el análisis de datos se han convertido en herramientas poderosas para la monitorización de procesos. Estas tecnologías permiten la captura de imágenes y videos en tiempo real, seguidos de un procesamiento automatizado para identificar defectos y anomalías. La implementación de redes neuronales para el procesamiento de imágenes es una de las innovaciones más prometedoras en este campo.

2.3 MONITORIZACIÓN DE PROCESOS DE TREFILADO

Tradicionalmente el monitorizado del proceso de trefilado se realiza de forma manual, con mucha dependencia de las cualidades y aptitudes del operario. Esto ha provocado la búsqueda de un proceso de monitorizado automatizado, que muestre de forma clara y efectiva la manera en la que está funcionando el equipo de trefilado, de cara a encontrar posibles problemas en él. A raíz de estas investigaciones han aparecido formas de monitorizar el proceso de trefilado [16]. Algunos de estos métodos son: “Eddy Current testing”, métodos ópticos, medida de emisión acústica, medida de la fuerza de trefilado y medida de la temperatura entre otros.

A continuación, se describen los métodos más interesantes que se han llevado a cabo en los últimos años:

- **Monitorizado por sensores de vibración** [22]: Este método aparece por la facilidad de emplear un acelerómetro en el ámbito industrial. Un acelerómetro es capaz de identificar cambios un proceso industrial, por lo que, a raíz de las lecturas de los datos adquiridos, se puede identificar si existe algún tipo de problema en el proceso de trefilado.

- **Monitorizado por emisión acústica** [4]: Al monitorizar el proceso de trefilado con un sensor de emisión acústica, se puede apreciar la sensibilidad de las señales de emisión acústica a la variación de los diferentes parámetros que afectan al proceso. La señal de emisión acústica se ve alterada por cambios en la velocidad de trefilado, cambios de temperatura y cambios en el lubricante entre otros.

- **Monitorizado por cámara térmica** [16]: En este caso, se utiliza una cámara térmica para analizar la superficie del cable de sección reducida. Analizando las imágenes proporcionadas por la cámara térmica, se pueden encontrar anomalías en el caso de mal funcionamiento del proceso de trefilado.

- **Monitorizado mediante web cam** [17]: Este método utiliza una cámara web para analizar el funcionamiento del proceso de trefilado. En este caso se analiza la reflectividad de las imágenes y se encuentra correlación con la fuerza de trefilado, de modo que se puede detectar cuando existe una lubricación deficiente.

2.4 PROCESAMIENTO DE IMÁGENES

El procesamiento de imágenes es una técnica que permite la manipulación y análisis de imágenes digitales mediante algoritmos computacionales [23]. Sus aplicaciones son muchas y van desde la medicina hasta la seguridad y la industria. En el contexto industrial, el procesamiento de imágenes se utiliza para la inspección y monitorización de procesos, permitiendo la detección automática de defectos y la medición precisa de características [21]. Las técnicas comunes de procesamiento de imágenes incluyen [7]:

- **Preprocesamiento de imágenes**: Consiste en mejorar la calidad de la imagen antes de aplicarle algoritmos más complejos. Algunas de las técnicas más comunes son:
    - **Redimensionado:** Ajustar las dimensiones de una imagen para cambiar su tamaño, útil cuando se necesitan imágenes de dimensiones específicas.
    - **Normalización:** Se utiliza para ajustar la intensidad de los píxeles de una imagen dentro de un rango estándar.
    - **Enfoque o filtrado:** Mejora los bordes o detalles importantes de la imagen, reduciendo el desenfoque o ruido.
- **Aumento de datos**: En muchos proyectos, especialmente en inteligencia artificial, se utiliza para generar nuevas imágenes a partir de las existentes, aumentando el tamaño del conjunto de datos de entrenamiento. Algunas técnicas comunes son:
    - **Espejo horizontal:** Reflejar la imagen sobre el eje horizontal para obtener una nueva versión.
    - **Zoom:** Ampliar o reducir la imagen para simular diferentes distancias del objeto capturado.
    - **Ruido gaussiano:** Añadir ruido a la imagen para simular imperfecciones y aumentar la robustez de los modelos.

- **Transformación de imágenes**: Cambios geométricos y de perspectiva para mejorar el análisis:
    - **Rotación:** Girar la imagen en ángulos específicos.
    - **Escalado:** Aumentar o reducir el tamaño de la imagen.
    - **Traslación:** Desplazar la imagen en el plano sin cambiar su forma o tamaño.
- **Filtrado y suavizado**: Utilizado para eliminar ruido o mejorar características. Algunas técnicas comunes son:
    - **Filtro gaussiano:** Suaviza la imagen reduciendo el ruido sin afectar demasiado los detalles importantes.
    - **Filtro mediana:** Se utiliza para eliminar ruido de tipo impulsivo ("salt-and-pepper noise") mientras preserva los bordes.
- **Segmentación**: Proceso de dividir una imagen en varias regiones o segmentos para identificar objetos o áreas de interés. Ejemplos:
    - **Segmentación por umbral:** Dividir la imagen en partes según niveles de intensidad de píxeles.
    - **Segmentación basada en contornos:** Identificar los bordes de los objetos en la imagen.
- **Detección de características**: Se refiere a la identificación de puntos clave en la imagen, como bordes, esquinas o texturas, que son fundamentales para análisis más avanzados como el reconocimiento de objetos o la reconstrucción 3D.

- **Compresión de imágenes**: Reducción del tamaño de las imágenes para su almacenamiento o transmisión. Ejemplos incluyen:

    - **Compresión con pérdida (JPEG):** Disminuye el tamaño eliminando detalles que el ojo humano difícilmente percibe.

    - **Compresión sin pérdida (PNG):** Comprime la imagen manteniendo toda la información original.

Cada una de estas técnicas tiene aplicaciones específicas dependiendo del objetivo, ya sea mejorar la visualización, preparar la imagen para análisis posterior, o generar nuevas versiones para entrenar modelos de aprendizaje automático

El uso de algoritmos avanzados, como las redes neuronales, ha revolucionado el campo del procesamiento de imágenes, permitiendo un análisis más preciso y eficiente.

2.5 REDES NEURONALES

2.5.1 Red Neuronal Simple

Una red neuronal simple [9], también conocida como red neuronal densa o red de perceptrón multicapa (MLP, por sus siglas en inglés), es la forma más básica de red neuronal. En una red neuronal simple, cada neurona de una capa está conectada a todas las neuronas de la capa siguiente, lo que se conoce como una estructura completamente conectada o "fully connected".

Estructura

Las redes neuronales simples constan de al menos tres capas:

- **Capa de entrada:** Esta capa es simplemente el punto de entrada de los datos crudos. Cada característica del conjunto de datos de entrada se asigna a una neurona en esta capa. Si estás trabajando con imágenes, cada píxel de la imagen sería una entrada individual (es decir, cada neurona en esta capa representaría un píxel si la imagen no se ha preprocesado). Por ejemplo, si tienes una imagen de 28x28 píxeles en escala de grises, habría 784 neuronas en la capa de entrada.

- **Capas ocultas:** Las capas ocultas son donde ocurre la mayor parte del aprendizaje en una red neuronal. Estas capas transforman las entradas crudas en representaciones más abstractas que son útiles para la tarea final (como la clasificación).

  - **Conexiones completamente conectadas:** En cada capa oculta, todas las neuronas están conectadas a todas las neuronas de la siguiente capa. Estas conexiones se representan mediante pesos, que son parámetros ajustados durante el proceso de entrenamiento.
    Cada neurona en una capa oculta realiza una combinación ponderada de las salidas de las neuronas de la capa anterior. Matemáticamente, esto se representa como:

$$z = \sum_i w_i * x_i + b$$

    Donde:

    w_i son los pesos de las entradas.

    x_i son las entradas de la neurona.

    b es el sesgo, un parámetro adicional que se ajusta para mejorar la capacidad de ajuste de la red.

  - **Función de activación:** Después de la combinación lineal de las entradas, el valor z se pasa a través de una función de activación que introduce no linealidad en el modelo, permitiendo que la red aprenda relaciones complejas. Las funciones de activación comunes son:

$$ReLU\ (z) = \max(0, z)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

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

- **Capa de salida:** La capa de salida produce las predicciones de la red. La cantidad de neuronas en esta capa depende del tipo de tarea:

  - En una tarea de **clasificación binaria**, la capa de salida suele tener una única neurona, que predice la probabilidad de una de las dos clases.

- En **clasificación multiclase**, hay tantas neuronas como clases, y normalmente se utiliza la función de activación **softmax** para convertir las salidas en probabilidades. De esta forma, la red puede predecir la probabilidad de pertenencia de cada clase.

$$softmax\ (z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

En la Figura 14 se muestra la arquitectura de una red neuronal simple.

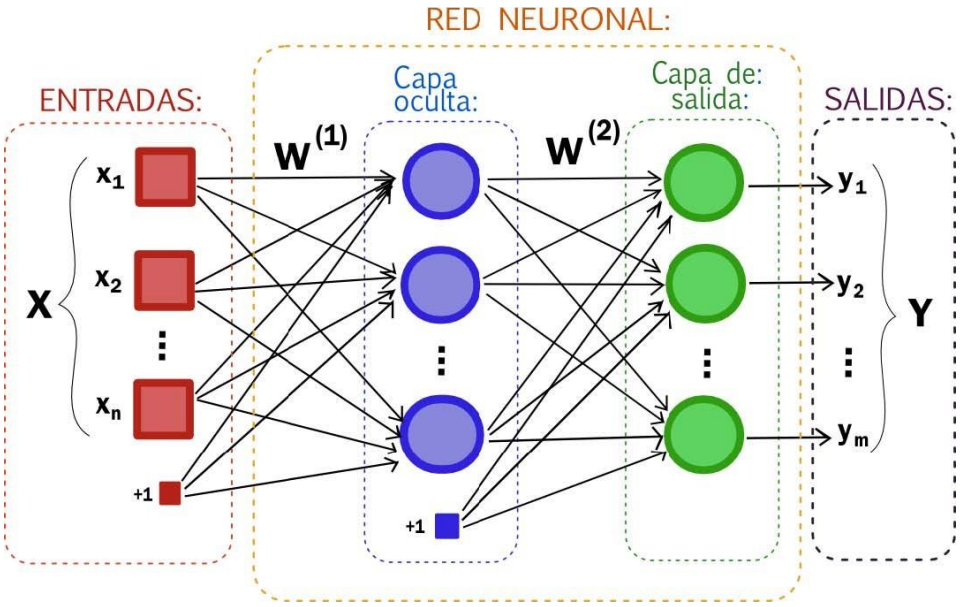


FIGURA 14. Arquitectura de una red neuronal simple

Funcionamiento

Cada neurona en la red toma una combinación lineal de las entradas que recibe de las neuronas anteriores y luego pasa esta combinación a través de una función de activación no lineal. Esta no linealidad es clave para que la red pueda modelar relaciones complejas en los datos. Durante el entrenamiento, la red ajusta los pesos de las conexiones entre las neuronas para minimizar el error cometido en la predicción, utilizando algoritmos como el descenso de gradiente y la retropropagación.

- **Paso hacia adelante (forward pass):** Durante este paso, los datos de entrada se pasan a través de la red, y cada capa transforma las entradas a través de combinaciones lineales y funciones de activación. Finalmente, la capa de salida genera una predicción.

- **Función de costo (pérdida):** La función de costo mide qué tan lejos está la predicción de la verdad, es decir, el valor real de salida. Para problemas de clasificación, una función de pérdida común es la entropía cruzada.

- **Retropropagación (backpropagation):** Este es el paso en el que la red aprende. Se calcula el gradiente de la función de costo con respecto a los pesos de la red, y luego, a través del descenso de gradiente (o un optimizador como Adam), se ajustan los pesos para minimizar la pérdida.

  La retropropagación se lleva a cabo utilizando la regla de la cadena, que permite calcular cómo la modificación de un peso en una capa afecta el error en la capa de salida.

2.5.2 Red Neuronal convolucional (CNN)

Las redes neuronales convolucionales (CNN) son una clase de redes neuronales especialmente diseñadas para el procesamiento y análisis de datos con una estructura de cuadrícula [9]. Las CNNs son extremadamente eficaces para tareas de visión por computadora, como clasificación de imágenes, detección de objetos y segmentación de imágenes. A continuación, se describe la estructura y funcionamiento de las rede neuronales convolucionales [8][25].

Estructura

Las CNNs tienen una estructura más compleja que las redes neuronales simples debido a las capas convolucionales. Los componentes clave de una CNN son:

- **Capas convolucionales:** La operación de convolución es el corazón de las CNNs. En lugar de conectar cada neurona de una capa a todas las neuronas de la siguiente, las capas convolucionales usan filtros (kernels) que se desplazan por la entrada para capturar características locales, como bordes o texturas.

  Los filtros (kernels) son pequeños "bloques" de neuronas, generalmente de 3x3 o 5x5, que se mueven por toda la entrada (imagen o característica). Cada filtro produce un mapa de características (feature map) al pasar sobre la entrada. La operación es una convolución 2D.

En las primeras capas, los filtros detectan características simples como bordes. En capas más profundas, los filtros detectan combinaciones de estas características, como formas y objetos completos.

- **Capa RELU**: Aplica una función de activación por elementos.

- **Capas pooling**: Se aplican para reducir la dimensionalidad de los mapas de características tomando pequeñas regiones de los mapas y seleccionando el valor más representativo. El pooling reduce la cantidad de parámetros, hace la red más eficiente y proporciona invariancia a traslaciones, lo que significa que pequeñas transformaciones en la entrada no afectan significativamente la predicción.
  El pooling más común es el **max pooling**, que selecciona el valor máximo en cada región de una ventana deslizante, reduciendo el tamaño del mapa sin perder información relevante.

- **Capas densas (fully conected):** Después de varias capas convolucionales y de pooling, las características aprendidas se "aplanan" (flattening) para formar un vector de características. Este vector se conecta a una o más capas densas como en las redes neuronales simples, y finalmente a la capa de salida, que produce las predicciones. Las capas densas actúan como clasificadores que utilizan las características aprendidas en las capas convolucionales para tomar decisiones finales.

En la Figura 15 se muestra la arquitectura de una red neuronal convolucional.

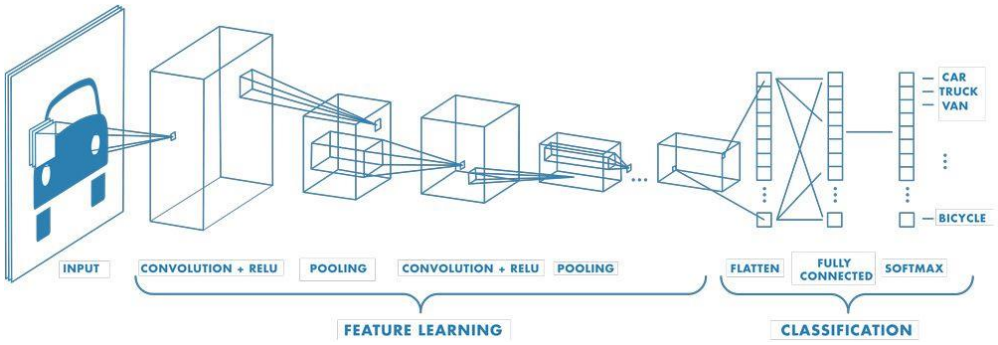


FIGURA 15. Arquitectura de una red neuronal convolucional.

Funcionamiento

El objetivo de las CNNs es aprender de forma jerárquica, comenzando por detectar patrones simples (bordes, esquinas) en las primeras capas y combinarlos progresivamente para reconocer formas y objetos más complejos en las capas más profundas. Las capas convolucionales actúan como detectores de características que pueden aprender automáticamente qué partes de la imagen son relevantes para la tarea de predicción.

- **Extracción de características:** Las capas convolucionales actúan como detectores de características locales. Los filtros se entrenan para identificar patrones clave en los datos de entrada, como bordes, texturas o formas. Estos filtros actúan de forma compartida, lo que significa que el mismo filtro se aplica a diferentes regiones de la entrada, lo que reduce drásticamente el número de parámetros.

- **Propagación de la información:** Al igual que en las redes simples, la información se propaga hacia adelante a través de las capas. Las características de bajo nivel detectadas en las primeras capas se combinan en características más abstractas en capas más profundas.

- **Clasificación:** Finalmente, las capas densas toman las características extraídas y las utilizan para clasificar las entradas en las categorías correspondientes. El entrenamiento ajusta los pesos en todas las capas, incluidas las capas convolucionales y las densas, para optimizar la precisión del modelo.

Ventajas frente a las redes neuronales simples

Las CNNs son mucho más potentes para manejar imágenes y otros datos donde las relaciones espaciales son fundamentales. A continuación, se presentan las ventajas más relevantes de las CNNs frentes a las redes neuronales convolucionales [19].

- **Mejor manejo de datos espaciales:** Las redes simples no consideran la estructura espacial de los datos de entrada (por ejemplo, la cercanía de píxeles en una imagen), lo que puede limitar su capacidad para reconocer patrones importantes. Las CNNs, al utilizar convoluciones locales, preservan la estructura espacial de las imágenes, permitiendo extraer características jerárquicas y hacer uso eficiente de la información de los píxeles vecinos.

- **Menos cantidad de parámetros:** En una red neuronal simple, todas las neuronas de una capa están conectadas a todas las neuronas de la capa siguiente, lo que puede generar una cantidad enorme de parámetros en problemas de entrada de alta dimensionalidad, como imágenes grandes.

  Las CNNs reducen este problema usando pequeñas ventanas convolucionales (filtros), que permiten trabajar con menos conexiones y parámetros, lo que hace más eficiente el entrenamiento y reduce el riesgo de sobreajuste.

- **Invarianza ante traslaciones y distorsiones:** Las redes convolucionales son más robustas a pequeñas variaciones en las entradas, como la traslación o rotación de un objeto en una imagen, gracias a las operaciones de pooling y convolución que capturan características locales con independencia de su posición exacta. Las redes simples, por otro lado, son sensibles a cambios de posición y requieren más datos o preprocesamiento para lograr esta invariancia.

- **Rendimiento superior en tareas de visión por computadora:** Las CNNs han demostrado ser muy eficaces en tareas como la clasificación de imágenes, el reconocimiento facial y la detección de objetos, superando ampliamente a las redes simples en términos de precisión y eficiencia en este tipo de problemas.

En este trabajo, se utiliza la red neuronal convolucional ResNet50.

2.5.3 ResNet50

En el campo del aprendizaje profundo (Deep Learning) [11], las redes neuronales convolucionales (CNNs) han demostrado ser altamente efectivas para tareas de visión por computadora, como la clasificación de imágenes, la detección de objetos y la segmentación. Sin embargo, a medida que se profundizan las redes (es decir, a medida que aumentan las capas), se enfrentan a ciertos problemas de rendimiento.

La arquitectura ResNet (Residual Network) fue propuesta para mitigar estos problemas. A continuación, se muestran las principales ventajas de utilizar bloques residuales en una red neuronal convolucional.

- **Problema del gradiente desaparecido:** En redes muy profundas, los gradientes (derivadas) necesarios para ajustar los pesos durante el entrenamiento tienden a volverse muy pequeños a medida que retropropagan desde las capas finales a las capas iniciales. Esto hace que las capas cercanas a la entrada aprendan muy lentamente. Las conexiones de salto en los bloques residuales permiten que el gradiente se propague directamente a las primeras capas, lo que mitiga el problema del gradiente desaparecido y facilita el entrenamiento de redes mucho más profundas (como ResNet50, que tiene 50 capas).

- **Aprendizaje más rápido:** En una red convolucional tradicional, cada capa debe aprender a transformar completamente la entrada de la capa anterior. En ResNet, en cambio, las capas aprenden los residuos, es decir, las correcciones necesarias para mejorar las predicciones, lo que simplifica la tarea de aprendizaje. Las conexiones residuales permiten que la red aprenda de manera incremental, ajustando solo los pequeños cambios necesarios sobre la identidad original.

- **Mayor profundidad sin pérdida de rendimiento:** En redes muy profundas, agregar más capas a menudo conduce a un peor rendimiento (degradación). Esto ocurre porque las capas adicionales no siempre mejoran la capacidad de la red para modelar las relaciones complejas en los datos. Con los bloques residuales, es posible aumentar la cantidad de capas sin que esto degrade el rendimiento, ya que las conexiones de salto preservan la información de las capas anteriores. Esto permite que arquitecturas como ResNet50 sean mucho más profundas que las CNN tradicionales, sin sufrir la degradación del rendimiento.

- **Mejor generalización:** Las redes residuales tienden a generalizar mejor en datos no vistos, ya que las conexiones de salto permiten que las capas más profundas se centren en aprender características más complejas, mientras que las capas más superficiales se encargan de aprender características básicas. Esta separación mejora la capacidad de la red para captar patrones en los datos.

Dentro de la familia de ResNets, la ResNet50 es una de las arquitecturas más conocidas y utilizadas. ResNet50 es una variante de las redes neuronales convolucionales desarrollada por Microsoft como parte de la familia de redes residuales (ResNet).

Bloques residuales

La arquitectura de ResNet50 consta de 50 capas, organizadas en bloques residuales. Cada bloque residual incluye conexiones de salto que permiten la combinación de la salida de una capa con su entrada original, mitigando el problema del gradiente desaparecido y mejorando la eficiencia del entrenamiento. Un bloque residual típico en la ResNet50 tiene la siguiente estructura:

- **Primera capa convolucional:** Se aplica una convolución 1x1, seguida de una normalización por lotes (Batch Normalization) y una activación ReLU.
- **Segunda capa convolucional:** Se aplica una convolución 3x3, seguida de normalización por lotes y activación ReLU.
- **Tercera capa convolucional:** Se aplica una convolución 1x1, seguida de normalización por lotes.
- **Conexión de salto (skip connection):** La entrada original del bloque se suma a la salida de la tercera capa convolucional.
- **Activación final:** Después de la suma, se aplica una activación ReLU.

En la Figura 16 se muestra un ejemplo de un bloque residual convolucional.

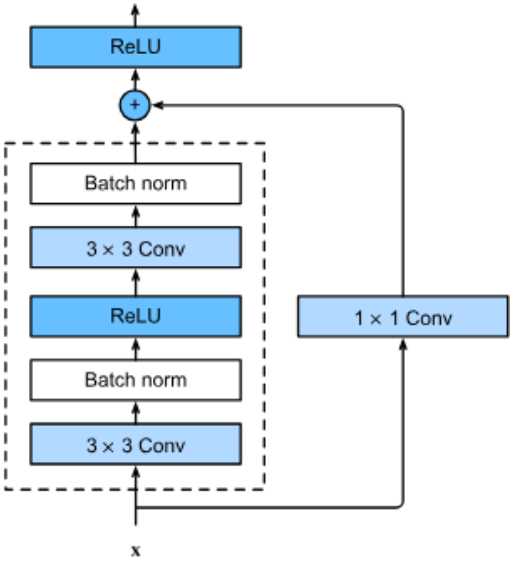


FIGURA 16. Bloque residual convolucional

Estos bloques permiten que la red aprenda a corregir los errores de la salida de la red a medida que profundiza.

Estructura de ResNet50

ResNet50 consta de 50 capas de profundidad y está compuesta principalmente por capas convolucionales, capas de pooling, capas de normalización por lotes (Batch Normalization), y capas de activación ReLU. La red sigue un patrón típico de CNN, pero lo que la hace única es el uso de los bloques residuales.

La arquitectura de ResNet50 se puede dividir en las siguientes secciones:

- **Primera capa convolucional y capa pooling (Conv1 y Max Pooling):** Se aplica una única capa convolucional de tamaño grande para capturar características básicas de la imagen.

  Convolución de 7x7 filtros, 64 canales de salida, con un stride de 2, seguido de una normalización por lotes (Batch Normalization) y activación ReLU. Finalmenmte la información pasa por la capa Max Pooling de 3x3 con stride de 2 para reducir la resolución de la imagen.

  El propósito de esta primera parte es reducir el tamaño de la imagen y extraer las características más básicas.

- **Bloques residuales:** Los bloques residuales son la estructura central de la ResNet50. Estos bloques están agrupados en varias capas, donde cada bloque tiene su propia configuración interna. Los bloques de la red ResNet50 siguen el siguiente patrón en términos de número de filtros y profundidad.

  - **Conjunto de bloques residuales 1 (Conv2_x):** Tres bloques residuales con 64 filtros en las capas convolucionales internas. Cada bloque residual tiene tres capas convolucionales: de 1x1, 3x3 y 1x1. El primer bloque en esta sección utiliza una conexión de proyección (convolución de 1x1 en la conexión de salto) para ajustar las dimensiones de la salida de la capa anterior a la entrada del bloque residual.

- **Conjunto de bloques residuales 1 (Conv2_x):** Cuatro bloques residuales con 128 filtros en las capas convolucionales internas. Los bloques siguen la misma estructura de tres capas: de 1x1, 3x3 y 1x1. Aquí también se utiliza una conexión de proyección en el primer bloque para ajustar las dimensiones de las características.

- **Conjunto de bloques residuales 1 (Conv2_x):** Seis bloques residuales con 256 filtros en las capas convolucionales internas. Los bloques mantienen la estructura de tres capas: de 1x1, 3x3 y 1x1. De nuevo, el primer bloque en esta sección incluye una conexión de proyección para manejar el cambio de dimensiones.

- **Conjunto de bloques residuales 1 (Conv2_x):** Tres bloques residuales con 512 filtros en las capas convolucionales internas. Similar a los bloques anteriores, con capas convolucionales de 1x1, 3x3 y 1x1. El primer bloque en esta sección también tiene una conexión de proyección para ajustar las dimensiones.

- **Capa final de la red:** Después de los bloques residuales, la red concluye con:

  - **Capa de Pooling Global (Global Average Pooling):** Esto reduce el tamaño espacial a 1x1 por canal.

  - **Cape densa (fully conected):** Una capa totalmente conectada de 1000 unidades (si se usa para clasificación en ImageNet), que produce la salida final.

  - **Capa Softmax:** Aplica la función softmax para convertir la salida de la red en probabilidades por clase.

En la Figura 17 se representa la arquitectura de la red ResNet 50.

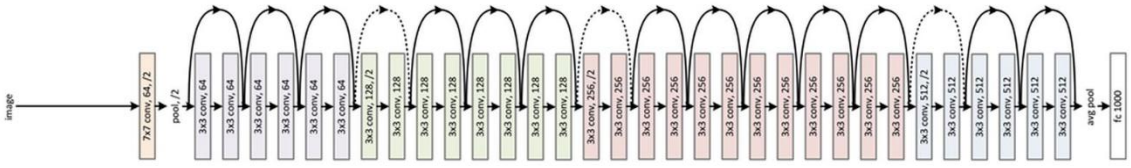


FIGURA 17. Arquitectura de ResNet50.

Aplicaciones de ResNet50

ResNet50 se ha utilizado ampliamente en diversas aplicaciones de visión por computadora, incluyendo:

- **Clasificación de imágenes:** En tareas de clasificación como ImageNet, ResNet50 ha alcanzado precisión de vanguardia.
- **Detección de objetos:** Se utiliza como backbone en modelos de detección como Faster R-CNN y YOLO.
- **Reconocimiento facial:** Muchas arquitecturas de reconocimiento facial utilizan ResNet50 como base.
- **Segmentación de imágenes:** También se utiliza en tareas de segmentación de imágenes, como en el modelo Mask R-CNN.

3 METODOLOGÍA

3.1 ADQUISICIÓN DE IMÁGENES

Para la adquisición de imágenes del proceso de trefilado, se utilizó un teléfono móvil de alta resolución. Las imágenes se capturaron en un entorno controlado para asegurar su consistencia y calidad. Las condiciones de iluminación fueron cuidadosamente ajustadas para minimizar las sombras y resaltar los detalles de la superficie del material trefilado.

- **Equipo utilizado:** Teléfono móvil Xiaomi 11T Pro (Figura 18), cuyas especificaciones técnicas de imagen y video [17] se muestran en la Tabla 1.

FIGURA 18. Dispositivo Xiaomi 11T Pro

Tabla 1. Especificaciones técnicas de imagen y vídeo del dispositivo Xiaomi 11T Pro.

Cámara principal	Triple 108 MP, f/1,8 26 mm (gran angular) + 8 MP, f/2.2, 120º (super gran angular) + 5 MP, f/2.4, 50 mm (telefotográfico macro)
Grabadora de vídeo	8K@30fps, 4K@30/60fps, 1080p@30/60/120/240/960, HDR10+
Flash	Sí

- **Condiciones de captura:** Iluminación artificial con lámparas LED para asegurar una iluminación uniforme.

- **Configuración de la cámara:** En la Tabla 2 se muestra las opciones de configuración del dispositivo para la adquisición de imágenes.

Tabla 2. Configuración de la cámara.

Modo	Automático
Resolución	1500x1500
Zoom	X2
Brillo	60/100
Flash	No

Los parámetros que no aparezcan en la Tabla 2 se han tomado por defecto.

- **Montaje para la adquisición de imágenes:** En la Figura 19 se muestra el montaje realizado para la adquisición de imágenes. Las cotas representadas en la imagen se facilitan en mm.

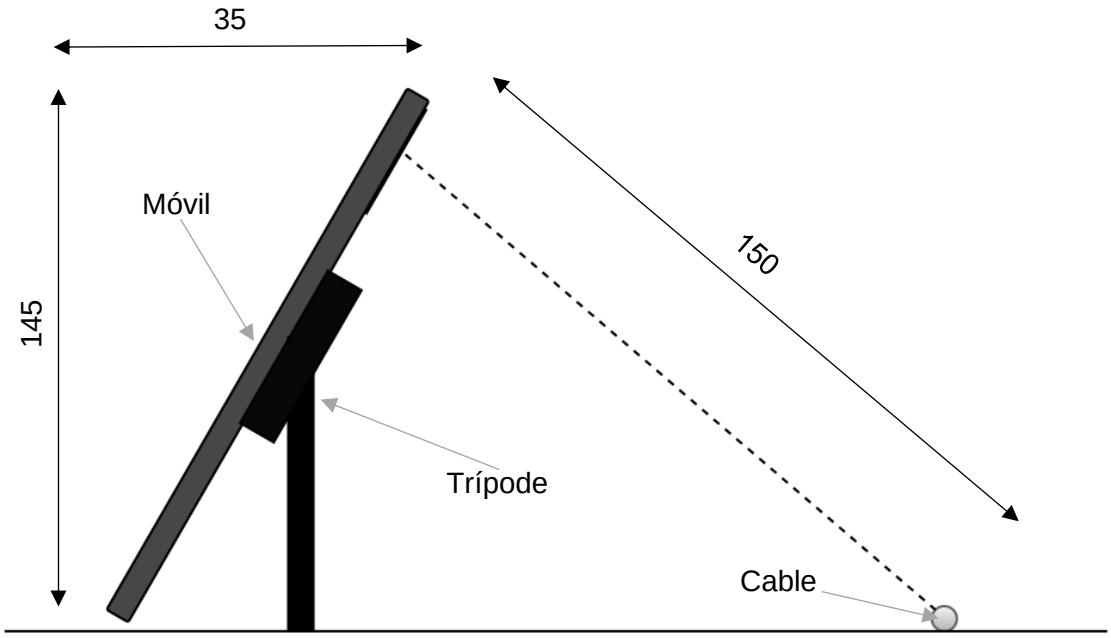


FIGURA 19. Montaje realizado para la adquisición de imágenes.

- **Proceso de adquisición de imágenes:** Las imágenes se tomaron de distintas muestras facilitadas por el laboratorio de la UC. Se tomaron imágenes de diferentes partes de una serie de cables obtenidos del proceso de trefilado para aumentar el conjunto de datos. Se capturaron un total de 284 imágenes, las cuales fueron etiquetadas manualmente como imagen con defectos e imagen sin defectos para su uso en el entrenamiento, evaluación y prueba de la red neuronal. Se obtuvieron 145 imágenes con defectos y 139 imágenes sin defectos. En la Figura 20 y Figura 21 se muestra un esquema de las divisiones que se han llevado a cabo longitudinal y transversalmente en los cables para poder tener un mayor número de datos con las muestras disponibles.

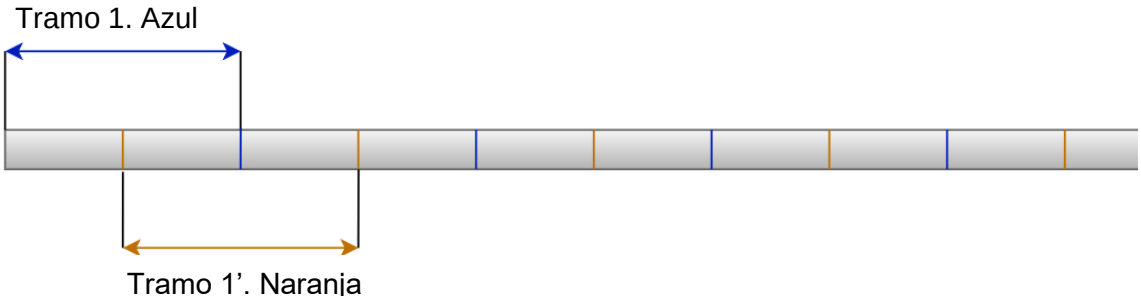


FIGURA 20. División longitudinal del cable.

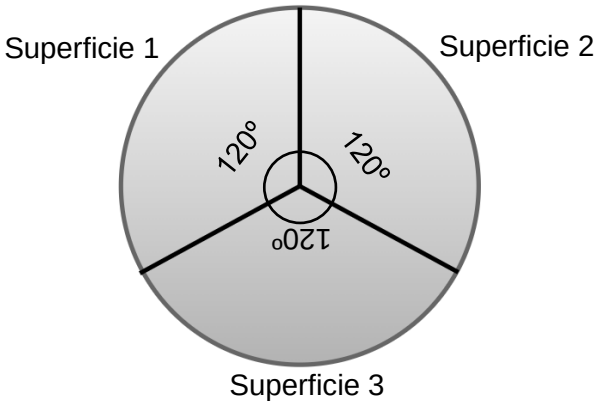


FIGURA 21. División transversal del cable.

Cómo se observa en la Figura 20, se divide el cable en tramos de 25 mm desde su inicio hasta el final (marcas azules, tramo 1). Para aumentar el conjunto de datos se realiza otra división del cable de 25 mm, tomando como primera marca la mitad del primer tramo azul (marcas naranjas, tramo 1').

En la Figura 21 se muestra la división transversal, en la cual se divide el cable en 3 secciones a 120º. Esto permite obtener datos de 3 superficies diferentes de cada cable (superficie 1, superficie 2 y superficie 3), aumentando de esta manera el conjunto de datos.
Con la combinación de ambas divisiones, se consigue obtener un gran número de imágenes partiendo de unas muestras reducidas.

3.2 PREPARACIÓN DE DATOS

Las imágenes capturadas se dividieron en dos y cuatro categorías para la detección de defectos y su clasificación respectivamente.

- **Categorías de detección de defectos:** “Con Defecto” y “Sin Defecto”

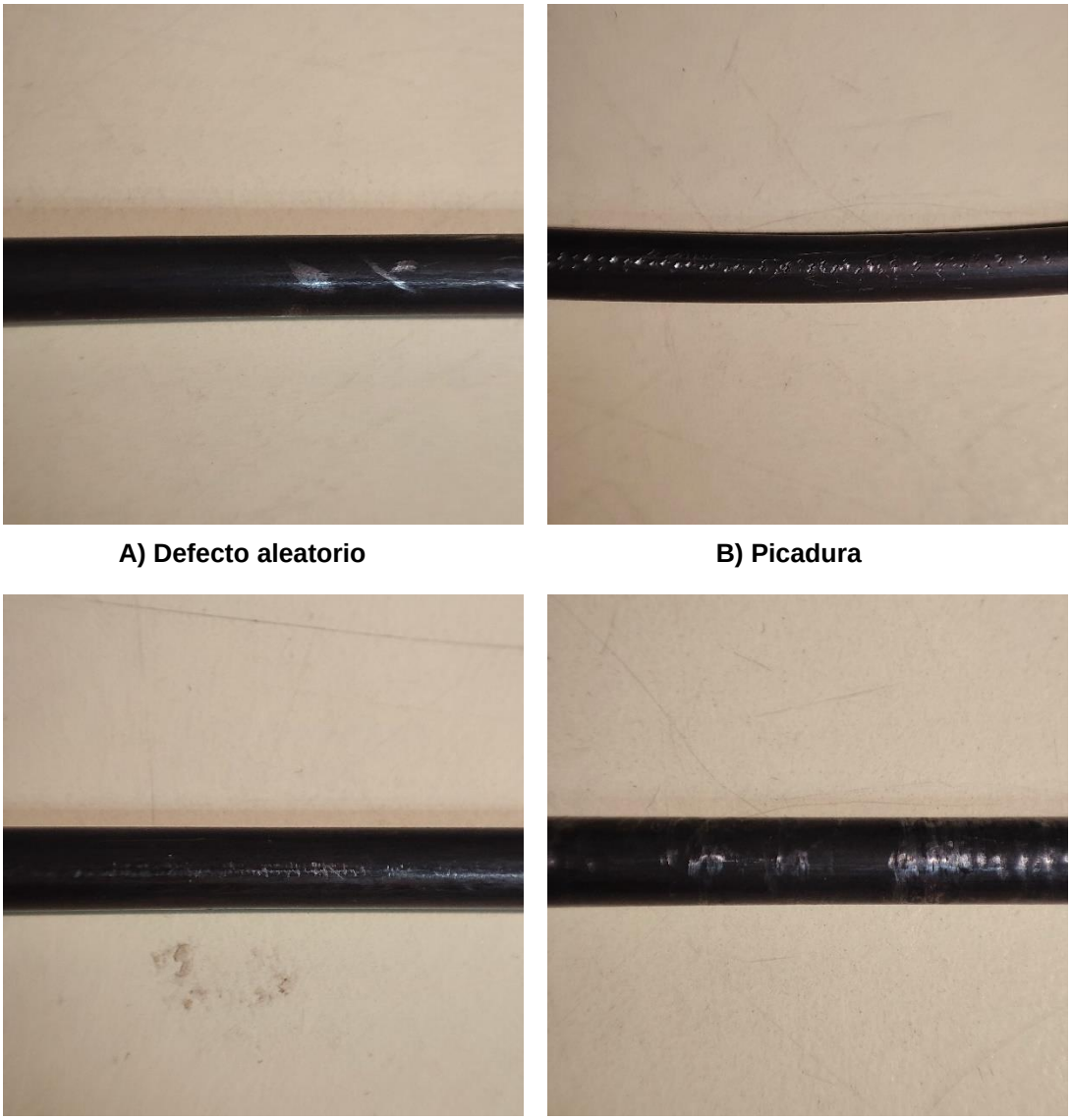
  En la Figura 22 se muestra la comparativa de imagen sin defecto (A) e imagen con defecto (B).

A) Imagen sin defecto **B) Imagen con defecto**

FIGURA 22. Comparativa imagen sin defecto e imagen con defecto

- **Categorías de clasificación de defectos:** “Defecto Aleatorio” (A), “Picadura” (B), “Rayado Longitudinal” (C) y “Rayado Transversal” (D).

  En la Figura 23 se muestra un ejemplo de cada tipo de defecto.



A) Defecto aleatorio **B) Picadura**

C) Rayado longitudinal **D) Rayado transversal**

FIGURA 23. Ejemplo de clasificación de defectos

Este etiquetado fue realizado manualmente para asegurar la precisión. Posteriormente, las imágenes originales (284 imágenes) se sometieron a técnicas para aumentar el conjunto de datos, obteniendo “artificialmente” 852 imágenes. Todas las imágenes obtenidas se preprocesaron para adecuarlas al formato ResNet50.

Finalmente, se dividió el número total de muestras en 3 subconjuntos para poner en funcionamiento las redes y validar su calidad.

3.2.1 Preprocesamiento de datos.

Se llevaron a cabo técnicas de preprocesado para aumentar el rendimiento de la red neuronal.

Enfoque

El enfoque es una técnica de procesamiento de imágenes que mejora los bordes y los detalles finos de la imagen, haciéndolos más nítidos. Se utiliza para resaltar características importantes, como contornos y defectos en las imágenes. Mejora la calidad visual de las imágenes y facilita que la red neuronal detecte bordes y diferencias sutiles, aumentando la precisión del modelo.

- **Proceso de enfoque:** Se ha utilizado el comando 'imsharpen' de Matlab para realizar esta operación. El comando imsharpen realza los bordes de una imagen aplicando un filtro de paso alto, que acentúa las diferencias de intensidad entre los píxeles cercanos. Este proceso es útil cuando se desea mejorar la claridad de los detalles en las imágenes antes de introducirlas en una red neuronal.

- **Técnicas de enfoque:** El filtro de enfoque detecta los bordes calculando la diferencia entre un píxel y sus vecinos. Los cambios bruscos en intensidad, como los bordes de objetos, son amplificados, mientras que las áreas suaves, como fondos uniformes, se dejan prácticamente intactas.

- **Control de parámetros:** Se han modificado los parámetros 'Radius' y 'Amount' para obtener un mejor resultado.

    - **Radius:** Controla el área en torno a cada píxel donde se busca un contraste para detectar bordes.

    - **Amount:** Controla cuánto se realzan los bordes.

Para elegir el valor de los parámetros de enfoque se han realizado 3 pruebas con 3 configuraciones diferentes con el propósito de escoger la mejor opción.

En la Tabla 3 se muestran los valores de los diferentes parámetros utilizados en cada prueba.

Tabla 3. Valor de los parámetros de enfoque en las diferentes pruebas.

	Prueba 1	Prueba 2	Prueba 3
Radius	1	2	3
Amount	2	3	4

En la Figura 24 se muestran los resultados de las pruebas. A), B) y C) se corresponden con la prueba 1, prueba 2 y prueba 3 respectivamente.

A) Prueba 1 **B) Prueba 2** **C) Prueba 3**

FIGURA 24. Resultados de las pruebas de enfoque.

Tras haber realizado las pruebas se ha elegido los valores de los parámetros de la **Prueba 2**. Se considera que son los valores adecuados para que el enfoque sea el necesario sin que se distorsione en exceso la imagen. Los valores elegidos se muestran en la Tabla 4.

Tabla 4. Parámetros de enfoque

Radius	2
Amount	3

- **Impacto en el desempeño de la Red Neuronal:** Aplicar enfoque puede ser útil cuando se busca que la red neuronal identifique bordes y características finas en las imágenes, lo que puede mejorar su capacidad de detección. Sin embargo, un enfoque excesivo puede generar ruido que podría confundir a la red y afectar negativamente su rendimiento.

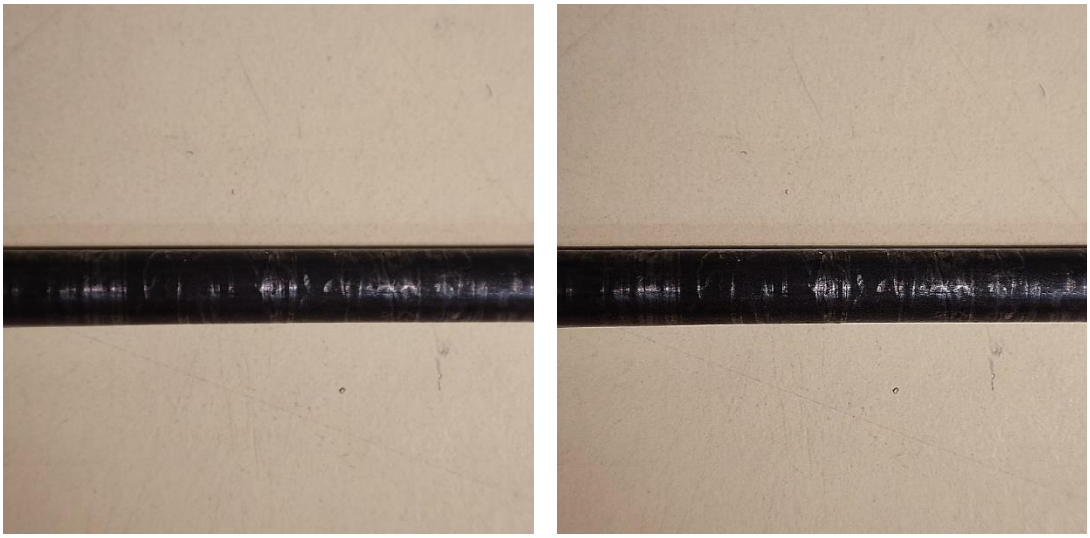
    - **Beneficios**: En tareas de detección de bordes o en imágenes donde los detalles finos son importantes, el enfoque puede mejorar la capacidad de la red para identificar características clave.

    - **Riesgos**: Si la imagen contiene ya mucho ruido, el enfoque puede amplificar este ruido, lo que puede afectar la precisión de la red.

Los parámetros elegidos son óptimos para que la aplicación de la técnica de enfoque tenga beneficios sobre la red, haciéndola más eficiente.
En las Figuras 25 y 26 se pueden ver dos ejemplos de imágenes sin enfoque y con enfoque. La imagen A) se corresponde con la imagen sin enfoque y la imagen B) se corresponde con la imagen con enfoque.

A) Imagen sin enfoque **B) Imagen con enfoque**

FIGURA 25. Ejemplo 1 comparativa imagen sin enfoque e imagen con enfoque.



A) Imagen sin enfoque **B) Imagen con enfoque**

FIGURA 26. Ejemplo 2 comparativa imagen sin enfoque e imagen con enfoque.

Normalizado

El normalizado implica ajustar los valores de los píxeles de la imagen a un rango determinado, normalmente entre 0 y 1 o centrarlos alrededor de la media 0 con una desviación estándar de valor 1. Las redes neuronales funcionan mejor cuando las entradas están en un rango predecible y estandarizado. Normalizar las imágenes ayuda a mejorar la eficiencia del entrenamiento y la convergencia del modelo. La normalización asegura que todos los píxeles contribuyan de manera uniforme, evitando que aquellos con valores altos dominen las activaciones. Esto facilita el entrenamiento y mejora la capacidad del modelo para generalizar.

- **Proceso de normalización:** En una imagen digital, los píxeles suelen tener valores que van de 0 a 255 (en imágenes tipo unit 8) o de 0 a 65535 (en imágenes tipo unit 16) para cada canal. En una imagen en escala de grises solo habría dos canales (blanco y negro) y en una imagen RGB habría tres (rojo, verde y azul). En este trabajo las imágenes son tipo unit 8 y RGB, por lo que los píxeles pueden tener un valor entre 0 y 255 para los canales rojo, verde y azul. La normalización convierte estos valores a un rango más manejable entre 0 y 1, lo que permite que los algoritmos de redes neuronales operen de manera más eficiente.

  Una forma sencilla de normalizar una imagen en MATLAB es dividir todos los valores de los píxeles por el valor máximo posible (255). Este proceso convierte los valores de los píxeles de la imagen de un rango de [0, 255] a un rango de [0, 1].

- **Significado del rango [0,1]:** En una imagen RGB normalizada el 0 representa el valor mínimo de intensidad en ese canal de color (ausencia total de color), mientras que el 1 representa el valor máximo de intensidad en ese canal de color (máxima presencia de color).

  - **Valor 0 en RGB:**

    - Si el valor en el canal **rojo (R)** es 0, significa que no hay rojo presente en ese píxel.
    - Si el valor en el canal **verde (G)** es 0, no hay verde presente en ese píxel.
    - Si el valor en el canal **azul (B)** es 0, no hay azul presente en ese píxel.

  - **Valor 1 en RGB:**

    - Si el valor en el canal **rojo (R)** es 1, significa que el color rojo está presente en su máxima intensidad.
    - Si el valor en el canal **verde (G)** es 1, significa que el color verde está en su máxima intensidad.
    - Si el valor en el canal **azul (B)** es 1, significa que el azul está en su máxima intensidad

  - **Algunos ejemplos prácticos:**

    - Píxel completamente rojo: [1,0,0]. Rojo máximo, sin verde ni azul.
    - Píxel completamente verde: [0,1,0]. Verde máximo, sin rojo ni azul.
    - Píxel completamente azul: [0,0,1]. Ausencia total de color.
    - Píxel completamente negro: [0,0,0]. Rojo máximo, sin verde ni azul.
    - Píxel completamente blanco: [1,1,1]. Rojo, verde y azul a su máximo valor.
    - Píxel gris medio: [0.5,0.5,0.5]. Intensidades medias en todos los canales.

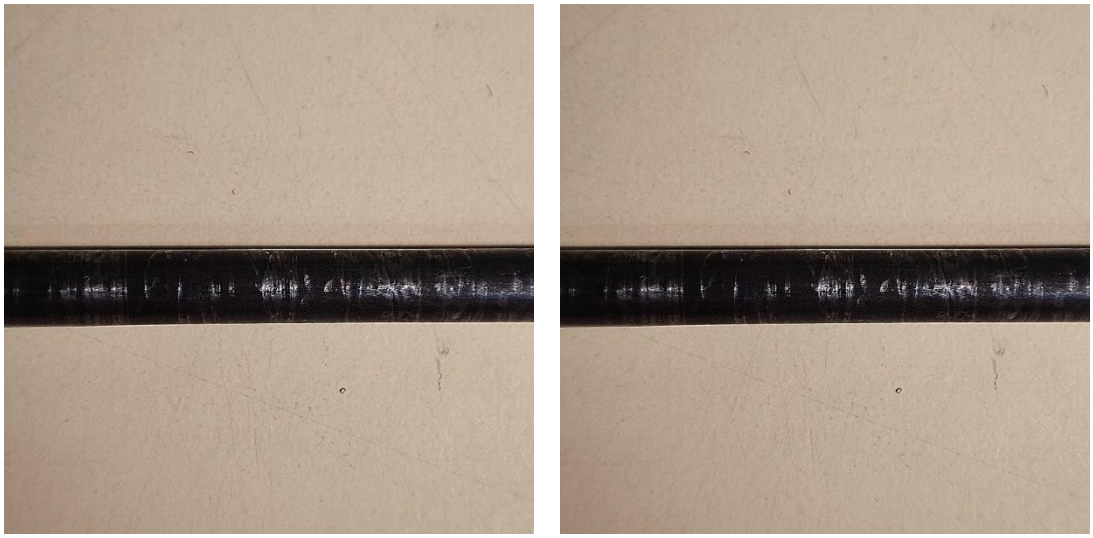
- **Impacto en el desempeño de la Red Neuronal:** La normalización tiene un impacto directo en el rendimiento de la red neuronal:

  - **Eficiencia en el entrenamiento**: Normalizar los valores de los píxeles ayuda a evitar que las activaciones de la red neuronal se saturen, lo que puede ralentizar o impedir el proceso de entrenamiento.

- **Mejora de la convergencia**: Los optimizadores utilizados en redes neuronales funcionan mejor cuando los datos de entrada tienen un rango controlado y homogéneo. Si los valores de los píxeles no están normalizados, algunas neuronas pueden aprender mucho más rápido que otras, lo que afecta la estabilidad y el rendimiento del entrenamiento.

- **Uniformidad entre imágenes**: Normalizar las imágenes asegura que todas tengan una escala similar, lo que permite que la red generalice mejor sobre los datos, sin estar sesgada por diferencias en la intensidad de los píxeles de las distintas imágenes.

A continuación, se muestra cómo afecta el proceso de normalizado a una imagen y como cambia sus valores. En la Figura 27 se muestra la imagen sin normalizar (A) y la imagen normalizada (B). Aparentemente la imagen no cambia, pero el valor de sus píxeles sí. Por ello, a continuación de cada figura, se muestran los valores de los píxeles de las primeras 10 filas y columnas para los tres canales.



A) Imagen sin normalizar **B) Imagen normalizada**

FIGURA 27. Imagen sin normalizar

A continuación, se muestran los valores de los píxeles de las 10 primeras filas y columnas para cada canal Rojo (1), Verde (2) y Azul (3) para la imagen sin normalizar (izquierda) y para la imagen normalizada (derecha)

(:,:,1) =

167	167	167	167	167	167	167	166	170	171
167	167	167	167	167	167	167	167	171	167
167	167	167	167	167	167	167	167	167	168
167	167	167	167	167	167	167	168	168	169
167	167	167	167	167	167	168	168	165	165
167	167	167	167	167	167	168	168	161	162
167	167	167	167	167	167	168	169	161	162
167	167	167	167	167	167	168	169	161	162
171	171	171	171	172	172	172	173	165	166
169	169	169	169	169	169	169	169	166	166

(:,:,2) =

148	148	148	148	148	148	148	147	151	152
148	148	148	148	148	148	148	148	152	148
148	148	148	148	148	148	148	148	148	149
148	148	148	148	148	148	148	149	149	149
148	148	148	148	148	148	149	149	146	146
148	148	148	148	148	148	149	149	142	143
148	148	148	148	148	148	149	150	142	143
148	148	148	148	148	148	149	150	142	143
152	152	152	152	152	153	153	154	146	147
150	150	150	150	150	150	150	150	147	147

(:,:,3) =

131	131	131	131	131	131	131	130	134	135
131	131	131	131	131	131	131	131	135	131
131	131	131	131	131	131	131	131	131	132
131	131	131	131	131	131	131	132	132	132
131	131	131	131	131	131	132	132	129	129
131	131	131	131	131	131	132	132	125	126
131	131	131	131	131	131	132	132	125	126
131	131	131	131	131	131	132	133	126	126
135	135	135	135	135	136	136	137	129	130
132	132	132	132	133	133	133	133	130	130

(:,:,1) =

0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6510	0.6667	0.6706
0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6706	0.6549
0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6588
0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6588	0.6588	0.6627
0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6588	0.6588	0.6471	0.6471
0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6588	0.6588	0.6314	0.6353
0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6588	0.6627	0.6314	0.6353
0.6549	0.6549	0.6549	0.6549	0.6549	0.6549	0.6588	0.6627	0.6314	0.6353
0.6706	0.6706	0.6706	0.6706	0.6745	0.6745	0.6745	0.6784	0.6471	0.6510
0.6627	0.6627	0.6627	0.6627	0.6627	0.6627	0.6627	0.6627	0.6510	0.6510

(:,:,2) =

0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5765	0.5922	0.5961
0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5961	0.5804
0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5843
0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5843	0.5843	0.5843
0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5843	0.5843	0.5725	0.5725
0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5843	0.5843	0.5569	0.5608
0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5843	0.5882	0.5569	0.5608
0.5804	0.5804	0.5804	0.5804	0.5804	0.5804	0.5843	0.5882	0.5569	0.5608
0.5961	0.5961	0.5961	0.5961	0.5961	0.6000	0.6000	0.6039	0.5725	0.5765
0.5882	0.5882	0.5882	0.5882	0.5882	0.5882	0.5882	0.5882	0.5765	0.5765

(:,:,3) =

0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5098	0.5255	0.5294
0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5294	0.5137
0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5176
0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5176	0.5176	0.5176
0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5176	0.5176	0.5059	0.5059
0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5176	0.5176	0.4902	0.4941
0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5176	0.5176	0.4902	0.4941
0.5137	0.5137	0.5137	0.5137	0.5137	0.5137	0.5176	0.5216	0.4941	0.4941
0.5294	0.5294	0.5294	0.5294	0.5294	0.5333	0.5333	0.5373	0.5059	0.5098
0.5176	0.5176	0.5176	0.5176	0.5216	0.5216	0.5216	0.5216	0.5098	0.5098

Redimensionado

El redimensionado implica ajustar las dimensiones de la imagen a un tamaño específico. En el contexto del uso de redes neuronales convolucionales (CNNs), como ResNet50 o VGG16, el redimensionado a 224x224 píxeles es un paso crítico, ya que estas redes están preentrenadas para procesar imágenes de esas dimensiones exactas. Sin este redimensionado, las dimensiones de la imagen no coincidirían con la estructura de la red, y el modelo no podría procesar los datos de entrada de manera adecuada.

El redimensionado asegura que la imagen tenga el tamaño correcto para alimentar las capas convolucionales y las capas de pooling de la red, manteniendo una estructura compatible con el entrenamiento original del modelo. Si las imágenes no se redimensionan, el modelo puede rechazar la entrada o funcionar incorrectamente.

Para realizar la técnica de redimensionado, se ha utilizado el comando ‘imresize’ de Matlab. El comando imresize (image, [224 224]) en MATLAB se utiliza para cambiar el tamaño de una imagen a dimensiones específicas. En este caso, la imagen original de 1500x1500 píxeles se redimensiona a un tamaño de 224x224 píxeles para que las redes neuronales funcionen de forma correcta. A continuación, se desglosa el proceso de redimensionado en MATLAB:

- **Interpolación:** Al cambiar el tamaño de una imagen, MATLAB utiliza técnicas de interpolación para determinar los valores de los píxeles en la imagen redimensionada. Dado que las imágenes están formadas por una cuadrícula de píxeles, al reducir o aumentar el número de píxeles, es necesario estimar los nuevos valores a partir de los existentes.

MATLAB ofrece varios métodos de interpolación para ajustar el tamaño de la imagen:

- **Interpolación bilineal:** Este método calcula el valor de cada nuevo píxel como un promedio ponderado de los cuatro píxeles más cercanos en la imagen original, teniendo en cuenta tanto las distancias horizontales como verticales.

- **Interpolación por vecino más cercano:** Este método es más simple y rápido, ya que asigna a cada nuevo píxel el valor del píxel más cercano en la imagen original. Sin embargo, puede producir imágenes redimensionadas de menor calidad, con bordes dentados o "pixelación".

- **Interpolación bicúbica:** Este método utiliza un rango mayor de píxeles vecinos (normalmente 16 píxeles) y calcula el valor del nuevo píxel con una fórmula cúbica. Produce una imagen más suave y de mayor calidad, aunque es más costoso computacionalmente.

Para elegir el modo de interpolación, se han redimensionado todas las imágenes originales (284 imágenes) con cada uno de los métodos descritos anteriormente. De esta manera se puede calcular el tiempo que tarda cada método en redimensionar las imágenes. Además, se ha comparado en una imagen los tres métodos con el propósito de comparar la calidad de las imágenes obtenidas.

En la Tabla 5 se muestran los resultados del tiempo tardado en redimensionar las imágenes del conjunto de muestra con cada interpolación.

Tabla 5. Tiempo de redimensionado según tipo de interpolación.

	Tiempo (s)
Bilineal	10
Vecino más cercano	8
Bicúbica	10

En la Figura 28 se muestra la comparativa del resultado de las diferentes interpolaciones en una imagen.

A) Bilineal **B) Vecino más cercano** **C) Bicúbica**

FIGURA 28. Comparativa de las diferentes interpolaciones

En este trabajo, ya que el tiempo utilizado para el procesado es similar en todos los métodos, la elección se basa en la calidad del redimensionado. Después de observar los resultados que aparecen en la Figura 26, se utilizará la interpolación **bicúbica** para mantener la calidad de la imagen en su mayoría.

- **Mantenimiento de la Relación de Aspecto:** Cuando redimensionas una imagen a dimensiones específicas, es posible que las proporciones (relación de aspecto) de la imagen original no coincidan con las nuevas dimensiones. En este caso, estamos forzando a la imagen a ser exactamente de 224x224 píxeles, independientemente de su tamaño o relación de aspecto original. Si la imagen original no es cuadrada, este redimensionado forzado podría distorsionar la imagen, estirando o comprimiendo los objetos en ella. Por ello, las imágenes adquiridas son cuadradas, de tamaño 1500x1500.

- **Antialiasing (Suavizado de Bordes)** Al reducir el tamaño de una imagen, puede aparecer un fenómeno llamado aliasing, donde se generan artefactos visuales, como bordes dentados o detalles de la imagen que se ven distorsionados. MATLAB aplica automáticamente un proceso de antialiasing para suavizar estos artefactos y mejorar la calidad visual de la imagen redimensionada. El suavizado de bordes se realiza aplicando un filtro de paso bajo (un filtro suavizante) antes de la interpolación para evitar que los píxeles cercanos a los bordes creen artefactos indeseados.

- **Impacto en el Desempeño de la Red Neuronal**

  - **Calidad de la imagen:** Un redimensionado adecuado mantiene la calidad visual de la imagen, lo que es crucial para que la red neuronal identifique correctamente los patrones visuales. Si se aplica una interpolación de baja calidad o no se utiliza antialiasing, la red puede recibir imágenes con artefactos, lo que podría reducir la precisión de las predicciones.

  - **Información perdida:** Al reducir el tamaño de una imagen, se pierde parte de la información, especialmente si la imagen original es mucho más grande que 224x224. Por esta razón, es importante que la imagen mantenga sus características clave tras el redimensionado para que la red neuronal pueda realizar un buen reconocimiento.

  - **Escalabilidad:** El uso de imágenes de tamaño estándar (como 224x224) también facilita el entrenamiento y la evaluación de la red, ya que permite que todas las imágenes entren de forma homogénea en el modelo, reduciendo la variabilidad introducida por imágenes de diferentes tamaños.

El redimensionado de imágenes se ha realizado de manera eficiente, manteniendo la calidad de la imagen y no perdiendo información crítica.

En la Figura 29 se muestra un esquema de cómo afecta el redimensionado a una imagen.

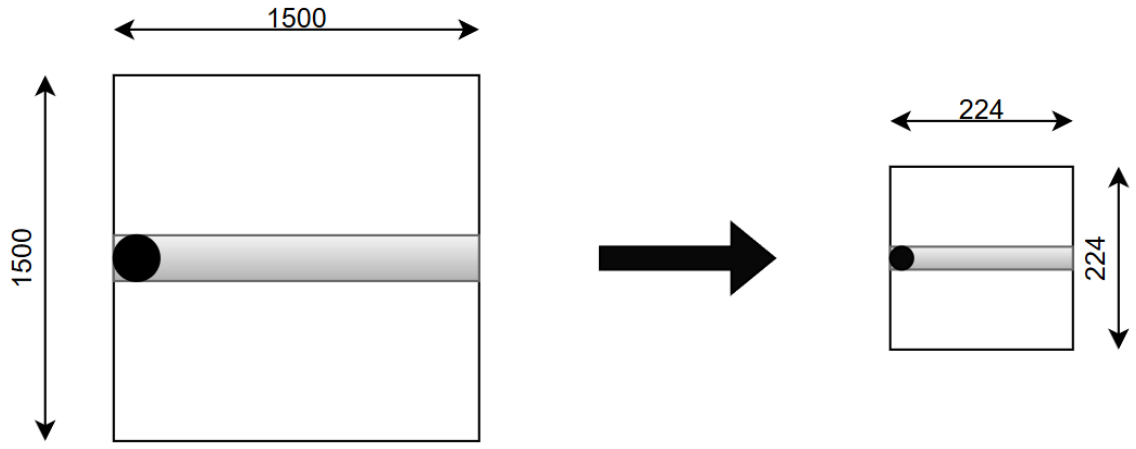


FIGURA 29. Esquema de redimensionado de una imagen.

En las Figura 30 se muestra el redimensionado de la Figura 27.

FIGURA 30. Imagen redimensionada.

3.2.2 Aumento del conjunto de datos.

Para obtener mejores resultados se aumentó el conjunto de datos, ya que se disponía de un número reducido de muestras. Se utilizaron tres técnicas diferentes:

Espejo horizontal

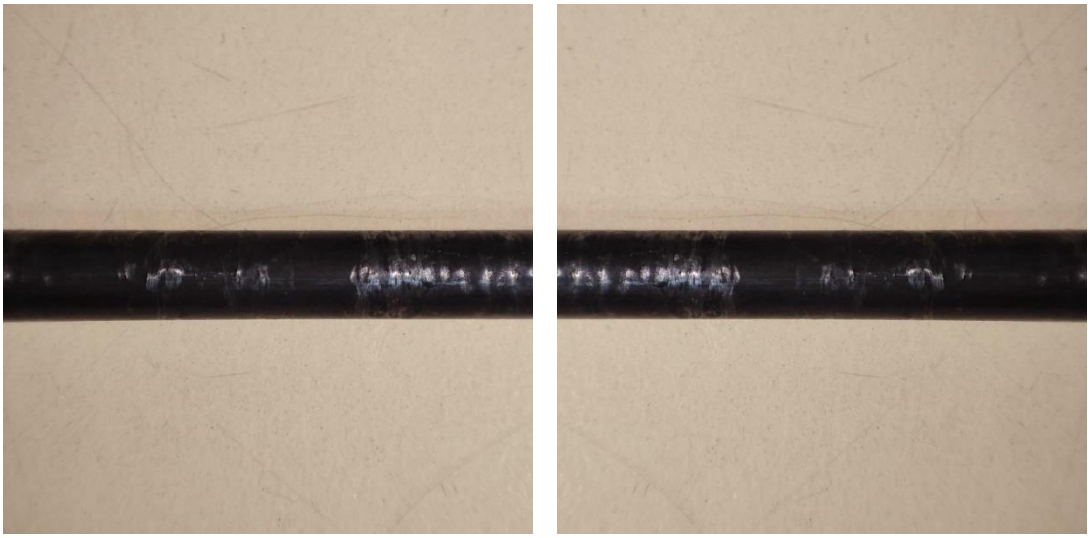
El espejo horizontal consiste en crear una imagen reflejada a lo largo del eje vertical, de modo que la imagen generada es una versión especular de la original. Este proceso es muy útil para crear nuevas versiones de las imágenes existentes y aumentar el tamaño del conjunto de datos sin tener que capturar nuevas imágenes.

- **Impacto en el rendimiento de la red neuronal**

    - **Beneficios**: El espejo horizontal es especialmente útil en tareas donde los objetos de la imagen pueden aparecer con diferentes orientaciones, pero la clase de la imagen no cambia (por ejemplo, imágenes de objetos simétricos como caras, vehículos, etc.). Esto ayuda a la red a generalizar mejor, permitiéndole aprender a reconocer patrones sin importar su orientación.
    - **Riesgos**: En algunos casos, no es apropiado usar espejo horizontal si la orientación tiene importancia en la clase de la imagen. Por ejemplo, en imágenes con texto, un espejo horizontal haría que el texto apareciera al revés, lo que podría confundir a la red.

En este caso, al presentarse el cable siempre de forma longitudinal en el centro de la imagen, es una técnica muy útil para aumentar el conjunto de datos.

En la Figura 31 se muestra un ejemplo de la técnica de espejo horizontal.



A) Imagen original **B) Imagen con espejo horizontal**

FIGURA 31. Representación de la técnica de espejo horizontal.

Zoom

El zoom implica escalar una porción de la imagen para hacerla más grande o más pequeña y luego recortarla o interpolarla para mantener el tamaño original. Con esta técnica se simula la existencia de cables con diferentes diámetros.
Para conseguirlo, se realiza un recorte centrado de la imagen original y se redimensiona la imagen recortada al tamaño de la imagen original.

En la Figura 31 se muestra un esquema del proceso seguido para conseguir una imagen con zoom a partir de una original.

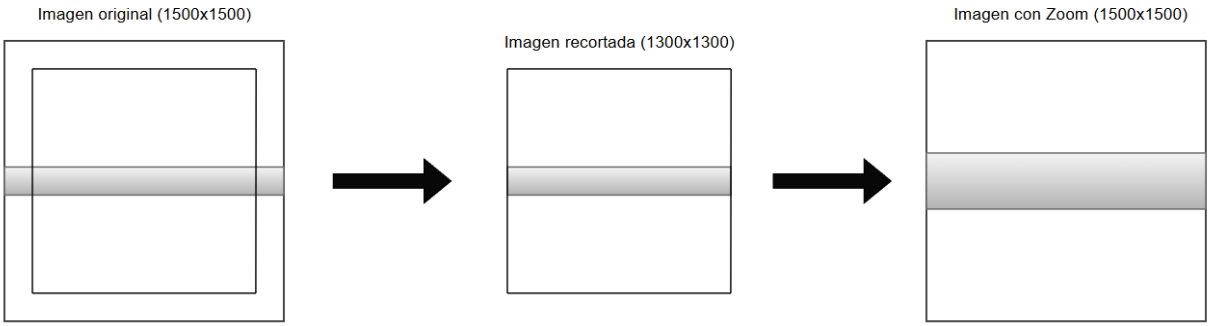


FIGURA 32. Esquema proceso de técnica de Zoom.

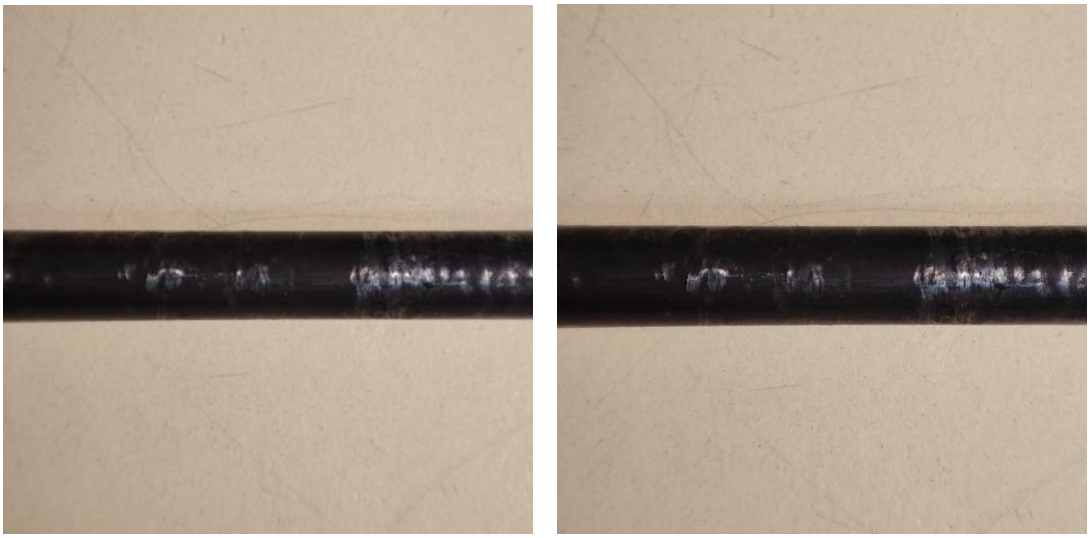
- **Impacto en el rendimiento de la red neuronal**

  - **Beneficios:** El zoom simula escenarios donde el objeto en la imagen aparece más cerca o más lejos de la cámara, lo que puede ayudar a la red neuronal a aprender a identificar objetos a diferentes distancias. Esto es útil cuando los objetos pueden aparecer a diferentes tamaños en las imágenes de prueba.

  - **Riesgos:** Si se realiza un zoom excesivo, se puede perder información clave en los bordes de la imagen o distorsionar la apariencia del objeto. Un zoom fuera de los límites razonables puede hacer que la red aprenda características irrelevantes.

Se ha tomado un aumento que permite aumentar el conjunto de datos sin riesgo de empeorar el desempeño de las redes neuronales.

En la Figura 33 se muestra un ejemplo de la técnica de zoom.



A) Imagen original **B) Imagen con zoom**

FIGURA 33. Representación de la técnica de Zoom.

Ruido Gaussiano

El ruido gaussiano introduce pequeñas variaciones aleatorias en los valores de los píxeles de la imagen, siguiendo una distribución normal o gaussiana. Este ruido simula imperfecciones en la captura de imágenes, como las que podrían surgir por condiciones de baja iluminación o interferencias en el sensor.

- **Impacto en el rendimiento de la red neuronal**

  - **Beneficios:** El ruido gaussiano ayuda a entrenar la red neuronal para ser más robusta frente a pequeñas perturbaciones y errores en las imágenes reales, como el ruido de la cámara o la compresión de las imágenes. Esto mejora la capacidad de la red para generalizar y evitar el sobreajuste a las imágenes perfectas de entrenamiento.

  - **Riesgos:** Si se añade demasiado ruido, se puede distorsionar demasiado la imagen, haciendo que los detalles importantes se pierdan o que la imagen se vuelva irreconocible para la red.

Estas técnicas introducen variaciones en las imágenes sin cambiar el contenido relevante, permitiendo que el modelo aprenda de diferentes configuraciones y sea más resistente a pequeños cambios en las imágenes de entrada.

Para elegir el valor de la varianza de ruido gaussiano adecuado, se han realizado pruebas con diferentes valores de este parámetro. El objetivo es obtener un valor de varianza que modifique la imagen de modo que se aumente el conjunto de datos sin introducir un ruido excesivo que distorsione demasiado la imagen. En la Tabla 6 se muestran los valores de varianza utilizados en las pruebas

Tabla 6. Valores de la varianza de ruido gaussiano en las diferentes pruebas.

	Valor de varianza
Prueba 1	0.005
Prueba 2	0.002
Prueba 3	0.0008

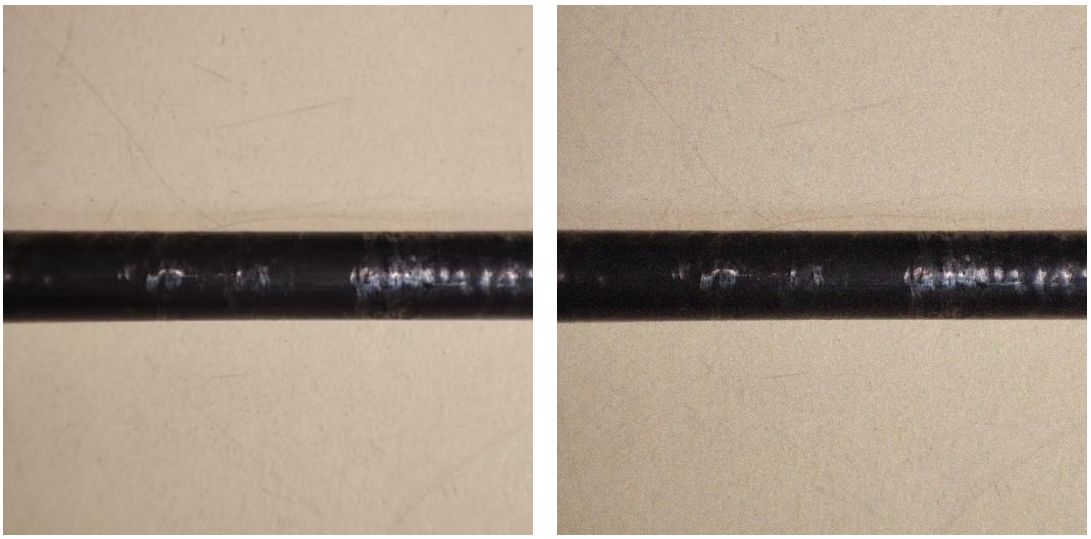
En la Figura 32 se muestran las imágenes obtenidas en cada prueba.

A) Prueba 1 **B) Prueba 2** **C) Prueba 3**

FIGURA 34. Resultados de las pruebas de introducción de ruido.

Analizando las imágenes obtenidas en las pruebas, se ha añadido un ruido gaussiano con una **varianza de 0.002**, valor que hace que se aumente el conjunto de datos y no introduce un ruido excesivo que pueda interferir en el buen funcionamiento de la red neuronal.

En la Figura 35 se muestra un ejemplo de la técnica de ruido gaussiano.



A) Imagen original **B) Imagen con ruido gaussiano**

FIGURA 35. Representación de la técnica Ruido Gaussiano

Una vez obtenidas estas imágenes se procesan con las técnicas de enfoque, normalizado y redimensionado para ser utilizadas por la red neuronal.

En la Tabla 7 se muestra el número de muestras obtenidas una vez aplicadas las técnicas de procesado y aumento del conjunto de datos.

Tabla 7. Desglose del conjunto de datos final

	Con Defecto	Sin Defecto
Muestras iniciales	145	139
Técnica Zoom	145	139
Técnica Espejo Horizontal	145	139
Técnia Ruido Gaussiano	145	139
TOTAL	**580**	**556**

En la Tabla 8 se muestra el desglose del número de imágenes obtenidas para cada tipo de defecto.

Tabla 8. Desglose de la clasificación de imágenes con defecto

	Muestras iniciales	Muestras finales
Defecto aleatorio	63	252
Picadura	21	84
Rayado longitudinal	47	188
Rayado transversal	14	56
TOTAL	**145**	**580**

3.2.3 Distribución del conjunto de datos.

- **Entrenamiento:** El conjunto de entrenamiento es la porción de los datos que se utiliza para entrenar la red neuronal. Contiene la mayor parte de las imágenes, y durante este proceso, la red ajusta sus pesos internos para aprender patrones y características clave de los datos. Sirve para enseñar a la red neuronal a reconocer patrones, en este caso, a identificar imágenes con y sin defectos en el proceso de trefilado. El objetivo es que la red aprenda a asociar las entradas (imágenes) con las salidas correctas (etiquetas de defectos o no defectos).

- **Validación:** El conjunto de validación es un subconjunto de datos que se utiliza para afinar el modelo mientras está en fase de entrenamiento. Estos datos no son utilizados directamente en el ajuste de pesos, sino que permiten evaluar el rendimiento del modelo en datos que no ha visto durante el entrenamiento. Sirve para ajustar hiperparámetros (como la tasa de aprendizaje o la arquitectura del modelo) y prevenir el sobreajuste. El modelo se evalúa periódicamente en este conjunto para determinar si está generalizando bien o si está sobreajustando al conjunto de entrenamiento.

- **Prueba:** El conjunto de prueba es un subconjunto de datos completamente independiente que se utiliza al final del proceso de entrenamiento para evaluar el rendimiento real del modelo en datos que no ha visto ni en el entrenamiento ni en la validación. Sirve para medir la capacidad real de generalización del modelo, es decir, su capacidad para hacer predicciones correctas en datos nuevos. Los resultados sobre el conjunto de prueba representan el desempeño final del modelo y su habilidad para ser aplicado a datos futuros.

Detección de defectos

El conjunto de datos final de 1136 imágenes se dividió en tres subconjuntos:

En la Tabla 9 se muestra la distribución del conjunto de datos para la detección de defectos, mostrando el número de imágenes por subconjunto, así como el porcentaje que representa cada uno del total de las imágenes.

Tabla 9. Distribución del conjunto de datos para la detección de defectos.

SUBCONJUNTO	Nº IMAGENES	PORCENTAJE
Entrenamiento	762	67%
Validación	254	22%
Prueba	120	11%

Clasificación de defectos

Para la clasificación de defectos se trabaja con el conjunto de datos con defectos. Los datos con defecto hacen un total de 580 imágenes.

En la Tabla 10 se muestra la distribución del conjunto de datos para la clasificación de defectos.

Tabla 10. Distribución del conjunto de datos para la clasificación de defectos.

SUBCONJUNTO	Nº IMAGENES	PORCENTAJE
Entrenamiento	415	71%
Validación	105	18%
Prueba	60	11%

3.3 CONFIGURACIÓN DE ResNet50 EN MATLAB

Para implementar y entrenar la red ResNet50 [14], se utilizó MATLAB debido a sus capacidades avanzadas de procesamiento de imágenes y su soporte para redes neuronales profundas. Se diferencian dos configuraciones diferentes de la red ResNet50, una para la detección de defectos (2 clases) y otra para la clasificación de los mismos (4 clases). De manera que, una vez detectado el defecto con la primera configuración de la red, se haga pasar la imagen por la segunda configuración para identificar qué tipo de defecto presenta el cable (Figura 36).

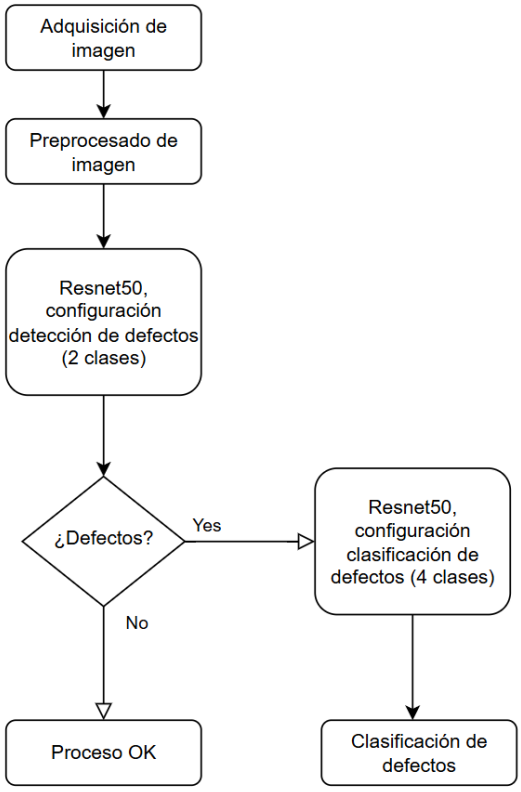


FIGURA 36. Diagrama de flujo simplificado del monitorizado del proceso de trefliado.

- **Carga de ResNet50:** Se utilizó la función 'resnet50' de MATLAB para cargar el modelo preentrenado.

- **Modificación de la red:** Se modifican capas finales de la red para adaptarlo a los problemas de clasificación de 2 y 4 clases. Se utilizan dos tipos de red ResNet50 de manera secuencial. La primera red detecta la existencia de defectos y la segunda red los clasifica.

    - **Clasificación 2 clases (detección de defectos):** La última capa de clasificación se modificó para adaptarse al problema de clasificación binaria (sin defectos/con defectos). Esto se logró reemplazando la capa fully connected y la capa softmax con nuevas capas adecuadas para dos clases.

    - **Clasificación 4 clases (clasificación de defectos):** Se reemplazan las últimas capas de la red (fc1000 y ClassificationLayer_fc1000) para ajustarse a la clasificación de 4 clases. La capa fullyConnected se cambia para que tenga 4 neuronas (una por cada clase) y la capa de clasificación se ajusta para la clasificación multiclase.

Se ajustaron los parámetros de entrenamiento de las redes para optimizar su rendimiento y garantizar un entrenamiento eficiente de la misma probando diferentes combinaciones de parámetros y analizando los resultados obtenidos, poniendo el foco en la precisión, el tiempo de entrenamiento y el grado sobreajuste.

3.3.1 Parámetros de entrenamiento

Tasa de aprendizaje (Learning Rate)

La tasa de aprendizaje controla el tamaño de los pasos que el optimizador da durante la actualización de los pesos.

- **Influencia en la red:**
  - **Tasa de aprendizaje alta:** Los pasos en la actualización de los pesos serán grandes.
    - **Ventajas:** Puede acelerar el entrenamiento y reducir el tiempo necesario para alcanzar una solución.
    - **Desventajas:** Si es demasiado alta, puede provocar que el modelo "salte" por encima del mínimo óptimo en la función de pérdida, causando que el modelo no converja y tenga una precisión inconsistente.
  - **Tasa de aprendizaje baja:** Los pasos en la actualización de los pesos serán pequeños.
    - **Ventajas**: Los pesos se ajustan lentamente, lo que puede ayudar a encontrar una solución más precisa y estable.
    - **Desventajas:** Un valor muy bajo puede hacer que el entrenamiento sea muy lento y que el modelo tarde mucho en converger o quede atrapado en mínimos locales.

Tamaño del Lote (Batch Size)

El tamaño del lote es el número de ejemplos que el modelo procesa antes de actualizar los pesos.

- **Influencia en la red:**

  - **Tamaño de lote pequeño (mini-batch):** Los gradientes se calculan a partir de menos ejemplos.

    - **Ventajas:** Puede introducir "ruido" en el entrenamiento que ayuda a escapar de mínimos locales, y requiere menos memoria.

    - **Desventajas:** Los gradientes pueden ser menos precisos y hacer que el modelo sea menos estable.

  - **Tamaño de lote grande:** Los gradientes se calculan usando más ejemplos.

    - **Ventajas:** Los gradientes son más estables y precisos, lo que puede mejorar la convergencia.

    - **Desventajas:** Requiere más memoria y, si es muy grande, puede resultar en un modelo que se ajuste demasiado a los datos de entrenamiento (sobreajuste).

Número de épocas

El número de épocas es el número de veces que todo el conjunto de datos de entrenamiento se pasa por la red. Se ajusta de manera que el modelo converja y no sobreajuste.

- **Influencia en la red:**

  - **Pocas épocas:**

    - **Ventajas:** Reduce el tiempo de entrenamiento.

    - **Desventajas:** Puede dar lugar a un modelo subentrenado, que no aprende adecuadamente los patrones de los datos y tiene un bajo rendimiento en el conjunto de prueba.

- **Muchas épocas:**

  - **Ventajas:** Da al modelo más oportunidades de aprender y ajustar sus pesos.

  - **Desventajas:** Puede llevar al sobreajuste, donde el modelo se ajusta demasiado a los datos de entrenamiento y pierde la capacidad de generalizar bien en datos nuevos.

Optimizador

El optimizador es el algoritmo que se encarga de ajustar los pesos de la red en función de los gradientes calculados por la función de pérdida. El optimizador influye directamente en cómo se ajustan los pesos y cómo avanza la red hacia un mejor rendimiento.

- **Tipos comunes de optimizadores y su influencia:**

  - **SGD (Stochastic Gradient Descent):** Ajusta los pesos según el gradiente de un lote de ejemplos.

    - **Ventajas:** Simple y efectivo.

    - **Desventajas:** Puede ser lento para converger y tiene dificultad con mínimos locales.

  - **Momentum:** Una mejora sobre SGD, que utiliza el gradiente acumulado para suavizar el entrenamiento.

    - **Ventajas:** Ayuda a evitar quedarse atrapado en mínimos locales y puede acelerar el aprendizaje.

  - **Adam (Adaptive Moment Estimation):** Combina las ventajas de RMSprop y Momentum, adaptando las tasas de aprendizaje para cada parámetro.

    - **Ventajas:** Uno de los más populares. Adapta dinámicamente la tasa de aprendizaje para cada peso, lo que generalmente lleva a una convergencia más rápida y estable.

- **Desventajas:** Puede no ser el más adecuado para todos los problemas, y en algunos casos puede requerir ajuste manual de parámetros como la tasa de aprendizaje.

3.3.2 Ajuste de los parámetros de entrenamiento.

Se probaron 4 configuraciones diferentes para ajustar los parámetros de entrenamiento en la red de detección de defectos (2 clases). Con los resultados obtenidos se eligió la configuración óptima atendiendo al tiempo de entrenamiento, el grado de sobreajuste, el grado de sobre entrenamiento, la precisión y la pérdida.

En la Tabla 11 se muestran las diferentes configuraciones de parámetros utilizadas en el ajuste.

Tabla 11. Configuraciones para el ajuste de parámetros.

	Configuración 1	Configuración 2	Configuración 3	Configuración 4
Tasa de Aprendizaje	0.001	0.0005	0.01	0.0001
Tamaño del Lote	32	64	32	16
Número de Épocas	10	8	6	15
Optimizador	ADAM	SGDM	ADAM	SGDM

Justificación de las configuraciones elegidas.

- **Configuración 1:** Se toma esta configuración como punto de partida, con un tamaño de lote moderado y una tasa de aprendizaje conservadora. Se usa como referencia base para ver si la red converge de manera estable.

- **Configuración 2:** Un tamaño de lote mayor puede estabilizar las actualizaciones de los pesos, pero reduce el número de iteraciones por época. La tasa de aprendizaje ligeramente menor y menos épocas deberían evitar un posible sobre entrenamiento en un tiempo razonable.

- **Configuración 3:** Se opta por una tasa de aprendizaje más alta y menos épocas, lo que puede ayudar a converger rápidamente en conjuntos de datos pequeños o moderados. Esto se traduce en un entrenamiento más rápido.

- **Configuración 4:** Una tasa de aprendizaje más baja y un número mayor de épocas permiten que el modelo aprenda más gradualmente. El lote pequeño puede ajustar más rápido los pesos, aunque tomará más tiempo en completar una época.

Resultados

A continuación, se muestran los resultados de las diferentes configuraciones utilizadas en la red de detección de defectos (clasificación den 2 clases). Se representan las gráficas de precisión y pérdida de entrenamiento y validación para evaluar las diferentes configuraciones y elegir la configuración óptima, atendiendo también al tiempo de entrenamiento empleado en cada caso.

- **Configuración 1**

  En las Figuras 36 y 37 se muestran las gráficas de precisión y pérdida respectivamente de la configuración 1.

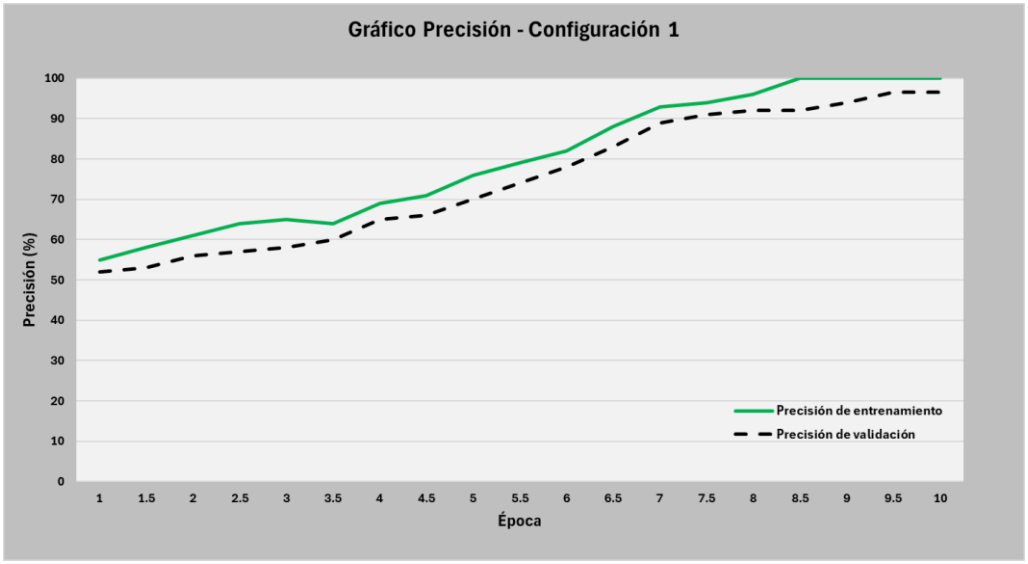


FIGURA 37. Gráfico de precisión de la Configuración 1

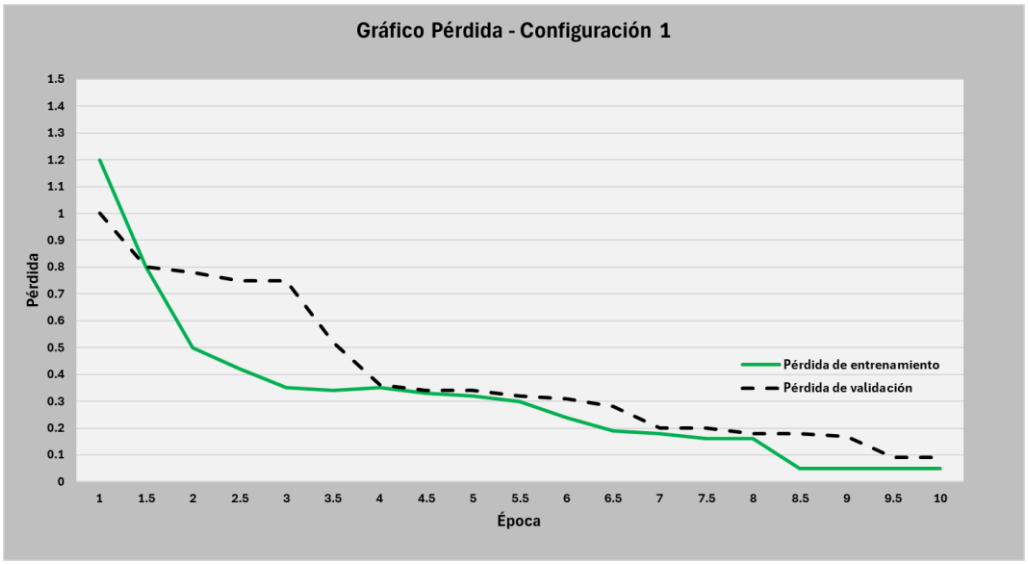


FIGURA 38. Gráfico de pérdida de la Configuración 1

En la Tabla 12 se muestran los resultados más representativos de la configuración 1.

Tabla 12. Resultados Configuración 1.

	Config 1
Precisión entrenamiento	100%
Precisión validación	96.5%
Pérdida entrenamiento	0.05
Pérdida de validación	0.09
Iteraciones	230
Tiempo	27 mins

- **Análisis:** En esta configuración, se ve un crecimiento gradual en la precisión de entrenamiento y en la precisión de validación. Sin embargo, hay una ligera brecha entre las curvas de entrenamiento y validación, lo que indica que la red está mejorando en el conjunto de entrenamiento más rápidamente que en el de validación. La pérdida disminuye en ambos conjuntos, pero la diferencia en la pérdida sugiere que puede haber un inicio de sobreajuste.

- **Conclusión:** El entrenamiento es estable, pero el rendimiento de validación podría ser mejor.

- **Configuración 2**

  En las Figuras 38 y 39 se muestran las gráficas de precisión y pérdida respectivamente de la configuración 2.

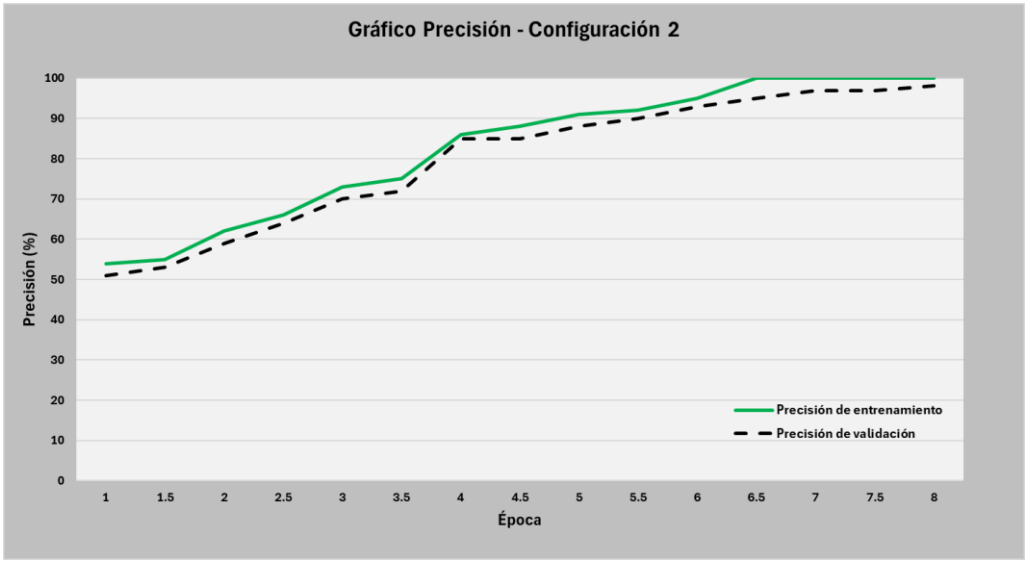


FIGURA 39. Gráfico de precisión de la Configuración 2.

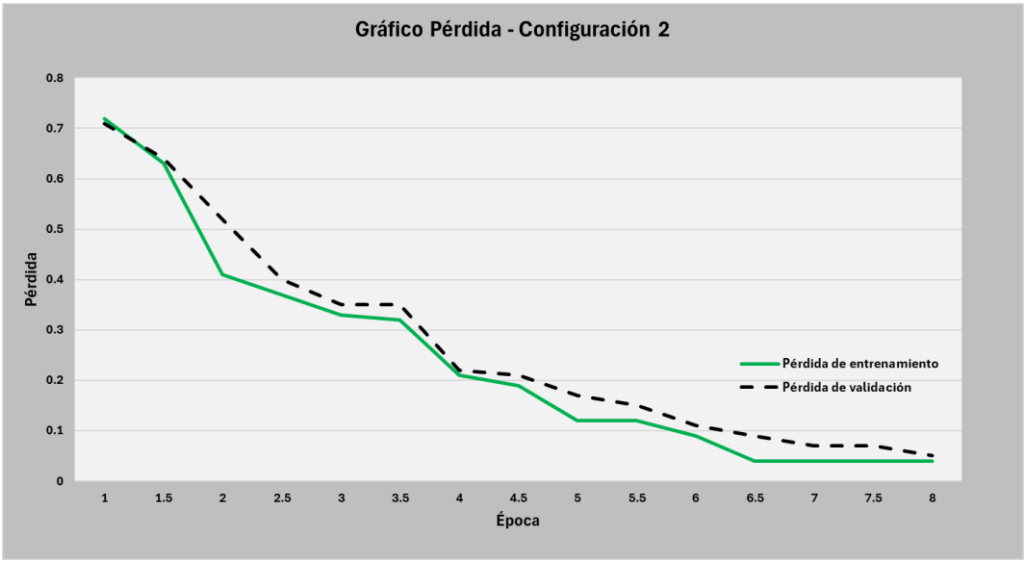


FIGURA 40. Gráfico de pérdida de la Configuración 2.

En la Tabla 13 se muestran los resultados más representativos de la configuración 2.

Tabla 13. Resultados Configuración 2.

	Config 2
Precisión entrenamiento	100%
Precisión validación	98.1%
Pérdida entrenamiento	0.04
Pérdida de validación	0.05
Iteraciones	88
Tiempo	19 mins

- **Análisis:** Esta configuración muestra un buen equilibrio entre la precisión de entrenamiento y validación. Ambas curvas de precisión crecen a un ritmo similar y convergen hacia valores cercanos al 98%, lo que indica que la red está generalizando bien. Las curvas de pérdida también disminuyen de manera constante, sin una brecha significativa entre el entrenamiento y la validación, lo que sugiere que el modelo no está sobreajustando. Además, el número de épocas es suficiente para alcanzar buenos resultados, pero sin excederse y correr el riesgo de sobreentrenar la red.

- **Conclusión:** La configuración 2 balancea adecuadamente el tamaño del lote, la tasa de aprendizaje y el número de épocas, obteniendo tanto altas métricas de precisión como una pérdida baja y bien balanceada entre los conjuntos de entrenamiento y validación.

- **Configuración 3**

  En las Figuras 40 y 41 se muestran las gráficas de precisión y pérdida respectivamente de la configuración 3.

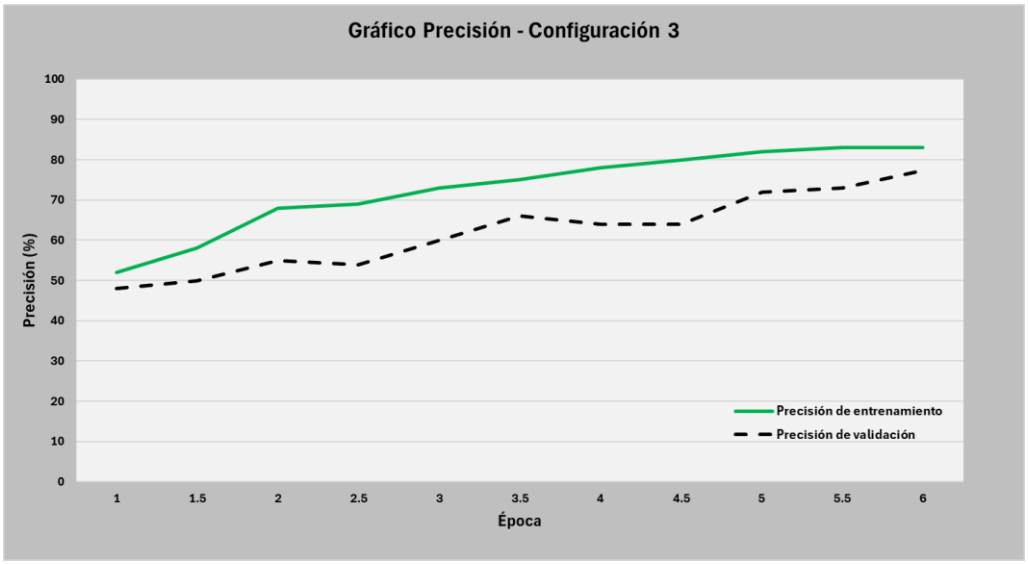


FIGURA 41. Gráfico de precisión de la Configuración 3.

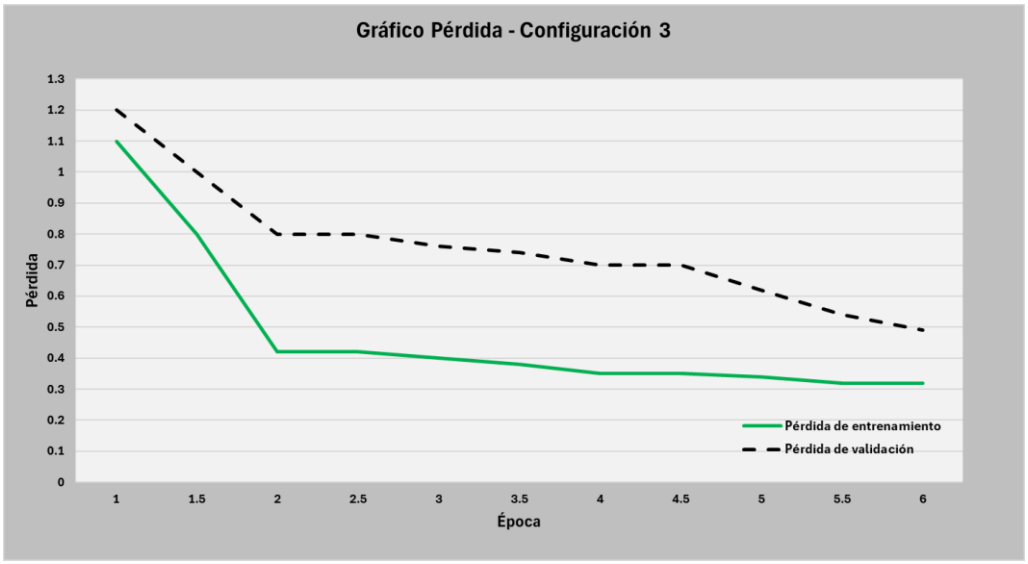


FIGURA 42. Gráfico de pérdida de la Configuración 3.

En la Tabla 14 se muestran los resultados más significativos de la configuración 3.

Tabla 14. Resultados Configuración 3.

	Config 3
Precisión entrenamiento	83%
Precisión validación	77.25%
Pérdida entrenamiento	0.32
Pérdida de validación	0.49
Iteraciones	138
Tiempo	14 mins

- **Análisis:** En esta configuración, la precisión del conjunto de entrenamiento crece rápidamente (en pocas épocas) y no obtiene un valor suficiente (subentrenamiento). Además, la precisión de validación muestra una brecha significativa. Esto indica que la red puede estar aprendiendo demasiado rápido en el conjunto de entrenamiento, sobreajustándose antes de que las curvas de validación puedan seguir el ritmo. La tasa de aprendizaje más alta probablemente esté haciendo que el modelo haga grandes saltos durante el entrenamiento, lo que afecta su capacidad de generalizar bien.

- **Conclusión:** El uso de una tasa de aprendizaje alta puede estar afectando la estabilidad del modelo, lo que se refleja en la brecha entre el rendimiento en el conjunto de entrenamiento y validación. El modelo no está generalizando bien.

- **Configuración 4**

  En las Figuras 42 y 43 se muestran las gráficas de precisión y pérdida respectivamente de la configuración 4.

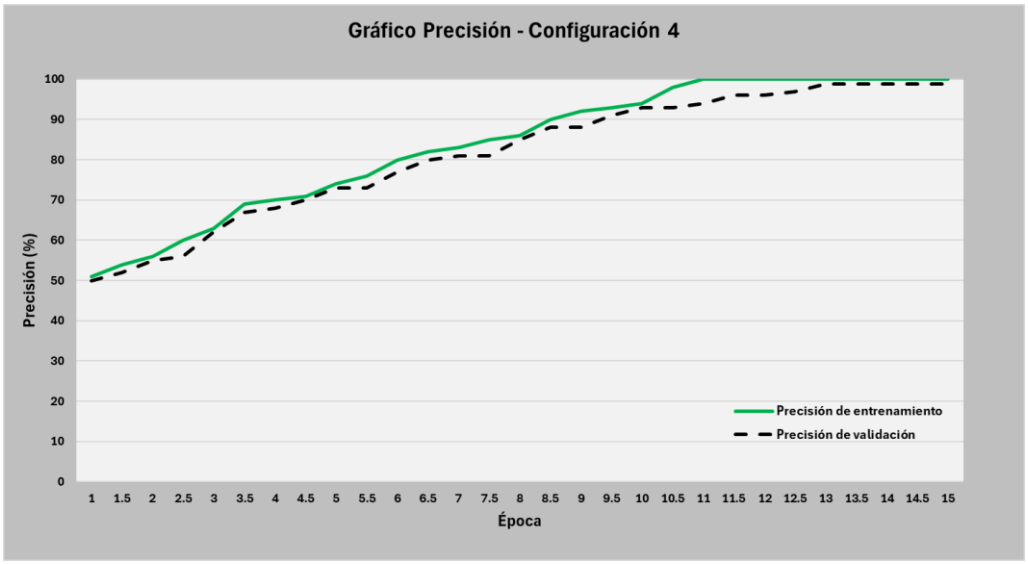


FIGURA 43. Gráfico de precisión de la Configuración 4.

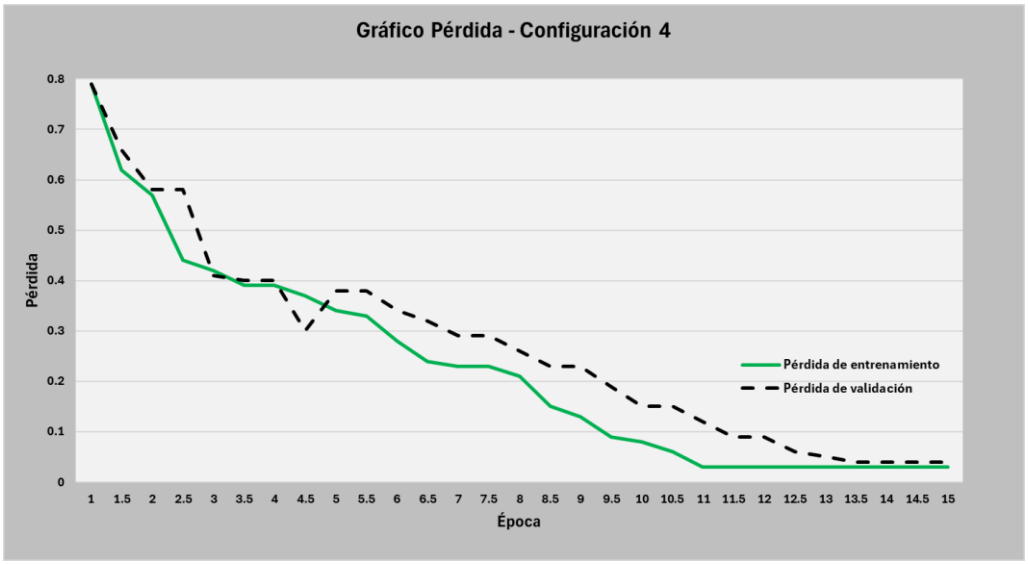


FIGURA 44. Gráfico de pérdida de la Configuración 4.

En la Tabla 15 se muestran los valores de los resultados más representativos de la configuración 4.

Tabla 15. Resultados Configuración 4.

	Config 4
Precisión entrenamiento	100%
Precisión validación	98.8%
Pérdida entrenamiento	0.04
Pérdida de validación	0.03
Iteraciones	705
Tiempo	50 mins

- **Análisis:** Esta configuración muestra un crecimiento lento pero estable en las curvas de precisión. La tasa de aprendizaje baja está haciendo que la red tarde más en aprender patrones útiles, lo que alarga el proceso de entrenamiento sin llegar a obtener mejoras significativas en comparación con otras configuraciones. Aunque no hay signos evidentes de sobreajuste, el número de épocas y el tamaño del lote relativamente pequeño hacen que la mejora sea muy lenta.

- **Conclusión:** El aprendizaje es muy lento, y aunque eventualmente el modelo llega a valores razonables, este método es ineficiente en cuanto a tiempo de entrenamiento y computación, sin aportar ventajas significativas en precisión o pérdida.

Elección de parámetros

Una vez obtenidos los resultados de todas las configuraciones, se elige la mejor opción comprando los resultados obtenidos. En la Tabla 16 se resumen los resultados obtenidos para todas las configuraciones.

Tabla 16. Resumen de resultados de todas las configuraciones.

	Config 1	Config 2	Config 3	Config 4
Precisión entrenamiento	100%	100%	83%	100%
Precisión validación	96.5%	98.1%	77.25%	98.8%
Pérdida entrenamiento	0.05	0.04	0.32	0.04
Pérdida de validación	0.09	0.05	0.49	0.03
Iteraciones	230	88	138	705
Tiempo	27 mins	19 mins	14 mins	50 mins

De las cuatro configuraciones, la Configuración 2 proporciona el mejor equilibrio entre el rendimiento del modelo y la eficiencia del entrenamiento. El tamaño del lote más grande (64 lotes) permite que la red aproveche mejor la actualización de los pesos en cada iteración, mientras que una tasa de aprendizaje más baja (0.0005) asegura que el entrenamiento sea estable y preciso. Además, el número de épocas (8 épocas) es suficiente para que la red aprenda bien sin sobreentrenarse ni sobreajustarse a los datos de entrenamiento.

Los datos indican que, en esta configuración, la red está generalizando bien a los datos de validación, ya que las curvas de precisión y pérdida no presentan diferencias significativas entre entrenamiento y validación. Este balance permite que el modelo obtenga un rendimiento alto tanto en entrenamiento como en validación, evitando los problemas de sobreajuste observados en la Configuración 3 y de entrenamiento ineficiente como en la Configuración 4.

En resumen, la Configuración 2 es la más eficiente porque maximiza el rendimiento en un tiempo de entrenamiento razonable, evitando los problemas de sobreajuste y falta de generalización.

3.4 ENTRENAMIENTO Y EVALUACIÓN DEL MODELO

El entrenamiento del modelo se llevó a cabo en MATLAB [14] utilizando la función `trainNetwork`. El conjunto de datos de entrenamiento se utilizó para ajustar los pesos del modelo, mientras que el conjunto de datos de validación se utilizó para ajustar los hiperparámetros y evitar el sobreajuste.

3.4.1 Proceso de entrenamiento

Tras haber realizado el proceso de ajuste de los parámetros de entrenamiento, se han elegido como óptimos los parámetros de la configuración 2. Estos parámetros se aplican tanto a la red de detección de defectos como a la red de clasificación de los mismos. En la Tabla 17 se muestran los parámetros de entrenamiento finales utilizados en el entrenamiento de las redes.

Tabla 17. Parámetros de entrenamiento finales tras ajuste.

Tasa de aprendizaje	0.0005
Tamaño del Lote	64
Número de épocas	8
Optimizador	SGDM

3.4.2 Evaluación del rendimiento

La evaluación del rendimiento se medirá por los siguientes parámetros [8].

- **Matriz de Confusión:** Proporciona una representación detallada de las predicciones correctas e incorrectas por clase. Es útil para identificar clases que la red confunde frecuentemente. En la Figura 44 se muestra una estructura tipo de la matriz de confusión.

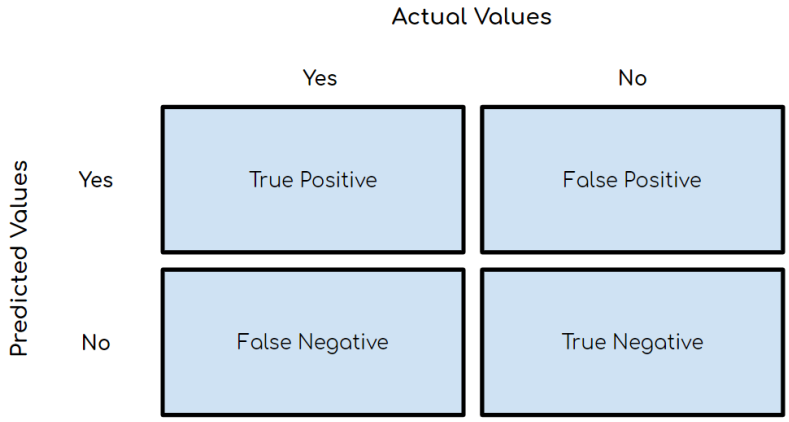


FIGURA 45. Estructura tipo de una matriz de confusión.

  - **Verdaderos Positivos (True Positives, VP):** Representa el número de ejemplos que son realmente positivos (pertenecen a la clase positiva) y que el modelo ha predicho correctamente como positivos.

    En el caso real, si se clasifican imágenes con y sin defectos, los verdaderos positivos serían aquellas imágenes sin defectos que fueron correctamente clasificadas como tales por el modelo.

  - **Falsos Negativos (False Negatives, FN):** Es el número de ejemplos que son realmente positivos, pero que el modelo ha predicho incorrectamente como negativos.

    En el caso del ejemplo de clasificación de imágenes, un falso negativo sería una imagen sin defectos que fue clasificada incorrectamente como con defectos. Este es un error de tipo "no reconocer" la clase positiva.

- **Falsos Positivos (False Positives, FP):** Es el número de ejemplos que son realmente negativos (pertenecen a la clase negativa), pero que el modelo ha predicho incorrectamente como positivos.

  En el ejemplo de clasificación de imágenes, un falso positivo sería una imagen con defectos que fue incorrectamente clasificada como sin defectos. Este es un error de tipo "falsa alarma", ya que el modelo detecta defectos donde no los hay.

- **Verdaderos Negativos (True Negatives, VN):** Representa el número de ejemplos que son realmente negativos y que el modelo ha predicho correctamente como negativos.

  En el ejemplo de clasificación de imágenes, los verdaderos negativos serían las imágenes con defectos que fueron correctamente clasificadas como tales por el modelo.

- **Precision (Precisión):** La precisión es la proporción de predicciones correctas sobre el total de predicciones positivas que el modelo ha realizado. Es una medida de cuántas de las instancias que se predijeron como positivas son realmente positivas.

$$\boldsymbol{Precision = \frac{TP}{TP + FP}}$$

  Una precisión alta indica que, cuando el modelo predice una clase positiva, es probable que sea correcta. Es especialmente importante en aplicaciones donde los falsos positivos son costosos.

- **Recall (Sensibilidad o Tasa de Verdaderos Positivos):** El Recall, o sensibilidad, es la proporción de verdaderos positivos sobre el total de instancias que son realmente positivas. Mide la capacidad del modelo para identificar correctamente todas las instancias positivas.

$$\boldsymbol{Recall = \frac{TP}{TP + FN}}$$

  Un alto Recall indica que el modelo es capaz de identificar la mayoría de las instancias positivas. Es crucial en aplicaciones donde los falsos negativos son costosos o peligrosos, como en el diagnóstico médico.

- **F1-Score:** El F1-Score es la media armónica de la precisión y el recall. Es una métrica que proporciona un balance entre precisión y recall, especialmente útil cuando hay un desequilibrio entre clases.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

  Un F1-Score alto significa que el modelo tiene un buen balance entre la precisión y el recall. Es especialmente útil cuando el costo de los falsos positivos y falsos negativos es similar, y es una métrica preferida cuando las clases están desbalanceadas.

- **Specificity (Especificidad):** La especificidad, o tasa de verdaderos negativos, mide la proporción de instancias negativas que el modelo identifica correctamente.

$$Specificity = \frac{TN}{TN + FP}$$

  La especificidad es importante en situaciones donde es crítico minimizar los falsos positivos.

- **Balanced Accuracy (Precisión Equilibrada):** La precisión equilibrada es la media aritmética de la sensibilidad y la especificidad. Es útil en conjuntos de datos desbalanceados.

$$Balanced\ Acuracy = \frac{Recall + Specificity}{2}$$

  La precisión equilibrada ofrece una visión más completa del rendimiento del modelo en escenarios donde las clases no están equilibradas.

- **ROC-AUC (Receiver Operating Characteristic - Area Under the Curve):** La curva ROC es un gráfico que muestra la tasa de verdaderos positivos frente a la tasa de falsos positivos a diferentes umbrales de decisión. El AUC (Area Under the Curve) es el área bajo esta curva.

  El AUC proporciona una medida del rendimiento general del modelo en la tarea de clasificación binaria. Un AUC de 0.5 indica un rendimiento aleatorio, mientras que un AUC de 1.0 indica un modelo perfecto.

- **Pérdida (Loss)**

  - Cross-Entropy Loss: Es la función de pérdida más común para tareas de clasificación. Mide la diferencia entre la distribución de las probabilidades predichas y la distribución verdadera de las clases. Un menor valor de pérdida indica una mejor predicción.

  - **Training vs Validation Loss:** Comparar la pérdida durante el entrenamiento y la validación ayuda a detectar si la red está sufriendo overfitting (pérdida de validación mayor que la de entrenamiento).

3.4.3 Resultados detección de defectos.

Gráfico de Precisión y Pérdida

Como se observan en las Figura 45 y 46, la precisión del modelo mejora progresivamente mientras que la pérdida disminuye, indicando un buen ajuste del modelo a los datos de entrenamiento.

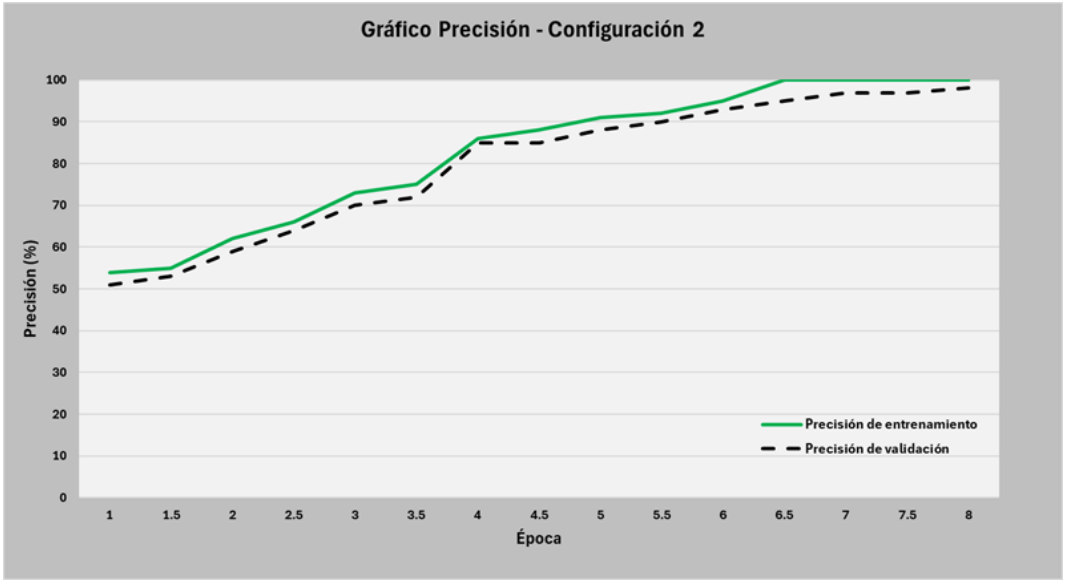


FIGURA 46. Gráfico de precisión de la identificación de defectos.

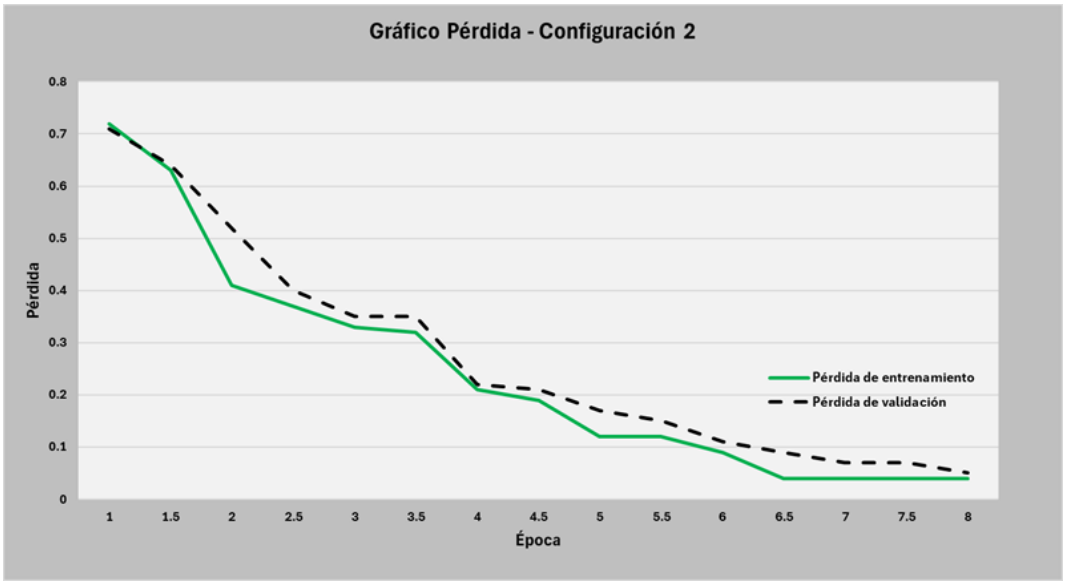


FIGURA 47. Gráfico de pérdida de la identificación de defectos.

Matriz de Confusión

La matriz de confusión en la Figura 47 muestra la distribución de clasificaciones correctas e incorrectas para las imágenes del conjunto de validación. Esta matriz ayuda a identificar posibles áreas de mejora en el modelo, como la reducción de falsos positivos o falsos negativos.

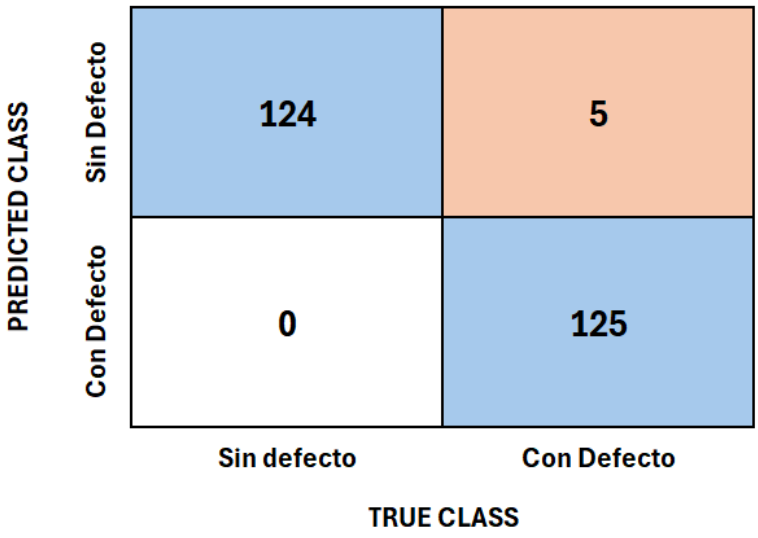


FIGURA 48. Matriz de confusión de la identificación de defectos.

Métricas de rendimiento del modelo.

La Tabla 18 resume las métricas de rendimiento del modelo. Estos resultados demuestran la capacidad del modelo para clasificar correctamente las imágenes con y sin defectos.

Tabla 18. Métricas de rendimiento del modelo.

Precisión	96.12%
Recall	100%
F1 Score	98.03%
Especificidad	96,15%
Precisión Equilibrada	98.1%

Clasificación del conjunto de prueba.

El modelo Resnet50 ha clasificado correctamente las 118 imágenes de las 120 que contiene el conjunto de prueba. Ha calificado 2 imágenes como “Sin Defecto” cuando la clasificación correcta era “Con Defecto”.

3.4.4 Resultados clasificación de defectos

Gráfico de Precisión y Pérdida

En las Figuras 48 y 49 se observa cómo la precisión de entrenamiento y la precisión de validación aumentan a medida que disminuyen las pérdidas, lo que significa que el modelo está bien ajustado a los datos de entrenamiento. Además, la diferencia entre la precisión de entrenamiento y la precisión de validación es mínima, por lo que el modelo garantiza un buen ajuste entre el entrenamiento y la validación.

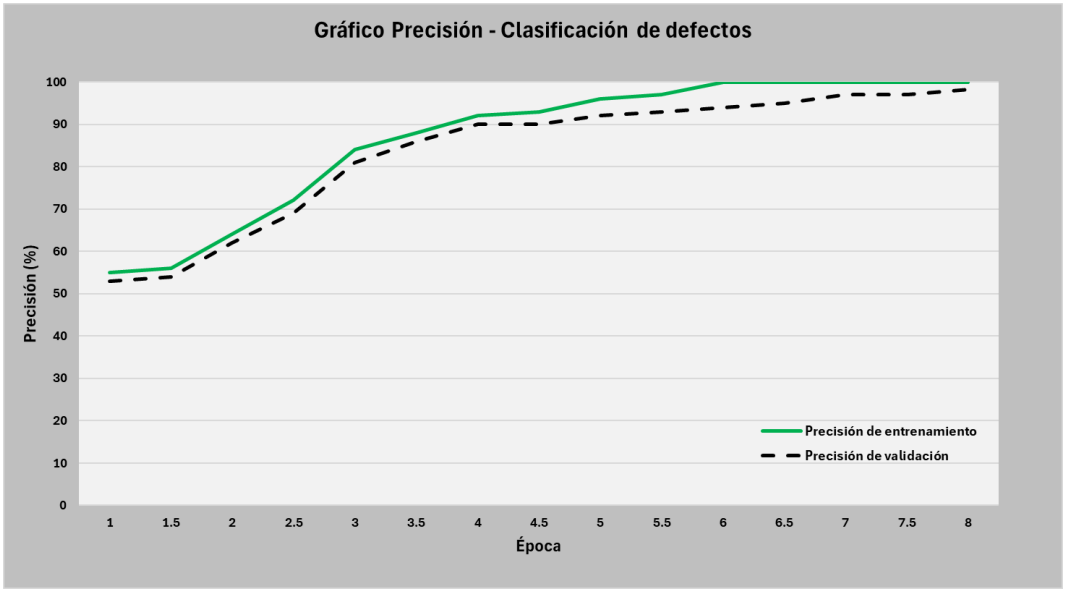


FIGURA 49. Gráfico de precisión de la clasificación de defectos.

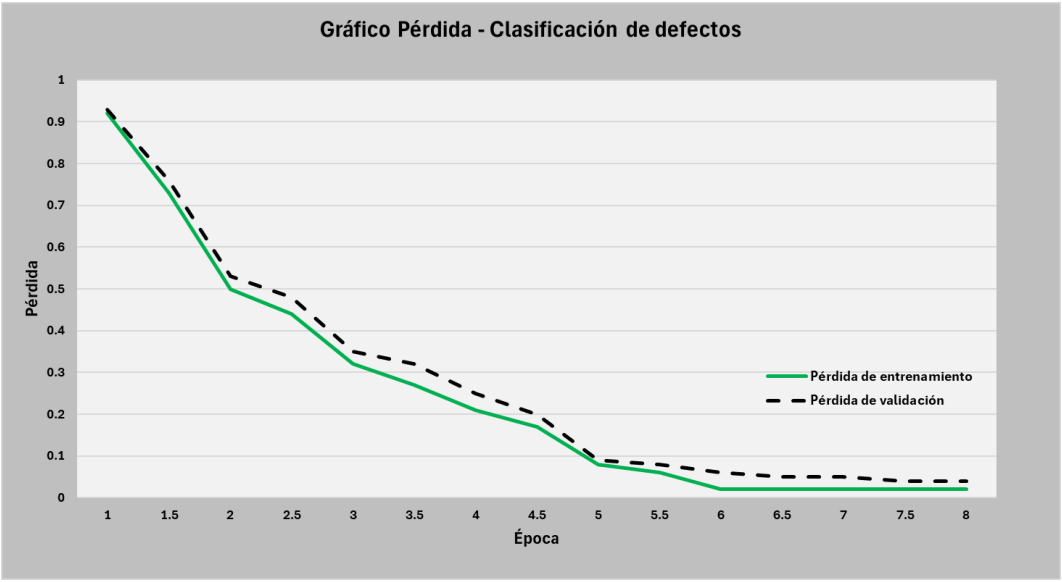


FIGURA 50. Gráfico de pérdida de la clasificación de defectos.

Matriz de Confusión

En la Figura 50 se muestra la matriz de confusión de la clasificación de defectos. Se observa que clasifica bien la mayoría de las imágenes, pero hay ciertas categorías que confunde eventualmente.

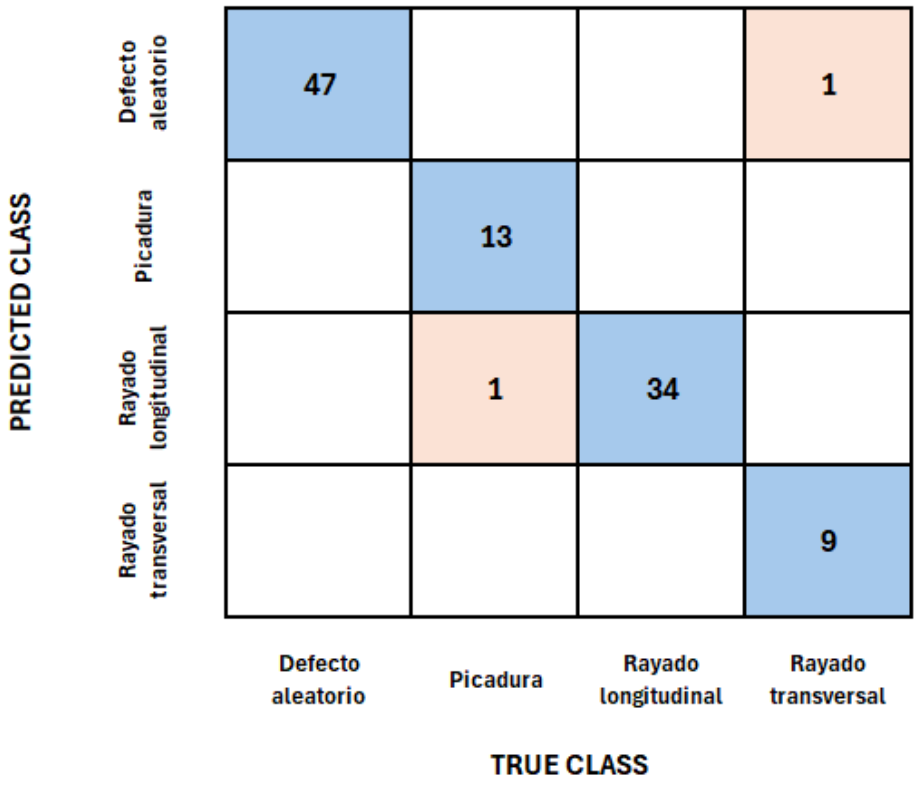


FIGURA 51. Matriz de confusión de la clasificación de defectos.

Precisión por categoría

En este caso de clasificación múltiple, la precisión se calculará como el cociente entre el número de clasificaciones correctas de una clase entre el número total de imágenes de esa clase.

$$Precisión = \frac{Nº\ imagenes\ bien\ calificadas\ clase\ x}{Nº\ imágenes\ total\ clase\ x}$$

En la Tabla 19 se muestran los resultados de las precisiones obtenidas para cada clase, así como la cantidad de imágenes bien y mal calificadas de cada una.

Tabla 19. Resultados de las precisiones por clase.

	Imágenes bien calificadas	Imágenes mal calificadas	Precisión
Defecto aleatorio	47	0	100%
Picadura	13	1	92.8%
Rayado longitudinal	34	0	100%
Rayado transversal	9	1	90%

La media ponderada de las precisiones de todas las clases es de **98.11%.**

La imagen mal calificada de la clase “Picadura” ha sido clasificada como “Rayado longitudinal” y las dos imágenes mal clasificadas de la clase “Rayado transversal” han sido clasificadas como “Defecto aleatorio”.

Clasificación del conjunto de prueba

La red clasifica correctamente 58 imágenes con defectos de 60, lo que significa que clasifica correctamente un 96.7%. En la Tabla 20 se muestra la clasificación por categoría. En las imágenes mal calificadas se añade como las calificó la red entre paréntesis.

Tabla 20. Clasificación de defectos del conjunto de prueba.

	Imágenes bien calificadas	Imágenes mal calificadas
Defecto aleatorio	16	0
Picadura	15	1 (Rayado longitudinal)
Rayado longitudinal	20	0
Rayado transversal	7	1 (Defecto aleatorio)

3.5 EVALUACIÓN DEL RENDIMIENTO

El rendimiento del modelo en la tarea de detección de defectos y su clasificación del proceso de trefilado se evaluó utilizando varias métricas [7]. Los resultados indican que el modelo es capaz de diferenciar con alta precisión entre imágenes con y sin defectos. Además, el modelo es capaz de clasificar el tipo de defecto que presentan las imágenes.

3.5.1 Detección de defectos

Análisis de la Matriz de Confusión

La matriz de confusión revela los siguientes resultados:

- **True Positives (TP = 124):** La red ha clasificado correctamente 124 imágenes de la clase positiva ("Sin Defecto").

- **True Negatives (TN = 125):** La red ha clasificado correctamente 125 imágenes de la clase negativa ("Con Defecto").

- **False Positives (FP = 5):** Cinco imágenes con defectos han sido clasificadas erróneamente como imágenes sin defectos.

- **False Negatives (FN = 0):** No hay ningún caso donde una imagen sin defectos haya sido mal clasificada como con defecto.

Evaluación de las métricas de rendimiento

- **Precisión (96.12%):** Esta métrica indica que de todas las imágenes que la red ha clasificado como "Sin Defecto", el 96.12% realmente lo son. Una precisión alta es importante en este caso porque queremos minimizar la cantidad de imágenes con defectos que se clasifican erróneamente como sin defectos (es decir, falsos positivos).

- **Recall (100%):** El recall del 100% indica que la red ha detectado correctamente todas las imágenes "Sin Defecto". Esto significa que no ha habido ningún falso negativo.

- **F1 Score (98.03):** El F1 score combina precisión y recall en una única métrica, y un valor de 98.03% indica un excelente equilibrio entre ambas. Esto refleja que la red está funcionando muy bien tanto en términos de detectar correctamente las imágenes sin defectos como en minimizar los falsos positivos.

- **Especificidad (96.15%):** La especificidad mide la capacidad de la red para identificar correctamente las imágenes con defectos. Un valor de 96.15% indica que la red tiene un buen rendimiento a la hora de distinguir imágenes con defectos, con solo cinco falsos positivos.

- **Precisión equilibrada (98.1%):** Esta métrica representa el promedio entre el recall y la especificidad, dándole un valor balanceado que tiene en cuenta tanto la capacidad para identificar imágenes "Sin Defecto" como para identificar correctamente las imágenes "Con Defecto". Un valor alto de precisión equilibrada indica que la red está funcionando de manera efectiva en ambas clases.

Evaluación de la clasificación del conjunto de prueba.

El modelo ha clasificado correctamente 118 imágenes de 120. Esto significa que ha clasificado bien el 98.3% de las imágenes. La red ha clasificado 2 imágenes “Con Defecto” como imágenes “Sin Defecto”.

En las Figuras 51 y 52 se muestran las imágenes que la red ha calificado de manera errónea.

FIGURA 52. Imagen mal calificada 1

FIGURA 53. Imagen mal calificada 2

La red no es capaz de detectar los defectos en estas imágenes, ya que las clasifica como “Sin Defecto” cuando en realidad son imágenes “Con Defecto”. La razón por la cual la red califica mal estas imágenes puede ser la calidad de la imagen, en la cual no se ve con claridad los defectos del cable.

En general, los resultados obtenidos demuestran la viabilidad de utilizar redes neuronales convolucionales para la monitorización del proceso de trefilado mediante la adquisición de imágenes.

3.5.2 Clasificación de defectos

La red es capaz de clasificar los tipos de defectos encontrados en las imágenes con un alto grado de precisión (98,11%). Aun así, la red confunde ciertas clases eventualmente. Las clases las cuales en ciertas ocasiones no es capaz de diferenciar son por un lado clase “Picadura” y clase “Rayado longitudinal” y clase “Rayado transversal” y clase “Defecto aleatorio” por otro.

Para intentar buscar una explicación a la confusión de la red, se analizan las fotos mal calificadas de la prueba. En la Figura 53 se muestra la imagen con picadura calificada como “Rayado longitudinal” y en la Figura 54 se muestra la imagen de rayado transversal clasificada como “Defecto aleatorio”.

FIGURA 54. Imagen con "Picadura" calificada como "Rayado longitudinal"

En este caso el error de la red puede deberse a que las picaduras se disponen de forma longitudinal, pudiéndose confundir con un rayado longitudinal.

FIGURA 55. Imagen con "Rayado transversal" como "Defecto aleatorio"

La red puede confundir la clase por no identificar el rayado transversal correctamente y encuentra parecido a las imágenes con defectos puntuales. Un aumento de las muestras para hacer cada clase más diferencial podría ayudar a resolver este problema

4 DISCUSIÓN

4.1 INTERPRETACIÓN DE LOS RESULTADOS

Los resultados obtenidos en la detección de defectos muestran que la red está obteniendo resultados muy buenos en todas las métricas clave, especialmente en el recall del 100%, lo que significa que clasifica correctamente todas las imágenes sin defectos. También, el valor de precisión del 96.12% muestra que la red está haciendo un buen trabajo minimizando los falsos positivos, aunque todavía hay un pequeño número de ellos (5 casos). En general, la red está funcionando de manera muy efectiva, mostrando un excelente balance entre precisión y recall.

En la Figura 55, Figura 56, Figura 57 y Figura 58 podemos observar los resultados que ofrece el modelo para un cable sin defectos. En la Figura 59, 60, 61 y 62 se muestran los resultados del modelo para un cable con defectos.

Sin Defecto

FIGURA 56. Ejemplo 1 de cable sin defecto.

Sin Defecto

FIGURA 57. Ejemplo 2 de cable sin defecto.

Sin Defecto

FIGURA 58. Ejemplo 3 de cable sin defecto.

Sin Defecto

FIGURA 59. Ejemplo 4 de cable sin defecto.

Con Defecto, Defecto aleatorio

FIGURA 60. Cable con defecto aleatorio.

Con Defecto, Rayado transversal

FIGURA 61. Cable con defecto de rayado transversal.

Con Defecto, Rayado longitudinal

FIGURA 62. Cable con defecto de rayado longitudinal.

Con Defecto, Picadura

FIGURA 63. Cable con defecto de picadura.

Además, la alta precisión obtenida en la clasificación de defectos (98,11%) muestra la solidez de la red para detectar que tipo de defecto se está encontrando en el proceso industrial

La alta precisión sugiere que el modelo es capaz de generalizar bien a nuevas imágenes, lo que es crucial para su implementación en un entorno de producción real.

En resumen, los resultados obtenidos cumplen con los objetivos planteados al inicio del proyecto, demostrando la viabilidad de utilizar una red neuronal convolucional como ResNet50 para la monitorización del proceso de trefilado. Sin embargo, se identifican áreas de mejora que podrían incrementar aún más la efectividad del sistema.

4.2 LIMITACIONES DEL ESTUDIO

A pesar de los resultados prometedores, este estudio presenta varias limitaciones que deben ser consideradas:

- **Calidad y Cantidad de Datos:** El conjunto de datos utilizado para entrenar y evaluar el modelo fue limitado en términos de cantidad y diversidad. Una mayor cantidad de imágenes y una mayor variedad de condiciones de captura podrían mejorar la capacidad del modelo para generalizar a diferentes escenarios.

- **Condiciones de Captura:** Las imágenes fueron capturadas en condiciones de iluminación controladas, lo que puede no reflejar las condiciones reales de un entorno industrial. Variaciones en la iluminación y el ángulo de captura pueden afectar la precisión del modelo.

- **Etiquetado Manual:** El etiquetado de las imágenes fue realizado manualmente, lo que puede introducir sesgos y errores humanos. Un sistema de etiquetado automatizado podría mejorar la consistencia y precisión del conjunto de datos.

- **Complejidad del Modelo:** ResNet50 es una red neuronal profunda con una gran cantidad de parámetros, lo que requiere un poder computacional significativo y puede no ser práctico para todas las aplicaciones industriales.

- **Velocidad del cable:** Las imágenes han sido tomadas de cables en estático. La adquisición en una trefiladora real podría causar problemas ya que el cable se encuentra en movimiento.

Estas limitaciones indican que, aunque los resultados obtenidos son prometedores, se requiere un trabajo adicional para mejorar y validar el modelo en un entorno de producción real.

4.3 PROPUESTAS DE MEJORA

Para superar las limitaciones identificadas y mejorar el rendimiento del sistema, se proponen las siguientes mejoras:

- **Aumento del Conjunto de Datos:** Capturar más imágenes en diversas condiciones de operación y con diferentes tipos de defectos.

- **Mejora en la Calidad de Imágenes:** Utilizar cámaras de mayor resolución y sistemas de iluminación avanzados para capturar imágenes con mayor detalle y consistencia.

- **Automatización del Etiquetado:** Implementar sistemas de etiquetado automatizado basados en algoritmos de preprocesamiento y segmentación para reducir el sesgo y errores humanos.

- **Modelos Alternativos:** Explorar modelos de redes neuronales más ligeros o técnicas de transfer learning para reducir la complejidad computacional sin comprometer la precisión.

- **Optimización de Hiperparámetros:** Realizar una búsqueda más exhaustiva de hiperparámetros utilizando técnicas como la búsqueda en cuadrícula o la optimización bayesiana para encontrar los parámetros óptimos para el modelo.

- **Implementación en Tiempo Real:** Desarrollar un sistema de implementación en tiempo real que permita la monitorización continua del proceso de trefilado y la detección instantánea de defectos, ajustando automáticamente las condiciones de operación para minimizar defectos.

Estas mejoras no solo podrían aumentar la precisión y eficiencia del sistema, sino también su aplicabilidad en un entorno industrial real.

5 CONCLUSIÓN

5.1 RESUMEN DE LOS HALLAZGOS PRINCIPALES

El objetivo principal de este trabajo fue desarrollar un sistema de monitorización del proceso de trefilado mediante la adquisición de imágenes y su análisis utilizando una red neuronal convolucional.

A través de un riguroso proceso de adquisición y preprocesamiento de datos, se capturaron y etiquetaron imágenes representativas del proceso de trefilado. Se utilizaron las clases “Sin defecto” y “Con defecto” para la detección de defectos y las clases “Defecto aleatorio”, “Picadura”, “Rayado longitudinal” y “Rayado transversal” para la clasificación de defectos. Utilizando MATLAB, se implementó y ajustó la red ResNet50 para clasificar estas imágenes con alta precisión.

Los resultados obtenidos mostraron que el modelo ResNet50 alcanzó una precisión balanceada de 98.1% en la detección de defectos y una precisión de 98.11% en la clasificación de defectos, demostrando su capacidad para monitorizar el proceso de trefilado.

5.2 RELEVANCIA DEL ESTUDIO

Este estudio representa un avance significativo en la aplicación de técnicas de visión por computadora y redes neuronales convolucionales para la monitorización de procesos industriales. La implementación de un sistema automatizado basado en ResNet50 para la detección de defectos y su clasificación en el proceso de trefilado tiene el potencial de mejorar considerablemente la eficiencia y la calidad de la producción.

La capacidad de identificar y clasificar defectos en tiempo real puede ayudar a reducir los costos asociados con el retrabajo y el desperdicio de materiales, además de asegurar la entrega de productos de alta calidad. Esta tecnología también puede ser adaptada a otros procesos industriales, lo que subraya su versatilidad y aplicabilidad en diferentes contextos de manufactura.

5.3 RECOMENDACIONES PARA FUTUROS TRABAJOS

Basado en las limitaciones identificadas y los hallazgos del estudio, se recomiendan las siguientes áreas para futuros trabajos:

- **Ampliación del Conjunto de Datos**: Capturar más imágenes bajo diversas condiciones de operación y con diferentes tipos de defectos para mejorar la capacidad del modelo de generalizar a diferentes escenarios.

- **Mejora en la Calidad de las Imágenes:** Utilizar equipos de captura de imágenes de mayor calidad y sistemas de iluminación avanzados para reducir la variabilidad y aumentar la precisión del sistema.

- **Desarrollo de Modelos Más Ligeros:** Investigar modelos de redes neuronales más ligeros que puedan ser implementados en tiempo real con menos requisitos computacionales, manteniendo o mejorando la precisión.

- **Integración en Sistemas de Producción:** Desarrollar e implementar el sistema en un entorno de producción real para validar su desempeño y realizar ajustes basados en retroalimentación continua.

- **Estudios Comparativos:** Realizar estudios comparativos con otras técnicas de detección de defectos para evaluar la eficiencia y precisión relativa de diferentes enfoques y tecnologías.

Estas recomendaciones tienen el potencial de mejorar significativamente la eficacia y aplicabilidad del sistema propuesto, contribuyendo al avance de la monitorización automatizada en la industria.

6 REFERENCIAS

[1] Altan, T., & Ngaile, G. (2012). *Cold and Hot Forging: Fundamentals and Applications*. ASM International.

[2] Bartelt T. Industrial automated systems: instrumentation and motion control. Boca Raton: CRC Press; 2003.

[3] Bequette BW. Process control: modeling, design, and simulation. Upper Saddle River: Prentice Hall; 2003.

[4] Caso Fernández E, García Fernández P, Fernández del Rincón A, de Juan de Luna A, Díez Ibarbia A. Monitorización del proceso de trefilado mediante emisión acústica: estudio del efecto del tipo de lubricante y la temperatura de hilera. Rev Metal. 2018.

[5] Coughanowr DR, LeBlanc S. Process systems analysis and control. New York: McGraw-Hill; 1991.

[6] Dunn WC. Fundamentals of industrial instrumentation and process control. New York: McGraw-Hill; 2005.

[7] Gonzalez RC, Woods RE. Digital image processing. Boston: Pearson; 2018.

[8] Géron A. Hands-on machine learning with Scikit-Learn & TensorFlow. Sebastopol: O'Reilly Media; 2019.

[9] Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge: MIT Press; 2016.
He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jun 27–30; Las Vegas, NV. IEEE; 2016

[10] Groover, M. P. (2012). *Fundamentals of Modern Manufacturing: Materials, Processes, and Systems* (5th ed.). Wiley.

[11] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jun 27–30; Las Vegas, NV. Piscataway (NJ): IEEE

[12] Hosford, W. F., & Caddell, R. M. (2011). *Metal Forming: Mechanics and Metallurgy* (4th ed.). Cambridge University Press.

[13] Instituto de Ingeniería del Conocimiento (IIC). Machine Learning y Deep Learning. [Internet]. [citado 2024 Aug 21]. Disponible en: https://www.iic.uam.es/inteligencia-artificial/machine-learning-deep-learning/

[14] Kim P. MATLAB deep learning: with machine learning, neural networks and artificial intelligence. Berkeley: Apress; 2017.

[15] Kalpakjian, S., & Schmid, S. R. (2014). Manufacturing Engineering and Technology (7th ed.). Pearson.

[16] Larsson J, Jansson A, Karlsson P. Monitoring and evaluation of the wire drawing process using thermal imaging. J Manuf Process. 2019.

[17] Larsson J, Jansson A, Pejryd L. Process monitoring of the wire drawing process using a web camera based vision system. J Manuf Process. 2017.

[18] Larsson J, Johansson-Cider H, Jarl M. Monitoring of the wiredrawing process. Wire J Int. 2015.

[19] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[20] ONU Medio Ambiente. ¿Cómo se mide la calidad del aire? [Internet]. 2019 May 17 [citado 2024 Aug 21]. Disponible en: https://www.unep.org/es/noticias-y-reportajes/reportajes/como-se-mide-la-calidad-del-aire

[21] Orange. Especificaciones Xiaomi 11T Pro y Apple iPad (9th Generation). [Internet]. [citado 2024 Aug 21]. Disponible en: https://guias.orange.es/xiaomi-11t-pro-android-11-0/especificaciones/apple-ipad-9th-generation-ipados-15-4/

[22] Pejryd L, Larsson J, Olsson M. Process monitoring of wire drawing using vibration sensoring. J Manuf Process. 2016.

[23] Pérez P. Sistemas de Manufactura Integrados por Computadora: Capítulo 2. [Internet]. [citado 2024 Aug 21]. Disponible en: https://www.fam

[24] Petrou M, Petrou C. Image processing: the fundamentals. Chichester: Wiley; 2000.

[25] Rueda C. Redes neuronales convolucionales para clasificación de imágenes CIFAR-10. [Internet]. [citado 2024 Aug 20]. Disponible en: https://dcain.etsin.upm.es/~carlos/bookAA/05.7_RRNN_Convoluciones_CIFAR_10_INFORMATIVO.html

[26] SEBIR. El proceso de trefilado de acero. [Internet]. [citado 2024 Aug 21]. Disponible en: https://sebirsa.com/procesos/el-proceso-de-trefilado-de-acero/

[27] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*

[28] Tassi O. Nonferrous wire handbook, Vol. 2: Bare wire processing. Connecticut: The Wire Association International, Inc.; 1981.

[29] Totten, G. E., & Funatani, K. (2004). *Handbook of Metallurgical Process Design*. Marcel Dekker.

[30] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Places: A 10 Million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6), 1452-1464.

7 ANEXOS

En este capítulo se muestran los códigos utilizados en MATLAB para la adquisición y procesamiento de imágenes, así como las modificaciones, entrenamiento e implementación de la red neuronal ResNet50 para conseguir una correcta clasificación de las imágenes (defectos/no defectos).

7.1 ADQUISICIÓN DE IMÁGENES

Se plantea un código capaz de tomar un número de imágenes determinado con una frecuencia de muestreo determinada. Estos parámetros podrían variar dependiendo de las necesidades, parámetros y características de cada proceso de trefilado.

```
% Lista las cámaras disponibles
camList = webcamlist;

% Selecciona la primera cámara (cámara incorporada en el ordenador)
cam = webcam(1);

% Define el intervalo de tiempo entre capturas (en segundos)
intervalo_tiempo = ;

% Número total de capturas que deseas realizar
num_capturas = ;

% Inicializar una matriz vacía
matriz_resultados = zeros(filas, columnas);

% Bucle para realizar múltiples capturas
for i = 1:num_capturas
    % Captura una foto
    imagen = snapshot(cam);
    % Pausa la ejecución durante el intervalo de tiempo especificado
    pause(intervalo_tiempo);
end
```

7.2 PREPROCESADO DE IMÁGENES Y AUEMNTO DE DATOS

Se presenta el código de MATLAB utilizado para el preprocesado de las imágenes y aumento del conjunto de datos. Se llevan a cabo técnicas de redimensionado, normalizado y enfoque para aumentar el rendimiento de las redes neuronales. Con las imágenes preprocesadas, se aplican técnicas de espejo horizontal, zoom e introducción de ruido para aumentar considerablemente el conjunto de datos.

Se carga el directorio donde están las imágenes a preprocesar y el directorio de destino de las imágenes preprocesadas.

```
% Directorio donde están las imágenes
input_folder = 'MUESTRAS/SIN DEFECTO'
output_folder = 'MUESTRAS PROCESADAS/SIN DEFECTO';  % Carpeta de salida


% Listar los archivos de imagen en la carpeta
image_files = dir(fullfile(input_folder, '*.jpg'));  % Cambia la extensión si es
necesario
```

Se preprocesa y se aplican las técnicas de aumento de datos para cada imagen de la carpeta. Una vez aplicadas las técnicas, se guardan las imágenes en la carpeta destino.

```
% Iterar sobre cada imagen en la carpeta
for i = 1:length(image_files)
    % Leer la imagen
    file_name = image_files(i).name;
    file_path = fullfile(input_folder, file_name);
    image = imread(file_path);

    size_image=size(image);
    T=size_image(1);
```

Se enfocan, normalizan y redimensionan las imágenes que han sido adquiridas.

```
    %% Preprocesado

    % Enfoque
    sharpened_image=imsharpen(image,'Radius',2,'Amount',3);

    % Normalizado
    normalized_image=im2double(sharpened_image);

    % Redimensionado

    resized_image = imresize(normalized_image, [224, 224],'bicubic');

    adjusted_image=resized_image;
```

Se aplican las técnicas de espejo horizontal, zoom y adicción de ruido gaussiano a las imágenes adquiridas. Una vez aplicadas estas técnicas, se preprocesan las imágenes para su uso en las redes neuronales (enfoque, normalizado y redimensionado)

```
    %% Aumento de muestras

    % 1. Espejeo horizontal
    imagen_flip = flip(image, 2);  % Voltear horizontalmente

    % Enfoque
    imagen_flip_enfoque=imsharpen(imagen_flip,'Radius',2,'Amount',3);

    % Normalizado
    imagen_flip_normalizada=im2double(imagen_flip_enfoque);

    %Redimensionado
    flipped_image=imresize(imagen_flip_normalizada,[224,224],'bicubic');


    % 2. Zoom (simulado con recorte y redimensionado)
    imagen_recortada0 = imcrop(image, [100 100 T-200 T-200]);
    imagen_recortada=imresize(imagen_recortada0,[T,T], 'bicubic');

    % Enfoque
    imagen_recortada_enfoque=imsharpen(imagen_recortada,'Radius',2,'Amount',3);

    % Normalizado
    imagen_recortada_normalizada=im2double(imagen_recortada_enfoque);

    %Redimensionado
    zoomed_image=imresize(imagen_recortada_normalizada,[224,224],"bicubic");


    % 3. Ruido gaussiano
    imagen_ruido = imnoise(image, 'gaussian', 0, 0.002);

    % Enfoque
    imagen_ruido_enfoque=imsharpen(imagen_ruido,'Radius',2,'Amount',3);

    % Normalizado
    imagen_ruido_normalizada=im2double(imagen_ruido_enfoque);

    %Redimensionado
    noisy_image=imresize(imagen_ruido_normalizada,[224,224],'bicubic');


    % Guardar las imágenes
    imwrite(adjusted_image, fullfile(output_folder, ['adjusted_' file_name]));
    imwrite(flipped_image, fullfile(output_folder, ['flipped_' file_name]));
    imwrite(zoomed_image, fullfile(output_folder, ['zoomed_' file_name]));
    imwrite(noisy_image, fullfile(output_folder, ['noisy_' file_name]));
end
```

Se aplica el mismo proceso a las imágenes con defecto. Simplemente se cambian los directorios. Se hace lo propio con el conjunto de prueba.

7.3 IMPLEMENTACIÓN DE RESNET50

7.3.1 Detección de defectos

A continuación, se presenta el código de MATLAB utilizado para la carga, preparación, entrenamiento e implementación de la red neuronal ResNet50 para la clasificación de imágenes (defectos/no defectos) del proceso de trefilado.

Se crea una ImageDatastore de la carpeta en la que se encuentran las imágenes, divididas en dos subcarpetas: SIN DEFECTO Y CON DEFECTO

```
% Crear un imageDatastore
imds = imageDatastore('Ruta Carpeta Imagenes','IncludeSubfolders',
true,'LabelSource', 'foldernames');
```

Se dividen las imágenes aleatoriamente en dos conjuntos de datos: imágenes para entrenamiento e imágenes para validación). El 75% de las imágenes utilizadas (Total – Imágenes prueba) va destinada a entrenamiento, y las restantes a validación.

```
% Dividir los datos en conjuntos de entrenamiento y validación
[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.75, 'randomized');
```

Se carga la red y se reemplazan las capas fully connected, softmax y la capa de clasificación para adaptar la red al problema de clasificación binaria.

```
% Cargar la red preentrenada
net = resnet50;

% Crear un grafo de capas
lgraph = layerGraph(net);

% Número de clases
numClasses = 2; % Cambiar esto al número de clases en tu problema



% Reemplazar la capa fully connected, la capa softmax y la capa de clasificación
newLayers = [fullyConnectedLayer(numClasses, 'Name', 'new_fc', 10,
'BiasLearnRateFactor', 10) ('Name', 'new_softmax') classificationLayer('Name',
'new_classoutput')];

% Eliminar las capas finales
lgraph = removeLayers(lgraph, {'fc1000', 'fc1000_softmax',
'ClassificationLayer_fc1000'});

% Añadir las nuevas capas
lgraph = addLayers(lgraph, newLayers);
```

```
% Conectar las nuevas capas al grafo de capas
lgraph = connectLayers(lgraph, 'avg_pool', 'new_fc');
```

Se obtiene el tamaño de entrada de la red para saber el tamaño que deben de tener las imágenes, posteriormente se redimensionan y se convierten a RGB si fuera necesario

```
% Obtener el tamaño de entrada de la red
inputSize = net.Layers(1).InputSize;

% Redimensionar las imágenes y convertir a RGB si es necesario
augimdsTrain=augmentedImageDatastore(inputSize(1:2),imdsTrain,'ColorPreprocessing'
,'gray2rgb');

augimdsValidation=augmentedImageDatastore(inputSize(1:2),imdsValidation,'ColorPrep
rocessing', 'gray2rgb');
```

Se especifican los parámetros de entrenamiento y se entrena la red.

```
% Especificar las opciones de entrenamiento
options      =      trainingOptions('sgdm','MiniBatchSize',      64,'MaxEpochs',
8,'InitialLearnRate',         0.0005,'ValidationData',         augimdsValidation,
'ValidationFrequency', 10, , false, 'Plots', 'training-progress');

% Entrenar la red
netTransfer = trainNetwork(augimdsTrain, lgraph, options);
```

Se evalúan los resultados. Se presenta la gráfica de precisión y pérdida y la matriz de confusión.

```
% Evaluar la red
YPred = classify(netTransfer, augimdsValidation);
YValidation = imdsValidation.Labels;
accuracy = sum(YPred == YValidation) / numel(YValidation);
fprintf('Precisión de Validación: %.2f%%\n', accuracy * 100);


% Mostrar la matriz de confusión
figure;
confusionchart(YValidation, YPred);
```

Se clasifican las imágenes del conjunto de prueba, mostrando por pantalla cuando una imagen está mal calificada.

```
%% Clasificar nuevas imágenes

% Comprobacion Calificacion imagenes sin defectos

% Directorio donde están las imágenes
input_folder_new = 'PRUEBA PROCESADAS/SIN DEFECTO';

% Listar los archivos de imagen en la carpeta
image_files_new = dir(fullfile(input_folder_new, '*.jpg'));  % Cambia la extensión
si es necesario

% Iterar sobre cada imagen en la carpeta
for i = 1:length(image_files_new)

    % Leer la imagen
    file_name = image_files_new(i).name;
    file_path = fullfile(input_folder_new, file_name);
    image = imread(file_path);
    newImage = imresize(image, inputSize(1:2)); % Redimensionar la imagen

    % Convertir la imagen a RGB si es necesario
    if size(newImage, 3) == 1
        newImage = repmat(newImage, [1 1 3]);
    end

    % Clasificar la nueva imagen
    YPredNew = classify(netTransfer, newImage);
    disp(['La nueva imagen es clasificada como: ', char(YPredNew)]);

    % Mostrar la nueva imagen con el resultado
    if char(YPredNew)== 'CON DEFECTO'
        disp('Imagen mal calificada, encuentra defectos cuando no los tiene');
        figure
        imshow(newImage);
        title('Imagen mal Calificada');
    else
        figure
        imshow(newImage)
        title('Imagen bien Calificada')
    end

end

% Comprobacion Calificacion imagenes con defectos
input_folder_new = 'PRUEBA PROCESADAS/CON DEFECTO';

% Listar los archivos de imagen en la carpeta
image_files_new = dir(fullfile(input_folder_new, '*.jpg'));  % Cambia la extensión
si es necesario


% Iterar sobre cada imagen en la carpeta
for i = 1:length(image_files_new)
    % Leer la imagen
    file_name = image_files_new(i).name;
```

```
    file_path = fullfile(input_folder_new, file_name);
    image = imread(file_path);
    newImage = imresize(image, inputSize(1:2)); % Redimensionar la imagen

    % Convertir la imagen a RGB si es necesario
    if size(newImage, 3) == 1
        newImage = repmat(newImage, [1 1 3]);
    end

    % Clasificar la nueva imagen
    YPredNew = classify(netTransfer, newImage);
    disp(['La nueva imagen es clasificada como: ', char(YPredNew)]);

    % Mostrar la nueva imagen con el resultado
    if char(YPredNew)== 'SIN DEFECTO'
        disp('Imagen mal calificada, no encuentra defectos cuando si tendria que
hacerlo');
        figure;
        imshow(newImage);
        title('Imagen mal Calificada');
    else
        figure
        imshow(newImage)
        title('Imagen bien Calificada')
    end
end
```

7.3.2 Clasificación de defectos

A continuación, se muestra el código de MATLAB para implementar la red ResNet50 en la clasificación de 4 tipos de defectos diferentes.

Se crea un ImageDatastore de la carpeta donde están las imágenes divididas en 4 subcarpetas: DEFECTO ALEATORIO, PICADURA, RAYADO LONGITUDINAL Y RAYADO TRANSVERSAL. Cada carpeta representa las diferentes clases de defectos.

```
dataFolder = 'TIPO DEFECTO PROCESADAS'; % Ruta a la carpeta de imágenes
imds = imageDatastore(dataFolder, 'IncludeSubfolders', true,'LabelSource',
'foldernames');
```

Se dividen las imágenes que van a ser utilizadas para entrenamiento y validación de la red. El 80% de las imágenes pertenecerán al conjunto de entrenamiento y el 20% pertenecerán al conjunto de validación

```
% Dividir los datos en subconjuntos de entrenamiento, validación y prueba
[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.8, 'randomized');
```

Se carga la red y se reemplaza la capa fullyconected y la capa de clasificación para adaptar la red al problema multiclase de 4 categorías.

```
% Cargar la red ResNet50 preentrenada
net = resnet50;

% Modificar las capas finales para 4 clases de defectos
lgraph = layerGraph(net);

% Reemplazar la capa totalmente conectada
numClasses = 4; % Número de clases de defecto
newFc = fullyConnectedLayer(numClasses, 'Name', 'new_fc', 'WeightLearnRateFactor',
10, 'BiasLearnRateFactor', 10);
lgraph = replaceLayer(lgraph, 'fc1000', newFc);

% Reemplazar la capa de clasificación
newClassLayer = classificationLayer('Name', 'new_classoutput');
lgraph = replaceLayer(lgraph, 'ClassificationLayer_fc1000', newClassLayer);
```

Se especifican los parámetros de entrenamiento y se entrena la red.

```
% Especificar opciones de entrenamiento
inputSize = net.Layers(1).InputSize;
augimdsTrain = augmentedImageDatastore(inputSize(1:2), imdsTrain);
augimdsValidation = augmentedImageDatastore(inputSize(1:2), imdsValidation);

options = trainingOptions('sgdm', 'MiniBatchSize', 64, 'MaxEpochs', 8 ,
'InitialLearnRate' , 0.0005 , ...ValidationData', augimdsValidation,
'ValidationFrequency' , 5, 'Verbose', false, 'Plots', 'training-progress',
'Shuffle', 'every-epoch');

%Entrenar la red
trainedNet = trainNetwork(augimdsTrain, lgraph, options);
```

Se evalúan los resultados obtenidos.

```
% Evaluar el rendimiento en el conjunto de validación
YPred = classify(trainedNet, augimdsValidation);
YValidation = imdsValidation.Labels;

accuracy = mean(YPred == YValidation);
disp(['Precisión en el conjunto de validación: ', num2str(accuracy*100), '%']);
```

Finalmente se evalúa el conjunto de prueba.

```
% Comprobacion Calificacion imagenes sin defectos
% Directorio donde están las imágenes
input_folder_new = 'TIPO DEFECTO PRUEBA PROCESADAS/RAYADO TRANSVERSAL';

% Listar los archivos de imagen en la carpeta
image_files_new = dir(fullfile(input_folder_new, '*.jpg'));  % Cambia la extensión
si es necesario

% Iterar sobre cada imagen en la carpeta
for i = 1:length(image_files_new)
    % Leer la imagen
    file_name = image_files_new(i).name;
    file_path = fullfile(input_folder_new, file_name);
    image = imread(file_path);
    newImage = imresize(image, inputSize(1:2)); % Redimensionar la imagen

    % Convertir la imagen a RGB si es necesario
    if size(newImage, 3) == 1
        newImage = repmat(newImage, [1 1 3]);
    end

    % Clasificar la nueva imagen
    YPredNew = classify(trainedNet, newImage);
    disp(['La nueva imagen es clasificada como: ', char(YPredNew)]);


end
```